Filière Actuariat ESSEC-ISUP **ESSEC / UPMC**

MÉMOIRE N° 2017 - 03

Mémoire présenté le mercredi 12 juillet 2017

par

Monsieur Juan José FRANCISCO MIGUÉLEZ

Diplômé de l'ESSEC en Juillet 2016
& étudiant en 3ème année d'Actuariat de l'ISUP pour l'obtention du

Titre d'Actuaire

délivré par l'Institut des Actuaires

sur le sujet :

Support Vector Machines: Machine Learning, the SVM algorithm and applications in Health Insurance Pricing

devant un jury composé de :

Emmanuel DUBREUIL

Magali Mahla Vigon

Marc ~~JUILLARD~~ (Pour ma déconfiture)

Marie KRATZ

M. Kratz

Note obtenue: 16 / 20

Responsable du stage dans l'entreprise :
Nicolas Wesner
Mazars Actuariat

CONFIDENTIEL ~~NON~~ OUI

En foi de quoi,

le Titre d'Actuaire lui a été décerné
~~ne lui a pas été décerné~~

INSTITUT DES ACTUAIRES

ESSEC BUSINESS SCHOOL

Mémoire présenté le 12 Juillet 2017
en vue de l'obtention du titre d'Actuaire de l'Institut des Actuaires
par **Juan José Francisco Miguélez**

suite à son stage effectué dans le cadre de la **filière Actuariat ESSEC-ISUP**

"Support Vector Machines: Machine Learning, the SVM algorithm and applications in Health Insurance Pricing"

Confidentialité ☒ NON ☐ OUI (durée : ☐ 1 an ☐ 2 ans)

Les signataires s'engagent à respecter la confidentialité indiquée ci-dessus.

Membres présents du jury de l'Institut des Actuaires :

- *Marie KRATZ*
-
-

- *Emmanuel DUBREUIL*

- *Marc JUILLARD*

- *Magali KELLE VIGON*

Entreprise :
Mazars Actuariat
Directeur du mémoire en entreprise :
Nicolas Wesner

Invité :

Signature :

Autorisation de publication et de mise en ligne sur site de diffusion de documents actuariels *(après expiration de l'éventuel délai de confidentialité)*

Signature du responsable entreprise :

ESSEC Business School

Memoir for the obtention of the
French Actuarial Qualification

Support Vector Machines:

Machine Learning, the SVM algorithm and applications in Health Insurance Pricing

Juan José Francisco Miguélez

supervised by:
Prof. Marie Kratz – ESSEC Business School
Dr. Nicolas Wesner – Mazars Actuariat

November 7, 2017

Abstract

The astonishing development of digital technologies coupled with the proliferation of measurement tools in all facets of life has provoked an explosion of the quantity of human-generated data over the last 20 years. In addition, the complexity and uncertainty of modern societies, resulting from phenomena such as demographic growth or economic globalization, have steadily increased the need for insurance arrangements, a trend which is expected to continue into the foreseeable future. Insurance providers have thus the opportunity to implement modern data analysis to leverage all biometric, economic and social data available in order to improve their pricing and risk management processes and cope with the increasing uncertainty they are facing.

This memoir is an attempt to contribute to this effort by exploring one particular data analysis model coming from the field of Machine Learning: we study the theory of Support Vector Machines and examine their applicability to the problem of pricing Health Insurance contracts in the French market.

First we introduce the discipline of Machine Learning to the reader. Starting with an historical perspective of the field, we then develop a cartography of Machine Learning algorithms based on learning styles and problems addressed. After comparing the discipline to sister fields such as Computer Science and Statistics, we introduce the formal mathematical theory underpinning Machine Learning. We finish off this chapter by discussing the opportunities as well as the threats of Machine Learning for the insurance business.

The following chapter is dedicated to the theory of Support Vector Machines and the analysis of their strong theoretical foundations. The theoretical presentation is followed by a review of algorithms and numerical techniques that enable the practical implementation of the Support Vector Machine model, such as Sequential Minimal Optimization or Stochastic Gradient Descent.

Chapter 3 is dedicated to the mechanics of the insurance industry and the presentation of the Health Insurance dataset we will be working with respectively. This chapter enables us to substantiate our modelling approach.

Finally, in chapter 4 we describe a series of experiments we have undertaken to assess both the predictive and the computational performance of the Support Vector Machine, which we benchmark against a cutting-edge Machine Learning algorithm, the Random Forest. Our findings show that, despite their theoretical soundness, Support Vector Machines are outperformed by Random Forests both in terms of computational cost and predictive power. We also test an ensemble linear model in which we combine the predictions of both the optimal Support Vector Machine and the optimal Random Forest.

Résumé

Le développement spectaculaire des nouvelles technologies numériques, combiné à la prolifération des instruments de quantification dans tous les aspects de la vie, ont provoqué sur les 20 dernières années une explosion de la quantité de données disponibles. En outre, la complexité et l'incertitude qui caractérisent les sociétés modernes, résultant de phénomènes divers tels que la croissance démographique ou la mondialisation, n'ont cessé d'augmenter les besoins des institutions et des personnes en produits d'assurance, une tendance que l'on s'attend à voir continuer. Les compagnies d'assurance ont donc aujourd'hui l'opportunité d'implémenter des techniques statistiques modernes pour exploiter efficacement toutes ces données biométriques, économiques et sociales, de façon à améliorer leurs processus de tarification et de gestion du risque et faire face avec succès à l'incertitude croissante.

Ce mémoire vise à contribuer à cet effort en analysant un modèle particulier issu de l'apprentissage automatique : on étudie la théorie des Machines à Vecteurs Support et on examine leur applicabilité à un problème de tarification en assurance santé sur le marché français.

Nous introduisons premièrement la discipline de l'apprentissage automatique. Partant d'une perspective historique, nous développons une cartographie des algorithmes d'apprentissage automatique basée sur les styles d'apprentissage et les problèmes qu'ils tentent de résoudre. Après avoir comparé la discipline à des domaines similaires tels l'informatique ou les statistiques, nous introduisons la théorie mathématique formelle qui dote de bases théoriques l'apprentissage automatique. Nous finissons ce chapitre par une discussion des opportunités et des menaces de l'apprentissage automatique pour l'industrie assurantielle.

Le chapitre suivant est consacré à la théorie des Machines à Vecteurs Support. Cette présentation est suivie d'une revue des algorithmes et des techniques numériques qui permettent l'application pratique des modèles de type Machines à Vecteurs Support, tels que l'algorithme de Minimisation Séquentielle ou l'algorithme de la Descente de Gradient Stochastique.

Le chapitre 3 est dédié à l'explication des mécanismes de l'industrie assurantielle ainsi qu'à la présentation de l'échantillon de données d'assurance santé sur lequel nous réaliserons des expériences. Ce chapitre nous permet de justifier notre approche de modélisation dans le dernier chapitre.

Finalement, nous décrivons au 4ème chapitre une série d'expériences que nous avons réalisés pour évaluer la performance prévisionnel ainsi que computationnelle des Machines à Vecteurs Support, que nous comparons à un algorithme de pointe en apprentissage automatique, les Forêts Aléatoires. Nos résultats montrent que, malgré leurs bonnes propriétés théoriques, les Machines à Vecteurs Support sont dépassées par les Forêts Aléatoires en termes de coût computationnel et de capacité prévisionnelle. Nous testons également un modèle linéaire de type *ensemble* qui nous permet de combiner efficacement les prévisions optimales de la Machine à Vecteurs Supports et de la Forêt Aléatoire.

Acknowledgements

First and foremost I want to thank the Actuarial department of Mazars for their support and making this memoir possible. I want to specifically thank Dr. Nicolas Wesner, my memoir company tutor, for suggesting me such an exciting subject and following through it despite my departure to the United Kingdom.

I thank ESSEC Business School for its outstanding technical teaching, which allowed me to improve my statistical knowledge hereby making possible the writing of this memoir. In this respect, I specifically thank Prof. Marie Kratz, my memoir academic tutor, for her classes at ESSEC as well as for encouraging me to enrol in the Actuarial track.

I also want to thank the faculty of the Institute of Statistics of Paris VI (ISUP) where I spent one year to follow actuarial and statistical classes and where I acquired an incredible amount of knowledge.

I also thank my friend Yunjing Xu who provided me with precious advice and tips on preparing and defending an actuarial memoir.

I thank my friend Borja Rivier whose knowledge on computing, Machine Learning and the Python programming language has been invaluable in successfully completing this memoir.

I want to thank the Stack Exchange community, particularly the Stack Overflow, Latex and Cross Validated sub communities, for their help on specific issues I encountered throughout the preparation and the writing of the memoir.

Finally, I thank my parents for their continued support throughout the writing of this work.

Contents

Contents **1**

1 Introduction to Machine Learning **6**
1.1 Historical background of Machine Learning . . . 6
1.2 What is Machine Learning? . . . 7
1.2.1 The learning process . . . 8
1.2.2 A cartography of Machine Learning algorithms . . . 10
1.2.3 Machine Learning and its related disciplines . . . 12
1.3 Introduction to Statistical Learning Theory . . . 14
1.3.1 Motivations behind Statistical Learning Theory . . . 14
1.3.2 Assumptions and formalization . . . 15
1.3.3 Brief introduction to generalization bounds . . . 16
1.4 Machine Learning in insurance . . . 19
1.4.1 The opportunities offered by Machine Learning . . . 19
1.4.2 The problems raised by Machine Learning . . . 20

2 Support Vector Machines **21**
2.1 Introductory comments . . . 21
2.2 The Support Vector Machine algorithm . . . 22
2.2.1 The Linear Maximal Margin classifier . . . 22
2.2.2 From Linear Maximal Margin classifiers to the Support Vector Machine . . . 24
2.2.3 The place of Support Vector Machines within Machine Learning . . . 33
2.3 Implementing Support Vector Machines . . . 34
2.3.1 Sequential Minimal Optimization . . . 35
2.3.2 Stochastic Gradient Descent . . . 37
2.3.3 Kernel approximation . . . 39

3 The Health Insurance Pricing Problem **41**
3.1 Introductory comments . . . 41
3.2 Principles of Non-Life insurance pricing . . . 41
3.2.1 Basic mechanics of the insurance industry . . . 41
3.2.2 Formalization of the insurance pricing problem . . . 42
3.2.3 The specific case of health insurance and the reimbursement principle . . . 43
3.3 The French health insurance system: the market and its participants . . . 45
3.4 Structure and main characteristics of the dataset . . . 47
3.4.1 Pretreatment of data . . . 47
3.4.2 Splitting data between a training set and a validation set . . . 48
3.4.3 Attributes of insurance policies . . . 48
3.4.4 Characteristics of the portfolio . . . 50
3.5 Preliminary study of the relationship between attributes and insurance claims . . . 54
3.5.1 The age effect . . . 54
3.5.2 The contract option effect . . . 57
3.5.3 The beneficiary and gender effect . . . 58

3.5.4 The contract effect . . . 58

4 Applying Support Vector Machines to Health Insurance 60
4.1 Introductory comments . . . 60
4.2 IT architecture . . . 60
4.2.1 IT general specifications . . . 60
4.2.2 Computation complexity of the Support Vector Machine . . . 61
4.2.3 Remote computing architecture . . . 62
4.2.4 Computational performance analysis . . . 64
4.3 Data preprocessing . . . 64
4.3.1 Categorical attributes binarization . . . 65
4.3.2 Quantitative attributes scaling . . . 65
4.4 General procedure for model specification and estimation . . . 66
4.4.1 Model specification . . . 66
4.4.2 Model estimation . . . 67
4.5 A Support Vector Machine for predicting occurrence . . . 68
4.5.1 Problem setting . . . 68
4.5.2 Analysis of results . . . 68
4.6 A Support Vector Machine for predicting cost . . . 75
4.6.1 Problem setting . . . 75
4.6.2 Analysis of results . . . 75
4.6.3 The contract effect: different models for different contracts . . . 79
4.6.4 Merging concepts: an Ensemble Linear Regression model . . . 80
4.7 Validation of results . . . 81

5 Conclusions and future work 83
5.1 Concluding remarks . . . 83
5.1.1 The Support Vector Machine model . . . 84
5.1.2 IT Infrastructure . . . 84
5.2 Future work . . . 85
5.2.1 Classification problems with unbalanced classes . . . 85
5.2.2 Probability calibration . . . 85
5.2.3 Algorithmic precision and computational efficiency . . . 86
5.2.4 Exploring ensemble models . . . 86

Bibliography 88

A Introduction to convex optimization 93
A.1 Main definitions and an important theorem . . . 93
A.2 The Karush-Kuhn-Tucker Theorem . . . 94
A.3 Duality in convex optimization problems . . . 95

B Mathematical analysis of the SVM 98
B.1 Study of the SVM optimization program . . . 98
B.1.1 Fundamental properties of the SVM problem . . . 98
B.1.2 Derivation of the simplified dual formulation . . . 100
B.1.3 Derivation of the decision function and the slack variables . . . 102
B.1.4 Characterisations of SVM solutions . . . 104
B.2 Uniqueness of the SVM solution . . . 105
B.2.1 Uniqueness of the primal solution . . . 105
B.2.2 Uniqueness of the dual solution . . . 106
B.3 The VC dimension of Support Vector Machines . . . 107

C The Sequential Minimal Optimization algorithm **110**
C.1 Derivation of the updated values of the working set 110
C.2 Derivation of the bounds L and H 112

D Overview of the Random Forests algorithm **113**
D.1 Decision trees 113
D.2 From decision trees to Random Forests 114

List of Abbreviations

kNN	k-Nearest Neighbor
AI	Artificial Intelligence
ALD	*Affectation de Longue Durée*
ANN	Artificial Neural Network
AODE	Average One-Dependence Estimator
AWS	Amazon Web Services
CART	Classification And Regression Tree
CHAID	CHi-squared Automatic Interaction Detector
CMU	*Couverture Maladie Universelle*
CNAMTS	*Caisse National de l'Assurance Maladie des Travailleurs Salariés*
CNN	Convolutional Neural Network
CSBM	*Consommation de Soins et de Biens Médicaux*
DBM	Deep Boltzmann Machine
DREES	*Direction de la Recherche, des Etudes, de l'Evaluation et des Statistiques*
EM	Expectation Maximisation
EMR	Empirical Risk Minimization
FDA	Flexible Discriminant Analysis
GLM	Generalized Linear Models
IEEE	Institute of Electrical and Electronics Engineers
INSEE	*Institut National de la Statistique et des Etudes Economiques*
KKT	Karush-Kuhn-Tucker
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
LOWESS	Locally Weighted Scatterplot Smoothing
LVQ	Learning Vector Quantization
ML	Machine Learning
MSE	Mean Squared Error

OLSR	Ordinary Least Squares Regression
P&C	Property & Casualty
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
QP	Quadratic-Programming
RAM	Random-Access Memory
RBF	Radial Basis Function
RF	Random Forest
SCP	Secure Copy
SGD	Stochastic Gradient Descent
SLT	Statistical Learning Theory
SMI	Supplementary Medical Insurance
SMO	Sequential Minimal Optimization
SOM	Self-Organizing Map
SSH	Secure Shell
SVC	Support Vector Clustering
SVM	Support Vector Machine
SVR	Support Vector Regression
VC	Vapnik-Chervonenkis

Chapter 1

Introduction to Machine Learning

From the dawn of Artificial Intelligence to modern Learning Theory

1.1 Historical background of Machine Learning

Machine Learning finds its roots in Artificial Intelligence (AI) research, which begun at the end of World War II and went extremely popular during the fifties and the first half of the sixties. When it started developing in the sixties, it was essentially the branch of AI concerned with devising methods – algorithms – to allow machines to learn from data. Since the onset of the discipline, researchers stressed particularly the notion of performance improvement, as more data should enable algorithms to improve their accuracy.

In the first years of AI and Machine Learning and following Turing's article "Computing machinery and intelligence" from 1950 (Turing, 1950), where he stated that AI should strive to "*carry through the organization of an intelligent machine with only two interfering inputs, one for pleasure or reward, and the other for pain or punishment*", the focus was on *Reinforcement* learning – we will discuss in greater detail different learning styles below – which succinctly aims at "*maximizing the sum along time of whatever rewards presented to the* [machine] *by the environment*" (Sebag, 2014).

In the second half of the sixties, a series of drawbacks affected popular AI and Machine Learning techniques, such as the Perceptron algorithm (Rosenblatt, 1957), which were showed to be much more limited than initially thought, while at the same time researchers started stumbling over the computational limitations of their time. As a consequence, AI underwent a transformation which resulted in a more pragmatic, modest and fragmented discipline.

At the same time, these drawbacks induced a shift in the approach of some researchers: AI experts realised that most of the AI undertook in the previous years, based on logic and knowledge representation, was deeply limited by the initial amount of knowledge a machine was equipped with. As a consequence, it became crucial to develop techniques that would be able to interact with the world, and as interacting with the world involves gathering data, some researchers started to shift their focus from logic to statistics.

These factors contributed to an increased independence of Machine Learning from its parent field of AI during the period 1980-1990:

- AI was essentially based on logic and symbolic representation of knowledge; as a consequence the field came to be dominated by *expert systems* – machines that imitate the decision-making capabilities of a human being in a particular field. On the other hand, Machine Learning had been gradually shifting its focus from logic to statistics in order to overcome limitations of early Machine Learning methods.
- The influence of pattern recognition techniques on Machine Learning drove the discipline to

specialize in regression and classification problems, which were the traditional problems faced by the pattern recognition community. As a consequence, traditional AI challenges such as reasoning or problem solving were set aside.

- The nature of the problems tackled changed, such that Machine Learning started addressing mostly well defined and restricted tasks through classification and regression. These simpler tasks lead the Machine Learning community to focus on *Supervised* and *Unsupervised* learning styles, which significantly differ from Reinforcement learning as they deal with an environment which is almost never modified during the learning process, and in any case never by the machine itself.

- The adoption of statistical and probabilistic techniques, as well as the availability of increasing amounts of data, resulted in another notable paradigm shift: from the original goal of learning quickly from very few examples like humans do, the Machine Learning community became interested in learning from extremely large data sets.

In addition, the eighties and nineties also saw the emergence of formal probabilistic theories underpinning Machine Learning, which developed independently from most of the research being done in algorithms:

- **The Probably Approximately Correct (PAC) framework**: proposed by L. Valeant in 1984 (Valeant, 1984), PAC theory is concerned with determining whether a given function f can be approximated by a function $\widehat{f}$ with an arbitrary accuracy and in an efficient way – both in terms of time and space.

- **Statistical Learning Theory (SLT)**: developed by Vapnik in the nineties, SLT builds on two distinct notions, the *empirical error* – the error between the predictions of our model and the data used to fit it – and the *generalization error* – the error between the predictions of our model and the expected error based on the whole unknown population. The goal of SLT is to study the relationship between the empirical error and the generalization error, particularly what can be inferred from the latter given the former.

By the year 2000, Machine Learning had become virtually a discipline of its own, spanning different subcategories – supervised machine learning, reinforcement machine learning, etc. – which were more or less close to the original nature of the field. Since its first steps in the 1950-1960 era, it has achieved success in many different areas, such as *speech recognition* and *computer vision*.

In recent years, Machine Learning has been rapidly gaining popularity thanks to the advent of the digital era: the improvement in computational power and the availability of vast amounts of data have made decision making techniques based on statistical tools an appealing instrument for domains such as marketing, finance, clinical research, etc. As the increase in available data is due to continue, Machine Learning is expected to develop further in the years to come.

1.2 What is Machine Learning?

Nowadays, Machine Learning has become a scientific field at the crossroads of statistical theory, mathematical optimization and computer science, concerned with developing techniques and algorithms that can learn from previous experience or data and possibly perform better on subsequent situations.

In the following paragraphs, we explore the notion of Machine Learning in detail to get a better understanding of the problems it tries to tackles and the techniques it implements to do so. Our discussion will focus on three points:

- Understanding what the term "learning" means in this environment and what are the different types of learning; we will particularly look at the three main types of learning found in the literature.
- Presenting Machine Learning methods and algorithms by similarity; this subsection will enable us to establish a detailed cartography of Machine Learning techniques as well as better understand the problems tackled and the techniques used by the Machine Learning community.
- Analysing the interactions and overlapping between Machine Learning and other closely related disciplines such as Statistics and Computer Science.

1.2.1 The learning process

First and foremost, it is important to clarify the concept of "learning" and to consider the different kinds of learning that are studied and implemented by the machine learning community.

In order to grasp the meaning of this word in the context of Machine Learning, it is useful to look at some definitions of the field given by people dedicated to it:

- Simon (1983) explained that the goal of any machine learning technique should be to improve the performance of a given task.
- Mitchell (1997) gave a formal definition of a learning problem:

  > "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*"

- In a book published in 2010, Alpaydin (Alpaydin, 2010) defines machine learning as "*programming computers to optimize a performance criterion using example data or past experience*".

From these definitions, learning could be interpreted in two ways:

- A machine could learn *statically*, e.g. learn how to divide data in two distinct groups from an available, fixed set of data – this interpretation would be closer to the 3$^{\text{rd}}$ definition above;
- It could also learn *dynamically*, improving its own performance and adapting its behaviour as new data becomes available – this interpretation would be closer to the 2$^{\text{nd}}$ definition above.

These interpretations are also to be found in a paper by Mitchell from 2006 (Mitchell, 2006). In his article, he gives two examples of learning tasks: autonomous robot cars that leverage their experience – i.e. recorded data – to improve their driving proficiency; extracting data from previous medical dossiers to predict how patients will react to specific treatments. His first example fall in the second category, which we designated by the name of dynamic learning, while the second one corresponds to what we called static learning.

Finally, we can refine the classification of learning styles by considering the two fundamental notions by which learning is achieved: data and algorithms. Data is the substance from which machines can learn something, while algorithms structure and guide the learning process. Hereafter we present the classical cartography of machine learning algorithms and techniques according to their learning style and the structure of the data they use, which is the one found in the literature:

- **Supervised learning**: in this setting, the data used by the algorithm is labelled. For example, we could have a training set of data composed of medical, biological and economic information about individuals – attributes or features – and whether they are or not affected by a given disease such as cancer – labels. The algorithm should learn how to identify afflicted individuals by analyzing the relationship between attributes and labels with the aim of making predictions.

The algorithm is said to be *supervised* because our training data is a set of examples $X_1, ..., X_n$ with their correct answers $Y_1, ..., Y_n$ from which the algorithm should learn.
This learning style is static: the algorithm learns from the data at hand the best parameters in order to make future predictions.
Linear and Logistic Regression are examples of supervised learning methods.

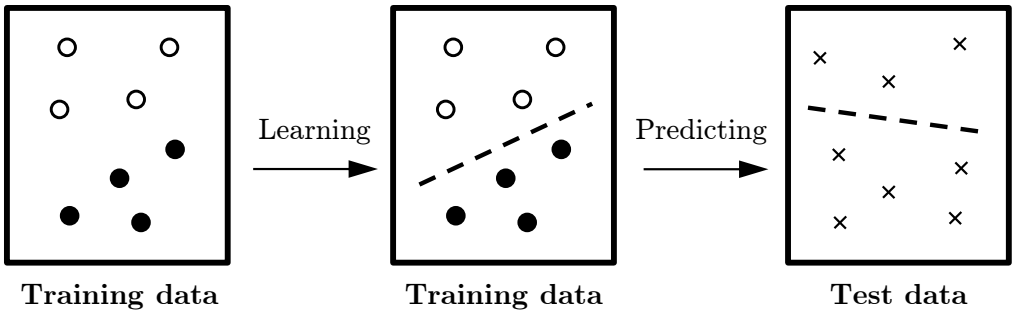

Figure 1.1: *The Supervised Learning principle*

- **Unsupervised learning**: in this setting, there is no label or result associated to the data, so we only have at our disposal input data $X_1, ..., X_n$. The goal of unsupervised learning algorithms is to bring to light patterns and structures underlying the data. For example, we could have data on voters of a given political party and we would like to know if we could cluster them in different socio-economic groups.
  Unsupervised learning is often used as a preparation step, in order to structure data that will subsequently be analyzed with supervised learning methods.
  Principal Component Analysis (PCA) and the k-Means algorithm are types of unsupervised learning.

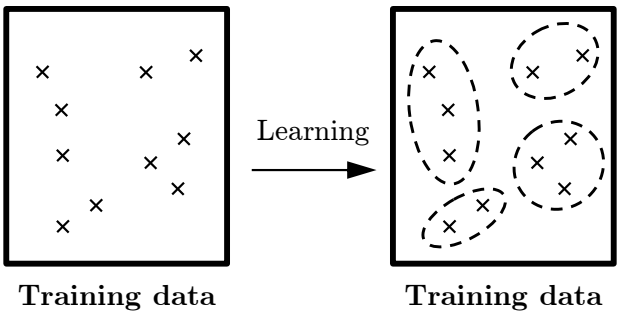

Figure 1.2: *The Unsupervised Learning principle*

- **Reinforcement learning**: in this setting, the output is a series of actions $a_1, ..., a_n$ and the aim of the algorithm is to determine the sequence of actions – known as a policy – which allows reaching a particular goal, e.g. winning a game. In this learning framework, the algorithm should be able to assess the goodness of its past actions according to rewards $r_1, r_2, ...$ and adapt its behavior accordingly to produce a good policy.
  This learning style can be considered more dynamic than the previous two and closer to the field of Artificial Intelligence, as the algorithm has the ability to improve itself.

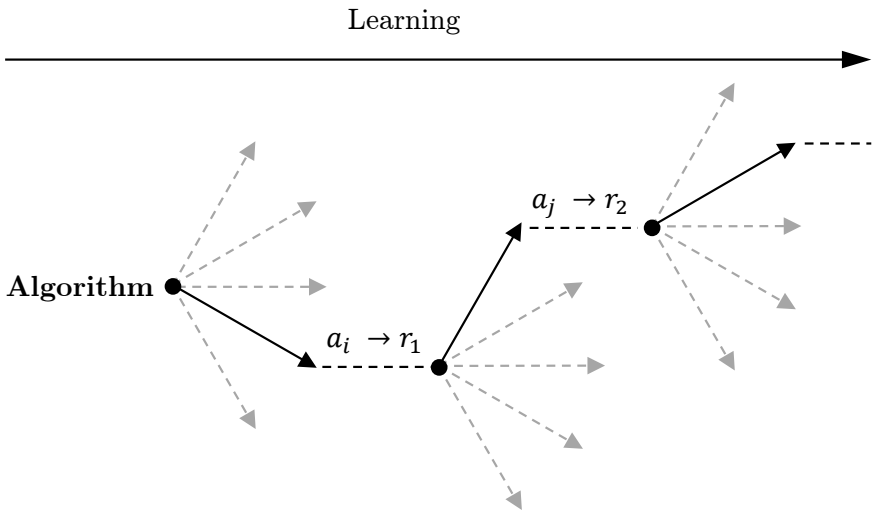

Figure 1.3: *The Reinforcement Learning principle*

Along these main learning styles, there are other subcategories which are more or less close to the three aforementioned, among which semi-supervised learning, learning-to-learn and developmental learning.

1.2.2 A cartography of Machine Learning algorithms

Another useful way to comprehend Machine Learning is to look at the mainstream algorithms and their functioning and classify them by similarity. This nomenclature should enable the reader to have a better hindsight on what range of techniques have been used in Machine Learning, as well as what kind of problems – regression, classification, etc. – the machine learning community has been trying to solve (Brownlee, 2013):

- **Regression algorithms**: these kinds of algorithms are concerned with modelling the relation f between a variable X and a variable Y. The accuracy of the modelled relation is assessed by defining an error measure Err, such as quadratic error, that we seek to minimize. They are probably the oldest style of machine learning technique, as first works on Ordinary Least Squares Regression (OLSR) were undertaken by Legendre and Gauss and date back to the XIX$^{\text{th}}$ century (Legendre, 1805; Gauss, 1809). Classical examples of such algorithms are Linear Regression, Logistic Regression and Local Regression (LOWESS).

- **Clustering algorithms**: these techniques aim at structuring and organizing data by creating homogenous clusters of observations. Examples of such algorithms are: k-Means, k-Medians, Expectation Maximisation (EM).

- **Bayesian algorithms**: here we group all techniques that explicitly exploit Bayes theorem; among those we can cite Naïve Bayes, Bayesian Networks and Averaged One-Dependence Estimators (AODE).

- **Decision Trees algorithms**: such algorithms construct decision trees based on data. A popular example of decision trees algorithms is the Classification And Regression Tree (CART).

- **Dimensionality Reduction algorithms**: the goal of these algorithms is to describe data with less information, determining which aspects are relevant and which are not by discovering the underlying structure of data. Very known examples are PCA, Linear Discriminant Analysis (LDA) and Flexible Discriminant Analysis (FDA).

- **Instance-based algorithms**: also known as "memory-based algorithms", these algorithms work by memorizing critical or important observations of the data – instances – in order to subsequently compare new data to these instances and assess their similarity to make a decision. A popular example of this technique is k-Nearest Neighbor (kNN). Other algorithms are Learning Vector Quantization (LVQ) or Self-Organizing Map (SOM).
- **Artificial Neural Networks (ANN) algorithms**: ANN have already a long history, as the first work on this technique was undertaken by McCulloch and Pitts in 1943 (McCulloch and Pitts, 1943). The ANN category includes hundreds of different algorithms which try to replicate the functioning of biological neural networks and which are particularly used to model functions or distributions $f : X \to Y$. Indeed, the Universal Approximation Theorem (Cybenko, 1989; Hornik, 1991) states that fairly simple neural networks can approximate continuous functions. Examples of such algorithms are the Perceptron algorithm and Radial Basis Function Networks (RBFN).
- **Association Rule Learning algorithms**: they work by devising rules of the form $X \Rightarrow Y$ – i.e. implications – that help explain the relationships between variables. The Apriori algorithm and the Eclat algorithm can be included in this category.
- **Deep Learning algorithms**: deep learning techniques are a complex variant of ANN which relies heavily on vast computations. Some examples are the Deep Boltzmann Machine (DBM), the Convolutional Neural Network (CNN) and the Stacked Auto-Encoders.
- **Regularization algorithms**: this category groups algorithms that are based on other techniques, such as Regression, but which have been modified in order to control a model's complexity. They allow to tradeoff between accuracy and simplicity. Popular examples of these algorithms are Ridge Regression and Least Absolute Shrinkage and Selection Operator (LASSO).
- **Ensemble algorithms**: these algorithms combine together multiple weaker models; predictions from each sub-model are then combined in order to output a unique, final prediction. Researchers in this field look particularly at which models to combine and how to combine them. We can cite Random Forests, Bagging and AdaBoost as examples of this category.

The graph below displays the 11 eleven categories aforementioned as well as a comprehensive list of algorithms belonging to them (Brownlee, 2013).

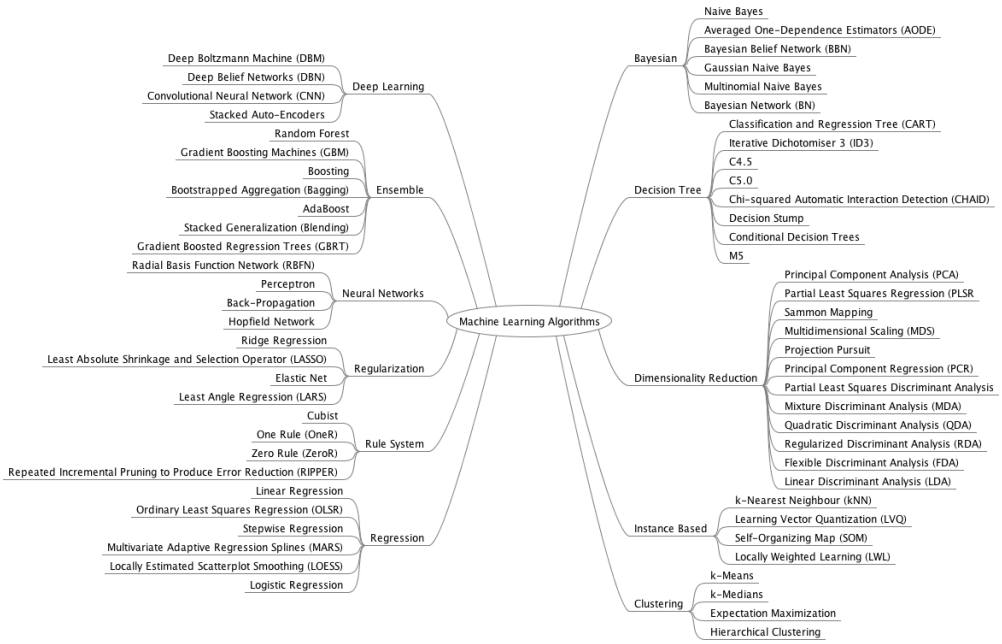

Figure 1.4: *A possible cartography of Machine Learning algorithms*

For a more specific account on successful and modern Machine Learning algorithms, the reader might refer to Wu, Kumar, Quinlan, Ghosh, Yang, Motoda, McLachlan, Ng, Liu, Yu, Zhou, Steinbach, Hand and Steinberg (2008) where the authors go through the most influential algorithms as identified by the IEEE International Conference on Data Mining in 2006.

1.2.3 Machine Learning and its related disciplines

To hone our comprehension of the Machine Learning field we have analysed the different learning styles it encompasses as well as traditional input data structures and mechanisms of different algorithms. Now we are going to explore the frontiers between Machine Learning and its related disciplines.

We stated above that Machine Learning was at the crossroad of statistics and computer science. To gain a better understanding of the discipline, it is important to pinpoint what the common points between these fields are and what the characteristics that set them apart are, all the more so that Machine Learning is deeply intertwined with both statistical theory and computer science.

The article from Mitchell (2006) is again very handy, this time to grasp the particularities of these three notions. He defines each one of them by the question they seek to answer:

- **Computer Science**: "*How can we build machines that solve problems, and which problems are inherently tractable/intractable?*"
- **Statistics**: "*What can be inferred from data plus a set of modelling assumptions, with what reliability?*"
- **Machine Learning**: "*How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?*"

These questions should serve as a base to reflect on the similarities and the differences between Machine Learning and its sister disciplines.

Machine Learning and Computer Science

From the above questions, we immediately see that Machine learning seeks to bring *autonomy* to computers: it seems as if the implicit ultimate goal of Machine Learning is to enable computers to program themselves, thus the term *machine* learning.

Moreover, Mitchell's article implies that Machine Learning brings *flexibility* to computers, as he highlights the following two examples in software development where Machine Learning methods are state-of-the-art:

- The desired software is so complex that a person would be unable to create an application to perform it – for example, developing an algorithm which would recognize your own picture.
- The software needs some degree of customization depending on the environment where it is going to be deployed, the person that is going to use it, etc.

Overall, Machine Learning strives to overcome the traditional rigidity of computer programming by bringing in increased flexibility as well as autonomy from human supervision.

Machine Learning and Statistics

Based on Mitchell's defining questions, the distinction between Machine Learning and statistics is more diffuse. Mitchell speaks of the "*the fundamental laws that govern all learning processes*", which brings to mind the field of asymptotic theory and convergence results from Statistics; moreover, if

we assimilate the term "'*experience*" to "*data*" we notice that both statistics and Machine Learning seem to have the same basic material.

Moreover, Machine Learning algorithms such as random forests or Bayesian methods involve a lot of statistical and probabilistic modelling, while Machine Learning authors and practitioners claim classical statistical methods such as logistic and linear regression also belong to the field of machine learning.

In addition this difficulty is compounded by the emergence of the term "statistical learning", which designates the formal theory which underpins popular Machine Learning methods such as Support Vector Machines and which is rooted in probability and statistical theory.

What is therefore the difference between the fields of Machine Learning and statistics?

The most held view among specialists is that the difference between the statistics community and the Machine Learning community is *how each one approach problems*: one way to put it could be that statistics is a *scientifically-minded* discipline while Machine Learning is an *engineering-minded* discipline. Thus, globally speaking, statisticians might be more interested in good theoretical properties for their models, while machine learning specialists would stress the operational side of models.

To better understand how statisticians and machine learners differ in their approach, we have come up with a non-exhaustive list of methodological differences between these two communities based on different readings (Alpaydin, 2010; Sebag, 2014):

- **Methodology vs. implementation and computational aspects**: statisticians stress rigorous methodology and data collection.
  Machine Learning experts, which usually deal with high dimensional problems, prioritize decision-making and are thus interested in the practical aspects of implementing an algorithm and the related computational issues.

- **Explanatory vs. predictive approach**: much of the statistics community is interested in explaining relationships. For example, the coefficients of a linear regression model can help understand how and to what degree an independent variable X affects a dependent variable Y.
  By contrast, members of the Machine Learning community seem to be much more concerned about good predictive power and care less about the interpretability of the parameters of the model.

- **Parametrization vs. non-parametrization approach**: most statistics rely on parametric models, such that statisticians normally work with a given probability distribution that models the underlying phenomena they are studying.
  By contrast, many Machine Learning algorithms do not assume any explicit representation of the underlying stochastic process governing the phenomena at hand.

- **Sample size**: a lot of statistics relate to asymptotic theory, which deals with properties of statistical models and estimators when sample size grows to infinity.
  In Machine Learning, practitioners stress finite-sample properties.

Thus the difference between statistics and Machine Learning does not seem to be a fundamental one, but rather a formal one, specially given the extensive knowledge transfers that have taken place between these two communities:

- Machine Learning has extensively imported methods from statistics such as regression analysis and is now underpinned by formal theories rooted in statistics.

- Statisticians are growingly becoming interested in techniques, such as cross-validation, and concerns, such as computational efficiency, which aroused within the Machine Learning community.

These observations echo Mitchell's article: he stresses that, relatively to statistics, Machine Learning is concerned with issues that arise when implementing forecasting and predictive data-based techniques, such as time efficiency, data storage and computability.

1.3 Introduction to Statistical Learning Theory

In the previous sections we have given the reader a brief overview of the historical origins and main developments of the Machine Learning discipline. We have also tried to make the term "Machine Learning" more meaningful by presenting the different learning styles that exist as well as some of the main classes of Machine Learning algorithms. Finally, we have also made a comparative analysis of Machine Learning by highlighting the similarities and differences of the field with respect to some closely related disciplines such as computer science and statistics.

Hereafter, our goal is to give a succinct introduction to one of the main theoretical frameworks that underpin algorithmic Machine Learning: the Statistical Learning Theory which is also known as Vapnik-Chervonenkis (VC) theory.

1.3.1 Motivations behind Statistical Learning Theory

SLT can be included in a wider field which is usually designated by the name of Computational Learning Theory. Computational Learning Theory, which also includes the Probably Approximately Correct framework, is devoted to the mathematical analysis of Machine Learning algorithms.

These theoretical foundations formalize concepts such as *learning*, *generalization* and *over-fitting*. On top of this, they also try to tackle the following fundamental question:

> "*What general laws constrain inductive learning?*"

Answering this question should enable to better understand what makes a good algorithm and devise better ones or improve existing ones. In particular, the concept of *generalization* is thoughtfully addressed by learning theory. Indeed, the basic idea behind the design of any Machine Learning algorithm is to look for regularities in the studied phenomenon, which should enable to make generalizations and thus predict the future behaviour. In this sense, learning theory aims at answering the question:

> "*Given a Machine Learning model f_n which has been fitted to a training dataset s_n, how accurate can I expect f_n to be when applied to a test set independent of the training set?*"

Let assume we want to estimate the non-linear, functional relationship between a one-dimensional dependent variable y and a one-dimensional independent variable x, a task for which we have a data sample $\{x_i, y_i\}_{i=1,\dots,n}$. As it will be made formal latter on, generalization is achieved through two different mechanisms:

- **Data**: given that our model is fitted or "trained" on an available data set, we would like the error between our model's prediction and the training set to be as small as possible. However, this cannot be the only way to ensure generalization, because we could end up with a model which perfectly fits the data at hand but that is afterwards unable to make accurate predictions on new data: for example, we know we could fit a polynomial f_n of degree $d = n$ that passes through every point $\{x_i, y_i\}_{i=1,\dots,n}$, however the polynomial would probably have a bad performance on new data; the estimated polynomial would also tend to be highly variable: estimating it with a different sample $\{x'_i, y'_i\}_{i=1,\dots,n}$ would likely yield a very different set of polynomial coefficients. We call this problem *over-fitting*.
- **Knowledge**: to counter the over-fitting issue, the person implementing the algorithm must leverage its knowledge of the problem at hand to restrict the set of possible functions or *model*

$\mathcal{F}$ from which the algorithm will be picking a function f_n to predict the value of one variable from the value of the other variable. As such, it is always preferable to have a simpler model than a complex one, because a complex model always involve more explicit or implicit hypothesis about the phenomenon being modelled and, as the complexity of the model grows, the range of phenomenon it is able to represent diminishes. However, there is also the risk of excessively restricting the set of possible function: for $d = 1$, our polynomial would probably be very biased, yielding significant errors on the dataset $\{x_i, y_i\}_{i=1,\dots,n}$. This is problem is known as *under-fitting*.

Based on the polynomial example, the goal of SLT is to provide a set of tools and results that allows to find a fairly decent value for d, higher than 1 to avoid *high bias* – levering the data available – but lower than n to *limit the variance* – limiting the model complexity. In tackling generalization, SLT formalizes the notion of errors and model complexity and studies how these can be optimized in order to have Machine Learning algorithms with good generalization properties.

To address this and other issues, the fields of Computational Learning Theory, SLT and PAC study the mathematical properties of algorithms, for example:

- Deriving probabilities or probability bounds of successful learning.
- Determining meaningful and optimal sizes of training examples.
- Analysing the accuracy to which the target function is approximated.
- Studying the size and complexity of the hypothesis space.

Now that we have explained the motivations behind SLT, we will introduce the formal theory in a more classical way. Our aim here is not to give a comprehensive overview, as SLT is a very vast theory, but explain the formal framework and an important result due to Vapnik. We will follow the seminal book by Vapnik (1995), *The Nature of Statistical Learning Theory*.

1.3.2 Assumptions and formalization

We consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ such that each $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ is a pair of random variables distributed according to an unknown probability distribution $\mathbb{P}$. We are interested in discovering the hidden functional relationship f between the two random variables, $f : \mathcal{X} \to \mathcal{Y}$, but we can only observe a data sample of n independent and identically distributed (i.i.d.) observations $\{(X_i, Y_i)\}_{1 \leq i \leq n}$. Therefore our goal is to construct a learning machine $f_n : \mathcal{X} \to \mathcal{Y}$ – where the subscript n specifies that the machine depends on the sample available – which approximates the true relationship f.

We consider the *loss function* $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ which measures the discrepancy between the available sample $\{Y_i\}_{1 \leq i \leq n}$ and the machine's predictions $\{f_n(X_i)\}_{1 \leq i \leq n}$. In order to select an appropriate machine f_n, we want to minimize the expected loss, also called the *risk* of the machine:

$$R(f_n) = \mathbb{E}^{\mathbb{P}}\left[L\left(Y, f_n(X)\right)\right] = \int L(y, f_n(x)) d\mathbb{P}(x, y)$$

Hence, if we specify a class of functions $\mathcal{F}$, also known as *hypothesis space* or *model*, to which we assume the true relationship f belongs and from which we pick f_n, the optimal learning machine f_n^* would be the following:

$$f_n^* = \underset{f_n \in \mathcal{F}}{\arg\min}\, R(f_n)$$

The loss function L can have different specifications depending on the learning task. In the case

of the classical Support Vector Machine, we are facing a *binary classification problem* – also known as a pattern recognition task –: the space $\mathcal{Y}$ is equal to $\{-1, 1\}$ meaning that each variable X_i can belong to two different classes, which we codify through the numbers -1 and 1. In this setting the traditional loss functional is the *0-1 loss*:

$$L\left(Y, f_n(X)\right) = \mathbb{1}_{Y \neq f_n(X)}$$

The risk function simplifies to the probability of error:

$$R(f_n) = \mathbb{E}^{\mathbb{P}}\left[\mathbb{1}_{Y \neq f_n(X)}\right] = \mathbb{P}(Y \neq f_n(X))$$

The problem we face is that the true distribution $\mathbb{P}$ is unknown and the only information we have is the data sample. To overcome this limitation, we need to define an empirical metric computable from the data available which approximates the true risk. The usual metric utilised in the Machine Learning community is the *empirical risk* R_n which, in our binary classification setting, corresponds to the rate of error on the data sample – here again the subscript n denotes the dependence on the available data:

$$R_n(f_n) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{Y_i \neq f_n(X_i)}$$

The principle consisting on choosing a function $f_n \in \mathcal{F}$ based on minimising $R_n(f_n)$ is known as *Empirical Risk Minimization* (ERM). However, we are now faced with the following question: what relationship exists between the true and the empirical risk?

The true risk can be decomposed in the following form, such as to introduce a specific relationship to the empirical risk:

$$R(f_n) = R_n(f_n) + (R(f_n) - R_n(f_n))$$

We are interested in the difference $R(f_n) - R_n(f_n)$: SLT aims among other things at establishing probabilistic bounds that enable us to have a better understanding of this difference between the uncomputable true risk $R(f_n)$ of the machine and its empirical risk $R_n(f_n)$ – this type of bounds are called *generalization bounds*. If we have a probabilistic bound on the difference between true and empirical risk, it is possible to obtain an upper bound on the value of the true risk.

In the following pages, we will present some crucial concepts and an important result due to Vapnik and Chervonenkis – refer for example to Vapnik (1995) – so that the reader can gain some familiarity with the field of SLT probabilistic bounds. But before that we make a last observation: assuming now that function f_n maps to the space of real numbers $\mathbb{R}$ and that we classify observations to either class -1 or 1 depending on the sign of the function, it is important to note that the SVM is not calibrated through a 0-1 loss but instead through the so called *hinge loss*:

$$L\left(Y, f_n(X)\right) = \max\left[0, 1 - Y f_n(X)\right]$$

The hinge loss can be interpreted as a smoothed version of the 0-1 loss: whereas the latter yields 1 as soon as an error is made by the classification machine, the hinge loss will take into account the degree to which the function f_n is wrong.

1.3.3 Brief introduction to generalization bounds

We will work in the setting described above: letting $Z = (X, Y)$ and $Z_i = (X_i, Y_i)$, we have at our disposal a single data sample $s_n = \{z_1, ..., z_n\}$. We assume the true functional relating X to Y

belongs to a model $\mathcal{F}$ from which we want to pick a machine. Throughout this subsection we assume we are working on a binary classification problem – $Y \in \{-1, 1\}$ – and hence that $\mathcal{F}$ is a model for binary classifying functions – most of the results can be adapted to more general cases. Hence we can define the 0-1 *loss class* $\mathcal{L}$, which is the set of loss functions associated to machines from $\mathcal{F}$:

$$\mathcal{L} = \{L : (\mathbf{x}, y) \to \mathbb{1}_{f(\mathbf{x}) \neq y}\,,\ f \in \mathcal{F}\}$$

We highlight that functions from both $\mathcal{F}$ and $\mathcal{L}$ are bounded from below by 0 and from above by 1 – boundness is fundamental for the theorem we present in this section – and that there exists a bijection between $\mathcal{F}$ and $\mathcal{L}$.

We are interested in the following set:

$$\mathcal{F}_{z_1,...,z_n} = \{f(z_1), ..., f(z_n) : f \in \mathcal{F}\}$$

The set $\mathcal{F}_{z_1,...,z_n}$ represents the total number of ways a given sample s_n can be classified by functions from the model $\mathcal{F}$. The cardinality of this set is necessarily finite and lower than or equal to 2^n. If the cardinality is 2^n then the sample s_n can be classified in all possible ways by using functions from $\mathcal{F}$.

We now give three important definitions that depend upon the mentioned set:

Definition 1 (Growth function) *The* growth function *of model $\mathcal{F}$ is the maximum number of ways into which n points can be classified by the model or function class:*

$$S_{\mathcal{F}}(n) = \sup_{\{z_1,...,z_n\}} |\mathcal{F}_{z_1,...,z_n}|$$

Please note that our definition of the growth function differs from Vapnik (1995), who includes a logarithm in the expression: $S_{\mathcal{F}}(n) = \log\left(\sup_{\{z_1,...,z_n\}} |\mathcal{F}_{z_1,...,z_n}|\right)$.

Definition 2 (Shattering) *We say a model $\mathcal{F}$* shatters *a set of size n s_n if it can generate any classification on these points. If $\mathcal{F}$ shatters a set of size n, then:*

$$S_{\mathcal{F}}(n) = 2^n$$

Definition 3 (VC dimension) *The* Vapnik-Chervonenkis dimension *or* VC dimension *of model $\mathcal{F}$ is the largest n such that there exists a dataset of size n that can be shattered by $\mathcal{F}$:*

$$V_{\mathcal{F}} = \max\{n : S_{\mathcal{F}}(n) = 2^n\}$$

If a model $\mathcal{F}$ can shatter samples of any size, then: $V_{\mathcal{F}} = \infty$. Notice that neither the growth function nor the VC dimension depend on the distribution P.

The growth function can be interpreted as a measure of the complexity of a model or function class: the higher the number of ways a sample s_n can be classified, the more complex we would think the functions $f \in \mathcal{F}$ to be. Think again about the polynomial estimation example: as we increase the number of degrees d of a polynomial, we improve our capacity to model different phenomena, but simultaneously the model's equation gets more and more complex. Ultimately, a model that shatters a sample s_n can classify it in as many ways as possible.

As for the VC dimension, it can also be interpreted as another complexity measure: Bousquet, Boucheron and Lugosi (2003) describe the VC dimension of a model as its *effective size*. Indeed, instead of looking at the number of functions which make up $\mathcal{F}$, VC dimension puts the stress on

how many ways the functions from $\mathcal{F}$ can *effectively* class finite samples.

The following lemma, required to state the main theorem, establishes a relationship between the growth function and the VC dimension (Vapnik, 1995):

Lemma 1 (Sauer-Shelah) *Let $\mathcal{F}$ be a model of functions and $n \in \mathbb{N}$ the size of a sample. Then:*

- *For $n < V_{\mathcal{F}}$:*

$$S_{\mathcal{F}}(n) \leq \sum_{i=0}^{V_{\mathcal{F}}} \binom{n}{i}$$

- *For $n \geq V_{\mathcal{F}}$:*

$$S_{\mathcal{F}}(n) \leq \left(\frac{en}{V_{\mathcal{F}}}\right)^{V_{\mathcal{F}}}$$

We can now state an important result due to Vapnik and Chervonenkis (Vapnik, 1995), which for simplicity we have rewritten for the specific case of the 0-1 loss – although there exists an expression valid for totally bounded loss classes:

Theorem 1 (Vapnik-Chervonenkis) *Let $\mathcal{F}$ a model of functions and n the size of a sample. Then for any $\delta > 0$:*

- *For all $n \in \mathbb{N}$:*

$$\mathbb{P}\left(\sup_{f \in \mathcal{F}} \left(R(f) - R_n(f)\right) \leq \frac{\log S_{\mathcal{F}}(2n) - \log \frac{\delta}{4}}{n}\right) \geq 1 - \delta$$

- *For all $n \geq V_{\mathcal{F}}$:*

$$\mathbb{P}\left(\sup_{f \in \mathcal{F}} \left(R(f) - R_n(f)\right) \leq \frac{V_{\mathcal{F}}\left(\log \frac{2n}{V_{\mathcal{F}}} + 1\right) - \log \frac{\delta}{4}}{n}\right) \geq 1 - \delta$$

This theorem leads us back to our initial discussion. By observing the above bounds we conclude that there are two levers to improve the generalization ability of an algorithm: data – represented by the sample size n which if increased, and ignoring its effect on $S_{\mathcal{F}}(\cdot)$ or $V_{\mathcal{F}}$ for simplicity, tightens the bound – and knowledge – represented by a complexity measure, such as the growth function or the VC dimension, which if decreased tightens the bound. An important consequence of these bounds is that, provided the VC dimension of $\mathcal{F}$ is finite, the empirical risk will converge uniformly towards the generalization risk.

We have presented some mathematical notions and technical results from Statistical Learning Theory. Extensive work has been devoted to this theory since the 90's; as a result extensions of the VC dimension – for example the *pseudo-dimension* and the *fat-shattering dimension* for regression tasks – and new concepts – such as *VC entropy*, *covering numbers* or *Rademacher averages* – for measuring the size of a model $\mathcal{F}$ have been developed and new generalization bounds have been derived – some of them being more operational and computable than the ones presented above.

Nonetheless, this presentation should enable the reader to have an appropriate understanding of the kind of mathematical properties a good learning algorithm is expected to have, particularly concerning its generalization ability. Its should also help to better grasp the intuition behind the algorithmic approach to Support Vector Machines, which will be presented afterwards.

1.4 Machine Learning in insurance

In this section we briefly discuss the applicability of Machine Learning in the realm of the insurance industry: we will see how Machine Learning can help insurers improve their businesses in areas such as pricing or commercial strategy, but also what are the challenges and threats that come with the application of these techniques.

1.4.1 The opportunities offered by Machine Learning

The 1st thing to note is that the application of Machine Learning to the insurance business is already a reality: according to a survey undertaken by the brokerage and advisory firm Willis Towers Watson in autumn 2015 (Friedman, Harley and Southwood, 2016), 42% of US Property & Casualty (P&C) insurers already used extensively analytics methods for pricing, underwriting and risk selection, whereas 41% of US Life insurers use Machine Learning related techniques for expanding customer relationships. According to this same study, some of the areas where insurers plan to expand their analytics capabilities in the next 2 years are claim management or enhancement of customer value proposition. But why and how can Machine Learning help insurers improve their businesses?

To begin with, from a technical point of view algorithms such as the Support Vector Machine or Artificial Neural Networks can free classical actuarial modelling from the rigidities of parametric statistical models – such as the Generalized Linear Model (GLM) which is probably the most widely used technique in actuarial pricing – due to the limited assumptions these algorithms make about the phenomena being modelled.

In addition, Machine Learning models depend on the risk factors – *i.e.* data attributes – with which the modeller feeds the algorithms, whereas parametric statistics can be utilised without attributes: indeed, an appropriate probability distribution can always be fitted to data observations – *i.e.* the objective variable – independently of their specific characteristics. Thus by enabling a better understanding of the risk factors driving policy pricing, Machine Learning could enable insurers to offer more customised products and prices to their clients, hence being more competitive and better hedged against future claims.

However, the insurance business is not limited to the pricing problem and Machine Learning can be leveraged to help decision makers and management by providing tools to analyse processes or threats that traditionally have been ignored or dealt with in a manual way. Some of these interesting areas are the following (Friedman *et al.*, 2016; Lagnaoui, 2015; Rao, Yoder and Busse, 2016; Taillieu, Delucinge and Bellina, 2014):

- **Fraud detection and processing**: the fraud detection process is currently very manual and devising fraud detection algorithms should enable insurers to both improve their detection rates and decrease their processing time. For a detailed example on this matter, the reader might consult Guha, Manjunath and Palepu (2015).
- **Policies portfolio management**: Machine Learning algorithms could help management to monitor the performance of a portfolio of insurance policies throughout its life and better understand the factors that drive its profitability or unprofitability.
- **Client behaviour monitoring**: with the proliferation of data recording devices and the build-up of large datasets, insurers can have at their disposal information about their clients that can help them enrich their relationship with them by devising *bonus-malus* schemes: for example, Lagnaoui (2015) uses in his actuarial memoir GPS driving routes data.

The above areas are only a fraction of the opportunities that Machine Learning can offer to the insurance industry: enhanced customer experience, policy robo-advising for clients, automated underwriting, etc. Generally speaking, provided the required data is available, quantitative techniques

can be developed and applied throughout the company's processes to make them more effective and more efficient. It will be up to the industry to make an intelligent use of Machine Learning capabilities in order to harness its true potential.

1.4.2 The problems raised by Machine Learning

Despite the potential of Machine Learning algorithms applied to insurance, the industry does not have to lose sight of the obstacles and threats that come with them. The survey from Friedman *et al.* (2016) shows that some of the concerns of US insurers are IT processing capacity, data availability or the lack of people with the right training and skills.

We will now briefly describe some of the main challenges that lie ahead for companies that want to equip themselves with powerful Machine Learning capabilities (Friedman *et al.*, 2016; Lagnaoui, 2015; Kaminska, 2016; Ralph, 2016):

- **Data availability and usability**: probably the most important problem faced by companies that want to develop an expertise in Machine Learning is whether they have the data and if it is usable. Insurers that want to seriously invest in Machine Learning will first have to tackle these issues, by making sure they have access to the right data and they have the necessary IT capabilities to exploit it.

- **Data confidentiality and client privacy**: to harness the power of Machine Learning algorithms, insurers will have to gather large amounts of data by either better exploiting client information they already have or by devising new ways to accumulate data: vehicle telematics for automotive insurance – there were 12m telematics-based insurance policies in the world at the end of 2015 according to the specialist consultancy Ptolemus (Ralph, 2016) –, health monitoring devices for life and health insurance, etc. However, the debate is growing on whether this strategy would undermine clients' privacy and expose insurers to legal liabilities for misusing personal data. Lagnaoui (2015) points out that the data can be modified such that it is still usable by the insurer while protecting the client's privacy – in his case, GPS data is subject to random journey rotations and departure and arrival information is deleted.
  Ultimately, insurers and regulators will probably have to collaborate together to develop a robust and comprehensive legal framework that enables the industry to confidently deploy models based on this kind of personal data.

- **Extreme price discrimination**: some experts argue that Machine Learning should allow insurers to sophisticate their pricing to the point where they could determine a unique fair price for each client. This extreme scenario would have important consequences: the insurance business model would probably be totally different, because the current risk pooling mechanism could no longer be necessary if insurers could predict with extreme accuracy the future cost associated to each client; additionally, clients carrying greater risk might be left out of the market because they could not afford the insurance premium demanded to them.
  In the end, algorithms will probably not be the final decision makers on the premium charged to each client and some sort of pooling mechanism – where low risk clients subsidize high risk clients – will still be maintained.

Again, the above list is not comprehensive as there are many more issues that insurers face, such as the lack of Machine Learning and analytics experts. However it should help make clear that, although this kind of quantitative-based decision making is full of promises, numerous dangers still lie ahead: insurers should be wary of turning a formidable asset into a painful liability.

Chapter 2

Support Vector Machines

Mathematical foundations and algorithmic refinements

2.1 Introductory comments

Support Vector Machines (SVMs) are a class of techniques belonging to the realm of Machine Learning. Although the SVM concept was introduced as soon as 1963, the current formalization of SVMs was developed in the nineties. Since then, they have been widely acclaimed as one of the best Machine Learning algorithms available, having being successfully applied in tasks such as handwritten character recognition, text categorization or machine vision.

The main contributor to the development of SVM theory has been Vladimir N. Vapnik, who incorporated two of the main characteristics of this technique:

- The notion of soft margin, a powerful improvement which enables to control the robustness of the model and thus its generalization ability (Cortes and Vapnik, 1995);
- Kernels, which allow to deal with non-linear problems (Boser, Guyon and Vapnik, 1992).

We will develop these notions in greater detail in the next pages.

SVMs were initially developed as classification tools, but have since been expanded to include regression problems (SVM-Regression, SV Regression, SVR or ϵ-SV Regression). Researchers have also developed a similar technique, called Support Vector Clustering (SVC), which as its name indicates allows clustering on data. In this dissertation we will only look at traditional SVMs, which allows regression and classification.

We will first present in a simplified manner the idea underlying SVMs for classification problems by studying the Linear Maximal Margin classifier.

Then we will look at the mathematical formulation and solution of SVMs, while trying to convey intuitively what are the main characteristics of this technique. We will analyse the SVM method with respect to the cartography of Machine Learning methods and algorithms we developed earlier.

Having finalised the presentation of the theory of the SVM, we will present briefly a class of numerical algorithms, *Sequential Minimal Optimization* (SMO), which have been widely adopted as a computational tool to solve SVM problems.

To finish the section, we will introduce some additional Machine Learning algorithms which, combined by the SVM, can greatly improve its computational efficiency.

2.2 The Support Vector Machine algorithm

2.2.1 The Linear Maximal Margin classifier

In this paragraph we will treat a simplified classification problem which should help to understand the overall concept of SVMs.

A *classification problem* consists on devising a rule that will allow the user to correctly classify data into a series of categories. Consider the case in which the data of interest has the following form:

$$S_n = \{\mathbf{x}_i, y_i\}_i = \{(x_{i,1}, ..., x_{i,p}), y_i\}_i, i \in \{1, .., n\}, \forall i\ \mathbf{x}_i \in \mathbb{R}^p, \forall i\ y_i \in \{-1, 1\}$$

As we explained in previous sections this is a *binary* classification problem. We have n observations or *instances* of the vector $\mathbf{x} = (x_1, ..., x_p)$, which has p distinct *attributes* – i.e. variables – and each one of these observations is allocated to a different class, coded through the variable y which takes only two values, meaning that vector $\mathbf{x}$ can only belong to two distinct classes. Vectors $\mathbf{x}$ are said to be *labelled* – supervised learning, to which the SVM belong, tackle problems where the data is labelled.

Letting $\mathbf{w} = (w_1, ..., w_p) \in \mathbb{R}^p$ and $b \in \mathbb{R}$, we are interested in obtaining a function $f : \mathbb{R}^p \to \mathbb{R}$ of the following form:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \\ \Leftrightarrow f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b \\ \Leftrightarrow f(\mathbf{x}) &= \sum_{l=1}^{p} w_l x_l + b \end{aligned}$$

The vector $\mathbf{x}$ would then be classified according to the sign of $f(\mathbf{x})$, such that the estimation $\widehat{y}$ of the class to which $\mathbf{x}$ belongs is given by the following:

$$\widehat{y} = \mathbb{1}_{\{f(\mathbf{x})>0\}} - \mathbb{1}_{\{f(\mathbf{x})<0\}}$$

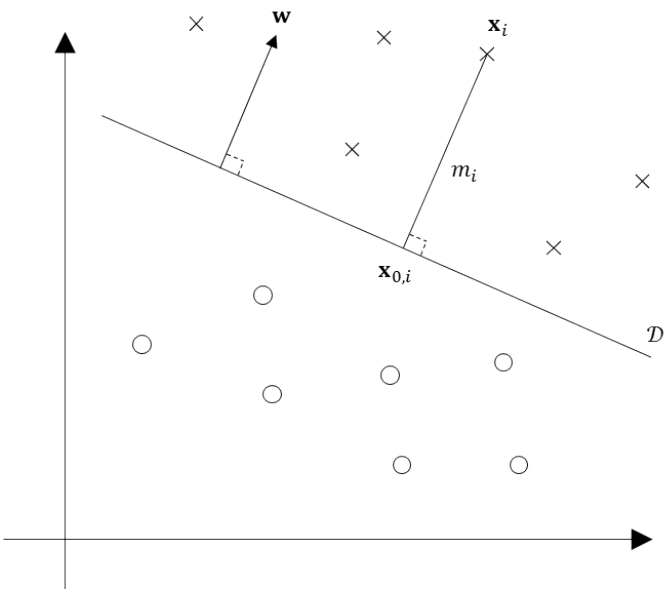

Figure 2.1: *Separating hyperplanes and margins*

We will designate by $\mathcal{F}_L$ the model of functions described above. In the presentation of the Linear Maximal Margin classifier we will assume that the two classes are *linearly separable*, which means that we can find a linear function f which can perfectly classify the data we have:

$$\exists\, f \in \mathcal{F}_L,\ \forall\, (\mathbf{x}, y) \in S_n \begin{cases} y = 1 & \Leftrightarrow\ f(\mathbf{x}) > 0 \\ y = -1 & \Leftrightarrow\ f(\mathbf{x}) < 0 \end{cases}$$

The decision frontier $\mathcal{D}$, also known as *classification frontier* or *separating hyperplane*, is defined by equation $f(\mathbf{x}) = 0$, pictured above, where crosses $(+1)$ and circles (-1) represent data points belonging to each of the two classes.

In the above graphic, m_i is the *geometric margin* of vector $\mathbf{x}_i$ to the separating hyperplane $\mathcal{D}$, which in our setting corresponds to the Euclidean distance: it represents the distance from $\mathbf{x}_i$ to the closest point on the hyperplane, which is its orthogonal projection $\mathbf{x}_{(0,i)}$, where: $\mathbf{w}^T\mathbf{x}_{(0,i)} + b = 0$.

Noting that the weight vector is orthogonal to the decision frontier $\mathcal{D}$, we can derive an expression for m_i by using the fact that $\mathbf{x}_{(0,i)}$ lies in the decision frontier. We also redefine the margin by multiplying it by y_i so that it is positive if the point is correctly classified, negative if not. We end up with the following expression:

$$m_i = \frac{y_i f(\mathbf{x}_i)}{\|\mathbf{w}\|}$$

We now define the *margin of the classifier* m^* to be the minimum over all individual margins:

$$m^* = \min_{i=1,\dots,n} m_i$$

The Linear Maximal Margin classifier as well as the SVM algorithm aims at maximizing this margin in order to determine the optimal frontier:

$$\max_{f \in \mathcal{F}_L} m^* = \max_{f \in \mathcal{F}_L} \left(\min_{i=1,\dots,n} m_i \right)$$

Indeed, consider the two figures below: in the left figure, the decision boundary $\mathcal{D}$ has been selected as to maximize the margin m^*, whereas the right figure presents another example of decision boundary. If we look at the position of point A with respect to the separating hyperplane, we conclude than the left function does a much better job at classifying observations than the right one. By maximizing the minimal distance between observations and the separating hyperplane, we are constructing a robust classifier that enables us to be more confident about our future predictions. We are back to our discussion on Statistical Learning Theory: a good algorithm is one that exhibits good generalization properties.

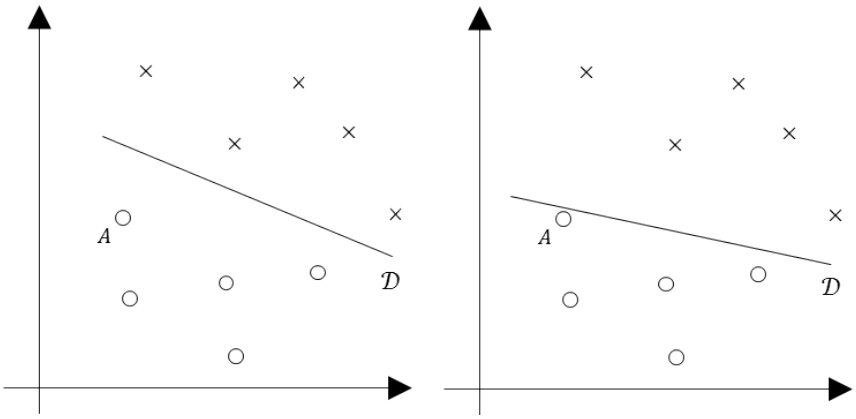

Figure 2.2: *The maximal margin principle*

Remark that by maximizing the lowest distance to the hyperplane among all points, the closest point to the optimal frontier labelled 1 will be at the same distance m_+^* from it than the closest point labelled -1 (m_-^*). Assume it was not the case so that for example $m_+^* > m_-^*$: then it would be possible to increase the minimum margin m_-^* while keeping it lower than m_+^*, which implies that it was not maximized in the 1st place.

m^* can be given a specific expression. Indeed, we have:

$$\begin{aligned} &\forall i \in \{1, \cdots, n\},\ m_i \geq m^* \\ &\qquad \Leftrightarrow\ y_i f(\mathbf{x}_i) \geq m^* \|\mathbf{w}\| \\ &\qquad \Leftrightarrow\ y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq m^* \|\mathbf{w}\| \end{aligned}$$

We notice that for any $(\mathbf{w}, b)$ that satisfies the constraint and given $k > 0$, $(k\mathbf{w}, kb)$ also satisfies it. So we can arbitrarily set:

$$m^* = \frac{1}{\|\mathbf{w}\|}$$

The Linear Maximal Margin optimization problem can then be formalized as follows:

$$\begin{aligned} &\max_{\mathbf{w},b}\ \frac{1}{\|\mathbf{w}\|} \\ &\text{s.t.:}\ \forall i \in \{1, ..., n\},\ y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \end{aligned}$$

For convenience, we can equivalently write:

$$\begin{aligned} &\min_{\mathbf{w},b}\ \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t.:}\ \forall i \in \{1, ..., n\},\ y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \end{aligned}$$

The advantage of the above formulation is that it is a problem with a convex quadratic objective with only linear constraints, which is simpler to solve than the original formulation and has some nice properties. If for some vector $\mathbf{x}_i$ we have $y_i f(\mathbf{x}_i) = 1$ we say that vector $\mathbf{x}_i$ *lies on the boundary of the margin.*

The Linear Maximal Margin classifier is a particular case of a Support Vector Machine, in which the data is assumed to be linearly separable. It is the simplest form of SVM and the general formulation of the SVM model is based on the same principles as those that have just been displayed.

2.2.2 From Linear Maximal Margin classifiers to the Support Vector Machine

The classifier presented in the previous paragraph is a particular example of a SVM, in which the data at our disposal is linearly separable. We now will show what the standard SVM formulation adds to the previous model.

Allowing error tolerance though soft margins

Recall that, as we discussed previously, one of the cornerstone of the SVM theory, introduced by Vapnik, is the *soft margin.* There are two main advantages to this approach:

- First, in the previous example we have assumed that our data was linearly separable. However this is a very restrictive condition that is very unlikely to be verified with real life data. In such a case, a linear classifier such as the one showed above will fail. Introducing a soft margin allows us to solve this issue.

- Secondly and most importantly, soft margin is a tool to avoid over-fitting. Soft margin enables us to allow some observations to be misclassified in exchange of greater robustness.

To preserve the convexity of the optimization problem, the conventional modification is the following:

$$m_i \geq m^*(1 - \xi_i)$$

ξ_i is a positive real number that represents the relative distance by which we allow observation $\mathbf{x}_i$ to violate the margin m^*.

The vector $\boldsymbol{\xi} = (\xi_1, ..., \xi_n)$ is known as the vector of *slack variables*. By forcing ξ_i to be positive for all i, ξ_i can only belong to 3 ranges of values. We explain what the different values of ξ_i mean in terms of classification:

- $\xi_i = 0$: observation $\mathbf{x}_i$ is classified in the right class and is situated at a distance greater than or equal to m^* of the decision frontier.
- $0 < \xi_i \leq 1$: observation $\mathbf{x}_i$ is classified in the right class but is situated at a distance lower than m^* of the decision frontier.
- $\xi_i > 1$: observation $\mathbf{x}_i$ is misclassified.

If $\xi_i = 0$ and $y_i f(\mathbf{x}_i) = 1$, we say again that $\mathbf{x}_i$ *lies on the boundary of the margin*. If $0 < \xi_i \leq 2$ then vector $\mathbf{x}_i$ *lies on the margin*, either correctly classified ($0 < \xi_i < 1$), on the separating hyperplane ($\xi_i = 1$) or misclassified ($1 < \xi_i \leq 2$).

To give a better understanding of slack variables, we have represented them in the following graph.

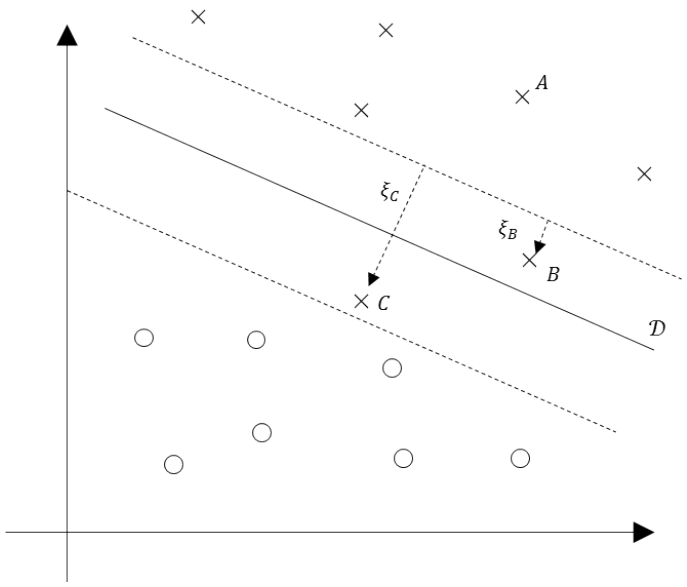

Figure 2.3: *Understanding slack variables* $\xi_1, ..., \xi_n$

Observation A is on the right side of the separating hyperplane and of the SVM margin. Observation B is rightly classified but it is situated within the margin; the distance to the margin boundary is represented by the slack variable ξ_B. Finally, observation C is misclassified and the distance to the margin boundary is ξ_C.

Keeping the classifier margin m^* equal to the inverse of the norm of the weight vector $\mathbf{w}$ as

previously, the constraint can be rewritten:

$$y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i$$

However, we need to modify further the optimization problem for it to make sense. Indeed, the optimization is now being undertook with respect to $\mathbf{w}$, b and $\boldsymbol{\xi}$, thus $\|\mathbf{w}\|$ can arbitrarily be minimized to 0 just by setting sufficiently high values for $\xi_1, ..., \xi_n$.

We must require a further constraint on $\boldsymbol{\xi}$ to control the overall proportional amount by which observations are classified on the wrong side of the margin – cases where $\xi_i > 0$. Therefore we inject the sum of the slack variables into the function to be minimized. The SVM problem becomes:

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right) \\ &\text{s.t. } \forall i \in \{1, ..., n\} \ : \ y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i \\ &\qquad\qquad \xi_i \geq 0 \end{aligned}$$

The hyper-parameter C allows us to determine our tolerance to errors and therefore to arbitrate between accuracy and generalization: if an observation is situated on the wrong side of the margin by a proportion ξ_i, the objective function will suffer a penalty equal to $C\xi_i$, hence the smaller the value taken by C, the wider we might expect the margin m^* to be because we are being more tolerant towards errors.

The Support Vector Machine optimization problem stated above is convex, hence we can apply the following theorem (Erästö, 2001):

Theorem 2 *Let $(\mathcal{P}_C)$ be a convex optimization problem characterised by an objective function f, an optimization variable $\mathbf{x}$ and constraints $g_k \leq 0$, $k \in S \subset \mathbb{N}$. Then every local solution $\mathbf{x}^*$ to $(\mathcal{P}_C)$ is a global solution.*

We are reassured that any solution $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$ will be a global one, thus even if there are multiple solutions $(\mathbf{w}^{*(1)}, b^{*(1)}, \boldsymbol{\xi}^{*(1)}), (\mathbf{w}^{*(2)}, b^{*(2)}, \boldsymbol{\xi}^{*(2)}), ...$ they would all yield the same global minimum on the SVM objective function:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

However, the uniqueness of the solution is not always guaranteed. For a discussion on this topic, the interested reader might consult Appendix B; appendix A is a quick refresher on optimization theorems relevant to the SVM problem.

The kernel trick

The second critical concept introduced by Vapnik in the nineties to improve the SVM model were *kernels*, although they had already been around for a considerable time – as early as the 1960 decade when they were introduced by Aizerman, Braverman and Rozoner (1964). As such, SVMs can be grouped in a class of algorithms called *kernel machines*, which also includes the Perceptron.

Whereas the soft margin allowed for greater flexibility and robustness in the model, kernels are a powerful computational tool that allows decreasing considerably the number of calculations involved in solving the SVM. They are also a way to *extract features*: creating new features from existing

ones. Finally, they allow the user to determine a classification function which is non-linear, therefore enlarging the set of problems that can be tackled with the SVM.

Although kernels can greatly complicate the structure of a Support Vector Machine – for example, *radial basis function* (RBF) kernels, also known as *Gaussian kernels*, have infinite VC dimension – this might not translate into a worsened accuracy: RBF kernelized SVMs have a track record of excellent performance.

To understand the utility of kernels, we first need to introduce the concept of *feature mapping*. Until now, we have assumed that our data could be separable with a linear function. However, it might be the case that such a function does not fit the problem at hand; in such a setting, a non-linear function might help to classify the data; the classifier function becomes:

$$f\left(\phi(\mathbf{x})\right) = \mathbf{w}^T\phi(\mathbf{x}) + b$$

The function ϕ is known as a *feature mapping* in the literature: letting $\mathcal{X}$ be the space of vectors $\mathbf{x}$ – in our case $\mathcal{X} = \mathbb{R}^p$ – and $\mathcal{H}$ a higher dimensional space, ϕ is mapping from $\mathcal{X}$ to $\mathcal{H}$. We could have for example:

$$\phi(\mathbf{x}) = \begin{pmatrix} \mathbf{x}^2 \\ \sqrt{\mathbf{x}} \end{pmatrix}$$

The idea behind applying a feature mapping is that it could allow us to project the data into a space where it might be linearly separable. The SVM learning algorithm will subsequently be applied to $\phi(\mathbf{x})$. To solve the problem, we just need to replace vector $\mathbf{x}$ by $\phi(\mathbf{x})$. Now, to understand the utility of kernels in this environment, consider the following 2 facts: first, it can be shown that the SVM problem might be reformulated as a *dual optimization program* – see Appendixes A and B for further technical details:

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \right)$$

$$\text{s.t. } \forall i \in \{1, ..., n\} \ : \ 0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Secondly, in Appendix B the reader will find the proof that the solution $\mathbf{w}^*$ of the problem has the following expansion in terms of Lagrangian coefficients:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}_i)$$

Thus the optimal decision function f_n^* might be written:

$$f_n^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b^*$$

The reader will notice that both the objective function of the dual formulation and the optimal decision function f_n^* depend solely on the *dot product* or inner product of vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$:

$$\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

The issue with these inner products is that they might be very expensive to calculate if the dimension of $\mathcal{H}$ is very high. However, it might be that there is a function K such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

K is known as the kernel associated to the feature mapping ϕ. The advantage of this kernel is that, instead of computing inner products, we can directly compute $K(\mathbf{x}_i, \mathbf{x}_j)$ which might be much more computationally efficient, without even having to explicitly represent $\phi(\mathbf{x}_i)$: this is why we call this technique the *kernel trick*. We directly equip our SVM with some kernel – in fact, even the vanilla SVM is equipped with a trivial kernel, the linear one – and implicitly map our data to a complex space $\mathcal{H}$ where the data can satisfactorily be separated linearly while greatly improving the time required to undertake computations. If the problem at hand is particularly difficult, we can construct new, more complex kernels by adding, multiplying, etc. previous kernels K_1, K_2, ..., K_k between them. In fact, given a kernel can be represented by a symmetric matrix defined as $K = (K_{i,j})_{0 \leq i,j \leq n} = (K(\mathbf{x}_i, \mathbf{x}_j))_{0 \leq i,j \leq n}$, the condition for a kernel to be valid is simply that its matrix is positive semi-definite:

$$\forall\, \mathbf{z} \in \mathbb{R}^n :\ \mathbf{z}^T K \mathbf{z} \geq 0$$

The reader might have noticed that the described strategy poses a relevant question: how can we know what mapping ϕ we are applying to our data $\mathbf{x}_{ii}$ when we choose a kernel K? Is the mapping implicitly defined by K meaningful and sensible given the nature of $\{\mathbf{x}_i\}_i$? Although apparently critical, these questions do not seem to be particularly addressed by the Machine Learning community. As explained in the 1st part of the memoir, the Machine Learning community might be characterised by the focus given to the predictive power of its models – as opposed to the interpretability of the parameters – hence as long as the kernel in question yields good predictions the question of knowing what mapping we are implicitly using is relegated to the background.

Fortunately, the abandonment of the feature mapping question can be justified by the characteristics of kernels. Although we are not going to delve into the theory of machine learning kernels, the following two examples should give the reader an idea of the breadth and complexity of the mappings kernels give access to. Let us first consider two vectors $(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^p \times \mathbb{R}^p$ and assume we have the following *polynomial kernel*:

$$K(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^T \mathbf{z}\right)^2$$

The kernel can be rewritten as follows:

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= \left(\sum_k x_k z_k\right)\left(\sum_l x_l z_l\right) \\ &= \sum_k \sum_l \left(x_k z_k\right)\left(x_l z_l\right) \end{aligned}$$

If $p = 2$ from the above representation we conclude the corresponding feature mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^4$ might be:

$$\phi(\mathbf{x}) = \begin{pmatrix} x_1 x_1 \\ x_1 x_2 \\ x_2 x_1 \\ x_2 x_2 \end{pmatrix}$$

We quickly note that the computational advantage of the kernel representation is clear: evaluating the mapping ϕ at a point $\mathbf{x} \in \mathbb{R}^p$ takes time $O(p^2)$, while computing the kernel takes time $O(p)$.

For this particular kernel, computation time is linear in the dimension of the data, i.e. the number of attributes.

The important takeaway from this example is that the mapping ϕ and hence the space $\mathcal{H}$ might not be unique for a given kernel. In this case, instead of $\mathcal{H} = \mathbb{R}^4$ we could have chosen $\mathcal{H} = \mathbb{R}^3$ and one of the following two mappings:

$$\phi_1(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

$$\phi_2(\mathbf{x}) = \frac{1}{\sqrt{2}} \begin{pmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

In fact, the mapping ϕ and the space $\mathcal{H}$ might be as complex as one wishes. Let now consider as a 2nd example, the *Gaussian* or Radial Basis Function kernel K:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right\}$$

Using the characterization of the exponential function, we might rewrite the Gaussian kernel as follows:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left\{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right\} \exp\left\{-\frac{\|\mathbf{z}\|^2}{2\sigma^2}\right\} \left(\sum_{k=0}^{+\infty} \frac{1}{k!} \left(\frac{\mathbf{x}^T\mathbf{z}}{\sigma^2}\right)^k\right)$$

Defining:

$$f(\mathbf{x}) = \exp\left\{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right\}$$

We obtain the following representation:

$$K(\mathbf{x}, \mathbf{z}) = \sum_{k=0}^{+\infty} \left\langle \sqrt[k]{\frac{f(\mathbf{x})}{\sigma\sqrt{k!}}}\, \mathbf{x}, \sqrt[k]{\frac{f(\mathbf{z})}{\sigma\sqrt{k!}}}\, \mathbf{z} \right\rangle^k$$

From this representation we conclude that the Gaussian kernel defines a mapping which projects the data into an infinite dimensional space $\mathcal{H}$.

These two examples should demonstrate that knowing the mapping is not critical. First, a given kernel K might be associated to multiple different mappings hence by training a SVM with K we are in fact potentially capturing hundreds or thousands of different transformations ϕ. In addition, as illustrated by the Gaussian example, kernels can represent arbitrarily complex mappings and spaces – even infinite dimensional ones.

The Support Vectors

We briefly explain where the name of this model comes from. More detailed mathematical derivations can be found on Appendix B.

Given that both the objective function of the SVM optimization program $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$ and its associated functional constraints $y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ are continuously

differentiable, any solution to the SVM must satisfy the *Karush-Kuhn-Tucker conditions* (KKT) – further technical details can be found in Appendixes A and B. Hereafter we will only focus on the *complementary slackness* KKT condition – recalling that $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b$:

$$\forall i \in \{1, ..., n\} : \ \alpha_i \left(1 - \xi_i - y_i f(\mathbf{x}_i)\right) = 0$$

$$\forall i \in \{1, ..., n\} : \ \beta_i \xi_i = 0$$

Taking into account the other constraints:

$$\forall i \in \{1, ..., n\} : \ y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

$$\forall i \in \{1, ..., n\} : \ \alpha_i \geq 0$$

We can derive the following implications

$$\alpha_i > 0 \ \Rightarrow \ y_i f(\mathbf{x}_i) = 1 - \xi_i$$

$$y_i f(\mathbf{x}_i) > 1 - \xi_i \ \Rightarrow \ \alpha_i = 0$$

In the mathematical optimization literature we say that the margin constraint $y_i f(\mathbf{x}_i) \geq 1 - \xi_i$ is *active* when equality holds. We define *support vectors* in the following way:

Definition 4 (Support Vector) *A vector $\mathbf{x}_i$ from sample S_n is a support vector if:*

$$\alpha_i > 0$$

Let $SV_{\mathbf{x}_1,...,\mathbf{x}_n} = \{\mathbf{x}_i : i \in \{1, ..., n\}, \alpha_i > 0\}$ be the set of all support vectors. For vectors from $SV_{\mathbf{x}_1,...,\mathbf{x}_n}$ the constraint $y_i f(\mathbf{x}_i) \geq 1 - \xi_i$ is active:

$$\forall \, \mathbf{x}_i \in SV_{\mathbf{x}_1,...,\mathbf{x}_n} \ , \ y_i f(\mathbf{x}_i) = 1 - \xi_i$$

From the above equations we also observe that support vectors are those for which the functional distance to the margin $y_i f(\mathbf{x}_i)$ is minimal and equal to $1 - \xi_i$ so they are the ones lying closest to the decision boundary $\mathcal{D}$ – at least in the separable case; in the non separable case there might be misclassified points that lie very far away from $\mathcal{D}$.

Note that the parameter C of the SVM has a crucial role in determining the number of support vectors: the higher the value taken by C, the more tolerant we are towards errors, hence more (support) vectors will be lying in the boundary or within the margin. We might conclude that the more we want to generalize – by increasing the value of C – the more support vectors our trained model will have.

Support vectors determine entirely the solution $(\mathbf{w}^*, b^*)$: removing all points which do not belong to $SV_{\mathbf{x}_1,...,\mathbf{x}_n}$ would not alter the solution. Indeed, recall that for a SVM equipped with a feature mapping ϕ the optimal decision function f_n^* is written:

$$f_n^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b^*$$

By definition, all training vectors $\{\mathbf{x}_i\}_i$ which are not support vectors are associated to a null alpha, thus their impact in the decision function is null. This property might help the reader to grasp intuitively the generalization ability of the Support Vector Machine: outlier observations cannot alter the trained classification function f, and adding further observations situated away from the margin will not alter our result. This property is in contrats with other classification techniques such as *Linear Discriminant Analysis* (LDA) which solution is determined by the whole training data.

Regularized formulation of the SVM

The SVM problem can also be interpreted as a *regularization problem*: regularized algorithms have a built-in mechanism to control for the model's complexity. The general formulation of a regularized optimisation objective is:

$$f_n = \arg\min_{f \in \mathcal{F}} \left(R_n(f) + \frac{\beta}{2} \|f\|^2 \right)$$

Where we recall that R_n is the empirical risk and β the regularization hyper-parameter. In the simple case of the linear SVM, functions from $\mathcal{F}$ are of the form $\mathbf{w} = (b, w_1, ..., w_p) \in \mathbb{R}^{p+1}$ and hence the regularizer would be $\|\mathbf{w}\|^2$. From the SVM optimization constraints, it can be shown that the empirical risk is written as follows – see Appendix B for a proof of this expression:

$$R_n(f) = \sum_{i=1}^{n} \max \left[0, 1 - y_i \mathbf{w}^T \mathbf{x}_i \right]$$

It can be showed that $\beta = 1/C$, hence the optimization objective of the SVM can be restated as:

$$\arg\min_{f \in \mathcal{F}} \left(\sum_{i=1}^{n} \max \left[0, 1 - y_i \mathbf{w}^T \mathbf{x}_i \right] + \frac{\beta}{2} \|\mathbf{w}\|^2 \right)$$

This formulation of the SVM will be helpful later on.

Extensions: unbalanced classes, multiple classes, regression, clustering

We briefly introduce some models that leverage the concept of margin maximisation and support vectors.

The first extension to the SV Machine is very simple. It consists on replacing the value of the hyper-parameter C by two values C_+ and C_-, associated respectively to positively-labelled and negatively-labelled examples. Letting $I_+ = \{i : y_i = 1\}$ and $I_- = \{i : y_i = -1\}$, the SVM objective function becomes:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C_+ \sum_{i \in I_+} \xi_i + C_- \sum_{i \in I_-} \xi_i$$

The main advantage of this change is to account for *unbalanced classes*: for example if our dataset has many more instances of the positive class, we might define C_- to be much higher than C_+ such that while training the SVM we give similar weights to errors on positively-labelled and negatively-labelled instances. Assuming that the number of errors in one class is proportional to the number of samples in that class and letting n_+ and n_- be respectively the number of positively-labelled and negatively-labelled examples, in order to have the same overall weight assigned to positive and negative errors we must simply satisfy – for further details see Ben-Hur and Weston (2009):

$$\frac{C_+}{C_-} = \frac{n_-}{n_+}$$

Another relatively popular parametrization of the SVM problem is known as *ν-SVM*. Although we will not discuss it further, we point out that the parameter $\nu \in]0, 1]$ is an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors with respect to the size of the training set.

Up to now our discussion and some of the results we have presented have been set in a particular framework: a binary classification problem. However SVMs can be extended to cope with *m classes*

classification problems: $\mathcal{Y} = \{1, ..., m\}$. However, one weakness of the SVM is that the separating hyperplane concept does not fit well to multi-class problems. Despite this limitation, two extensions to the SVM have been developed that deal well with multiple classes. We briefly explain them hereafter – we will assume there are $m > 2$ classes in total:

- **One vs. One classification**: under this approach, we fit $N_{\text{pairs}} = \binom{m}{2}$ different SVMs, one for each possible pair of classes. We obtain a set of classifiers $\{f_1, ..., f_{N_{\text{pairs}}}\}$ such that the prediction of a SVM f_k is a class $j \in \{1, .., m\}$. Let consider a new data point $\mathbf{x}$, then the *One vs. One SVM* has the following classification function:

$$f_{O\,vs\,O}(\mathbf{x}) = \underset{j \in \{1,...,m\}}{\arg\max} \left(\sum_{k=1}^{N_{\text{pairs}}} \mathbb{1}_{f_k(\mathbf{x})=j} \right)$$

  The vector $\mathbf{x}$ is assigned to the class to which it is the most frequently assigned during the N_{pairs} pairwise classifications.

- **One vs. All classification**: under this approach, we only fit m different SVMs, one for each possible class – during the training procedure of SVM f_j, we consider only two classes: class j and another class grouping all the rest of the classes. We obtain a set of classifiers $\{f_1, ..., f_m\}$ such that the prediction of a SVM f_j is a real number – the sign of that number determines whether a vector has been classified into j or the group of the rest of the classes by SVM f_j; the higher the absolute value of that number, the more confident we are about the prediction. Let consider a new data point $\mathbf{x}$, then the *One vs. All SVM* has the following classification function:

$$f_{O\,vs\,A}(\mathbf{x}) = \underset{j \in \{1,...,m\}}{\arg\max} \left(|f_j(\mathbf{x})| \right)$$

  The vector $\mathbf{x}$ is assigned to the class for which we have a greater degree of confidence.

As it has already been mentioned, SVMs can also be expanded further to tackle regression and clustering problems:

- **ϵ-SV Regression**: we will briefly state the optimization program that defines Support Vector Regression, as we will implement this model in our insurance pricing application; the interested reader might refer to Smola and Schölkopf (2003) for further details, but the main takeaway is that SVMs for classification and regression share the same basic principles.
  Let $\epsilon \in \mathbb{R}_+$ be a real number and assume that the variable y now takes a numerical value in $\mathbb{R}$. The model has the same specification as the SVM for classification:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

  However, there are now two vectors of slack variables $\boldsymbol{\xi}^{(1)} = (\xi_1^{(1)}, ..., \xi_n^{(1)})$ and $\boldsymbol{\xi}^{(2)} = (\xi_1^{(2)}, ..., \xi_n^{(2)})$. The *$\epsilon$-SV Regression* is defined by the following optimization program:

$$\begin{aligned}
&\min_{\mathbf{w}, b, \boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \left(\xi_i^{(1)} + \xi_i^{(2)} \right) \right) \\
&\text{s.t. } \forall i \in \{1, ..., n\} \; : \; y_i - \left(\mathbf{w}^T \mathbf{x}_i + b \right) \leq \epsilon + \xi_i^{(1)} \\
&\qquad\qquad \left(\mathbf{w}^T \mathbf{x}_i + b \right) - y_i \leq \epsilon + \xi_i^{(2)} \\
&\qquad\qquad \xi_i^{(1)} \geq 0 \\
&\qquad\qquad \xi_i^{(2)} \geq 0
\end{aligned}$$

The ϵ-SV Regression model generates a linear function which is located within a margin of length 2ϵ. The objective is to generate predictions which are at most at a distance ϵ from their true value; vectors $\boldsymbol{\xi}^{(1)}$ and $\boldsymbol{\xi}^{(2)}$ allow this maximal error constraint to be violated but by doing so they inflict a penalty to the objective function.

- **Support Vector Clustering**: we will not present Support Vector Clustering as we will not be using this method when studying our insurance data; the interested reader might refer to Ben-Hur, Horn, Siegelmann and Vapnik (2001) for an introduction to this technique.

Aside from classification, regression and clustering tasks, the popularity of the Support Vector Machine since the mid 90' has spanned a lot of research to emulate their success in other Machine Learning problems; as such, the concept of Support Vectors and maximizing a certain margin have been transposed to tasks such as *novelty detection* or *probability density estimation*. Specific SVM models have been designed for these tasks.

Now that we have explored the technical intricacies of the Support Vector Machine, we will put this model in perspective with respect to the field of Machine Learning, levering our earlier discussion on the discipline.

2.2.3 The place of Support Vector Machines within Machine Learning

One of the main current limitations of Machine Learning is the lack of an appropriate general and practitioner-oriented framework for selecting an algorithm in the face of a specific problem. As stated by Sebag (2014), "*Algorithm selection has been viewed as a* [Machine Learning] *bottleneck since the 80s*". Although our purpose here is far from developing such a framework for SVMs, we aim at reflecting on the model and situate it within the numerous divisions and areas in which the Machine Learning algorithm portfolio can be split into. Doing so should enable the reader to hone his understanding of the situations in which Support Vector Machines can be used. We will rely on the cartographies of Machine Learning we elaborated in the first part of this memoir.

As a classification method, SVMs requires the training instances to be labelled. Given that the difference between supervised and unsupervised learning is the labelling of instances, under the formulation presented here above the SVM algorithm belongs to the realm of supervised learning. The same can be said of ϵ-SV Regression as training an SVR requires to have a numerical value assigned to each training instance. However, we also mentioned the existence of a modification of the SVM called Support Vector Clustering. This third type of SVMs falls under the category of unsupervised learning because clustering algorithms seek to group data into clusters by similarity and no additional information (label) is used.

With regards to our second cartography of Machine Learning techniques presented in 1.2.2, Support Vector Machines are more difficult to classify. The different categories were: (1) regression algorithms; (2) clustering algorithms; (3) Bayesian algorithms; (4) decision Trees; (5) dimensionality Reduction algorithms; (6) instance-based algorithms; (7) Artificial Neural Networks algorithms; (8) association rule learning algorithms; (9) Deep Learning algorithms; (10) regularization algorithms, and (11) ensemble algorithms.

It is rather clear that SVMs do not belong to categories 3, 4, 5, 8, 9 and 11. However, SVMs are in some way or another related to the rest of the categories:

- Trivially, ϵ-SV Regression is to be included in the "Regression" class, whereas SVC belongs to the category of "Clustering" algorithms. However this characterization pertains only to these two extensions of the Support Vector concept.
- "Instance-based algorithms" might be the best fit for the Support Vector method. We have explained that these algorithms work by memorizing critical or important observations of the

data in order to subsequently compare new data to these instances and assess their similarity to make a decision. In the case of SVMs these critical points would be the support vectors, as they are the only points from the whole training set that determine the separating hyperplane function. The similarity assessment between a new data point $\mathbf{x}_{\text{new}}$ and support vectors can be explicitly explained using the popular Gaussian kernel K. Let consider a support vector $\mathbf{x}_i^{(SV)}$:

$$K\left(\mathbf{x}_{\text{new}}, \mathbf{x}_i^{(SV)}\right) = \exp\left\{-\frac{\|\mathbf{x}_{\text{new}} - \mathbf{x}_i^{(SV)}\|^2}{2\sigma^2}\right\}$$

From the properties of the exponential function we notice that:

$$\lim_{\|\mathbf{x}_{\text{new}} - \mathbf{x}_i^{(SV)}\| \to 0} K\left(\mathbf{x}_{\text{new}}, \mathbf{x}_i^{(SV)}\right) = 1$$

$$\lim_{\|\mathbf{x}_{\text{new}} - \mathbf{x}_i^{(SV)}\| \to +\infty} K\left(\mathbf{x}_{\text{new}}, \mathbf{x}_i^{(SV)}\right) = 0$$

So the closer the new observation is to the support vector, the higher the value of the kernel will be.

- SVMs is closely related to the Perceptron algorithm, which we included in the category "ANN algorithms", because both consist on a classification problem in which the objective is to find a separating hyperplane. Moreover both methods can be tuned through the incorporation of a kernel. In fact, the SVM can be viewed as an optimal and stable Perceptron. However, the similarities between SVMs and Neural Networks might not go beyond the Perceptron model and again this characterization would only apply for the classification SVM, not for the whole SVM class (classification, regression, clustering).
- We explained that the SVM objective function can be reformulated as a regularized version of some other function. This characteristic lead us to consider that the SVM might be viewed as a "Regularization algorithm".

Considering the critical role played by support vectors in the formulation of the SVM optimal weight vector $\mathbf{w}^*$, as well as their influence in the uniqueness property of the solution, our view is that the SVM class of models should be considered as an *instance-based* algorithm. As such, SVMs might be useful particularly when the user believes that in the task at hand the solution might depend on some crucial instances.

In addition to being an instance-based supervised learning method, the SVM can also be included in another category of learning machines that we mentioned earlier: *kernel machines*. Here we group all algorithms that can incorporate a kernel function to implicitly map features into high dimensional spaces. Apart from the SVM, other potential kernel methods include the Perceptron, PCA, ridge regression and spectral clustering. Hence SVMs might prove useful to solve some classification or regression problem when the available features are insufficient to determine a good decision function f and further features must be extracted.

Having briefly discussed the place of the model within the Machine Learning realm and given some clues about the range of problems to which it can be applied, we have now concluded our overview of the Support Vector Machine. In the next pages, we will turn our attention to the technicalities related to its practical implementation.

2.3 Implementing Support Vector Machines

The current section will be divided in two parts.

First we will discuss about *Sequential Minimal Optimization* (SMO) which is a numerical analysis tool enabling to solve the SVM in an efficient way both in terms of accuracy and time. It was first proposed by J.C. Platt in 1998 (Platt, 1998). It is now one of the most widely used solving algorithm for the SVM, and some of our experimental results were obtained using the `LIBSVM` solver (Chang and Lin, 2013), which is based on the SMO algorithm.

Secondly, although SMO is critical for the applicability of SVMs on large datasets, computation time might still be prohibitively large in some cases. We will present two Machine Learning techniques which allow to further improve time efficiency: Stochastic Gradient Descent and Kernel approximations.

2.3.1 Sequential Minimal Optimization

The SMO algorithm is a type of *decomposition method* which solves the dual formulation of the SVM problem. Decomposition methods seek to split a quadratic programming (QP) problem – such as the SVM – into a sequence of smaller QP problems by selecting some elements $\alpha_{d_1}, ..., \alpha_{d_m}$ from $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$, with $m < n$, and optimizing the objective function with respect to this subset. The set $\{\alpha_{d_1}, ..., \alpha_{d_m}\}$ is usually called the *working set* and in the SMO case its size is equal to 2: $m = 2$. Despite increasing significantly the number of QP problems to be solved, each sub-problem is solved in a timespan that ensures SMO brings computation time gains.

Much of the research done on SMO algorithms in the last 15 years has been devoted to working set selection, which is the 1$^{\text{st}}$ step of the SMO algorithm. Indeed, computational efficiency can be dramatically improved if the rules guiding the selection process are carefully designed. There are many heuristic techniques that might be used to select a working set: as long as at least one of the coefficients from the working set violated the KKT conditions before the optimization step, Osuna's theorem (Osuna, Freund and Girosi, 1997) ensures convergence of the SMO algorithm to the optimal solution.

We will not discuss further about working set selection – the interested reader might consult Platt's original paper, Bottou and Lin (2007) or Chen, Fan and Lin (2005) for further discussion. Instead our presentation of SMO will focus on the 2$^{\text{nd}}$ and 3$^{\text{rd}}$ steps: optimization with respect to the working set and update of the parameter b. These 3 steps are repeated until some convergence criterion is satisfied.

Optimization of the objective function w.r.t. the working set

We are interested in a SVM which has been equipped with a kernel K. We will assume the SMO algorithm has already selected a working set $W_{i,j} = (\alpha_i, \alpha_j)$. The algorithm optimizes at each step the dual objective function with respect to the two Lagrangian coefficients α_i and α_j. Note that this is the size of the minimum working set because of the following linear constraint of the SVM program:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Without loss of generality, let fix $i = 1$ and $j = 2$ and introduce the following notation:

$$\zeta \equiv -\sum_{i=3}^{n} \alpha_i y_i$$

There are two constraints on $W_{1,2}$:

- "*Box*" *constraints*: coefficients from the working set must lie between 0 and C – recall this was a constraint from the SVM dual formulation. Their values are then constrained to be in a box.

- "*Diagonal*" *constraints*: coefficients from the working set must verify the following linear equality:

$$\alpha_1 y_1 + \alpha_2 y_2 = \zeta$$

  Their values are hence constrained to be in a diagonal line.

From these two constraints we conclude that the coefficients from $W_{1,2}$ must lie in a diagonal segment. Furthermore, we can formulate the optimization problem as a one-variable problem by setting:

$$\alpha_1 = y_1 (\zeta - \alpha_2 y_2)$$

First the SMO algorithm is going to optimize the dual function with respect to α_2 without considering the "box" constraint – while the "diagonal" constraint is implicitly taken into account by expressing α_1 in terms of α_2. Then the resulting updated value of α_2 will be compared to some bounds L and H that ensure the satisfaction of the "box" constraint, such that if the new value for α_2 is lower than L then its value will be replaced by L, and if it is higher than H it will be replaced by H. The new value for α_1 is then derived from α_2.

Let a superscript $^{\text{old}}$ denote the old value of a coefficient, $^{\text{new}}$ its optimized value ignoring the "box" constraints $0 \leq \alpha_i \leq C$ and $^{\text{new, clipped}}$ it optimized value taking into account the "box" constraints. Then the SMO algorithm derives the updated values of the coefficients α_2^{new}, $\alpha_2^{\text{new, clipped}}$ and $\alpha_1^{\text{new, clipped}}$ using the following analytic equations:

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + y_2 \left(\frac{(f(\mathbf{x}_1) - y_1) - (f(\mathbf{x}_2) - y_2)}{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)} \right)$$

$$\alpha_2^{\text{new, clipped}} = L \mathbb{1}_{\{\alpha_2^{\text{new}} \leq L\}} + \alpha_2^{\text{new}} \mathbb{1}_{\{L < \alpha_2^{\text{new}} < H\}} + H \mathbb{1}_{\{\alpha_2^{\text{new}} \geq H\}}$$

$$\alpha_1^{\text{new, clipped}} = \alpha_1^{\text{old}} + y_1 y_2 \left(\alpha_2^{\text{old}} - \alpha_2^{\text{new, clipped}} \right)$$

When $y_1 = y_2$ bounds L and H are equal to:

$$L = \max(\zeta - C, 0), \qquad H = \min(C, \zeta)$$

When $y_1 = -y_2$ bounds L and H are equal to:

$$L = \max(-\zeta, 0), \qquad H = \min(C - \zeta, C)$$

It is important to note that the SMO algorithm can handle the cases where $K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2) \leq 0$. The derivations for all these results can be found in Appendix C. Note that the above equations are an advantage of the SMO technique, as the solution is analytical and thus numerical optimization is unnecessary.

Computing the value of b

Once the working set $W_{1,2}$ has been optimized, a value for the intercept b must be selected in order to verify the KKT conditions. In his paper, Platt suggests the following values depending on the new values of α_1 and α_2:

- If for some i, $i \in \{1,2\}$, α_i is not at bounds ($0 < \alpha_i < C$) then the following value for b is valid and it forces the trained function to output y_i when the input is $\mathbf{x}_i$, $j \in \{1,2\}$, $j \neq i$:

$$b^{\text{new}}(i) = b^{\text{old}} - (f(\mathbf{x}_i) - y_i) - y_i \left(\alpha_i^{\text{new, clipped}} - \alpha_i^{\text{old}}\right) \mathbf{x}_i{}^T \mathbf{x}_i - y_j \left(\alpha_j^{\text{new, clipped}} - \alpha_j^{\text{old}}\right) \mathbf{x}_i{}^T \mathbf{x}_j$$

- If both α_1 and α_2 are not at bounds, then $b^{\text{new}}(1) = b^{\text{new}}(2)$ – as defined above – and both values are valid.
- If both α_1 and α_2 are at bounds, either 0 or C, then any linear combination of $b^{\text{new}}(1)$ and $b^{\text{new}}(2)$ will satisfy KKT conditions. Platt chooses the average between both values.

Convergence of the algorithm

After having updated the value of b, the algorithm repeats the previous step by selecting a new working set $W_{i,j}$ until convergence has been achieved for some tolerance criterion ϵ. For the SVM problem, KKT conditions are necessary and sufficient for any solution – see Appendix B for details. Hence SMO convergence to the optimal solution might be checked by verifying the following conditions:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \\ 0 < \alpha_i < C &\Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) = 1 \\ \alpha_i = C &\Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \leq 1 \end{aligned}$$

The reader might refer to Appendix B to have a detailed account on how to easily check the satisfaction of KKT conditions.

2.3.2 Stochastic Gradient Descent

In what follows we assume that we want to train some function f_w which depends on a parameter w, either a real number or a vector of real numbers. We also introduce a *loss function* Q to reformulate empirical risk:

$$R_n(f_w) = \frac{1}{n} \sum_{i=1}^{n} Q\left(f_w(x_i), y_i\right)$$

For example, in the case of a binary classification problem, we might select the *0-1 loss function*:

$$Q\left(f_w(x), y\right) = \mathbb{1}_{\{f_w(x) \neq y\}}$$

Alternatively, the SVM uses the *hinge loss*:

$$Q\left(f_w(x), y\right) = \max[0, 1 - y f_w(x)]$$

Numerous Machine Learning techniques, among which the SVM, involve a minimization program which seeks to determine an optimal function f_w^* in terms of empirical errors by reaching the corresponding optimal value for w – a maximization program is equivalent to the minimization of the negative or the inverse of the objective function. As a consequence, solutions for these problems require very often numerical approximations that work by constructing sequentially a series $w^{(1)}, w^{(2)}, ...$ such that:

$$f_{w^{(t)}} \xrightarrow[t \to +\infty]{} f_w^*$$

To construct the sequence $\{w^{(t)}\}_t$, an algorithm updates the value of w at each iteration according to some *direction of descent* D_t multiplied by $\eta > 0$:

$$w^{(t+1)} \equiv w^{(t)} + \eta D_t$$

A common choice for the descent direction is the negative of the gradient of the function with respect to w, $\nabla_w f_{w^{(t)}}$. This method is simply known as *gradient descent*. The sequence $\{w^{(t)}\}_t$ is defined as follows:

$$w^{(t+1)} \equiv w^{(t)} - \eta \nabla_w R_n \left(f_{w^{(t)}}\right) = w^{(t)} - \frac{\eta}{n} \sum_{i=1}^{n} \nabla_w Q \left(f_{w^{(t)}}(x_i), y_i\right)$$

where η is a hyper-parameter known as the *learning rate* or *step size*.

The main issue of gradient descent is that it might be very computationally expensive to achieve: at each step of the optimization algorithm, we are computing n different gradients, one for each instance in the data sample. An important computational improvement to this technique which is widely used in the presence of very large data sets is *stochastic gradient descent* (SGD). In SGD, for some randomly picked $I \in \{1, ..., n\}$ the sequence $\{w^{(t)}\}_t$ is defined as:

$$w^{(t+1)} \equiv w^{(t)} - \eta \nabla_w Q \left(f_{w^{(t)}}(x_I), y_I\right)$$

The updated weight vector $w^{(t+1)}$ computed under SGD is an unbiased estimator of $w^{(t+1)}$ computed under gradient descent. We also note that by making the learning rate a decreasing function of the step size $\eta(t)$, with the rate decreasing at an adequate pace, we will ensure that the algorithm will converge almost surely to the optimal solution.

The SGD algorithm can be summarized as follows:

Algorithm 1 *Stochastic Gradient Descent*

1: Input $\{(x_i, y_i)\}_i$, η, $w^{(0)}$ and ϵ
2: $w \leftarrow w^{(0)}$
3: Randomly select $I \in \{1, ..., n\}$ with uniform distribution
4: Repeat until $|\nabla_w Q \left(f_{w^{(t)}}(x_I), y_I\right)| < \epsilon$:
   - Randomly select $I \in \{1, ..., n\}$ with uniform distribution
   - $w \leftarrow w - \eta \nabla_w Q \left(f_{w^{(t)}}(x_I), y_I\right)$
5: Output w

The advantage of the SGD algorithm is obvious: consider the case where $n = 100,000$, then after a single iteration SGD has already made a step towards the optimal value while gradient descent needs to make $100,000$ computations before improving the value of w. Although SGD might increase the total number of steps, each step is computed so quickly that it normally converges more rapidly than gradient descent.

To finish off, we state the main equations that characterize the Stochastic Gradient Descent implementation that we have used in our SVM training procedure. Recalling the regularized formulation of the SVM – where $f(\mathbf{x}_i) = \mathbf{w}^T \phi(\mathbf{x}_i)$:

$$\arg\min_{f \in \mathcal{F}} \left(\sum_{i=1}^{n} \max\left[0, 1 - y_i f(\mathbf{x}_i)\right] + \frac{\beta}{2} \|\mathbf{w}\|^2 \right)$$

In this case we are not just optimizing with respect to a loss function – corresponding to the 1$^{\text{st}}$

term – but also with respect to a regularizer – corresponding to the 2nd term. The differential of $f(\mathbf{x}_i)$ with respect to $\mathbf{w}$ is simply given by $\mathbf{x}_i$. Hence, for some random I the SGD SVM equations are given by:

$$\max\left[0, 1 - y_I f(\mathbf{x}_I)\right] = 1 - y_I f(\mathbf{x}_I) \quad \Rightarrow \quad \mathbf{w}^{(t+1)} \equiv \mathbf{w}^{(t)} - \eta(t)\left(\beta\,\mathbf{w}^{(t)} - y_I \mathbf{x}_I\right)$$

$$\max\left[0, 1 - y_I f(\mathbf{x}_I)\right] = 0 \quad \Rightarrow \quad \mathbf{w}^{(t+1)} \equiv \mathbf{w}^{(t)} - \eta(t)\beta\,\mathbf{w}^{(t)}$$

The learning rate is updated with the following equation:

$$\eta(t+1) = \frac{1}{\beta(t_0 + t)}$$

t_0 is chosen at the start of the algorithm based on a heuristic suggested by L. Bottou – see Bottou and Bousquet (2011):

$$t_0 = \eta(0) = \beta^{-\frac{3}{4}}$$

2.3.3 Kernel approximation

A critical motivation for incorporating kernels into the Support Vector Machine was that they allowed to avoid the problem of computing dot products between new features $\phi(\mathbf{x}_i)$, $\phi(\mathbf{x}_j)$ by directly calculating the kernel value $K(\mathbf{x}_i, \mathbf{x}_j)$.

Despite the computational efficiency achieved through the "kernel trick", it turns out that for very large datasets the number of computations involved in calculating the kernel matrix $K = (K_{i,j})_{0 \leq i,j \leq n} = (K(\mathbf{x}_i, \mathbf{x}_j))_{0 \leq i,j \leq n}$ might be prohibitively high: indeed considering that kernels are symmetric, determining the whole kernel matrix K requires $\frac{1}{2}n(n+1)$ computations; if the dataset has $100,000$ instances, that involves more than $5,000$ million operations on vectors of size p to compute a single kernel matrix – the final number is compounded by the fact that we have to make further computations to find optimal values for hyper-parameters such as C. For extremely large datasets, there might not even be enough memory for the kernel matrix to fit in.

Furthermore, intuitively it is easy to understand that computing the whole matrix K is relatively wasteful: for example, when applying the trained SVM to a new vector, we are only interested in the kernel values with respect to support vectors, because for the rest of training data coefficients alpha are equal to 0. Therefore it seems reasonable to try to optimize the calculation of the matrix K.

A possible solution to this issue is to avoid the calculation of the kernel matrix, which implicitly maps the data into higher dimensional spaces, and instead focus on obtaining upstream the mapping of the data ϕ, to which we train a linear SVM – which are computationally more efficient than kernelized SVMs. However, instead of explicitly calculating the mapping ϕ, an interesting option is to approximate it with another, simpler mapping Z. Ideally, we could also lever the mapping Z, not only for training the SVM, but also for testing it on new data.

We now succinctly present a method which achieves both objectives when working with the Gaussian kernel.

Approximating the Gaussian kernel's implicit mapping with random features

Our implementation of the SVM tests a method for approximating feature mappings efficiently. Also known as *Random Kitchen Sinks* (Rahimi and Recht, 2007), this technique allows to approximate upstream features implicitly mapped through *shift-invariant kernels*, a category to which the

Gaussian kernel belongs. Shift-invariant kernels have the following defining property:

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i - \mathbf{x}_j)$$

The Random Kitchen Sinks method works by implementing a *randomized* feature map $Z : \mathbb{R}^p \rightarrow \mathbb{R}^q$, $q < p$, which allows to approximate the kernel value:

$$Z(\mathbf{x}_i)^T Z(\mathbf{x}_j) \approx \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \approx K(\mathbf{x}_i, \mathbf{x}_j)$$

To understand how Z is constructed, recall that a valid kernel K is one for which the corresponding kernel matrix is positive semi-definite. We can hence apply the following theorem to valid kernels, which underpinds the Random Kintchen Sinks method:

Theorem 3 (Bochner) *A continuous kernel $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i - \mathbf{x}_j)$ on $\mathbb{R}^p$ is positive semi-definite if and only if $K(\mathbf{x}_i - \mathbf{x}_j)$ is the Fourier transform of a non-negative measure $\mathbb{P}$.*

From the above theorem, it comes:

$$\begin{aligned} K(\mathbf{x}_i - \mathbf{x}_j) &= \int_{\mathbb{R}^p} e^{\iota \boldsymbol{\omega}^T (\mathbf{x}_i - \mathbf{x}_j)} d\mathbb{P}(\boldsymbol{\omega}) \\ &\approx \frac{1}{q} \sum_{k=1}^{q} e^{\iota \boldsymbol{\omega}_k^T (\mathbf{x}_i - \mathbf{x}_j)} \\ &= \sum_{k=1}^{q} \sqrt{\frac{1}{q}} e^{\iota \boldsymbol{\omega}_k^T \mathbf{x}_i} \sqrt{\frac{1}{q}} e^{-\iota \boldsymbol{\omega}_k^T \mathbf{x}_j} \end{aligned}$$

where ι stands for the imaginary unit. With appropriate scaling the kernel value can hence be interpreted as an expectation over a probability distribution, which can be approximated simply through a Monte Carlo method by the Law of Large Numbers. More precisely, it can be shown that for the Gaussian kernel K the probability distribution $\mathbb{P}$ is given by:

$$d\mathbb{P}(\boldsymbol{\omega}) = \frac{1}{(2\pi\sigma^2)^{q/2}} \exp\left\{ -\frac{\|\boldsymbol{\omega}\|^2}{2\sigma^2} \right\}$$

The corresponding randomized feature map Z is simply:

$$Z(\mathbf{x}_i) = \sqrt{\frac{1}{q}} \left(e^{\iota \boldsymbol{\omega}_1^T \mathbf{x}_i}, ..., e^{\iota \boldsymbol{\omega}_q^T \mathbf{x}_i} \right)$$

where $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_q$ have been drawn randomly from distribution $\mathbb{P}$.

One additional advantage of this method is that, once the linear SVM has been trained, evaluation at a new data point $\mathbf{x}$ is also speeded up by applying mapping Z to $\mathbf{x}$ and then simply computing $f(Z(\mathbf{x})) = \mathbf{w}^T Z(\mathbf{x}) + b$, instead of having to compute kernel values.

Theoretical results ensure that these randomized feature mappings approximate uniformly shift-invariant kernels such as the Gaussian one with a very small error $err(q)$ which depends on the dimension q.

Chapter 3

The Health Insurance Pricing Problem

Presentation of the insurance framework and preliminary analysis of data

3.1 Introductory comments

The purpose of this chapter is twofold: we first give the reader a quick overview of the formal insurance pricing problem, with a particular emphasis in health insurance, as well as of the structure of the French market. Afterwards, we describe and undertake a preliminary analysis of the dataset that will be used for testing the empirical performance of Support Vector Machines.

We will equivalently use the expressions "SMI institution", "insurance institution", "insurance company" and similar terms from now.

3.2 Principles of Non-Life insurance pricing

In this section we will give a very high-level overview of the functioning of the insurance industry in order to understand the fundamental problem that every insurance company faces when doing business. After that, we will formalize the insurance pricing problem in the context of Non-Life insurance and health insurance. This should help the reader to better understand how the Support Vector Machine can be implemented in the context of insurance pricing.

3.2.1 Basic mechanics of the insurance industry

The insurance industry is a social institution which aim is to allow individuals and companies to protect themselves against risk, like for example a car accident or a storm: in exchange for a premium π_C paid by the individual, the insurance company will bear any financial loss derived from the realization of the insured risk.

One important particularity of the insurance industry is what we call the *inverted production cycle*: whereas standard companies will know their cost before they know their revenues, insurance companies must offer a price π_C before knowing what the cost associated to that transaction will be. It is the insurance company responsibility to assess the likely financial consequences of the realization of the aforementioned risk and compute a fair price π_C to cover such potential future losses. Of course, as any company's goal is to generate a profit, the price π_C should be made up of a *pure premium* π, which captures the likely financial loss if the risk materializes, a quantity c to cover related costs and a margin m which should enable the company to generate a profit.

The insurance business is governed by an essential mathematical principle: the *law of large numbers*. By insuring and pooling together a large number of equivalent risks that are independent, the company is reassured that the sum of pure premiums will allow it to break even with respect to the payments made to clients.

To finish off, let characterize a risk as a triplet of random variables (I, T, X) where I indicates whether the risk materializes, T the time when it does so and X the payout due by the insurance company to the insured individual. The insurance industry might be split in 2 big areas:

- **Life Insurance**: in life insurance, the insured risk relates to the *survival of a person*. A life insurance contract might specify that a lump sum X will be paid to the family of the client at the time of his death T.
  The risk is mainly concentrated on T because I is non random and X is relatively well defined at the contract signature.
- **Non-Life Insurance**: as its name indicates, Non-Life Insurance includes basically all insurance companies which business does not depend on the survival of clients – car insurance, health insurance, income protection insurance, etc.
  In this case, the risk is mainly concentrated on X and I.

We will now formalize mathematically the insurance pricing problem. We will only be interested in the pure premium π: the calibration of the commercial premium π_C is primarily driven by the volatility of the risk as well as by business strategy considerations, matters that we will not discuss further in this memoir.

3.2.2 Formalization of the insurance pricing problem

We first derive abstractly the optimal pure premium π when the insurer seeks to minimize its expected squared loss. We then present a common modelling approach in Non-Life insurance to represent a random financial claim X.

The optimal pure premium

X represents the payment due by the insurance company to a client in case the risk specified in the insurance contract materializes. We assume here that the quantity X is a random variable, which is the standard case in non-life insurance and, more particularly, in health insurance. The insurer must nonetheless offer to its client a fixed price π at the signature of the contract – assume for now that $\pi_C = \pi$. Its objective is to find the best possible value for π.

To achieve this goal, the insurer might want to minimize its expected quadratic loss, which penalizes over and under-evaluations of π. It is straightforward to show that:

$$\mathbb{E}\left[(X - \pi)^2\right] = \mathbb{V}\text{ar}[X] + (\mathbb{E}[X] - \pi)^2$$

The insurer must therefore sets the following price to minimize its expected quadratic loss:

$$\pi = \mathbb{E}[X]$$

The problem then becomes the development of a probabilistic model that can reasonably represent the random variable X in order to compute the above expectation. Although the insurer might model directly the total payment X, it is common practice in the Non-Life insurance business to deconstruct X in two parts. We will now present such an approach, which is the one we have adopted in our study.

The Occurrence-Severity model

Let 1, ..., n be the insurance policies, $N_i \in \mathbb{N}$ the random number of claims for policy i on a given year, $C_i \in \mathbb{R}_+$ its total cost *conditional on occurrence – i.e.* $N_i > 0$ – and X_i the payment to be made by the insurer. We model whether there is *at least one* claim per year and the average total cost C_i given occurrence.

More formally, occurrence I is a random indicator function – we drop subscripts:

$$I = \mathbb{1}_{\{N>0\}}$$

The expectation of payment X is given by:

$$\begin{aligned}\mathbb{E}[X] = \mathbb{E}[\,IC\,] &= \mathbb{E}[\mathbb{E}[\,IC \mid I\,]] \\ &= \mathbb{P}(I=1)\mathbb{E}[\,IC \mid I=1\,] + \mathbb{P}(I=0)\mathbb{E}[\,IC \mid I=0\,] \\ &= \mathbb{P}(I=1)\mathbb{E}[\,C \mid I=1\,] \\ &= \mathbb{P}(N>0)\,\mathbb{E}[\,C \mid N>0\,]\end{aligned}$$

The last equation above helps us understand the utility of the occurrence-severity model: when the reliability of the specific number of claims data is dubious but we are confident that a strictly positive number of claims is robust – i.e. $N > 0$ is a trustworthy indicator to whether there has been at least one claim but not to the total number of claim there has been – modelling the risk X as the product IC allow us to exploit the information contained in N while circumventing its drawbacks.

3.2.3 The specific case of health insurance and the reimbursement principle

Given the nature of our dataset, in this paper we will be interested in health insurance, a type of Non-Life Insurance related to medical expenses. More specifically, we will be modelling *medical expenses reimbursements*, which is an essential feature of any *supplementary medical insurance* (SMI) contract.

In France, SMI contracts are structured such that the contract signatory is reimbursed fully or partially of any medical expense non covered by the mandatory Health Social Security. Medical expenses are generated by medical acts, for example the acquisition of an antibiotic or a medical consultation. These acts are very precisely defined by the Social Security.

Each medical act is associated to a total cost or expense E, which is a random variable. For each specific medical act the Social Security defines a *Reimbursement Base* RB to which it applies a *reimbursement rate* r: the Social Security reimburses the amount rRB. The difference between the Reimbursement Base and the Social Security reimbursement is called the *copayment* and is trivially equal to $RB(1-r)$. The fraction of the expense which is above RB is the *surcharge*. In addition, for some specific medical acts a *fixed participation* is deducted from the Reimbursement Base: for example, since 2005 any consultation with a General Practitioner or a specialist which is covered by the Social Security has a fixed participation of 1€ which must be paid by the individual. The fraction of expense E which is financed by the individual is equal to $E - rRB$ and is made up of the fixed participation if applicable, the copayment and the surcharge. We will designate this quantity, which is a random variable, by X. The individual might decide to protect himself against the expense X and take out an SMI policy. The SMI contract will reimburse partially or totally the quantity X to the insured individual, depending on the coverage of the contract – the fixed participation is normally excluded. This coverage might be expressed in euros or as a fraction of E or RB. If the contract does not cover fully X, the remaining sum is usually called the *remaining balance* and must be financed by the individual.

The objective of the insurance company offering the SMI contract is to model the expected reimbursement $\mathbb{E}[X]$, such that the (pure) price of the contract is equal to π. In this case, N corresponds to the number of medical acts covered by the contract and C the cost of a medical act.

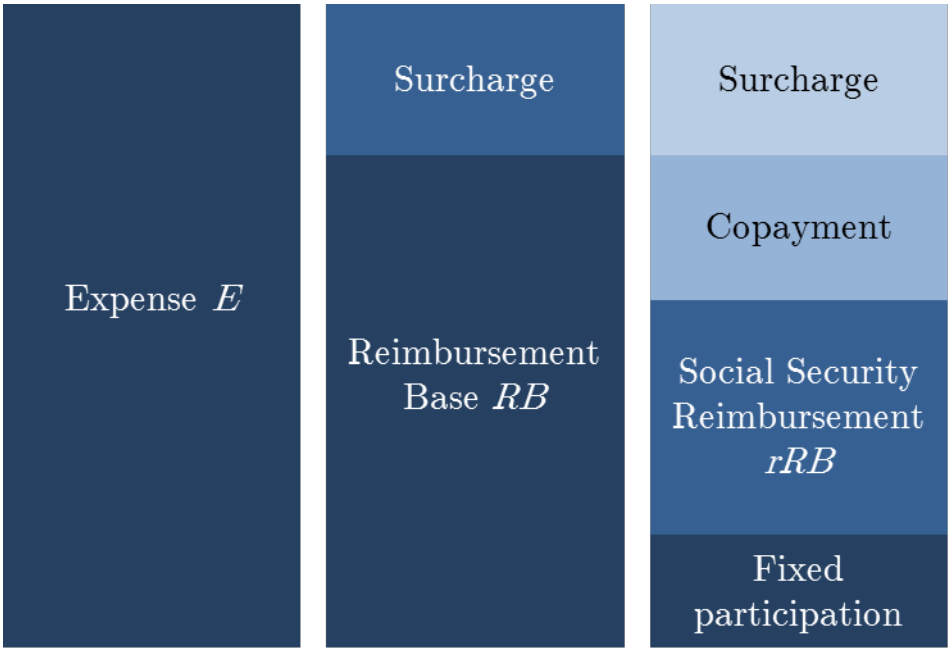

Figure 3.1: *Decomposition of expense E with respect to Social Security reimbursement mechanics*

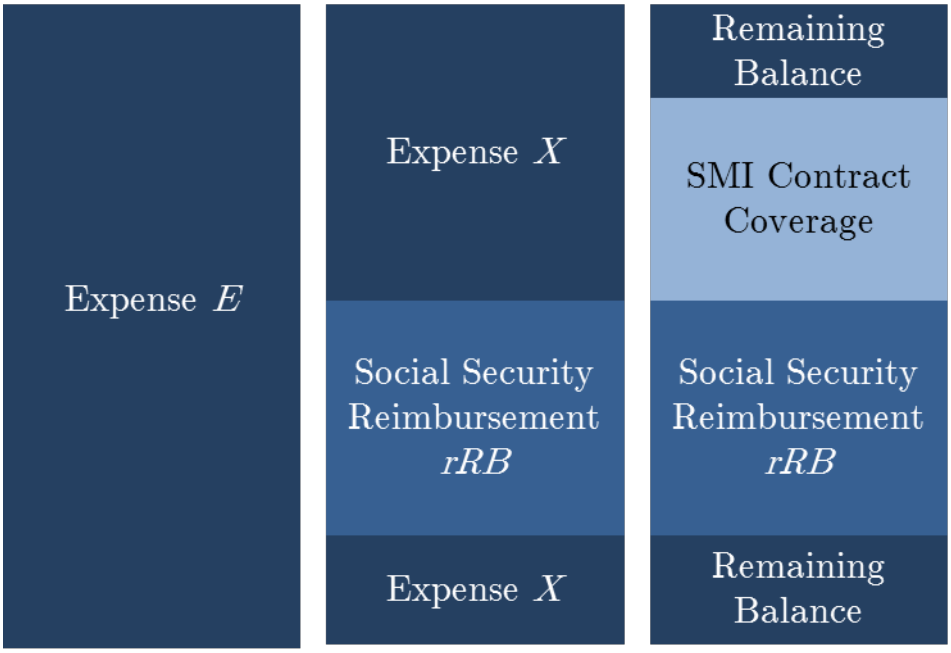

Figure 3.2: *Decomposition of expense E with respect to the SMI contract coverage*

Note that if we model a particular type of act a, conditional on the legislation remaining unchanged, the random expense X_a associated to a has the same distribution as the total expense E_a because the Social Security Reimbursement is known for each medical act:

$$\mathbb{P}(X_a \leq x) = \mathbb{P}(X_a + r_a RB_a \leq x + r_a RB_a) = \mathbb{P}(E_a \leq x + r_a RB_a)$$

However this is not the case of our modelling framework, as we will not distinguish types of medical acts due to the lack of data.

The next section is devoted to a brief overview of the French market.

3.3 The French health insurance system: the market and its participants

The concept of *Social Protection* relates to the set of social institutions which aim is to protect individuals from the financial consequences derived from *social risks*, which can be grouped in 4 main domains: health, retirement, family and employment. Such institutions include the State's *Social Insurance system* as well as the private insurance industry. In this paper we are only interested in health-related risks. Considering the reimbursement principle that we expounded above, it is then evident that the health insurance industry is France is highly dependent on the the State's Social Insurance system, thus it is important to understand the role it plays in ensuring the population is covered against health risk as well as how other market participants supplement the Social Security.

We will now describe the structure of the health insurance market in France. Our discussion will be illustrated with consumption figures, which come mainly from the "2014 Health Expenditures" report (*Les dépenses de santé en 2014 - Résultats des Comptes de la Santé, édition 2015*) published in 2015 by the French Directorate of Research, Studies, Evaluations and Statistics (*Direction de la Recherche, des Etudes, de l'Evaluation et des Statistiques*, DREES) which is attached to the Ministry of Social Affairs and Health (DREES, 2015).

The main metric used by the DREES to assess the size of the health market in France is the "consumption of medical care and goods" (*Consommation de Soins et de Biens Médicaux*, CSBM) which is defined as "*the total value of goods and services engaged in the treatment of a temporal health perturbation*". CSBM can be split is 5 subgroups:

- Consumption of hospital goods (46.5% of CSBM in 2014);
- Consumption of city care (medical practices, clinics, laboratories, dental-related expenses, etc., 26.2%);
- Medicaments consumption (17.8%);
- Consumption of other medical goods (which includes among other things optical-related expenses, 7.2%);
- Patient transport expenses (2.3%).

The latest figures available at the time of writing were for the year 2014, during which the total amount for CSBM was 190.6€ billion, which represents approximately 2,900€ per person and 8.9% of French GDP. The growth of this figure has slowed in recent years: whereas in 2007 CSBM grew at an annual rate of +4.0%, the rate was +3.1% in 2008 and 2009 and has been inferior to +3.0% since then (+2.5% in 2014).

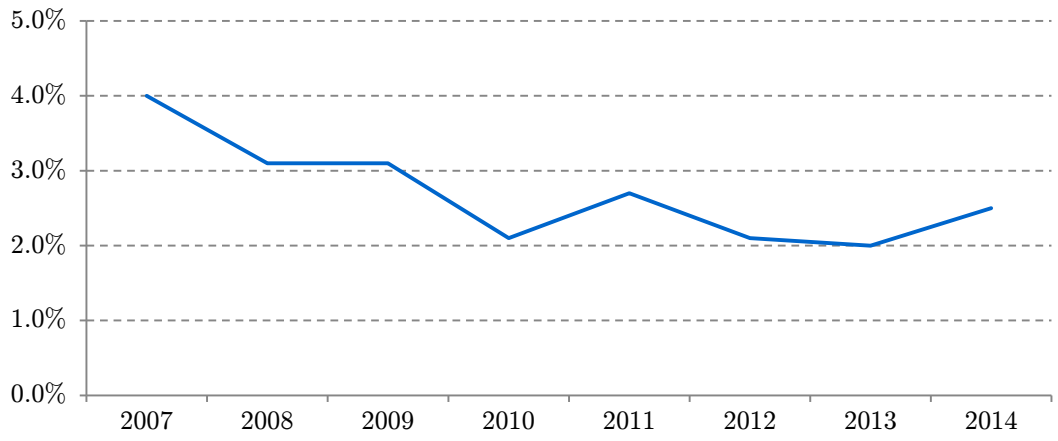

Figure 3.3: *Growth rate of CSBM*

The financing of CSBM is ensured by 4 main actors: the Social Security, the State, SMI institutions and households.

The *French Social Security* provides a basic insurance to the whole population in exchange for mandatory contributions, protecting contributors against social risks. It financed 76.6% of CSBM in 2014, from 76.2% in 2010 and 76.7% in 2007. This financing is distributed among the different branches of the Social Security system:

- The general regime (*régime général*): it is the principal component of Social Security, as it covers approximately 80% of the population;
- The agricultural regime (*régime agricole*), which covers farmers and employees of the agricultural sector;
- Self-employed regime (*régime social des indépendants*);
- Special regimes (*régimes spéciaux*): there are multiple professional special regimes in France – sailors, miners, employees from the Bank of France, etc.

The *State* covers directly a small fraction of CSBM – 1.4% in 2014, from 1.3% in 2010 and 1.4% in 2007. This financing relates principally to medical expenses for illegal immigrants, war handicapped persons and the payment of the Universal Health Coverage (*Couverture Maladie Universelle*, CMU) aimed at people with low economic resources.

SMI institutions cover the fraction of expenses non reimbursed by the Social Security. SMI institutions are particularly crucial in the domains of dental-related expenses, optical-related expenses and medicaments: for example, in 2014 21.7% of city care expenses – which includes dental expenses – and 38.9% of expenses in other medical goods – which includes optical expenses – were covered by SMI institutions; the figure for medicaments was 13.7%. On the other hand, only 5.3% of hospital-related expenses were covered by these companies. In total, SMI institutions covered 13.5% of CSBM in 2014, up from 13.4% in 2010 and 13.0%. The group of SMI institutions can be split in 3 different categories. Each kind of institution is governed by different legislation and has a distinct functioning:

- *Mutual insurance companies* (*Mutuelles*): their distinctive trait is that they are non-profit organisations. They operate on the base of solidarity between members and their financing comes mainly from members' contributions. Having no equity holders, members control the company such that each member has the same weight in votes. In France they are governed by the *Code de la Mutualité*.
- *Provident Society* (*Institution de Prévoyance*): as mutual insurance companies, they are non-profit organizations; however the ownership system is different: the management of the society is undertaken by the social partners (*partenaires sociaux*) – namely employees' unions and employers' organizations. Moreover, they only offer group insurance plans. They are governed by the *Code de la Sécurité sociale*.
- *Insurance companies*: insurance companies in France are private organizations which seek to generate a profit. They are governed by a specific legislation aimed at insurance companies.

The remaining balance of CSBM is paid by households. They represented 8.5% of CSBM financing in 2014, down from 9.1% in 2010 and 9.0% in 2007.

Individuals in our database have received benefits – i.e. reimbursements – mainly from the general regime. It is nonetheless worth noting that a sizeable fraction of the studied population is also covered by a mandatory supplementary regime which complements the general regime: the *local regime of Alsace-Moselle*. The Alsace-Moselle regime is a heritage from the period during which the region belonged to Germany (1870-1945) and hence its inhabitants were covered by the German Social Security system, which was more favourable. This last information is crucial when modelling the pure

premium, as individuals covered by the Alsace-Moselle regime will have greater total reimbursements from the Social Security, hence the expected reimbursement paid by SMI institutions will be lower. The price charged to inhabitants of this region should be adapted to reflect this feature.

Now that we have expounded the main concepts underpinning insurance pricing, we will shift to the next section where we present the proper health insurance data set.

3.4 Structure and main characteristics of the dataset

The dataset consists on a pool of SMI policies – 159,374 in total – coming from databases from various SMI institutions with French operations. For each policy we have annual data on the expense reimbursed by the insurance institution as well as on the number of medical acts, figures which we will try to predict. In addition, we also have various characteristics – age, gender, place of residence, type of contract, etc. – about the policy's beneficiary. Policies come from group insurance plans as well as from individual insurance contracts.

3.4.1 Pretreatment of data

We first note that Mazars Actuariat had already pretreated gross data to structure it in an meaningful and exploitable way (Huther, 2014): for example, given that expense reimbursement data spans the years 2009, 2010 and 2011, Mazars Actuariat accounted for price inflation by inflating reimbursements from 2009 and 2010 according to French inflation data. Mazars Actuariat has also analysed the underlying insurance policies to create a standardised nomenclature for the contracts in the database – see below.

We have analysed and cleaned further the database on top of what Mazars Actuariat had already done. The dataset has undergone a rigorous scanning to identify corrupt, incomplete and unexploitable data. For simplicity, these data points have been directly excluded from our analysis. For example:

- For a negligible fraction of the portfolio, the annual reimbursed expense was negative; in the absence of further details on this phenomenon, we have eliminated from our database all policies for which the reimbursed expense was negative.
- For some policies, the number of annual medical acts was null whereas the annual reimbursed expense was strictly positive. In the absence of explanations for this phenomenon we have suppressed all policies with this characteristic.

We have also monitored the quality of the attributes available. Although the initial number of attributes was relatively high, we did not have enough information on some of them to be able to identify what they represented. In other cases, although the attribute was understandable, the majority or all of the values it could take were not.

Incomprehensible attributes were eliminated from the database. Attributes presenting a majority of unknown values were also erased from the database; for some attributes for which the number of unknown values was negligible, only the unknown values were eliminated.

We end up with an observation regarding outliers treatment. As it has been explained, one of the advantages of the Support Vector Machine is that it is insensitive to outlier data, support vectors being the only data points that matter. Thus it is neither necessary to undertake an outlier analysis nor to eliminate these instances from our database.

The portion of the initial database affected by this pretreatment is negligible: after this procedure, the size of the remaining dataset is 153,673, a decrease of only 5,701 (-3.58%) data points with respect to the initial size of the database.

3.4.2 Splitting data between a training set and a validation set

In Machine Learning, the common strategy to assess the generalization ability of a given model or algorithm is to split the available data into two separate datasets:

- A **training set** which is used to determine the optimal parameters of the machine;
- A **validation set** which is used to test the accuracy of the trained machine by applying it on data points which it has not yet "seen".

The hypothesis underpinning this approach is that both the training set and the validation set have been generated through the same underlying stochastic law and that they are both independent. We have followed a standard approach which is to set the size of the training set equal to 90% of the total size of the dataset and the validation set to the remaining 10%. The allocation of instances to either set is determined randomly through a uniform distribution. However, the statistics presented in the following pages pertain to the whole dataset.

3.4.3 Attributes of insurance policies

In the following, we designate by "internal data" data which was already available within the original database; moreover we designate by "external data" data that has been obtained from external sources and that we have added to the original database.

Internal data

As we have remarked, some of the available attributes were eliminated during the pretreatment phase. The remaining attributes of contracts are listed hereafter – we only state the meaningful attributes, ignoring variable such as the policy number. Unless otherwise specified, all attributes refer to the *beneficiary* of the policy and not to the policyholder:

- **Beneficiary**: the beneficiary of a policy might be the policyholder, his spouse or his child.
- **Social Security regime of the policyholder**: the Social Security regime of the policyholder, either the general regime, the local regime of Alsace-Moselle or the special regime of the French State-owned National Railway Company (*Société Nationale des Chemins de Fer*, SNCF). The regime of the policyholder should be the same regime as the beneficiary except for very few cases.
- **Birth date**: the birth date allows us to compute the rounded integer age of the beneficiary.
- **Gender**: either male or female.
- **French region**: this attribute specifies the region where the beneficiary lives. Until December 31st 2015, France was divided in 27 regions. From January 1st 2016, the country has seen the number of regions decreased to 18. Our data being for years 2009, 2010 and 2011, this political change had to be taken into account when introducing some external variables – see below.
- **French department**: this attribute specifies the department where the beneficiary lives. Each French region is divided in departments; there are 101 in total, a number which was not modified by the region reform cited above.
- **Insurance contract**: the insurance contract the policyholder has subscribed. Each contract has different guarantees and coverage.
- **Insurance contract option**: a detailed analysis of insurance contracts has been undertaken by Huther (2014). A major pattern emerged: each contract offered a "normal" version, an "improved" version and a "maximal" version of the contract. For those contracts for which the list of options seemed different, guarantees were analysed and the options linked to the 3 options cited.

- **Type of contract**: either a group insurance plan or an individual insurance contract.

The dataset also contains expense data. Although we will not be using this variable for prediction for obvious reasons, we will analyse its behaviour in this chapter.

External data

In addition to the 9 attributes described below, we have decided to include external attributes that might help explaining the reimbursed expense for a particular policy. By adding further attributes not only will we be able to sophisticate the modelling of the reimbursement but also we will test the ability of the Support Vector Machine to identify patterns and relationships between the target variable and attributes which might not be intuitively grasped by a person. This is all the more significant in an environment where insurance companies have improved access to large pools of biometric, social, economic and environmental data.

We make some methodological precisions concerning the treatment of external data:

- The additional attributes' values have been determined for each policy by looking at the department or the region where the policyholder lives.
- If there is differentiated data for years 2009, 2010 and 2011 it has been taken into account when assigning a value to the attribute for a given policy, if not the most recent figure has been kept.
- In case external data is available for a geographic division which does not correspond to a region, one of the following two approaches has been followed:
  - If the geographic division G for which data is available is bigger than a region, regions belonging to G have been identified and the attribute value for each region has been set to be equal to the value for G;
  - If the geographic division g for which data is available is smaller than a region, smaller geographic divisions $g_1, ..., g_k$ forming a region have been identified and the attribute value for the region has been set to be equal to the average of values for $g_1, ..., g_k$.

Data sources are specified next to the attribute; INSEE stands for the French Statistics Institute (*Institut National de la Statistique et des Etudes Economiques*) and CNAMTS for the Employees Health Insurance National Fund (*Caisse National de l'Assurance Maladie des Travailleurs Salariés*). The external variables included in the procedure are:

- **Long term sickness** (*Affectation de Longue Durée*, ALD; CNAMTS, Directorate of strategy, studies and statistics): long term sickness are precisely defined by the Social Security, which covers 100% of the related medical expenses. The data is expressed as a percentage over the population belonging to the general regime for a specific department.
- **Functional Limitation** (INSEE): functional limitations are defined by the INSEE as the "*the difficulty to achieve elementary acts*" such as walking 500m. The data is expressed as a percentage over the population for a specific region.
- **Pharmacies** (the French National Order of Chemists): the number of pharmacies per department. The data is expressed as a percentage over the population for a specific department.
- **Gross mortality** (INSEE): the number of deaths with respect to the annual average population of the department. The data is expressed as a percentage over the population for a specific department.
- *Consumption habits* (INSEE): we have gathered data from INSEE about French households average annual expenses for different classes of goods and services. The data is expressed as a percentage over the total household average annual expense for a specific region:

- **Oil, fat, sugars and related products**
- **Fruits and vegetables**
- **Alcohol**
- **Tobacco**
- **Higher education and books**

- **Unemployment** (INSEE): the unemployment rate. The data is expressed as a percentage over the population for a specific department.
- **Real GDP per capita** (INSEE): the real GDP divided by the average population for the year. The data is expressed in euros for a specific department.
- **Coastal department** (no specific source): a indicator variable that is equal to 1 if the department has direct access to the sea, 0 otherwise.

Including the above attributes brings the total number of variables of our model to 21, which remains a reasonable number.

3.4.4 Characteristics of the portfolio

In this section we study the main characteristics of the portfolio of policies in order to have a better comprehension of the population being studied.

Distribution by type of beneficiary and gender

As expected, most beneficiaries correspond to the holder of the insurance contract (55.16%). Moreover, we notice that the total proportions of beneficiaries that are either the spouse or the child of the policyholder are equivalent.

	Male	*Female*	**Total**
Policyholder	31.51%	23.65%	**55.16%**
Spouse	4.31%	18.38%	**22.69%**
Child	11.09%	11.05%	**22.15%**
Total	**46.91%**	**53.09%**	**100%**

Table 3.1: *Portfolio composition by beneficiary and gender*

The data is consistent: we observe that a higher proportion of men among policyholders – 31.51% men against 23.65% women – is echoed by a higher proportion of females spouses – 18.38% women against 4.31% men.

Distribution by age and gender

Beneficiaries from SMI policies are concentrated in the 51-70 age tranche. The most populated age for both women and men is 64 – 1.94% of female population and 2.22% of male population are 64 years old – which is a striking coincidence. There is also an over-representation of teenagers, the 11-20 age tranche, particularly when compared with the adjacent age tranches: this is consistent with 35% of the population being between 51 and 70 years old, as these people are probably policyholders which have subscribed to a contract that covers themselves, their spouses and their children.

There are 21,918 people younger than 18 years – 14.26% of the total population – and 31 people older than 99 years – 0.02% of the total population.

	Male	*Female*	**Total**
1-10	3.31%	3.23%	**6.53%**
11-20	5.88%	5.67%	**11.55%**
21-30	3.68%	4.15%	**7.83%**
31-40	3.47%	3.97%	**7.44%**
41-50	5.34%	6.54%	**11.90%**
51-60	8.82%	9.22%	**18.04%**
61-70	8.32%	8.65%	**16.97%**
71-80	5.58%	6.82%	**12.41%**
81-90	2.30%	4.31%	**6.60%**
91-100	0.20%	0.51%	**0.71%**
>100	0.00%	0.01%	**0.01%**
Total	**46.91%**	**53.09%**	**100%**

Table 3.2: *Portfolio composition by gender and age tranche*

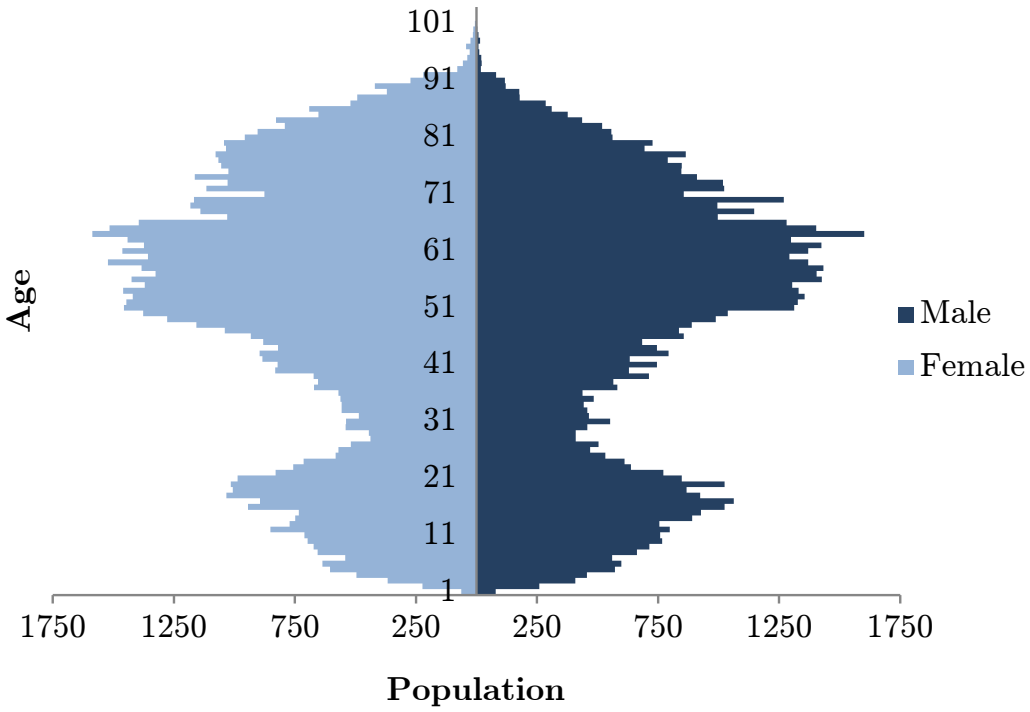

Figure 3.4: *Population pyramid by gender*

The population pyramid structure differs from the whole French population pyramid: there is a steep rise in the number of people from age 40 until 50, and around 1/4 of the population is concentrated in the 50-65 range, with a deep cavity in the range 25-35; the French pyramid does not present these characteristics, as it is relatively stable from age 5 up to age 65. Using a similar argument as above, this shape can be explained as follows: considering that 25% of the population is in the 50-65 age tranche, all these people might have their children covered by the policy; this would explain the relative trough of the pyramid in the 25-35 tranche, as the over-representation of the 50-65 tranche results in an over-representation of the 5-20 tranche through family ties.

Distribution by region and department

The geographic distribution of our portfolio is very uneven. As it has already been highlighted, most contracts beneficiaries are located in the region of Alsace-Moselle; moreover, more than 85% of all policies are located within only 3 regions – there are 22 in total. We have represented hereafter the geographical distribution of policies by region – department limits are shown for information. Darker shades of blues indicate a higher number of policies.

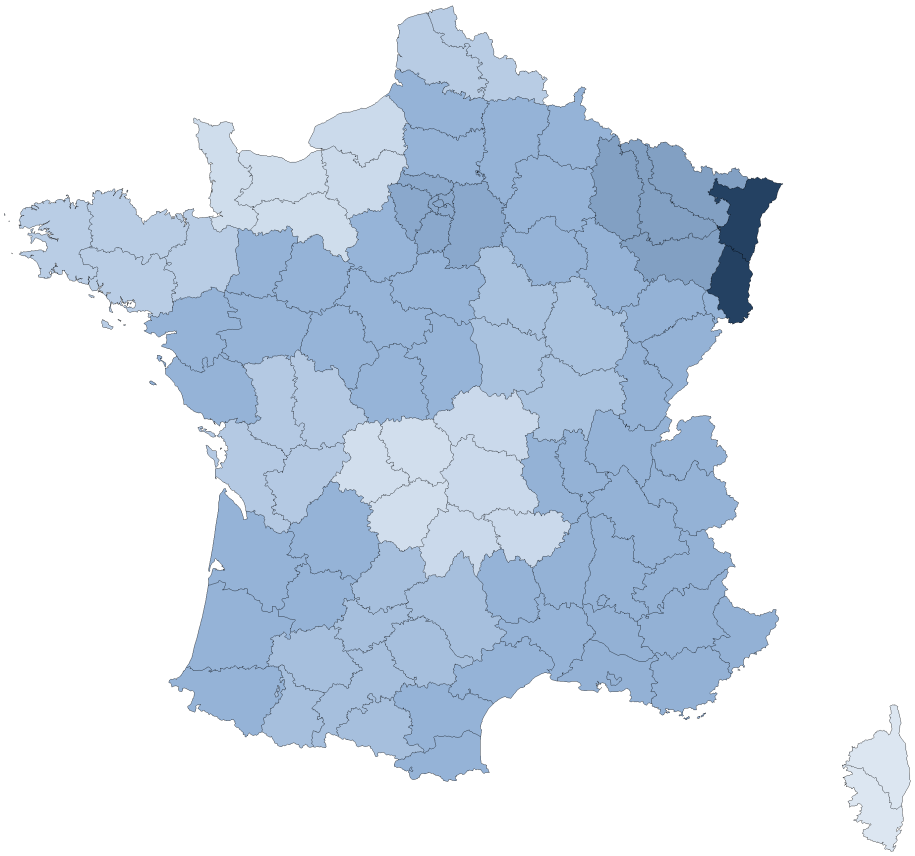

Figure 3.5: *Geographical distribution of the portfolio by region*

The darkest region – Alsace – contains 67.80% of all policies, which amounts to 104,195 contracts. It is followed by Lorraine, which is situated next to it, with 12.14% of policies, and Ile-de-France – the Paris region – with 7.22% of contracts. The region with the fewest contracts is the island of Corsica, on the south east, with just 0.02% of all contracts – 30 contracts.

Distribution by regime and type of contract

The local regime of Alsace-Moselle is overrepresented in our database, with 58.65% of the database population adhering to it. This is consistent with the data above, as 79.94% of the population lives either in Alsace or in Lorraine. There is also a significant number of policyholders that adhere to the SNCF special regime.

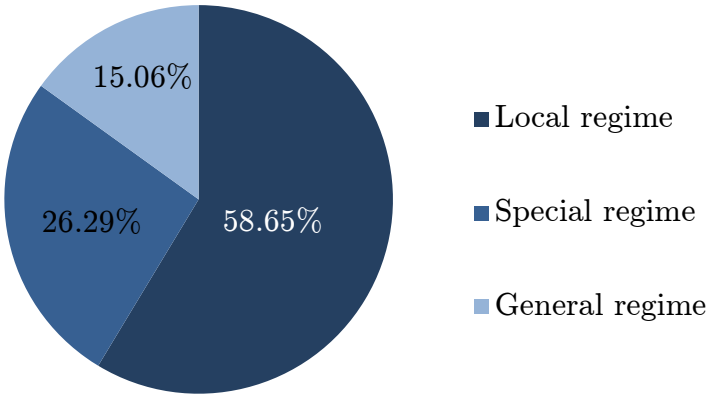

Figure 3.6: *Portfolio composition by Social Security regime of the policyholder*

In terms of individual or group plan contracts, we observe a over-representation of individual contracts, while group plans account for merely 7% of our portfolio.

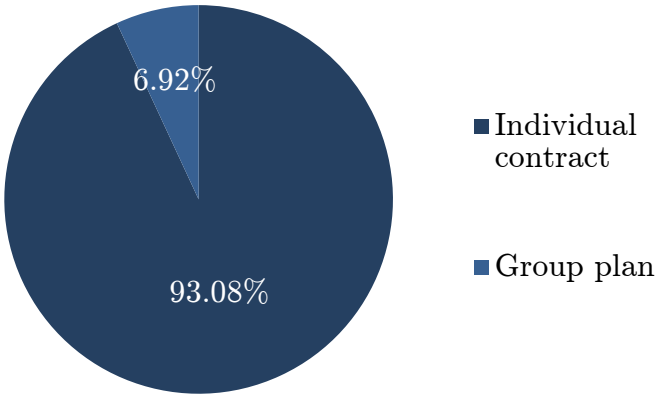

Figure 3.7: *Portfolio composition by type of contract*

Distribution by type of option

We explained that one of the attributes of an insurance policy was the contract's option. There are three of them: the *normal* version of the insurance contract, with basic coverage; an *improved* version, with further guarantees and coverage; and finally a *maximal* contract version, which provides the most complete guarantees and coverage. Our portfolio of insurance policies only has 14.8% of normal version contracts, whereas improved and maximal versions are evenly subscribed – 40.9% and 44.3% respectively.

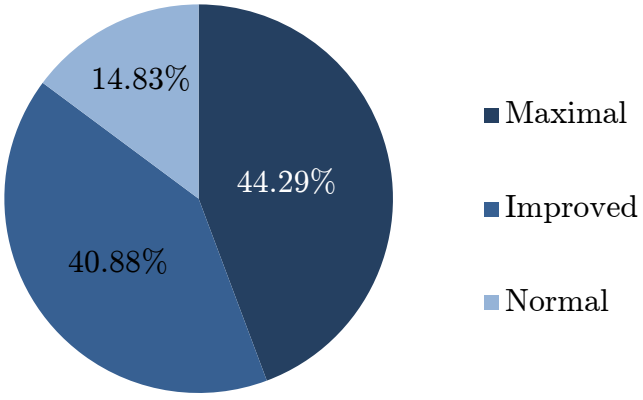

Figure 3.8: *Portfolio composition by type of contract*

Distribution by contract

Our health insurance portfolio contains 15 different contracts in total, although most policies (67.92%) correspond to one of the following 2 contracts: Isoles (35.33%) or SNCF (32.59%). Among the remaining contracts, the most populated one is the C4 contract, accounting for 9.15% of all policies. The SNCF contract is a supplementary health insurance contract linked to the SNCF special regime: 99.60% of the policyholders adhering to the special regime also subscribe to the SNCF product.

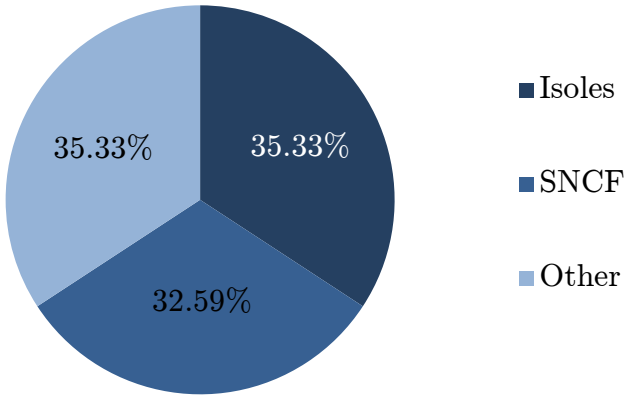

Figure 3.9: *Portfolio composition by contract*

We can assume that the population adhering to the SNCF contract has a significant degree of internal homogeneity and specificity with respect to the rest of the population: these policyholders work or are related to someone who works for the stated-owned railway company SNCF, and they are adherents to the SNCF special regime. Given the significant portion of the database population that adheres to this special regime and/or subscribes to the SNCF contract, the SNCF population should deserve specific attention to determine whether a separate modelling process could improve the overall performance of our models. We will be discussing this later on.

3.5 Preliminary study of the relationship between attributes and insurance claims

Now our goal is to analyse expense and reimbursement figures with respect to some of the attributes. This preliminary work should help us comprehending what are the main drivers of reimbursements as well as establishing a benchmark against which the Support Vector Machine should be compared.

3.5.1 The age effect

As in any actuarial study relating to life or health risk, the first factor to look at is age as experience has largely proven that this is the primordial variable driving health expenses. In the following tables and graphs we have displayed average values for ages 1 to 90 – above 90 years, as the population steadily decreases, values for expenses and reimbursements start to become very volatile. Our analysis is consistent with the general behaviour of medical expense statistics relative to age – trends are particularly visible in the reimbursement graph.

- We observe a significant spike in claims at ages 13-18, corresponding to the teenage years. This phenomenon is largely explained by optical and dental expenses. Indeed, eyesight issues usually manifest during teenage years, triggering the acquisition of glasses and lenses. Moreover, during adolescence people with eyesight problems will transition from child frames to adult frames which are in general more expensive. In addition, dental expenses, particularly dental braces,

are also incurred principally during teenage years. Finally, these effects are compounded by the fact that the Social Security has a limited coverage of optical and dental expenses, hence transferring the bulk of the reimbursement burden to the insurance company.

- From age 40 to 70 there is a steady increase in expenses and reimbursements, which is globally due to the general ageing process and the related expenditures – e.g. appearance of presbyopia.
- Finally, after stabilising during the 70-75 years period, expenses and reimbursements start to steadily decrease. A first explanation is that the main optical and dental expenses were incurred at younger ages so no further costly expenses on these health issues are necessary. Moreover, from 70 years on medical expenses are particularly concentrated on the hospitals, medicaments and city care categories, which are largely financed by the Social Security – respectively 91.1%, 69.1% and 63.9% in 2014.

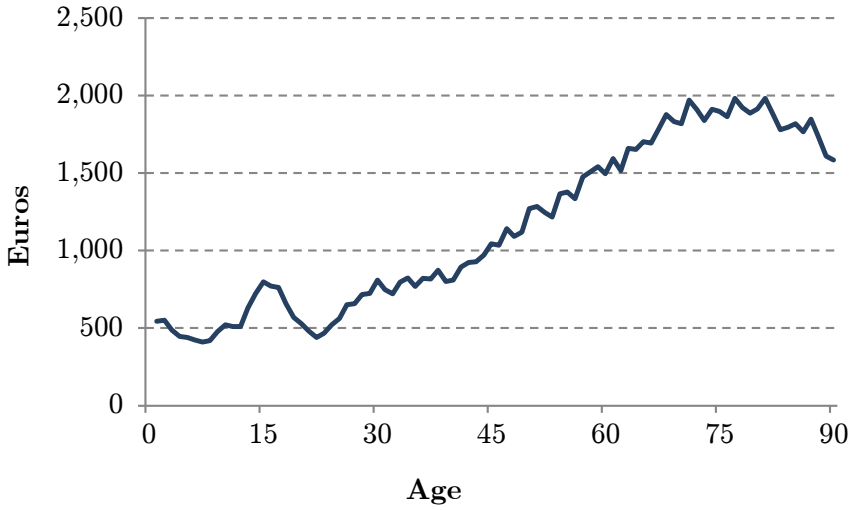

Figure 3.10: *Average medical expense per age, 1-90 years*

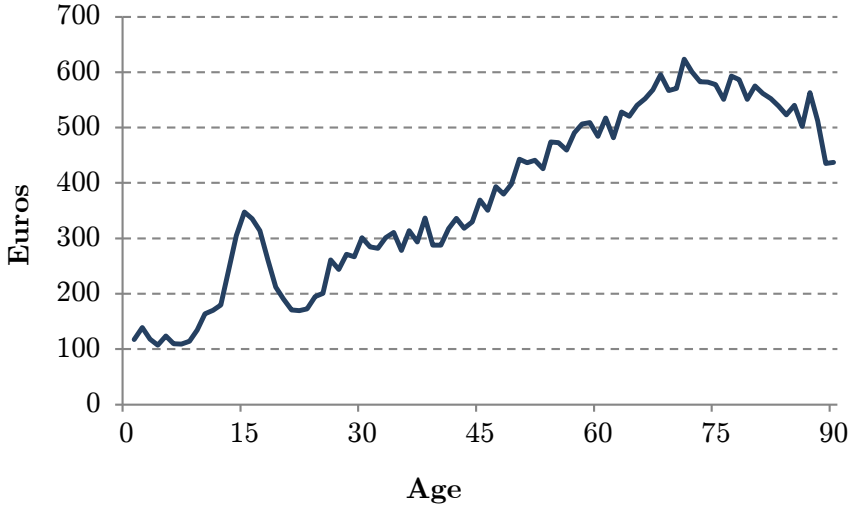

Figure 3.11: *Average expense reimbursement per age, 1-90 years*

We have also computed the ratio between the reimbursed expense and the total expense to understand reimbursement patterns. In this case, the ratio sharply increases during the first years of life, experiencing again an adolescence spike and reaching its maximum at 16 years, 43.6%, from 21.6% during the 1st year of life. However, from this point on, the ratio seems to stabilise at around

37% for the tranche 20-35 years, then starts to steadily decrease in a linear fashion, down to just 27.6% at 90 years. These trends in the reimbursement/expense ratio validate our view that the policies from our portfolio protect particularly well against expenses lightly covered by the Social Security, such as optical and dental treatments, which are concentrated on younger ages – the 13-18 age tranche.

Figure 3.12: *Average ratio reimbursement/expense per age, 1-90 years*

Finally, we have also analysed the frequency of occurrence: as explained previously, occurrence is equal to 1 if the number of medical acts for a policy is equal to or higher than 1, 0 otherwise. In this particular case, values do not become extremely volatile after the 90 years threshold, hence we have decided to include the whole population. Volatility does spike for the last 5 years – 102, 103 (no data), 104, 105 and 106 – however these ages represent a negligible fraction of the population so we have decided to keep them in our graph.

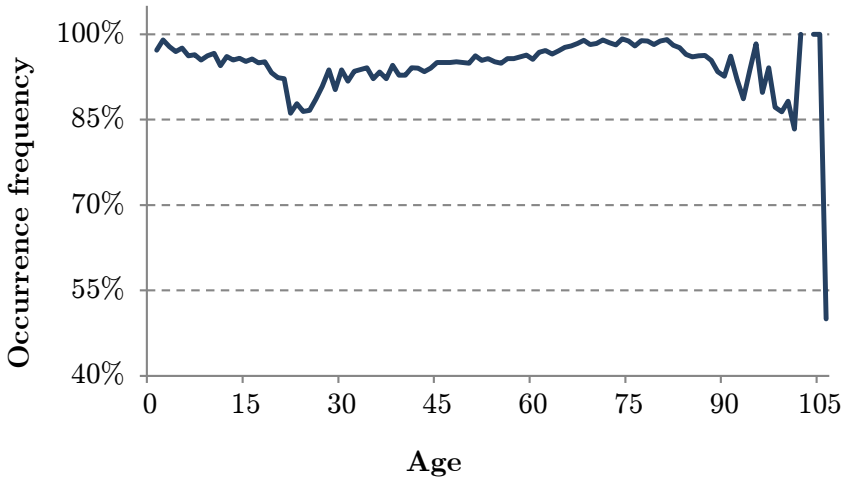

Figure 3.13: *Occurence frequency per age, 1-106 years*

The most prominent characteristic is the trough in the 20-27 age tranche: a possible explanation is at that age, optical and dental expenses might decrease significantly – they are mostly incurred during the 13-18 years old period – and additionally people in their twenties might tend to be fitter thus have a lower need for health consultations and medicines.

3.5.2 The contract option effect

Another analysis that must be undertaken is with respect to the contract option. First, as the coverage and guarantees of a contract improve, we expect the reimbursement from the insurance company to mechanically increase. Secondly, some areas of insurance, among which health insurance, are prone to a *moral hazard* phenomenon: in case the realization of some risk X might be influenced by a person, the latter will alter his behaviour in case he is insured against X. For example, a person with a supplementary medical insurance will tend to spend more on dental-related medical acts than a person that does not benefit from such an insurance. Hence we might expect the reimbursed expense for people subscribing the Maximal option to be higher than those subscribing options Improved and Normal; the same can be said of people with an Improved contract with respect to those with a Normal policy.

	Normal	**Improved**	**Maximal**
Average total expense	1,108.3€	1,217.7€	1,308.5€
Average SMI reimbursement	301.9€	387.6€	461.5€

Table 3.3: *Analysis of reimbursement patterns for each contract option*

Our intuition turns out to be accurate: we effectively observe that a better option – in terms of guarantees and coverage – leads to an increase in average total expense and average SMI reimbursement. More specifically, the average total expense increases 9.9% and 7.5% by switching from Normal to Improved and from Improved to Maximal respectively; in the case of total SMI reimbursement these figures are 28.4% and 19.1% respectively.

Hereafter we show the annual reimbursed amount with respect to the age of the beneficiary for all 3 options.

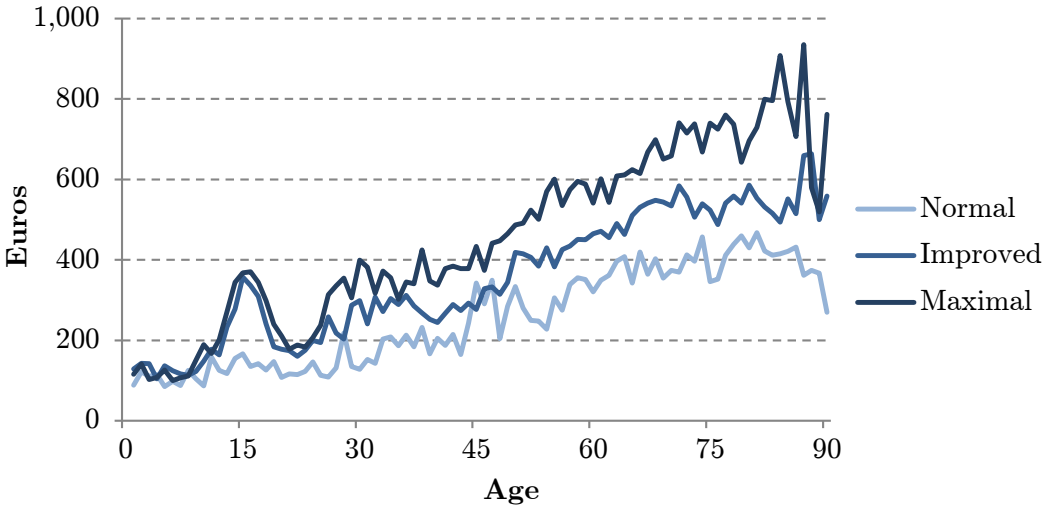

Figure 3.14: *Average expense reimbursement per option and age, 1-90 years*

As expected, the curve for the Maximal option is predominantly above the curve for the Improved version of the policy, which is itself above the Normal option curve. The most striking feature of these curves is the lack of the adolescence spike in the normal curve; if we look at the beneficiaries for these 3 options, there are 15.1% which are children in the normal case, whereas there are 20.9% and 25.7% in the improved and maximal case respectively. Hence policyholders might anticipate medical expenses for their children and therefore subscribe contracts with broader coverage.

3.5.3 The beneficiary and gender effect

Regarding reimbursement statistics with respect to beneficiary and gender, we notice first that the average annual reimbursement is 23.0% higher for women than for men. This might be due for example to pregnancy. On the other hand, the average reimbursement in case the beneficiary is the policyholder or the spouse is relatively similar.

	Male	*Female*	**Total**
Policyholder	419.1€	512.4€	**459.1€**
Spouse	401.7€	502.7€	**483.5€**
Child	189.4€	213.7€	**201.5€**
Total	**363.2€**	**446.9€**	**407.6€**

Table 3.4: *Average annual reimbursement by beneficiary and gender*

Surprisingly, the average reimbursement for child – 201.5€ – seems low compared to the other two, specially considering our previous discussion on adolescence optical and dental expenses. However, if we restrict ourselves to children in the teenage band – 13 to 18 years old included – the average annual reimbursement spikes to 301.8€, almost a 50% increase with respect to the total average. On the other hand, the above averages for policyholders and spouses are independent of age, so they capture the higher expenses incurred from 50 years onward.

3.5.4 The contract effect

We observed previously that the population adhering to the SNCF contract deserved greater attention to assess whether a separate model for them is necessary, due to the specificity of this population and the strong linkage between the SNCF special regime and the SNCF contract. Here we evaluate the impact of the value "SNCF" from attribute "Product" in occurrence and reimbursement statistics. In order to be exhaustive, we also analyse the contract "Isoles" which, as was shown, above represents also a sizeable portion of the database: the latter is roughly equally divided between SNCF contracts, Isoles contracts and the rest of the contracts.

In the following table we show average occurrence and reimbursement data for these 2 contracts as well as for combinations of the remaining contracts:

	Occurrence	*Avg. reimbursement*
SNCF	96.34%	430.66€
Isoles	95.94%	437.05€
Other ex. SNCF	95.19%	396.45€
Other ex. Isoles	95.36%	391.51€
Other ex. SNCF and Isoles	94.37%	351.74€
All contracts	95.57%	407.60€

Table 3.5: *Average annual occurrence and reimbursement by contract*

We observe that both the Isoles and SNCF contracts push upward occurrence and reimbursement statistics: whereas the impact on average occurrence is small – 1.27% relative increase with respect to the average occurrence excluding these 2 contracts – the impact on reimbursement is much more significant as the average increases by 15.88% when including these 2 contracts in the sample.

As a consequence, we believe it might be worthwhile to generate 3 different reimbursement models: one for Isoles contract holders, one for SNCF contract holders and one for the remaining population. If the heterogeneity between these subgroups is sufficient, having 3 different reimbursement models might yield a significant improvement in predictive capacity with respect to a situation in which we only train a model on the whole population.

Given that volatility is much higher for reimbursement data than for occurrence data – measured through the coefficient of variation, *i.e.* standard deviation over mean, it is equal to 21.54% for the latter and 149.14% for the former – and that the difference from excluding SNCF and Isoles contracts is much lower, we do not build separate occurrence models.

We will be undertaking this test and showing the results in the following chapter.

Chapter 4

Applying Support Vector Machines to Health Insurance

Pricing a supplementary medical insurance contract for a portfolio of policyholders

4.1 Introductory comments

In this last chapter, we will apply Support Vector Machine models to price the health insurance contracts that were described in the previous chapter:

- We start off by discussing the computational complexity of training a Support Vector Machine and how we have limited the cost related to this complexity through our IT architecture and capabilities.
- We will then explain the final data preparation methodology, necessary to make the raw data exploitable by SVM algorithms.
- We finally describe the construction and analyze the results of two SVM models, one for claim occurrence and the other for claim severity.

In order to assess the performance of our main models, we will benchmark those against a competing, powerful Machine Learning algorithm: the *Random Forest* (Breiman, 2001). Like SVM, Random Forests (RF) can be used for both classification and regression tasks, and have historically proven to be a very successful technique; the previous study conducted with this same health insurance dataset concluded that Random Forests were the best predictive model amongst a suit of other Machine Learning techniques (Huther, 2014), which has motivated our choice as a benchmark model. A brief theoretical overview of RF can be found on the appendix.

4.2 IT architecture

In the following subsections, we start by describing the computation capabilities we have used for our experiments. Afterwards we give an overview of specific computational issues and costs of the SVM optimisation program. These limitations motivate the setup of a remote computing architecture to overcome them, which we describe afterwards. We then analyse the empirical computational cost of our experiments.

4.2.1 IT general specifications

We give some general technical details about the hardware and the software deployed to carry out this memoir and its experiments.

Most computations have been undertaken on a 64-bit Operating System laptop computer with a 4-core processor Intel® Core™ i7-6500U – 2.50 GHz frequency – and 8.00 GB of RAM memory. Some computations have been parallelized by using the Python module `multiprocessing` which supports parallelization. Additionally, some computations for the Gaussian SVM regression have been executed remotely by using Amazon Web Services (AWS) computing capabilities: CPUs can be rented on an hourly basis to speed up calculations. These computations were done with Intel® Xeon® E5-2666 v3 processors and 60.00GB of RAM; Python `multiprocessing` module was leveraged to distribute calculations across 9 processors.

The programming language used for experiments has been Python 2.7.12 and most sophisticated computations have been undertaken through `scikit-learn` functionalities – version 0.17.1 – which is a free Machine Learning library for Python. We make extensive use of classes `SGDClassifier`, `SGDRegressor`, `RBFSampler`, `RandomForestClassifier` and `RandomForestRegressor`.

For Support Vector Machines, `scikit-learn` implements the `LIBSVM` solver, which is based on a variation of the SMO algorithm. Their Stochastic Gradient Descent algorithm is based on the Stochastic Gradient SVM algorithm developed by L. Bottou (Bottou and Bousquet, 2011).

4.2.2 Computation complexity of the Support Vector Machine

One of the drawbacks of SVMs is their computational complexity: they are very expensive to train in terms of time and memory. The interested reader might consult Bottou and Lin (2007) or Chang and Lin (2013) for further details. To give some flavour on the general complexity of any SVM solving algorithm, we follow a similar argument as the one given by Bottou and Lin to derive a lower bound on the solving cost:

- Assume the highest possible accuracy rate that can be achieved for a data sample S_n of size n mapped into a feature space $\mathcal{H}$, independently of the algorithm, is a^*. We want to train on this data sample a SVM equipped with a kernel implicitly mapping to $\mathcal{H}$. As our sample size n increases, we might expect our model to perform better – the more data it has to learn from, the better it can get at predicting labels. Hence, we can expect the number of misclassified data points to be $(1-a^*)n$, all of which are support vectors. We conclude that asymptotically the number of support vectors scales at least linearly with the sample size.

- Now assume that we want to verify whether a given vector $\boldsymbol{\alpha}$ is a solution to the SVM dual optimization program: the most efficient way is to check the KKT conditions, which are necessary and sufficient for $\boldsymbol{\alpha}$ to be a solution. There are multiple ways to simplify KKT conditions, here we explain our reasoning based on the KKT conditions displayed in appendix B:

$$\begin{aligned} 0 \leq \alpha_i &\leq C \\ 0 \leq \alpha_i < C &\Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1 \geq 0,\ \xi_i = 0 \\ \alpha_i = C &\Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - (1 - \xi_i) \geq 0,\ \xi_i \geq 0 \\ \sum_{1 \leq j \leq n} \alpha_j y_j &= 0 \end{aligned}$$

  These simplified conditions require calculating ξ_i for all $i \in \{1,...,n\}$, which as shown in the appendix is such that: $\xi_i = \max[0, 1 - y_i f(\mathbf{x}_i)]$. Thus in order to verify the optimality conditions it is necessary to compute the decision function for each one of the n data point in the sample; given that the decision function depends only on support vectors, each evaluation of the decision function requires computing N_{SV} dot products for each data point. Hence we need a number of operations proportional to nN_{SV} to verify the optimality of $\boldsymbol{\alpha}$.

Putting these 2 claims together, we conclude that as the sample size grows larger, the computational complexity scales polynomially with the size of the data sample in the order of n^2. Bottou

and Lin derive an additional computational cost in the order of n^3 such that the actual cost depends on whether the error tolerance parameter C is small – square cost – or large – cubic cost. Moreover, according to their paper modern SVM solvers generally verify these "*scaling laws*": among others it is the case of the SMO-based `LIBSVM` algorithm which underpins `scikit-learn`'s SVM library, as specified in their documentation.

Another point to note is that computing the kernel matrix is also computationally costly: we already highlighted this when we introduced kernel approximation techniques as a solution to this drawback. Additionally, for very large datasets we can face kernel matrices so large that they might not fit in the memory.

4.2.3 Remote computing architecture

The computational limitations exposed above led us to consider setting up a remote computing infrastructure to carry out part of the model training: more specifically, we trained the regression Support Vector Machine equipped with a Radial Basis Function kernel remotely because it is the longest model to train – due to the presence of 3 hyper-parameters to cross-validate and the inclusion of an approximated RBF kernel with 3,000 features.

We have illustrated our remote computing set up in Figure 5.1. We will explain briefly in the next paragraphs each element of the illustration so as to give to the reader a basic understanding on configuring and running remote computing architectures.

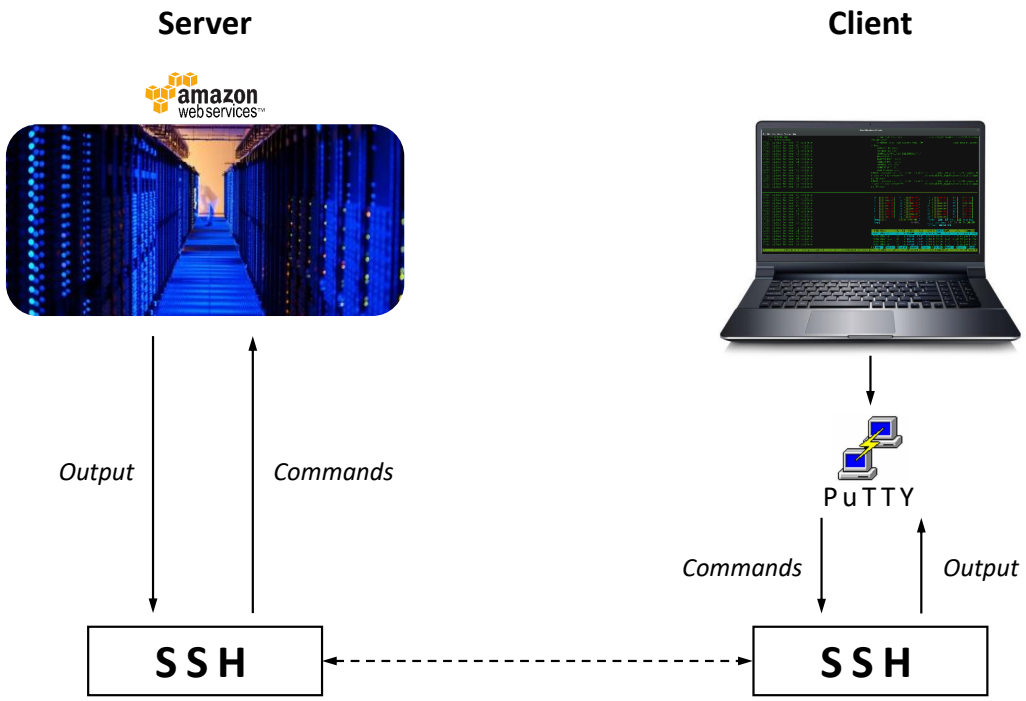

Figure 4.1: *Remote computing architecture*

Setting up a remote computing solution might enable the user to achieve one or more of the following:

- **Gain access to more powerful processors**: computing speed is principally determined by the available processors. Long and complex calculations can be speed up simply by executing them with more powerful processors.

- **Increasing the available Random-Access Memory (RAM)**: RAM is a type of short-term memory necessary to carry out operations such as estimating a Machine Learning model. RAM is particularly important when training an SVM, for example to store the matrix representing the kernel.
- **Parallelizing the training procedure**: by connecting to a remote server we can increase the overall number of processors at our disposal – independently of their performance. With more processors, the training phase of the algorithm can be easily parallelized: because cross-validation loops are independent from one each other, instead of carrying them out sequentially we can distribute them among a suit of processors.

The client

Our remote computing setup corresponds to a *client-server communication model*. In such as setting, the *client* is the entity which sends a request to a server: in our case, we are the client and we will be sending a request to execute a series of Python scripts – these Python scripts contain the code to train the algorithms.

Given that we chose to launch a Linux machine on Amazon servers, the interface with the server was done through Linux command window. Python scripts are executed from the command window by first accessing the folder where they are located – see below on how to transfer files to the remote server. We use the handy `htop` functionality to monitor our processors and memory consumption.

Connecting through the SSH protocol

Secure Shell (SSH) is a secured communication protocol which allows among other things to establish a secure connection with a remote computer, sending commands to the server and receiving the resulting output: in our case, SSH allows us to establish a safe communication channel between our laptop and the remote Amazon server. The advantage of SSH is that it allows to establish a safe connection through an insecure network, the Internet. To achieve this, SSH uses a *private key-public key cryptographic system*. The basic mechanics of this cryptographic solution is the following:

- The client generates both a public key and a private key. The private key must be kept confidential, whereas the public key can be distributed to servers to which we wish to connect.
- When connecting to a server possessing the public key, the server verifies whether the client which is trying to connect has the corresponding private key. If so, the connection is established, and the remote server can be controlled from the client.

In order to set up a secure SSH connection to our Amazon server we used *PuTTY*, a free and open-source SSH client which also supports other communications protocols.

SSH is the basis of the *Secure Copy* (SCP) transfer mechanism: SCP allows to remotely transfer files between a client and a server. In our memoir we used SCP to transfer data and Python scripts to the Amazon server, and import the results back. We used a user friendly interface for Windows called WinSCP.

Amazon Web Services server

For our memoir we decide to use Amazon Web Service (AWS) to set up a remote computing solution. AWS is a subsidiary of Amazon which offers cloud computing services such as compute power, database storage, applications, etc. In our case, we want to have access to greater computing power. The specific AWS service we used is called *Amazon Elastic Compute Cloud* (Amazon EC2): it is an on-demand web service that provides extra computing capacity by renting servers. Within their EC2 service, we selected their "Compute Optimized" instances which have the highest performing processors amongst their servers, carrying out computations in their c4.8xlarge model.

Their "Memory Optimized" solutions, displaying as much as 128 processors and 1,952.00GB of RAM, would have allowed to reduce even further the computational cost of the training phase, but the pricing is significantly higher.

4.2.4 Computational performance analysis

Throughout the training phases of our models, we monitored and kept track of computational issues, mainly computation time and memory requirements. We summarise the technical specifications as well as average cross-validation times for each training procedure in the following table – "CV." stands for cross-validation, "Clf." for classification and "Rgr." for regression:

Algorithm	**Location of computations**	**Parallel**	**Processors**	**RAM**	**CV. average time**	**Total time**	**Real CV. average time**
Linear Clf. SVM	Laptop	No	Intel® Core™ i7-6500U	8.00 GB	1m1s	1h1m12s	1m1s
Gaussian Clf. SVM	Laptop	No	Intel® Core™ i7-6500U	8.00 GB	8m2s	22h37m55s	8m2s
Clf. Random Forest	Laptop	No	Intel® Core™ i7-6500U	8.00 GB	10m14s	10h14m1s	10m14s
Gaussian Rgr. SVM	AWS servers	Yes (9 processors)	Intel® Xeon® E5-2666 v3	60.00 GB	12m43s	$\sim$ 47h45m	$\sim$ 1m25
Rgr. Random Forest	Laptop	No	Intel® Core™ i7-6500U	8.00 GB	7m39	17h50m23s	7m39s

Table 4.1: *Computational performance*

The cross-validation average time corresponds to the pure average time each cross-validation loop lasted. On the other hand, the real average time corresponds to the total computation time divided by the number of loops and processors: in other words, it incorporates the effect of parallelization.

It is important to keep in mind this is not a rigorous performance comparison but rather a preliminary analysis. For example, when comparing the time cost of training between our laptop hardware and the Amazon server we can compare training times for the classification and regression Gaussian SVM. This can give us a rough idea of the advantage of the server over the laptop, however we are mixing 3 effects: we are training 2 different models – although arguably extremely close; we are using different processors in each case; the regression training is parallelized whereas the classification training is not. However, considering that the classification and the regression SVM optimization program are very similar, we observe that calculating in the Amazon server with parallelization yields very significant time gains.

4.3 Data preprocessing

Before proceeding to the application of Machine Learning algorithms to our data, it is crucial to preprocess our data matrices to make them exploitable and prevent issues during computations. Preprocessing procedures applied depend on the type of data, which can be either *categorical*, i.e. qualitative, or *quantitative*.

We now briefly expose our preprocessing methodology.

4.3.1 Categorical attributes binarization

Raw categorical attributes such as the "Beneficiary" or the "French department" must first be numbered to be exploitable by a mathematical model: for example, if the "Beneficiary" variable can take values "Policyholder", "Spouse" or "Child", we represent each beneficiary by digits 0, 1 and 2 respectively.

In the case of the variables used in our model, this procedure is not enough: by coding categorical attributes with sequences of numbers $0, 1, 2, 3, 4, ...$ we are implicitly introducing an *order*: $0 < 1 < 2 < 3 < 4 < ....$ This characteristic of numbered attributes might be desirable for categorical variables which convey notions of "good" and "bad", however this is not the case for our data.

To eliminate any implicit order in our categorical data, we need to *binarize* it: let x_i be some categorical attribute which can take c different values; binarizing x_i consists on replacing x_i with c new binary attributes $x_{i,1}, ..., x_{i,c}$ such that:

$$\forall j \in \{1, ..., c\} \ , \ x_{i,j} = \mathbb{1}_{\{x_i = j\}}$$

Attribute $x_{i,j}$ is equal to 1 if the original attribute x_i takes value j, 0 otherwise.

The number of new variables is equal to the total number of categories from the old variables. In our case, we transform 9 categorical variables into 147 binary variables.

4.3.2 Quantitative attributes scaling

It has been widely documented that training a SVM with quantitative attributes which value ranges greatly differ can have harmful consequences on the accuracy of the model – see for example Ben-Hur and Weston (2009) or Hsu, Chang and Lin (2003) for further comments:

- An attribute x_i with values between 30,000 and 50,000 might dominate in the trained decision function over an attribute x_j which takes values in $[0, 5]$.
- Attributes which take very large values can provoke numerical difficulties during computations – recall that the SVM optimization program involves many dot products.

Best practice is to apply some type of transformation to the data to avoid these issues. A common choice is *standardization* which consist on subtracting to each quantitative attribute its mean and dividing the result by the standard deviation. In this memoir we have decided to apply a *scaling* on the data, namely bringing all quantitative attributes to a common range $[r_1, r_2]$. Letting S_{x_i} be the set of all possible values taken by attribute x_i on the data sample – *including both the training and the test set* – we have chosen $r_1 = 0$ and $r_2 = 1$, hence:

$$\forall i \in \{1, ..., p\} \ , \ x_i^{\text{scaled}} = \frac{x_i - \min S_{x_i}}{\max S_{x_i} - \min S_{x_i}}$$

The choice of the interval $[0, 1]$ is motivated by the binarization of our categorical variables: the combination of these two transformations yields a dataset where all numbers are between 0 and 1.

We stress again that scaling must be applied consistently across the train set and the test set: either scaling is applied upstream to the whole data set, or else the scaling factors from the training data must be saved and applied posteriorly to the test set. Failure to proceed so will result in inaccurate results.

4.4 General procedure for model specification and estimation

In this section we describe the models we have tested as well as the general procedure for estimating their parameters: our discussion is applicable to both the classification problem – *i.e.* predicting occurrence – as well as the regression problem – *i.e.* estimating average cost given occurrence.

4.4.1 Model specification

We will be working on 3 different algorithms, which we will label for concision – a subscript "C" indicates the model has been used for classification, while "R" indicates regression:

- L-SVM-SGD: Linear Support Vector Machine, trained with the Stochastic Gradient Descent algorithm;
- G-SVM-SGD: Support Vector Machine equipped with an approximated Gaussian kernel, trained with the Stochastic Gradient Descent algorithm;
- RF: Random Forest;
- Ensemble Linear Regressor: for the regression task – predicting cost – we have tested an hybrid model which combines linearly the predictions of an optimal SVM and Random Forest model.

We have not trained a linear SVM on the regression task because it is clearly a non-linear problem: it suffices to consider the relationship between age and reimbursement – which we analysed in the previous chapter – which displays specific, non-linear behaviour at different age ranges.

The reader might notice that we have not even considered a plain SVM algorithm, trained with SMO-based techniques: our initial experiments with this model – classes `LinearSVC` and `SVC` from `scikit-learn` – showed that training time was disproportionately high, particularly for a machine equipped with a Gaussian kernel (`SVC`). As the additional computation time was not compensated by a meaningful accuracy gain, in our final experiments we omitted SMO-based machines.

For the classification task, we will also benchmark these models to a dummy majority classifier, which simply allocates all observations to the largest class, which in this case is class 1 – at least one medical act. The expected accuracy of this dummy classifier is 95.55% on the training set – see next subsection.

To measure the quality of the classification models, we might use the following metrics:

- **Accuracy rate**: the accuracy rate is simply equal to the proportion of samples that have been correctly labelled by a model M with decision function f:

$$a_M = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{y_i = f(\mathbf{x}_i)}$$

  The accuracy – or similarly the error rate, which is simply equal to 1 minus a_M – is the main metric in assessing the performance of a classifier and benchmarking it against other, competing models (Yom-Tov, 2003).

- **Cohen's Kappa** (Cohen, 1960): we will simply use Cohen's Kappa as an indicator of over-performance over the dummy classifier. Letting a_M be the accuracy of a model M and p_1 the proportion of samples labelled 1, we have:

$$\kappa_M = \frac{a_M - p_1}{1 - p_1}$$

  Negative values of κ indicate the model M is doing a worse job than the dummy classifier; any

value above 0 indicates the model is performing better, with a maximum at 1 – equivalent to a perfect model.

- **Youden's J Statistic** (Youden, 1950): letting a_M^l be the accuracy of a model M on label l, $l \in \{0, 1\}$, we have:

$$J_M = a_M^0 + a_M^1 - 1$$

The J Statistic can range from -1 to 1. 1 indicates a perfectly discriminating model whereas a value 0 indicates that the model is not learning any pattern and does not discriminate.

For the regression task, model quality has been measured with the classical metrics:

- **Mean Squared Error** (MSE): the MSE of a model M is simply equal to the mean of the squared errors between the observed values and the model predictions:

$$MSE_M = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$$

- **Coefficient of determination** R^2: the R^2 coefficient is expressed as a function of the MSE and can be interpreted as the fraction of the data variability that is explained by the model M. Letting σ_n^2 be the variance of the sample y_1, ..., y_n, we have:

$$R_M^2 = \frac{\sigma_n^2 - nMSE_M}{\sigma_n^2} = 1 - \frac{nMSE_M}{\sigma_n^2}$$

4.4.2 Model estimation

As explained in the data description chapter, we will be working with a randomly chosen training set of size equal to 9/10 of the total dataset size – 138,300 training observations – the remaining samples forming the validation or test set – 15,373 test observations. In this training set, the fraction of policies from class 0 is equal to 4.45%, hence class 1 represents 95.55% of policies, the ratio between the two classes being 21.49. In the validation set, the proportion is 4.30% and 95.70% respectively, the ratio being 22.26.

For all our models, we include as attributes the whole set of internal and external variables that we described in the previous chapter. Supervised learning algorithms tend to robustly under-weight attributes and features which are not significant for explaining the targeted label. This is particularly true for the Random Forest algorithm, which can be tuned so as to evaluate the relative importance of each attribute: in the specific case of library `scikit-learn`, let $g_C(l_{T_m}, f^{\mathbf{x}})$ be the error gain achieved according to the Random Forest loss criterion C on the node l_{T_m} of tree T_m which depends on attribute or feature $f^{\mathbf{x}}$, $n(l_{T_m})$ the number of samples routed through that node and F_p the set of attributes or features, then the algorithm computes the importance I of $f^{\mathbf{x}}$ as:

$$I_{f^{\mathbf{x}}} = \sum_{f^{\mathbf{x}} \in F_p} n(l_{T_m}) g_C(l_{T_m}, f^{\mathbf{x}})$$

We will be using this capability to have an idea of the importance of each original attribute in our models.

SVM solving algorithms compute optimal solutions, however the user still has to select the value for the model's hyper-parameters $(H_{\text{Param}_1}, H_{\text{Param}_2}, ..., H_{\text{Param}_m})$ – such as for example C or other kernel-specific parameters: each possible vector of values $(h_1, h_2, ..., h_m)$ for the set of hyper-parameters $(H_{\text{Param}_1}, H_{\text{Param}_2}, ..., H_{\text{Param}_m})$ defines a unique machine. Hence determining the best machine consists on testing numerous values for the hyper-parameters that define it and picking the

one with the best performance on the data training set.

To achieve this goal, we implement a standard *grid search strategy*: assuming our machine is hyper-parametrized by a couple $(H_{\text{Param}_1}, H_{\text{Param}_2})$, we exhaustively sweep through different possible values for the pair of hyper-parameters $(h_1^{(1)}, h_2^{(1)})$, $(h_1^{(2)}, h_2^{(2)})$,... each time optimizing the machine with respect to internal parameters $(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. If we select m_1 possible values for H_{Param_1} and m_2 possible values for H_{Param_2}, we train in total $m_1 \times m_2$ machines. The main disadvantage of this technique is its computational cost. Alternative strategies might involve the random selection of values to be tested, whereas some Machine Learning models can be equipped with bespoke algorithms to select the best values for hyper-parameters.

Each machine is trained through a *ten-fold cross-validation procedure*: the training dataset is split into 10 equally sized stratified subsets, such that the proportion of class 0 and class 1 examples is kept relatively unchanged between subsets; for a given value(s) for the model's hyper-parameter(s), the machine is trained 10 times, each time on 9 subsets making up 124,470 observations, then tested accordingly 10 times on the remaining 13,830 examples subset – which we will call cross-validation test set – hence obtaining for each possible value of the hyper-parameters 10 training errors and 10 cross-validation test errors. We look at the average of these 10 training and cross-validation test errors when choosing an appropriate model.

4.5 A Support Vector Machine for predicting occurrence

The first phase in developing an insurance pricing model is to construct a function which enables us to predict the frequency of medical acts. In this section we will explain our modelling choices and analyze the results obtained. As explained above, results will be assessed against a dummy classifier and a Random Forest.

4.5.1 Problem setting

As we have explained in chapter 3, the moderate quality of our data has convinced us to chose an occurrence-severity model: we do not know from which insurance company each policy comes from; hence, given that each insurer has its own methodology for recording medical acts, two policies with the same number of medical acts might not be comparable, hence the consistency of the number of medical acts across the database is not verifiable. Consider for example the reimbursement of new glasses: an insurer A might record this as a single medical act, whereas a second insurer B might record two medical acts, one for the frame and one for the proper glass. Another example relates to medicaments bought in a pharmacy: the medical act might be recorded according to either the prescription or the medicament bought; hence, whereas for a given prescription A might record a single act, B might record as many acts as the number of medicaments bought.

Therefore the choice of a occurrence-severity model seems sensible: given the realization of an event X covered by the SMI contract, although each insurer might have different methodologies for recording the number of acts triggered by X, we can confidently assume that all insurance company will at least record one medical act. Under this setting, the frequency modelling task becomes a binary classification problem: we can implement a standard two-classes Support Vector Machine to predict occurrence. We will naturally designate by class 0 the policies for which no medical acts were declared; equivalently, class 1 refers to policies for which at least one medical act was declared during the year.

4.5.2 Analysis of results

In this subsection we discuss the results obtained with all 3 models. Globally speaking, results are not particularly encouraging as all 3 models seem to have difficulties learning an occurrence pattern from the data available.

Model L-SVM-SGD$_C$

The first model we estimated is a linear SVM trained with an SGD algorithm. The only hyper-parameter we optimised is the error tolerance parameter C; under `scikit-learn` specification, the model is formulated in its regularization form, hence the hyper-parameter to optimise is proportional to the inverse of C, and we will call it beta β – not to be confounded with the Lagrange multipliers $\boldsymbol{\beta} = (\beta_1, ..., \beta_n)$ from the SVM Lagrangian. In our training procedure, we conducted a grid-search over an exponentially growing sequence of betas $\{2^{-100}, 2^{-98}, \cdots, 2^{18}, 2^{20}\}$ for a total of 61 possible values. As explained above, for each possible value for beta we performed a ten-fold cross-validation, keeping record of accuracy metrics on cross-validation training and test sets; importantly we kept track of conditional test accuracy on both labels 0 and 1. The average computation time for each value of beta was 1 minute and 1 second, which remains reasonable.

Another free parameter of this model is the number of iterations of the Stochastic Gradient Descent algorithm: `scikit-learn`'s implementation of SGD scans the whole dataset n_{iter} times; at each loop, it optimises the model with respect to each data point, one by one, in a random order, hence the total number of operations is $n_{\text{iter}} \times n$, and the user can control parameter n_{iter}. However, as this parameter essentially allows to trade accuracy and computation time, we have not cross-validated it and simply set it equal to 30 – `scikit-learn` recommendation is to set it at least to 10^6 divided by the number of training data points, which in our case would have yielded a number around 9 or 10. We undertook some initial experiments on this parameter, exploring the range between 5 and 30, and we did not identified any meaningful computation time gains or losses by changing its value.

Our results can be visualised in the below figure. The main takeaway is that the model did not achieve any meaningful learning: both the best training accuracy score as well as the best test score were equal to the proportion of label 1 samples, meaning that in those cases the model had degenerated into a majority classifier, like our dummy benchmark. In Figure 5.1 we can observe that in the cases where the test accuracy on label 0 improved, this was at the expense of the accuracy on label 1 and hence on overall accuracy. The figure clearly shows that overall accuracy trails accuracy on label 1, which is expectable given the greater proportion of records for which there was at least one medical act declared. On the other hand, the SVM does not seem to have been victim of over-fitting, given that the maximum train accuracy achieved is equal to the proportion of samples labelled 1.

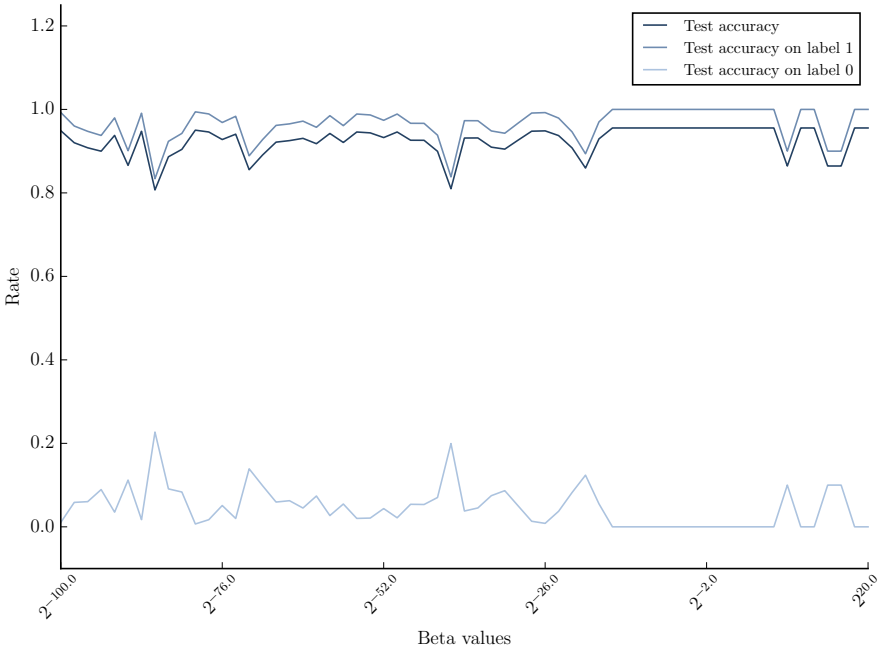

Figure 4.2: L-SVM-SGD$_C$, *test accuracy rates on cross-validation test sets*

The linear SVM does not seem to accurately capture, if any, the underlying pattern governing occurrence of medical acts. The next model we turn to is the SVM equipped with an (approximated) Gaussian kernel.

Model G-SVM-SGD$_C$

In order to train the Gaussian SVM, we executed a 2-dimensional grid-search over an exponentially growing sequence of betas $\{2^{-100}, 2^{-90}, \cdots, 2^{10}, 2^{20}\}$ and sigmas $\{2^{-100}, 2^{-90}, \cdots, 2^{10}, 2^{20}\}$ for a total of 169 possible values. Again, a ten-fold cross-validation was undertook to test every possible pair (β, σ). In addition, instead of working directly on the Gaussian kernel, we approximated the implicit mapping it defines through the Random Kitchen Sinks methodology which, as explained in chapter 2, estimates the mapping through a Monte-Carlo-based technique. We chose the number of components q – *i.e.* the number of Gaussian-generated features – to be equal to 3,000: the greater this number, the better we approximate the true mapping; however a bigger number also increases significantly the computation time.

The average computation time for each pair (β, σ) was 8 minutes and 2 seconds, which remains reasonable, although with some variability as the computation time was increasing in the value of sigma. Due to the additional computational burden of the Gaussian SVM with respect to the linear one, the number of iterations of the SGD algorithm was decreased from 30 to 20.

Results can be seen in Figures 5.2 and 5.3. Conclusions are similar to the ones from the linear model: the machine does not manage to generate an improvement over the dummy classifier, and no clear pattern on the effect of hyper-parameters values on accuracy emerges: the behaviour in Figures 5.2 and 5.3 seems random, apart from the range 2^{-10} to 2^{10} for beta – the error tolerance hyper-parameter – where the SVM degenerates into a majority classifier. Again, better accuracy on label 0 is achieved at the expense of overall accuracy. It is also difficult to distinguish a clear pattern between accuracy and hyper-parameter values, apart from values of beta equal to 2^{-10}, 1 and 2^{10}.

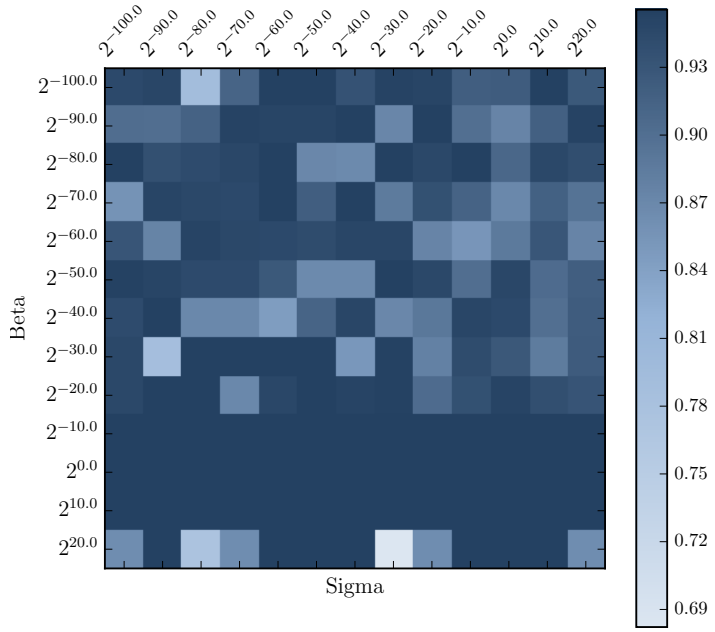

Figure 4.3: G-SVM-SGD$_C$, *total test accuracy rate on cross-validation test sets*

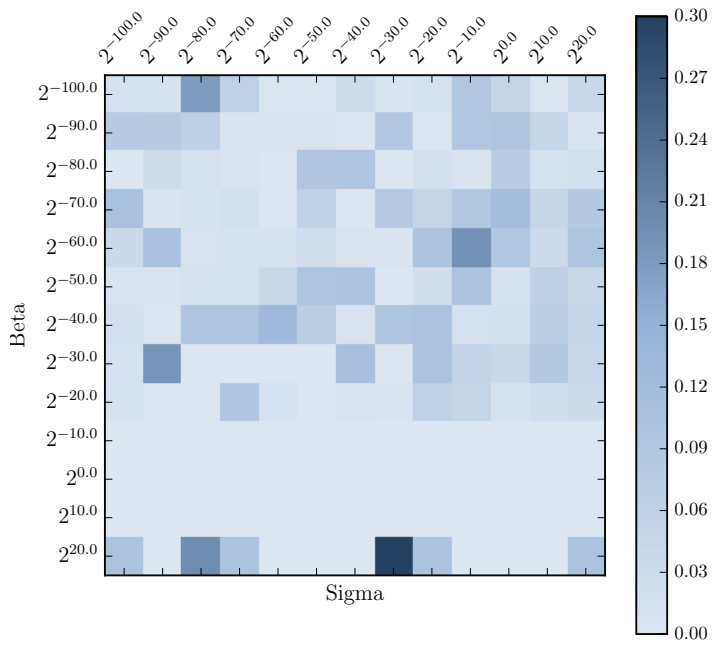

Figure 4.4: G-SVM-SGD$_C$, *test accuracy rate on label 0 on cross-validation test sets*

Finally, as the beta parameter reached a very high value – 2^{20} – the classifier started to show very erratic behaviour: on the bottom line of both heat maps, we observe that the accuracy rate fluctuates significantly, after being constant for the previous 3 values of beta – 2^{-10}, 1 and 2^{10}. The reason is that each one of the bottom 13 accuracy rates is an average value across 10-fold cross-validations and each cross-validation resulted in a majority classifier, *but* not always on the same label: for example, for the brighter square, corresponding to $\beta = 2^{20}$ and $\sigma = 2^{-30}$, 7 cross-validations resulted in a majority classifier to label 1, and 3 in a majority classifier to label 0. Unfortunately, the SVM, as other Machine Learning algorithms, is rather opaque and it is difficult to understand this behaviour.

Model RF$_C$

In order to benchmark our SVM models, we have also trained a Random Forest model on the training dataset. In a classification Random Forest, there are 4 main parameters to estimate – we have specified their names under `scikit-learn` terminology:

- The number of estimators N_T – `n_estimators`;
- The number of randomly chosen features N_F to consider when determining the optimal split – `max_features`. The lower the value of this parameter, the lower the correlation between the trees in the forest – since each split is done by randomly selecting N_F features;
- The maximum depth of each tree $D_{\max}$ – `max_depth`. Setting a restriction on the depth of trees can increase the bias of the forest;
- The minimum number of samples $S_{\min}$ which we require in order to split an internal node of a tree – `min_samples_split`.

We have not cross-validated neither N_T nor $D_{\max}$. Indeed, the number of trees behaves in the same way as parameter n_{iter} in the SGD algorithm: increasing it improves accuracy while augmenting the computation time. Huther (2014) finds that $N_T = 25$ yields already very good results, here we have decided to improve this parameter by increasing it to 100. As for $D_{\max}$, we do not have placed any restriction on the depth of each tree, hence we have set the parameter's value to `None`: in this case, each tree grows until leaves are pure or contain less than $S_{\min}$ samples.

As for the other two parameters, we have made N_F vary from 10% of the total number of features (16) to 100% (159), including 20% (32), 30% (48), ..., 90% (143) of features. For $S_{\min}$, we trained the Random Forest on values in $\{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 50, 75, 100\}$ for this parameter. The average computation time for each couple of possible values for N_F and $S_{\min}$ was 10 minutes and 14 seconds, although there was a great variation depending on the value of N_F: while for $N_F = 16$ the average computation time was 3 minutes and 34 seconds, for $N_F = 159$ – *i.e.* all features considered – the computation time was on average 17 minutes and 2 seconds.

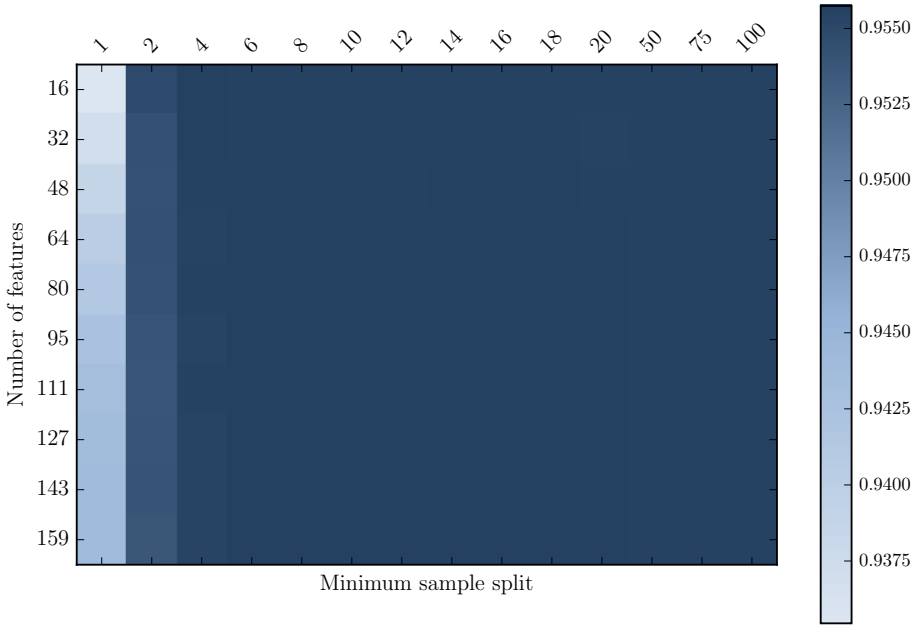

Figure 4.5: RF_C, *total test accuracy rate on cross-validation test sets*

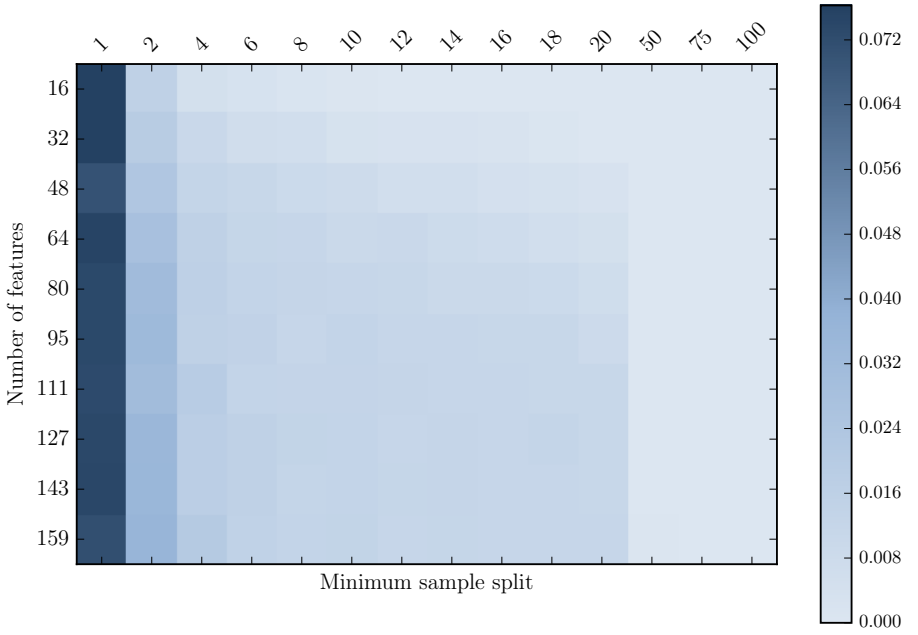

Figure 4.6: RF_C, *test accuracy rate on label 0 on cross-validation test sets*

Our results can be visualised in the above figures. The first thing to note is the same accuracy trade-off phenomenon identified before: an increased accuracy on label 0 is achieved at the expense

of global accuracy; given the unbalanced nature of label proportions, all algorithms either degenerate into a majority classifier or have their accuracy rate tightly fluctuating around the proportion of label 1 samples. The structure of the data makes this choice, which corresponds to the dummy classifier, a decent choice as we are sure of obtaining a 95.55% accuracy. On the other hand, compared to the SVM models the Random Forest has managed to obtain a test accuracy higher than the proportion of label 1 samples for some set of parameters, although the improvement is virtually negligible: the best test score, corresponding to $N_F = 159$ and $S_{\min} = 10$, obtained is 95.57%, yielding a Cohen's Kappa of 0.47% and a J statistic of 1.32%. We also note that for some set of parameters, the Random Forest seems to have over-fitted the train data: this happens all the more so that the value of $S_{\min}$ is low; the best train score was obtained for $N_F = 159$ – all features – and $S_{\min} = 1$, yielding an train accuracy rate of 98.52% but a test accuracy of just 94.40% – 98.46% on label 1 and 7.12% on label 0.

Both heat maps seem to suggest that the minimum sample split dominates over the number of features.

Feature importance analysis with the Random Forest algorithm

We remind the reader the Random Forest algorithm has a built-in mechanism to automatically evaluate the importance of each feature in the model. The importance of a given feature $f^{\mathbf{x}}$ is calculated by summing the products of 1) error gains achieved at each node which depends on feature $f^{\mathbf{x}}$ for each tree of the forest and 2) the number of samples routed through that node. We have represented the top 15 features in the below graph, where the blue bars represent plus or minus one inter-tree standard deviation:

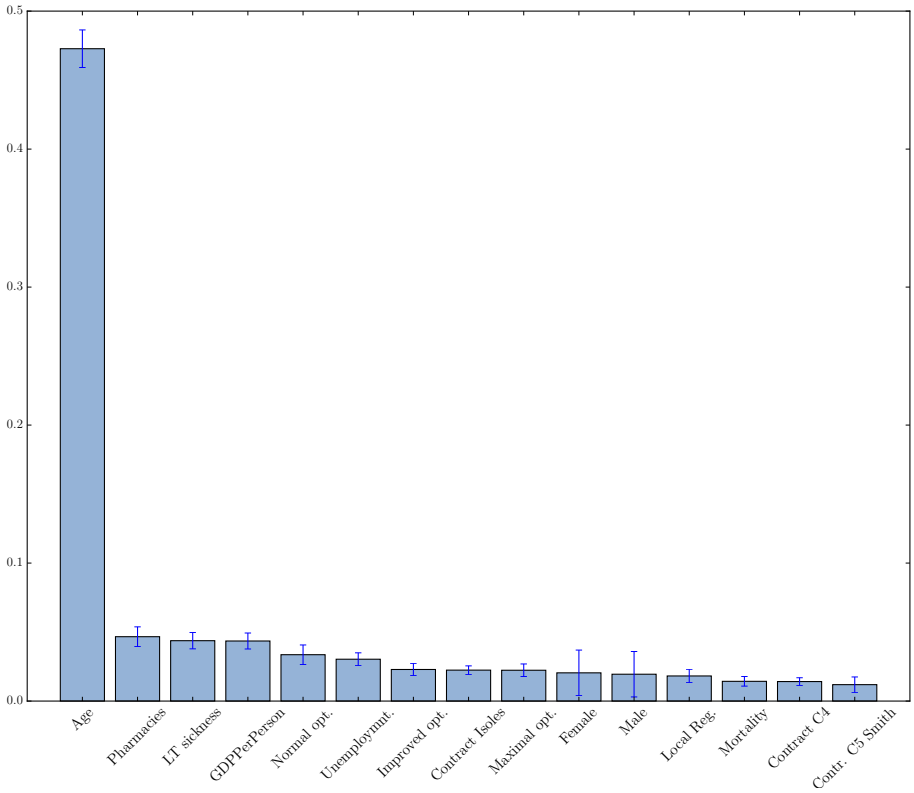

Figure 4.7: RF_{R}, *attribute importances for classification*

Age is the main predictive attribute, with a relative importance which is almost 10 times higher than the one of the next top attribute. It is interesting to see that, albeit not as important as age, the 2nd, 3rd and 4th more explanatory attributes are external: the number of pharmacies might be predictive as the more the pharmacies, the more beneficiaries might have the time or the opportunity to buy medicaments; it is more surprising to see long term sicknesses in third position, as in theory

medical expenses for people afflicted by them are 100% covered by the Social Security, however it might be the case that these sicknesses generate additional but not bijectively related issues which are thus not covered by the Social Security.

Summary

We display in the next tables the performance of the best SVM classifiers, benchmarked against both the best Random Forest and the dummy classifier – "best" is to be understood with respect to the accuracy rate achieved. As we have explained, none of the models has a significant performance – p_1 corresponds to the empirical proportion of label 1 in the relevant data set.

	L-SVM-SGD$_C$	**G-SVM-SGD$_C$**	**RF$_C$**	**Dummy$_C$**
Train sets	p_1	p_1	95.61%	p_1
Test sets	p_1	p_1	95.57%	p_1
Test sets (1)	100%	100%	99.96%	100%
Test sets (0)	0%	0%	1.37%	0%

Table 4.2: *Accuracy statistics for optimal models – based on test accuracy*

	L-SVM-SGD$_C$	**G-SVM-SGD$_C$**	**RF$_C$**	**Dummy$_C$**
Kappa	0%	0%	0.47%	0%
J Stat	0%	0%	1.32%	0%

Table 4.3: *Quality metrics for optimal models – based on test accuracy*

In addition to these poor results, the SVM seemed to display an extremely negative feature, which is at odds with its theoretical properties: instability. To test for it, we conducted a performance stability analysis by randomly selecting for each SVM model a set of hyper-parameters for which we had not obtained a degenerate majority classifier: $\beta = 2^{-50}$ for the linear SVM and $\beta = \sigma = 2^{-20}$ for the Gaussian SVM; whereas for the Random Forest we picked the best set of hyper-parameters with respect to test accuracy: $N_F = 159$ and $S_{\min} = 10$. Additionally, for the linear SVM we increased the number of iterations of the SGD algorithm from 30 to 500, to test whether this was having an effect on the instability. Then, we estimated these 3 models 20 times each – the training was done with the same training data through a 10-fold cross-validation, exactly as explained above. Once the computations were finished, we recorded the standard deviation of the 20 train and test accuracy rates obtained, as well as the minimum and maximum accuracy on the train test. Results are displayed below.

	L-SVM-SGD$_C$	**G-SVM-SGD$_C$**	**RF$_C$**
St. dev. train sets	3.648%	2.902%	0.001%
St. dev. test sets	3.669%	6.385%	0.003%
Min. train sets	80.289%	85.856%	95.607%
Max. train sets	95.389%	p_1	95.610%

Table 4.4: *Standard deviation, minimum and maximum of accuracy scores for selected models*

The fitted SVM is highly variable as demonstrated by the standard deviation of the accuracy rate achieved on both the train set and the test set. Additionally, looking at the minimum and maximum accuracy rate achieved on the train set is even more surprising: in the linear case, these go from 80.3%

up to 95.4%; as for the Gaussian SVM, the maximum accuracy corresponds to a degenerate model. Stability performance of the SVM is all the more disappointing if we compare it to our Random Forest classifier, which proves to be incredibly stable despite the fact that they are trained with only 100 trees.

4.6 A Support Vector Machine for predicting cost

The second phase in developing our pricing model is to construct a function which enables us to estimate the expected annual cost given occurrence. We will now explain the procedure followed and analyse results.

4.6.1 Problem setting

Compared to occurrence data, we cannot include the whole population in our estimation procedure. As explained in chapter 3, cost data starts to become highly volatile from 91 years onwards as the number of contracts per age steadily declines. Hence we have decided to exclude data from beneficiaries whose age is higher than 90 years. On the other hand, we do not have the reliability problem we had with occurrence data and we can work with it more confidently.

For this regression task we have not considered the linear model – L-SVM-SGD – as the data clearly displays non-linear behaviour – consider for example the average cost curve against age – which a linear SVM will not capture.

4.6.2 Analysis of results

Here we discuss the results obtained with a Gaussian SVM for regression and a Random Forest. Although the results for the Random Forest are encouraging, the Gaussian SVM generates a regression function which is probably not adapted to a health insurance pricing problem; we discuss in detail these findings in the following paragraphs.

Model G-SVM-SGD$_{\mathbf{R}}$

Our Gaussian SVM for regression has been trained by executing a 3-dimensional grid-search over parameters β – the error-tolerance –, σ – the Gaussian bandwidth – and ϵ – half the width of the regression line's margin. We have chosen the same search values as before for parameters β and σ, whereas for the ϵ hyper-parameter we have selected an exponentially growing sequence in base 10, with powers ranging from -5 to 5, to which we have added the value 0: $\{0, 10^{-5}, 10^{-4}, 10^{-3}, \cdots, 10^{5}\}$, yielding a total of 2,028 triplets $(\beta, \sigma, \epsilon)$ to test. The rest of the methodology is equivalent to the one applied to the classification task: the Gaussian kernel is approximated and the optimization program is numerically solved with SGD.

As explained at the beginning of this chapter, we executed these computations on a remote machine through Amazon Web Services to speed them up. Average computation time was 12 minutes and 43 seconds, which is longer than the average time we obtained for this same model on the classification task (8 minutes and 2 seconds) for which the training was undertook on the laptop; however because we parallelized calculations on 9 processors the process duration was actually around 9 times quicker that it would have been on our laptop.

Unfortunately results were disappointing: the maximum R^2 test score achieved was -7.11%, meaning that all scores were negative. In addition, low values of σ resulted in an extremely erratic regression function, yielding disproportionately low values of R^2. If we plotted the model's average reimbursement per age against the observed average reimbursement per age, we would observe an almost flat curve situated around a level of 200€, showing none of the stylized facts that are normally observed on this curves – teenage spike, increasing trend throughout life, etc.

To understand this result, we undertake a heuristic mathematical digression. We situate ourselves in the same setting in which we presented Statistical Learning Theory, where (X, Y) are jointly distributed according to a probability distribution P. We define the *ϵ-insensitive loss* as:

$$L\left(Y, f_n(X)\right) = \max(|Y - f_n(X)| - \epsilon, 0)$$

Assuming for clarity that Y takes its values in $\mathbb{R}$, the risk function for the ϵ-insensitive loss is given by:

$$\begin{aligned} R(f_n) &= \mathbb{E}^{\mathbb{P}}\left[\max(|Y - f_n(X)| - \epsilon, 0)\right] \\ &= \int_{Y - f_n(X) \leq -\epsilon} (f_n(x) - y - \epsilon) d\mathbb{P}(x, y) + \int_{Y - f_n(X) \geq \epsilon} (y - f_n(x) - \epsilon) d\mathbb{P}(x, y) \end{aligned}$$

Differentiating the risk with respect to f_n, we obtain:

$$\frac{\partial R}{\partial f_n}(f_n) = \mathbb{P}(Y - f_n(X) \leq -\epsilon) + \mathbb{P}(Y - f_n(X) \leq \epsilon) - 1$$

Now, if we choose $\epsilon = 0$ and set the derivative equal to 0 to determine the optimum, we get:

$$\mathbb{P}(Y \leq f_n(X)) = \frac{1}{2}$$

The optimal decision function corresponds to the *median* in this case. Hence we observe that the Regression SVM has a loss function that results in a *quantile* decision function: in general, the optimal machine f_n is found by "solving" the following equation:

$$P(Y - f_n(X) \leq -\epsilon) + P(Y - f_n(X) \leq \epsilon) = 1$$

It is important to highlight that we did obtain more interesting results before launching the cross-validation procedure: previous to any rigorous calculation – and this applies to all models tested, for both classification and regression tasks – we obviously undertook an informal training procedure with small subsets of data – around 10,000 samples – without any cross-validation to avoid lengthy computation times. The main objective of this informal preliminary phase is to become accustomed to the model, assess its sensitivity to changes in hyper-parameters values and try to identify ranges of values for β, σ, etc. for which we obtain good results. However, because these preliminary experiments are done with significantly smaller samples and without cross-validation, results obtained posteriorly through a full cross-validation training procedure can be very different – as seen here. Due to this divergence, it could be argued that our cross-validation procedure was not complete and that we should have expanded the range of triplets $(\beta, \sigma, \epsilon)$ to test for in order to detect values for which we can obtain results similar to those obtained in the preliminary phase. However we think this is a wrong approach in this situation:

- First, as explained those results were obtained with significantly smaller data sets, without cross-validation and with a lot of manual tweaking: they might have suffered from over-fitting. It would then be impossible to reproduce them through fully rigorous cross-validation on a full dataset.
- Secondly and most importantly, assuming these results could have been replicated through a rigorous procedure, our memoir remains an academic *as well as practical exercise*: as opposed to purely Machine Learning academic research which aim tends to be to explore the full potential of a model, we have to stress the practical applicability of algorithms. Therefore, we believe that the computational complexity of the regression SVM cannot be justified with regards to results obtained first informally and then rigorously, especially when compared to the Random Forest algorithm which managed to achieve very good results with minimal tuning and in a quick way, as we will see just below.

Model $\mathbf{RF_R}$

The training procedure of a regression Random Forest has been exactly the same as the one done for a classification Random Forest – we have used class `RandomForestRegressor()` instead of `RandomForestClassifier`. We have also tested the same set of values for hyper-parameters N_F and $S_{\min}$. The average computation time has been 7 minutes and 39 seconds, which remains quite low. As for the classification task, we have displayed a heat map to visualize the effect of the model hyper-parameters on its predictive power.

We first note that, as for the occurrence model, the parameter $S_{\min}$ dominates over N_F. However, a detailed analysis on the top 10 test scores achieved shows that the highest R^2 corresponds to relatively low values of N_F – 32, 48, 64 – whereas there does not seem to be a clear pattern for $S_{\min}$. The highest R^2 score is achieved for $N_F = 48$ and $S_{\min} = 50$, corresponding to 11.18%; additionally, the fitted models do not seem to suffer from significant over-fitting, as the average test score on the top 10 Random Forests is 11.11%, compared to a train score of 14.62% on those same models – whereas for some models for which the train R^2 was almost 70% the same score of the test set was normally below 0%.

This last heat map validates a phenomenon we have already observed in the occurrence task: whereas SVM heat maps and graph showed fluctuating behaviour – practically random – the impact of changes to hyper-parameter values on model performance seems to follow a pattern for Random Forest. For example, here we observe that increasing $S_{\min}$ increases the score of the model, whereas increasing N_F also tends to improve the model performance. On the other hand, a similar assessment for the SVM is not possible as the impact of each hyper-parameter does not seem to follow a clear trend.

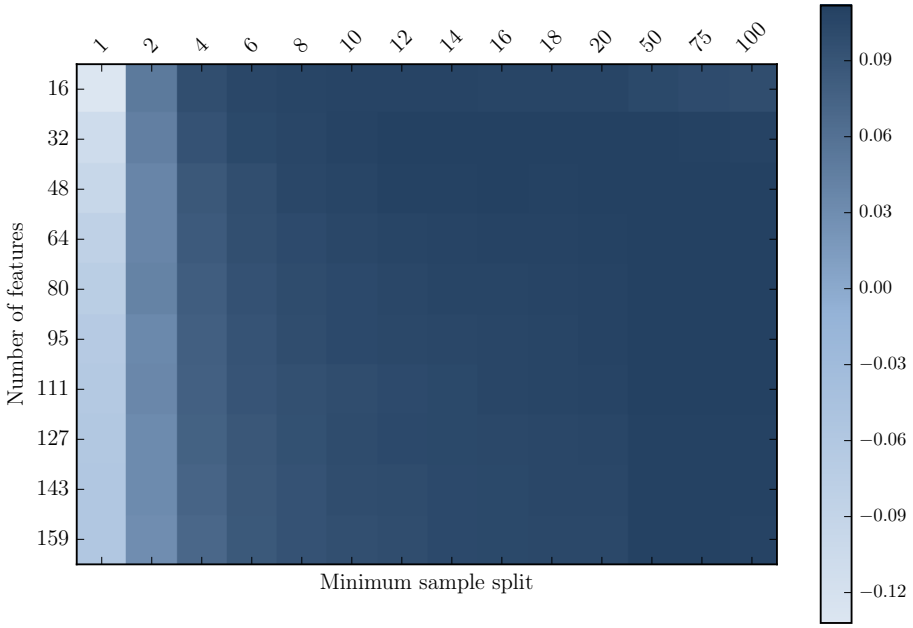

Figure 4.8: $\mathrm{RF_R}$, *total test* R^2 *on cross-validation test sets*

In order to visualize the best Random Forest, we train it on the whole training set without any cross-validation. Then, we compute the average expected cost per age and we compare it against the empirical data. In this case, we obtain an R^2 score of 13.63% compared to the pure cross-validation test score of 11.18%. We can observe in the figure that the Random Forest model renders quite accurately the reimbursement curve while at the same time slightly smoothing it. Our results seem to corroborate those obtained by Huther, which achieved similar performance with this same algorithm.

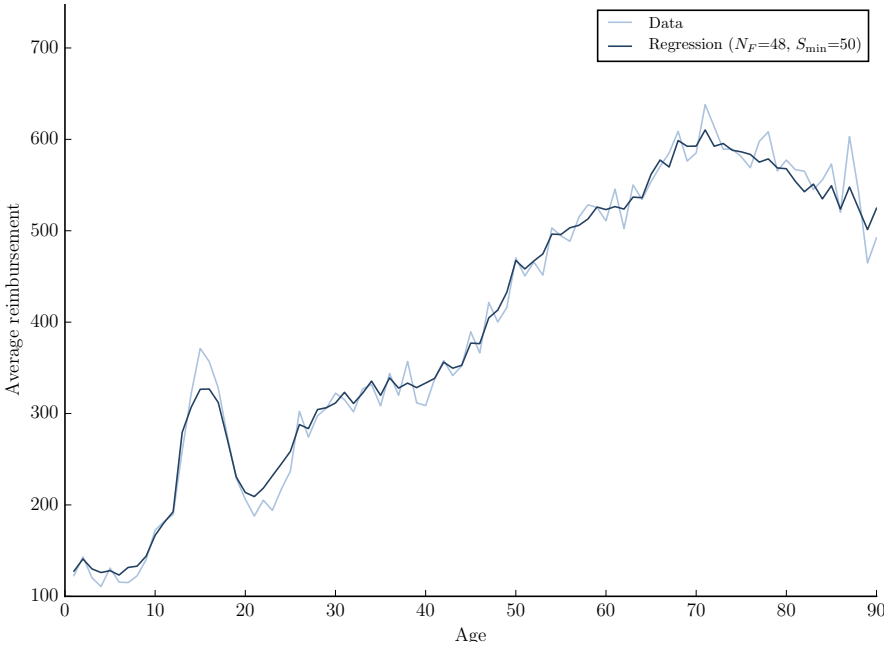

Figure 4.9: RF_R, *average reimbursement given occurrence per age*

Feature importance analysis with the Random Forest algorithm

As previously, we use the Random Forest capability to measure a feature's relative importance. Results are shown in the next figure.

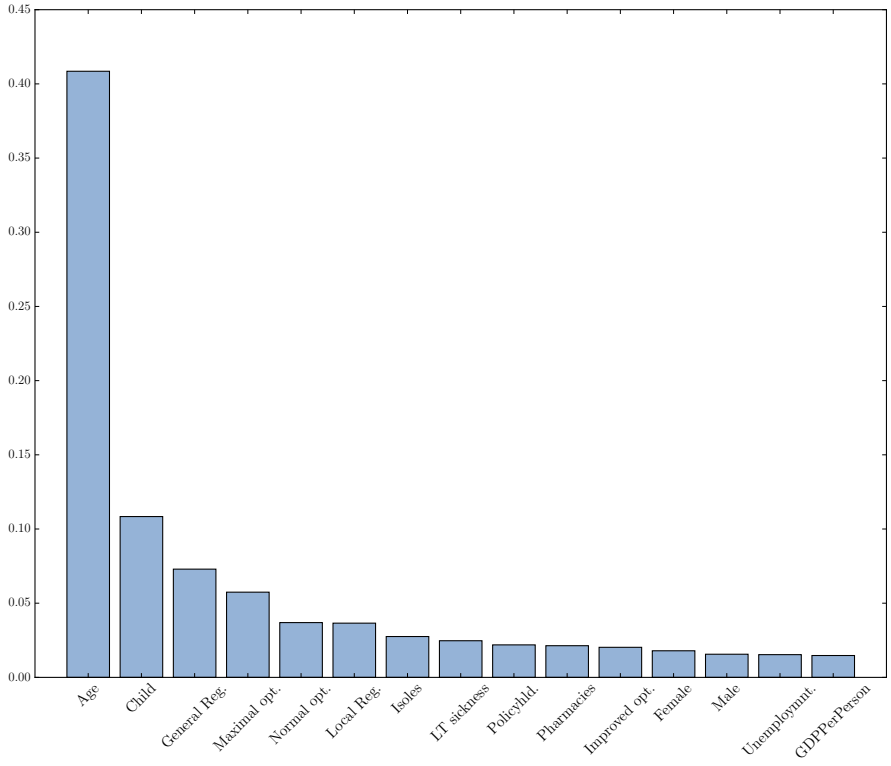

Figure 4.10: RF_R, *attribute importances for regression*

As expected, age is the main driver by a large difference. We also observe that the dummy variable indicating whether the person is a child or not has also strong predictive power, which is consistent with our previous observations regarding optical and dental expenses of teenagers. All 3 dummy variables indicating the contract option – normal, improved or maximal – are also amongst

the top 10 explanatory features.

Summary

Needless to say, results for the Random Forest are much better than those obtained for the regression SVM, in all aspects – hyper-parameter tuning, calculation time and quality of the prediction. In fact, with the Random Forest algorithm we have actually managed to obtain a reasonably operational model, which could support the pricing process of health insurance contracts:

- Random Forest hyper-parameters are easier to tune than those of the SVM, in the sense that either the impact of hyper-parameters on the quality of the model is unambiguous or the region where their values can lay is bounded: the number of trees N_T is undoubtedly the better the higher, subject to computation power constraints; N_F is bounded below by 1 and above by the number of features q; and the minimum number of samples $S_{\min}$ is also bounded below by 1 and above by a number which must be inferior to the sample size. On the other hand, values for β, σ and ϵ lay respectively on $\mathbb{R}_+$, $\mathbb{R}$ and $\mathbb{R}_+$; moreover as observed in our experiments their impact on the model is relatively ambiguous: our graphics on the classification task showed highly fluctuating behaviour.

- Calculation time for the regression SVM seems prohibitively high, at least on a laptop. This is mainly due to the presence of 3 hyper-parameters that we need to optimize and highlights an issue of the cross-validation method: its complexity is polynomial in the number of hyper-parameters. On the other hand, not only was the Random Forest algorithm relatively quick, but the main hyper-parameter to control complexity – the number of trees N_T – seems to admit low values while maintaining a good performance: as stated before, Huther obtained very promising results with $N_T = 25$, 4 times lower than our parameterisation here.

- Finally, the quality of the prediction does not require a long discussion: the SVM model did not managed to produce an R^2 above 0, whereas our best test score for the Random Forest algorithm was 11.18%.

Due to the calculation cost of the SVM and the unambiguously superior performance of the Random Forest, we did not undertaken a stability analysis on the solutions.

4.6.3 The contract effect: different models for different contracts

In the previous chapter we highlighted an important feature of our data: the sample is roughly divided in 3 subgroups of equal size, one group corresponding to policyholders subscribing to the "SNCF" contract, another group including those who adhere to the "Isoles" contract and the rest. We noted that SNCF policyholders might have different reimbursement patterns due to their specificity – this subgroup majoritarily adheres to the special regime. We therefore deemed necessary to generate reimbursement models specifically targeted at each one of these subgroups: the specificity of the SNCF population, resulting in significantly different reimbursement patterns, and its internal homogeneity might corrupt the overall performance of a model trained and applied indiscriminately to the whole population; by segmenting our database in 3 different subgroups and training a bespoke model for each one, we might be able to improve the performance of each model with respect to a global model.

Due to the unappealable superiority of the Random Forest results obtained above, we limited ourselves to training a Random Forest regressor on each one of the 3 subgroups we described. We retained the same training set we have used previously, except we divided it by contract. We also followed the same training procedure as before, testing the same set of values for the model's hyper-parameters. In the following table we show the optimal R^2 test score we obtained for each subgroup, including our previous result for the model applied to the whole population, as well as the relative

difference (RD) with respect to the base case – *i.e.* the model trained on the whole dataset.

	Test R^2	**RD vs. All contracts**
SNCF	12.59%	+12.61%
Isoles	9.01%	−19.41%
Other ex. SNCF and Isoles	11.71%	+4.74%
All contracts	11.18%	0%

Table 4.5: *Quality metrics for optimal models per subgroup – based on test R^2*

Overall, our results show that there does not seem to be a significant improvement in performance from separating our population in 3 distinct groups: an independent model targeted specifically at the subset of the population for which variable "Product" equals "SNCF" achieves a better test R^2 with respect to the model trained on the whole population, with an improvement equal to 12.61%; on the other hand, modelling separately subscribers to the "Isoles" contract results in a decrease of −19.41% of test R^2; finally, the performance achieved on the remaining population (excluding both holders of either the "SNCF" or the "Isoles" contract) is 11.71%, a meagre 4.74% higher than the base case.

4.6.4 Merging concepts: an Ensemble Linear Regression model

In the beginning of this work, we explained that ensemble algorithms work by combining the predictions of weaker models. In order to benchmark our SVM model we have implemented such an algorithm, the Random Forest, that had already proved very accurate on the previous study undertook with this database.

Here, we introduce a bespoke ensemble linear regression model: our goal is to optimally combine the predictions of both the regression SVM and the regression Random Forest. Letting f_{Ens} be the ensemble prediction function, f_{SVM} the SVM prediction function and f_{RF} the Random Forest prediction function, the ensemble model is:

$$f_{\text{Ens}}(\mathbf{x}) = af_{\text{SVM}}(\mathbf{x}) + bf_{\text{RF}}(\mathbf{x})$$

Where a and b are the regression coefficients that need to be estimated through the ordinary least squares method. The model does not include an intercept because each individual model f_{SVM} and f_{RF} has already an intercept term. To estimate the ensemble model, we follow the following procedure:

Algorithm 2 *Ensemble Linear Regression Algorithm*

1: Estimate the optimal decision functions f^*_{SVM} and f^*_{RF} with the whole training set;
2: Generate predictions for the training set with both models;
3: Estimate the optimal ensemble decision function $f^*_{\text{Ens}} = (a^*, b^*)$ with the previous predictions;
4: Generate validation predictions with the validation dataset.

Note that in this case, there are no hyper-parameters to tune, hence there is only a training and a validation phase. We found optimal values of $a^* \approx -0.2191$ and $b^* \approx 1.1402$, resulting in a train R^2 score of 14.02%. As expected, the optimal decision function overweights the Random Forest prediction while it assigns a negative coefficient to the SVM-predicted value.

4.7 Validation of results

As we have explained, the last phase of our experiments is to validate our results by training our models on the whole training dataset, then applying them to the validation data we had left out – corresponding to 10% of the whole data sample. Unfortunately, results obtained for the classification SVM during the cross-validation phase were disappointing and we have not been able to retain a single pertinent model given the "best" model converged to a majority classifier. Under these circumstances, we had then no choice but to execute the classification validation only on the Random Forest model.

In the table below we summarise the specifications of the optimal models, selected based on their test scores.

	β	σ	ϵ	n_components	n_iter
Regression	2^{-40}	2^{-10}	100	3,000	20

Table 4.6: *Hyperparametrisation of the optimal SVM algorithm*

	N_T	N_F	$D_{\max}$	$S_{\min}$
Classification	100	159	`None`	10
Regression	100	48	`None`	50

Table 4.7: *Hyperparametrisation of optimal Random Forest algorithms*

	a	b
Regression	−0.2191	1.1402

Table 4.8: *Hyperparametrisation of the optimal ensemble algorithm*

It is worth to note that the best classification model corresponds to $N_F = 159$, which seems to defeat the purpose of the Random Forest concept: picking a value for N_F which is lower than the total number of features q is a way to decrease the overall variance of the model by controlling the correlation between the trees forming the forest. This is another indication that applying a classification model to the occurrence variable of this dataset is a difficult exercise, and that there might not be any underlying pattern to learn.

	Model $\mathbf{RF_C}$	**$\mathbf{Dummy_C}$**
Train accuracy	95.61%	p_1
Train Kappa	3.16%	0%
Validation accuracy	95.69%	p_1
Validation Kappa	−0.15%	0%
Validation J Stat	0.72%	0%

Table 4.9: *Validation metrics for the optimal classification Random Forest*

	Model G-SVM-SGD$_\mathbf{R}$	**Model RF$_\mathbf{R}$**	**Ensemble model**
Train R^2	-0.75%	13.82%	14.02%
Validation R^2	-8.75%	11.02%	10.80%

Table 4.10: *Validation metrics for the optimal regression models*

In the above tables we show validation scores for both problems. Results are very close to those obtained during the cross-validation phase:

- The classification Random Forest struggles to learn any pattern from the data in order to predict occurrence and significantly outperform the dummy classifier – which consists on classing all data points to class 1.
- The regression SVM achieves very poor results on both the training and the validation set. On top of that, the significant difference between the train and the validation score is an indication that the model might have over-fitted the data.
- The regression Random Forest manages to achieve a R^2 score above 10% for both the train and the validation data sample. The validation score obtained (11.04%) is very close to the best test score reached during cross-validation (11.18%).
- The Ensemble linear regression model has a performance which is close to the Random Forest performance. However, it suffers more from over-fitting than the Random Forest algorithm, given the difference between the train and the validation score. Due to the weakness of the SVM regression, we cannot realistically expect this model to consistently outperform the Random Forest on the validation set because in this case the SVM prediction drags down the overall performance of the ensemble technique.

Chapter 5

Conclusions and future work

Thoughts on the applicability of Support Vector Machines in an insurance framework

5.1 Concluding remarks

In this memoir, we have first given a brief overview of the Machine Learning discipline, discussing its history and describing some cartographies of the field based on learning styles and types of algorithms. We have also made a brief introduction to the mathematical theory that underpins Machine Learning algorithms so that the reader has a better understanding of the discipline, its aim and its tools. We have also taken the occasion to discuss about Machine Learning in the context of insurance: the opportunities it offers as well as the threats it poses.

In a second chapter we have made a deep dive into the theory of Support Vector Machines, starting from the Linear Maximal Margin classifier and finishing with the concepts of soft margins and kernels. After having discussed about the situation of the Support Vector Machine within Machine Learning, we have shifted to practical considerations, presenting a series of algorithms aimed at making more efficient the numerical solving of the model.

Following the SVM presentation, we have familiarised the reader with the health insurance industry, by presenting its working principles and the particularities of the French health insurance market. We have also described the dataset at our disposal and how we have enlarged it by considering additional, external variables. We have made a preliminary analysis to understand the impact of each variable in occurrence and cost.

Finally, in chapter 4 we have described the experiments we have carried out, in which we have tried to model both claim occurrence and claim reimbursement. We have benchmarked the SVM model to the popular Random Forest algorithm; results for the SVM model were not encouraging, whereas Random Forest performance was reasonable. We also tested a very simple linear ensemble model to combine the SVM's and the Random Forest's prediction.

In the next subsections we briefly summarise our main takeaways from this memoir: the relative weakness of the Support Vector Machine against the Random Forest algorithm, and the importance of computational issues in implementing Machine Learning algorithms. We will then discuss future work based on these conclusions.

5.1.1 The Support Vector Machine model

The Support Vector Machine is probably the Machine Learning model with the strongest theoretical foundations. It is based on the maximal margin principle, ensuring the right balance between learning from the available data and generalizing to new samples; its solution is obtained through a convex quadratic optimization program, a type of optimization program which is very well understood and for which the solution is always global; finally, when equipped with a Gaussian kernel, the SVM is an extremely powerful machine which, provided with enough computation and memory capacity, can learn any pattern in the data.

Despite its theoretical soundness, in practice our experience shows the SVM is outperformed by the Random Forest algorithm in terms of hyper-parameter interpretability, predictive prowess, stability of the solution and computational complexity:

- Random Forest hyper-parameters are much easier to interpret and hence to tune;
- Our experiments have showed that Random Forests are more accurate than the SVM for both tasks – although it can be argued that both are non-comparable in the regression task due to the nature of their loss functions – ;
- In addition, the analysis we undertook on the classification task showed clearly that the SVM suffered from very high estimation variance, whereas the Random Forest was extremely stable;
- Finally, the SVM is computationally very expensive compared to the Random Forest – particularly in the regression case where cross-validation has to be done for 3 parameters – and this is after tuning it with Stochastic Gradient Descent and kernel approximation to speed up calculations; the standard SVM with SMO-based solving algorithms revealed itself to be absolutely unpractical for a dataset size like ours, at least with the laptop computing power we had available – 4-core Intel® Core™ i7-6500U processor with 8.00GB of RAM memory.

5.1.2 IT Infrastructure

Throughout this memoir, one thing has been made clear: practical challenges related to IT capabilities are central to Machine Learning and Support Vector Machines have proved they are no exception to the rule. These algorithms extract relevant information by analysing vast quantities of data: the more data they have the better their predictive power will be. It is thus crucial to have the necessary IT infrastructure to store and manipulate this data as well as to execute the estimation procedures of the models.

In our limited exposure to IT challenges, we have identified 2 critical variables to control for: the *volatile memory – i.e.* the Random-Access Memory – and the *computation speed*. Broadly speaking, volatile memory depends on the RAM size, measured in GB, whereas computation speed is determined by the processors.

In our experiments, computation speed was a critical factor: the complexity of standard SVM estimation schemes led us to implement numerical tricks – SGD, kernel approximation – to speed up the estimation of the model; we also set up a remote computing solution to carry out the training phase of the regression SVM more quickly.

Generally computation times were quite long for both the SVM and Random Forests. This limitation can be tackled physically with more powerful processors: when undertaking calculations in Amazon Web Services we clearly noted the difference in speed. As it has also been mentioned, numerical tricks as those mentioned above can help solve the problem from a purely mathematical perspective.

Memory was also an important bottleneck. When we turned to AWS to rent CPUs, we selected an

instance with 36 processors and 60.00GB of RAM; however, due to the volatile memory requirements of the SVM, we were only able to parallelize computations on 9 processors – it is necessary to account for memory spikes that can trigger memory errors in the middle of calculations. Hence, based on these calculations as well as those done on the laptop, we roughly estimate the memory requirements for training a SVM model – tuned with SGD and kernel approximation – with `scikit-learn` on our data set to be between 5.00GB and 7.00GB of RAM, while the dataset itself has a size of 0.02GB in CSV format. To tackle memory issues, recipes are similar to those for computation speed: a greater quantity of RAM and mathematical tricks that allow to reduce the size of the required volatile memory of the model.

5.2 Future work

We finish off by discussing additional exploration paths for subsequent works in relation to the issues we encountered when working on this memoir.

5.2.1 Classification problems with unbalanced classes

Although our models achieved reasonable results on the regression task, they essentially failed to predict the occurrence of medical acts – a classification task. This failure was partly explained by the highly unbalance nature of the problem: label 1 samples represented approximately 95.5% of all samples, whereas label 0 only represented 4.5%. In this setting, Machine Learning models might display a degenerate behaviour and converge to a majority classifier, as it has been observed for the Support Vector Machine.

This shortcoming can be dealt with through different methods, which generally speaking allow to weight more or less the different labels:

- **Class weights**: the SVM can be tuned such that different classes have different error tolerance hyper-parameters C_0 and C_1; it could even be possible to construct a sequence $\{C_1, ..., C_n\}$ of error penalties for each individual data point. Assigning different hyper-parameters modifies the relative cost of errors and instructs the machine to concentrate on avoiding mistakes on labels with high costs.
- **Threshold selection**: other classification metrics and tools, such as the Receiver Operating Characteristic (ROC) curve and the ROC Area Under the Curve (AUC), might be used to modify the threshold which separates label 0 from label 1 samples – and hence target label 0 and label 1 accuracy rates – once a model has been trained.

However, our data structure is too limited to be able to apply these remediations: in order to intelligently select class weights or desired accuracy rates on each label, we would need more information to determine our preference towards one or the other – as explained previously, error tolerance hyper-parameters C_0 and C_1 can be tuned so as to give the same overall weight to both classes, but this solution lacks a proper business-based ground. Given this hypothetical additional information, such as for example profit per contract, we could answer questions such as "*do we want to be more confident on the beneficiaries predicted to not incur any medical act, or on those predicted to incur at least one act?*".

5.2.2 Probability calibration

There exists methodologies to extract probability estimates from the values of a classifier decision function; these techniques are collectively known as *probability calibration* and consist on mapping a classifier confidence score in $\mathbb{R}$ into a probability vector $(p_0, 1-p_0) \in [0,1]^2$ where p_0 is the probability of belonging to class 0. However due to the poor results obtained on our classification task, we did not calibrate probabilities as results were irrelevant. When reviewing the relevant literature we identified 2 main techniques to achieve this:

- **Platt scaling** (Platt, 1999): this method was developed by Platt specifically for SVMs, although it can be used for other classifiers. Schematically, it consists on fitting a logistic regression to confidence scores.
- **Isotonic regression**: this method fits a stepwise, non-decreasing function to rescaled scores (Zadrozny and Elkan, 2002).

With a classification model with better results than those obtained in this work, it might be interesting to explore these probability calibration techniques.

5.2.3 Algorithmic precision and computational efficiency

In this memoir, we introduced some additional techniques from the realm of Machine Learning – the Stochastic Gradient Descent (SGD) algorithm, kernel approximation methods – which have a small impact on the SVM performance while significantly improving computation time and sample scalability. As already mentioned, these techniques can be extended and modified in multiple ways to tackle learning tasks.

There are two extensions to the SGD algorithm that should deserver further work: *Average Stochastic Gradient Descent* (ASGD) which is a slight variation of the plain SGD that updates parameter $\mathbf{w}$ the same way, but which returns an average value over all previous estimations of the weight vector; and *Mini-Batch Stochastic Gradient Descent* (Mini-Batch SGD) which updates the weight vector by computing gradients for B observations, $1 < B < n$, which are chosen randomly according to a uniform distribution.

Regarding kernel approximations, we studied a particular method, Random Kitchen Sinks, which allows to approximate kernels that can be interpreted as a random variable expectation over some probability measure. A more general method exists, the *Nyström method*, which allows to approximate the kernel matrix by randomly picking rows and columns from it. In principle the Nyström method allows to approximate any kind of kernel. The interested reader might consult Williams and Seeger (2000) for a comprehensive overview of this technique.

On the IT side, as we have explain the main factors to control for are volatile memory and computing speed through RAM size and processors respectively. Apart from increasing RAM size and processor power, there are many other technical solutions to these limitations, including optimized matrix sparsity, efficient parallelization schemes, leveraging non-volatile memory for computation purposes, etc. However these techniques are outside the scope of this memoir and we will not discuss them further.

5.2.4 Exploring ensemble models

In this memoir we have tested a very simple ensemble model that combined linearly the predictions of both the Support Vector Machine and the Random Forest algorithms for the regression task. Our model obtained a performance which was very close to that obtained by the Random Forest.

Subsequent works could explore more complex ensemble techniques. For example, it might be interesting to develop an *Ensemble SVM* out of weaker SVMs, merging the support vector concept with Random Forests. Indeed, the stability analysis showed that our SVM had high variance whereas its train and test accuracy on the classification task were similar, which is generally a sign of low bias, making our SVM a perfect candidate for ensemble learning.

We sketch some ideas related to the implementation of such an algorithm:

- An Ensemble SVM could be made up of linear SVMs, which are quicker to estimate that SVMs with complex kernels. If computation power and time allows or linear SVMs prove to be insufficiently precise, kernels can be introduced.

- To tackle computation speed, each SVM could be trained on a bootstrapped sample of *smaller size* than the original sample. To understand the advantage of this methodology, consider that SMO-based solvers – like `LIBSVM` – tend to scale polynomially in the size of the data sample, whereas a procedure consisting on training a series of SVMs to combine them afterwards would scale linearly on the number of SVMs: thus training more SVMs with smaller bootstrapped samples could yield a computational gain over training less SVMs with bigger bootstrapped samples. Empirical analysis should be undertaken to confirm this hypothesis and analyse the impact on the model predictive power.
- It could also be interesting to borrow the feature bootstrapping technique from Random Forests and test it on the Ensemble SVM. This should decrease the correlation between the different SVMs and hence decrease the overall variance of the Ensemble SVM.

Different aggregation schemes – *i.e.* how to combine the predictions of the SVM – could also be explored: for example, an idea could be to train a classification SVM to choose which one of the weaker linear SVMs should be used to make a prediction for a specific data point $\mathbf{x}$.

Of course, the ideas sketched out here should only serve as an initial inspiration for further work as different ensemble techniques might be considered, for example the popular *boosting* technique – instead of the bagging principle, which is the category to which Random Forests belong. Indeed, the degree of tuning of any Machine Learning model is immense and it is up to the modeller to combine algorithms in efficient and innovative ways so as to yield improved predictions. A good place to start would be the paper by Kim, Pang, Je, Kim and Bang (2003), "*Constructing support vector machine ensemble*", where the authors explore both the bagging and boosting methods applied to SVMs.

Bibliography

Accord National Interprofessionnel du 11 Janvier 2013 pour un nouveau modèle économique et social au service de la compétitivité des entreprises et de la sécurisation de l'emploi et des parcours professionnels des salariés (2013).

Aizerman, M.A., Braverman, E.M. and Rozoner, L.I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". *Automation and Remote Control*, 25:821–837.

Alpaydin, E. (2010). *Introduction to Machine Learning*. MIT Press, 2nd ed.

Anthony, M. and Biggs, N. (1995). "PAC Learning and Artificial Neural Networks". Tech. Rep. NC-TR-95-023, NeuroCOLT, Royal Holloway.

Aouizarate, J.M. (2010). *Alternative Neuronale en Tarification Santé*. Memoir for the obtention of the French Actuarial Qualification, Conservatoire National des Arts et Métiers (CNAM).

Assemblée nationale et Sénat (1989). "Loi n°89-1009 du 31 décembre 1989 renforçant les garanties offertes aux personnes assurées contre certains risques". Journal Officiel n°1 du 2 janvier 1990, page 13.

Assemblée nationale et Sénat (2004). "Loi n°2004-810 du 13 août 2004 relative à l'assurance maladie". Journal Officiel n°0190 du 17 août 2004, page 14598.

Ben-Hur, A., Horn, D., Siegelmann, H.T. and Vapnik, V.N. (2001). "A Support Vector Method for Clustering". *Journal of Machine Learning Research*, 2:125–137.

Ben-Hur, A. and Weston, J. (2009). *Biological Data Mining*, chap. A User's guide to Support Vector Machines, pp. 223–239. Springer Protocols.

Boser, B.E., Guyon, I. and Vapnik, V.N. (1992). "A training algorithm for optimal margin classifiers". In "Proceedings of the Fith Annual Workshop of Computational Learning Theory", ACM, Pittsburgh, PA, USA.

Bottou, L. (2012). "Stochastic Gradient Descent Tricks". In "Neural Networks, Tricks of the Trade, Reloaded", Lecture Notes in Computer Science, pp. 430–445. Springer.

Bottou, L. and Bousquet, O. (2011). "The Tradeoffs of Large Scale Learning". In "Optimization for Machine Learning", pp. 351–368. MIT Press.

Bottou, L. and Lin, C.J. (2007). "Large Scale Kernel Machines". chap. Support Vector Machine Solvers, pp. 301–320. MIT Press.

Bousquet, O., Boucheron, S. and Lugosi, G. (2003). "Introduction to Statistical Learning Theory". In "Advanced Lectures on Machine Learning", Lecture Notes in Computer Science, pp. 169–207. Springer.

Boyd, S. and Vandenberghe, L. (2009). *Convex Optimization*. Cambridge University Press, 1st ed.

Breiman, L. (2001). "Random Forests". *Machine Learning*, 45(1):5–32.

Brownlee, J. (2013). "A Tour of Machine Learning Algorithms". `http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/`.

Burges, C.J.C. (1998). "A Tutorial on Support Vector Machines for Pattern Recognition". *Data Mining and Knowledge Discovery*, 2:121–167.

Burges, C.J.C. and Crisp, D. (2000). "Uniqueness of the SVM Solution". *Advances in Neural Information Processing Systems*, 13:223–229.

Chang, C.C. and Lin, C.J. (2013). "LIBSVM: a Library for Support Vector Machines". *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27. This document was created in 2001 and since then has been actively maintained/updated.

Chen, P.H., Fan, R.E. and Lin, C.J. (2005). "Working Set Selection Using Second Order Information for Training Support Vector Machines". *Journal of Machine Learning Research*, 6:1889–1918.

Chen, P.H., Fan, R.E. and Lin, C.J. (2006). "A Study on SMO-type Decomposition Methods for Support Vector Machines". *IEEE Transactions on Neural Networks*, 17:893–908.

Ciuperca, I.S. (2012). "Cours d'optimisation". Lecture notes from the Institut de Science Financière et d'Assurances (ISFA).

Cohen, J. (1960). "A coefficient of agreement for nominal scales". *Educational and Psychological Measurement*, 20(1):37–46.

Cortes, C. and Vapnik, V.N. (1995). "Support-Vector Networks". *Machine Learning*, 20:273–297.

Council Directive 2004/113/EC on equal treatment between men and women in access to and supply of goods and services (2004). Official Journal of the European Union, L 373/37.

Cybenko, G. (1989). "Approximations by superpositions of sigmoidal functions". *Mathematics of Control, Signal, and Systems*, 2(4):303–314.

DREES (2015). *Les dépenses de santé en 2014 - Résultats des Comptes de la Santé*. DRESS, 2015 ed.

Erästö, P. (2001). *Support Vector Machines – Backgrounds and Practice*. Academic Dissertation for the Degree of Licentiate of Philosophy, Rolf Nevanlinna Institute, University of Helsinki.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In "KDD–96 Proceedings", Association for the Advancement of Artificial Intelligence, Portland, OR, USA.

Friedman, E., Harley, A. and Southwood, K. (2016). "Insurance big data – float like a butterfly, sting like a bee". Willis Towers Watson.

Friedman, J. (1997). "Data mining and statistics: What's the connection?" In "Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics", Houston, TX, USA.

Gauss, C.F. (1809). *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. Perthes.

Gretton, A., Strathmann, H. and Jitkrittum, W. (2016). "Reproducing kernel Hilbert spaces in Machine Learning". Lecture notes from the University College of London.

Guha, R., Manjunath, S. and Palepu, K. (2015). "Comparative analysis of machine learning techniques for detecting insurance claims fraud". Wipro.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2nd ed.

Herbrich, R. and Williamson, R.C. (2002). *The Handbook of Brain Theory and Neural Networks*, chap. Learning and Generalization: Theoretical Bounds, pp. 619–623. MIT Press, 2nd ed.

Hornik, K. (1991). "Approximation Capabilities of Multilayer Feedforward Networks". *Neural Networks*, 4(2):251–257.

Hsu, C.W., Chang, C.C. and Lin, C.J. (2003). "A Practical Guide to Support Vector Classification". Tech. Rep. July 2003, Department of Computer Science of the National Taiwan University. Version from 2010.

Huang, H.Y. and Lin, C.J. (2016). "Linear and kernel classification: when to use which?" In "SIAM International Conference on Data Mining", Miami, FL, USA.

Huther, C. (2014). *Tarification et suivi de portefeuille en assurance non-vie : analyse comparative de modèles de prévisions en assurance Santé*. Memoir for the obtention of the French Actuarial Qualification, Institut de Science Financière et d'Assurances (ISFA) & Mazars Actuariat.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

Kaminska, I. (2016). "Breaking insurance models with big data". *Financial Times*.

Keerthi, S.S. and Lin, C.J. (2003). "Asymptotic behaviors of support vector machines with Gaussian kernel". *Neural Computation*, 15:1667–1689.

Kim, H.C., Pang, S., Je, H.M., Kim, D. and Bang, S.Y. (2002). "Support vector machine ensemble with bagging". In "Pattern Recognition with Support Vector Machines", Lecture Notes in Computer Science, pp. 397–408. Springer.

Kim, H.C., Pang, S., Je, H.M., Kim, D. and Bang, S.Y. (2003). "Constructing support vector machine ensemble". *Pattern Recognition*, 36:2757–2767.

Lagnaoui, M.J. (2015). *Machine learning et nouvelles classes de problèmes actuariels : une illustration par l'étude de cas*. Memoir for the obtention of the French Actuarial Qualification, EURo Institut d'Actuariat (EURIA).

Langley, P. (2011). "The changing science of machine learning". *Machine Learning*, 82:275–279.

Legendre, A.M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. Didot, F.

Liang, P. (2016). "Statistical Learning Theory". Lecture notes from Stanford University.

Lin, C.J. (2001a). "Formulations of support vector machines: a note from an optimization point of view". *Neural Computation*, 13(2):307–317.

Lin, C.J. (2001b). "On the convergence of the decomposition method for support vector machines". *IEEE Transactions on Neural Networks*, 12:1288–1298.

Manton, J.H. and Amblard, P.O. (2015). "A Primer on Reproducing Kernel Hilbert Spaces". *Foundations and Trends in Signal Processing*, 20(20):1–130.

McCulloch, W.S. and Pitts, W.H. (1943). "A logical calculus of the ideas immanent in nervous activity". *Bulletin of Mathematical Biophysics*, 5:115–133.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill, 1st ed.

Mitchell, T. (2006). "The Discipline of Machine Learning". School of Computer Science, Carnegie Mellon University.

Ng, A. (2016). "Machine learning (CS229)". Lecture notes from the University of Stanford.

Niedzwiedz, M. (2014). *Utilisation des supports vecteurs machines pour l'accélération du calcul du capital économique*. Memoir for the obtention of the French Actuarial Qualification, École Nationale de la Statistique et de l'Administration Économique (ENSAE) & Actuaris.

Nonne, H. (2015). "Une petite histoire du Machine Learning". `http://www.quantmetry.com/single-post/2015/10/28/Une-petite-histoire-du-Machine-Learning/`.

O'Connor, B. (2008). "Statistics vs. machine learning, fight!" `https://brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight/`.

Osuna, E., Freund, R. and Girosi, F. (1997). "An Improved Training Algorithm for Support Vector Machines". In "Proceedings of IEEE Neural Networks for Signal Processing 1997", Amelia Island, FL, USA.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research*, 12:2825–2830.

Platt, J.C. (1998). "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines". Tech. Rep. MSR-TR-98-14, Microsoft Research.

Platt, J.C. (1999). "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". *Advances in large margin classifiers*, 10(3):61–74.

Pontil, M. and Verri, A. (1998). "Properties of support vector machines". *Neural Computation*, 10:955–974.

Rahimi, A. and Recht, B. (2007). "Random Features for Large-Scale Kernel Machines". *Advances in Neural Information Processing Systems*, 20.

Rai, P. (2011). "Learning theory". Lecture notes from the School of Computing of the University of Utah.

Ralph, O. (2016). "Insurers test the limits of telematics' big data". *Financial Times*.

Rao, A., Yoder, J. and Busse, S. (2016). "AI in Insurance: Hype or reality?" PricewaterhouseCoopers.

Rosenblatt, F. (1957). "The perceptron – a perceiving and recognizing automaton". Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory.

Rossi, F. (2008). "An introduction to statistical learning theory". Lecture notes from Telecom ParisTech University.

Rudin, C. (2012). "Prediction: Machine Learning and Statistics". Lecture notes from the Massachusetts Institute of Technology, available on MIT OpenCourseWare.

Schölkopf, B. and Smola, A.J. (2005). *Encyclopedia of Biostatistics*, vol. 8, chap. Support Vector Machines and Kernel Algorithms, pp. 5328–5335. John Wiley & Sons.

Sebag, M. (2014). "A tour of Machine Learning: an AI perspective". *AI Communications*, 27(1):11–23.

Sécurité Sociale (2015). *Les chiffres clés de la Sécurité Sociale 2014*. Sécurité Sociale, 2015 ed.

Shashua, A. (2008). "Introduction to Machine Learning". Lecture notes from the Hebrew University of Jerusalem.

Simon, H. (1983). *Reason in Human Affairs*. Stanford University Press.

Smola, A.J. and Schölkopf, B. (2003). "A Tutorial on Support Vector Regression". *Statistics and Computing*, 14(3):199–222.

Taillieu, F., Delucinge, S. and Bellina, R. (2014). "Méthode d'apprentissage statistique – Machine Learning". Milliman.

Turing, A.M. (1950). "Computing machinery and intelligence". *Mind*, 59:433–460.

Valeant, L. (1984). "A theory of the learnable". *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V.N. (1995). *The Nature of Statistical Learning Theory*. Springer, 1st ed.

Williams, C.K.I. and Seeger, M. (2000). "Using the Nyström Method to Speed Up Kernel Machines". *Advances in Neural Information Processing Systems*, 13:682–688.

Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J. and Steinberg, D. (2008). "Top 10 algorithms in data mining". *Knowledge and Information Systems*, 14:1–37.

Yom-Tov, E. (2003). "An Introduction to Pattern Classification". In "Advanced Lectures on Machine Learning", Lecture Notes in Computer Science, pp. 1–20. Springer.

Youden, W. (1950). "Index for rating diagnostic tests". *Cancer*, 3:32–35.

Zadrozny, B. and Elkan, C. (2002). "Transforming classifier scores into accurate multiclass probability estimates". In "Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining – KDD'02", ACM, Edmonton, AB, Canada.

Appendix A

Introduction to convex optimization

In this appendix we review some of the main results of convex optimization theory which will allow us to study the support vector machine optimization problem. These concepts underpin the strong mathematical properties of the SVM:

- Global solutions of convex problems;
- The Karush-Kuhn-Tucker theorem; and
- Dual representation of problems.

A.1 Main definitions and an important theorem

We start by stating a few basic definitions:

Definition 5 (Convex optimization program) *Let* $\mathbf{x} \in \mathbb{R}^p$ *be the optimization variable,* $f : \mathbb{R}^p \to \mathbb{R}$ *a convex function,* $g_1, ..., g_{n_g} : \mathbb{R}^p \to \mathbb{R}$ *a sequence of convex functions and* $h_1, ..., h_{n_h} : \mathbb{R}^p \to \mathbb{R}$ *a sequence of affine functions. A* convex optimization program $(\mathcal{P}_C)$ *has the following form:*

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } g_k(\mathbf{x}) \leq 0, \quad k \in \{1, \cdots, n_g\} \\ &\qquad\qquad h_k(\mathbf{x}) = 0, \quad k \in \{1, \cdots, n_h\} \end{aligned}$$

Definition 6 (Convex quadratic optimization program) *We say a convex optimization program* $(\mathcal{P}_C)$ *is a* convex quadratic optimization program *if its objective function* f *is convex quadratic and the inequalities constraints* $g_1, ..., g_{n_g}$ *are affine functions.*

Definition 7 (Feasibility) *A point* $\mathbf{x}$ *is* feasible *for an optimization program* $(\mathcal{P}_C)$ *if it satisfies the constraints* $g_1, ..., g_{n_g}, h_1, ..., h_{n_h}$ *of* $(\mathcal{P}_C)$*.*

Definition 8 (Local optima) *A point* $\mathbf{x}$ *is* locally optimal *for problem* $(\mathcal{P}_C)$ *if it is feasible and if there exists some* $R > 0$ *such that all feasible points* $\mathbf{z}$ *with* $\|\mathbf{x} - \mathbf{z}\| \leq R$ *satisfy* $f(\mathbf{x}) \leq f(\mathbf{z})$*.*

Definition 9 (Global optima) *A point* $\mathbf{x}$ *is* globally optimal *for problem* $(\mathcal{P}_C)$ *if it is feasible and for all feasible points* $\mathbf{z}$*,* $f(\mathbf{x}) \leq f(\mathbf{z})$*.*

The main crucial result related to convex optimization, which makes convex problems very convenient to solve, is the following:

Theorem 4 *Every local solution* $\mathbf{x}^*$ *to a convex optimization problem* $(\mathcal{P}_C)$ *is a global solution.*

This theorem is extremely powerful and explains in part the popularity of the Support Vector Machine – which is a convex quadratic optimization program, as the reader might have observed. There are some Machine Learning algorithms, as for example Neural Networks, that do not solve a convex optimization program and hence they can get "stuck" in local optima.

A.2 The Karush-Kuhn-Tucker Theorem

After having stated the basic definitions and the global solution theorem, we now introduce one of the core results in convex optimization theory; this theorem is extremely useful for solving the SVM program, both from a purely mathematical as well as algorithmic point of view.

First, we observe that for all k we can transform equality constraints into inequality constraints as follows:

$$h_k(\mathbf{x}) = 0 \quad \Leftrightarrow \quad h_k(\mathbf{x}) \leq 0,\ -h_k(\mathbf{x}) \leq 0$$

Hence any convex program $(\mathcal{P}_C)$ can be restated with inequality constraints only. We will further assume that functions f, g_1, ..., g_{n_g} are of *class* C^1 – a condition which is fulfilled in the case of the SVM:

Definition 10 (Function of class C^1) *A function $f : E \to F$ is said to be of* class C^1 *over E if it is differentiable over E and its derivative f' is continuous over E.*

We introduce a few more definitions.

Definition 11 (Feasible region) *The* feasible region *O of a program $(\mathcal{P}_C)$ is the set of all points that satisfy the constraints of the problem:*

$$O = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^p,\ g_k(\mathbf{x}) \leq 0,\ k \in \{1, \cdots, n_g\}\}$$

Definition 12 (Lagrangian function) *Let $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n_g}) \in \mathbb{R}_+^{*\,n_g}$ be the* Lagrange multipliers. *The* Lagrange function *of the convex program $(\mathcal{P}_C)$ is defined as follows:*

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{k=1}^{n_g} \alpha_k g_k(\mathbf{x})$$

Definition 13 (Constraint activation) *A constraint $g_k(\mathbf{x}) \leq 0$ is said to be* activated *in $\mathbf{x} \in O$ if:*

$$g_k(\mathbf{x}) = 0$$

We define the following set of indexes:

$$I(\mathbf{x}) = \{k : g_k(\mathbf{x}) = 0, k \in \{1, \cdots, n_g\}\}$$

This set allows us to define a *variety* of the feasible region O:

$$\tilde{O}(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in \mathbb{R}^p,\ \forall k \in I(\mathbf{x})\ g_k(\mathbf{y}) = 0\}$$

Note that $\mathbf{x} \in \tilde{O}(\mathbf{x})$. We introduce the last definition we need:

Definition 14 (Regular point) *A point $\mathbf{x} \in O$ is said to be* regular *if either:*

1. *$I(\mathbf{x}) = \emptyset$*
2. *For $\mathbf{x} \in \tilde{O}(\mathbf{x})$ the gradient vectors $\nabla g_1(\mathbf{x}), ..., \nabla g_{n_g}(\mathbf{x})$ are linearly independent in $\mathbb{R}^p$.*

With all these definitions in mind, we now state another crucial theorem of convex optimization. The conditions listed in the theorem are known as the *Karush-Kuhn-Tucker* (KKT) *conditions*:

Theorem 5 (Karush-Kuhn-Tucker for a convex problem) *Let $(\mathcal{P}_C)$ be a convex optimization program such that functions f, g_1, ..., g_{n_g} are of class C^1 and $\mathbf{x}^*$ a regular point of the feasible region O of $(\mathcal{P}_C)$. Then $\mathbf{x}^*$ is a local optima of f over O if and only if there exists a vector $\boldsymbol{\alpha}^* \in \mathbb{R}_+^{*n_g}$ such that:*

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) = 0$$

$$\forall k \in \{1, ..., n_g\} : \alpha_k^* g_k(\mathbf{x}^*) = 0$$

This theorem is particularly important in the SVM setting, as we have already explained. The KKT conditions give a necessary and sufficient condition for a vector $\mathbf{x}^*$ to be a global optima of a convex optimization program $(\mathcal{P}_C)$, hence to determine an optima it suffices to find a regular point $\mathbf{x}^* \in O$ and a vector $\boldsymbol{\alpha}^* \in \mathbb{R}_+^{*n_g}$ that fulfils the KKT conditions. This property is widely used by algorithms that aim at solving the SVM program efficiently.

An alternative formulation which is more simple is the following: "*let $(\mathcal{P}_C)$ be a convex optimization program such that functions f, g_1, ..., g_{n_g} are of class C^1. Then $\mathbf{x}^*$ is a local optima of f over $\mathbb{R}^p$ if and only if there exists a vector $\boldsymbol{\alpha}^* \in \mathbb{R}^{n_g}$ such that:*

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) = 0$$

$$\forall k \in \{1, ..., n_g\} : g_k(\mathbf{x}^*) \leq 0$$

$$\forall k \in \{1, ..., n_g\} : \alpha_k^* \geq 0$$

$$\forall k \in \{1, ..., n_g\} : \alpha_k^* g_k(\mathbf{x}^*) = 0$$

Now that we have briefly exposed the KKT theorem, the 3rd element we require from convex optimization theory to study the SVM is the set of results related to duality, which we discuss in the next section.

A.3 Duality in convex optimization problems

To finish off this appendix, we present the main notions and results that link the *primal* and *dual* representations of a convex optimization program – we already discussed these representations in the case of the SVM in the main body of the paper. We first consider a convex problem $(\mathcal{P}_C)$ and we define both representations – we will use the notation $P(\mathcal{P}_C)$ to designate the primal representation and $D(\mathcal{P}_C)$ to designate the dual one.

Definition 15 (Primal representation) *Let $(\mathcal{P}_C)$ be a convex optimization program and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n_g}) \in \mathbb{R}^{*n_g}$ the Lagrange multipliers. Its primal representation is:*

$$P(\mathcal{P}_C) : \min_{\boldsymbol{\alpha}} \left(\max_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) \right)$$

Definition 16 (Dual representation) *Let $(\mathcal{P}_C)$ be a convex optimization program and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n_g}) \in \mathbb{R}^{*n_g}$ the Lagrange multipliers. Its dual representation is:*

$$D(\mathcal{P}_C) : \max_{\mathbf{x}} \left(\min_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) \right)$$

Definition 17 (Duality gap) *Let p^* be the optimal value to the primal representation of a convex optimization program and d^* the optimal value to its dual representation. The* duality gap *is equal to the difference $p^* - d^*$.*

Definition 18 (Strong duality) *A problem* $(\mathcal{P}_C)$ *admitting a primal and a dual solution is* strongly dual *if and only if:*

$$p^* = d^*$$

If strong duality holds, p^* is said to be a *saddle point*:

Definition 19 (Saddle point) *Let* $\mathcal{X}$ *and* $\mathcal{Y}$ *be two sets and* $f : \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$. *We say* $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ *is a* saddle point *of* f *over* $\mathcal{X} \times \mathcal{Y}$ *if:*

$$\max_{y \in \mathcal{Y}} f(x^*, y) = f(x^*, y^*) = \min_{x \in \mathcal{X}} f(x, y^*)$$

In terms of a convex problem with primal and dual representation, we have:

$$f(x^*, y) = \min_{x \in \mathcal{X}} f(x, y)$$

$$f(x, y^*) = \max_{y \in \mathcal{Y}} f(x, y)$$

Strong duality implies that it is equivalent to solve the primal or the dual representation of a problem $(\mathcal{P}_C)$ because they will both yield the same optima. For strong duality to hold $(\mathcal{P}_C)$ normally has to fulfil some *constraint qualifications*. One of the most commonly invoked qualifications are *Slater's conditions*. Before stating them, we introduce a few more definitions, where we will focus on some subset $S \subseteq \mathbb{R}^p$.

Definition 20 (Ball) *A* closed ball $N_\epsilon(\mathbf{x})$, $\epsilon \in \mathbb{R}_+^*$ *and* $\mathbf{x} \in \mathbb{R}^m$, *is the set of vectors which distance to* $\mathbf{x}$ *is equal to or inferior to* ϵ*:*

$$N_\epsilon(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in \mathbb{R}^m,\ \|\mathbf{x} - \mathbf{y}\| \leq \epsilon\}$$

If the inequality is strict, we call it an open ball.

Definition 21 (Affine combination) *A linear combination of vectors* $\mathbf{x}_1$, ..., $\mathbf{x}_m \in S$ *with coefficients* $\lambda_1, ..., \lambda_m \in \mathbb{R}$ *is said to be an* affine combination *if:*

$$\sum_{i=1}^{m} \lambda_i = 1$$

Definition 22 (Affine hull) *The* affine hull $\text{aff}(S)$ *of a set* S *is the set of all affine combinations of elements of* S*:*

$$\text{aff}(S) = \left\{ \sum_{i=1}^{m} \lambda_i \mathbf{x}_i : m > 0, \mathbf{x}_i \in S, \lambda_i \in \mathbb{R}, \sum_{i=1}^{m} \lambda_i = 1 \right\}$$

Definition 23 (Relative interior) *The* relative interior $\text{relint}(S)$ *of a set* S *is its interior with respect to the affine hull* $\text{aff}(S)$*:*

$$\text{relint}(S) = \left\{\mathbf{x} : \mathbf{x} \in S,\ \exists\, \epsilon \in \mathbb{R}_+^*,\ N_\epsilon(\mathbf{x}) \cap \text{aff}(S) \subseteq S\right\}$$

With these definitions in mind we can state the general formulation of Slater's conditions:

Definition 24 (Slater's conditions) *Let $(\mathcal{P}_C)$ be a convex problem with objective function f and constraints $g_1, ..., g_{n_g}, h_1, ..., h_{n_h}$. Define the domain D as:*

$$D = \text{dom}(f) \cap \left(\bigcap_{k=1}^{n_g} \text{dom}(g_k) \right)$$

Then Slater's conditions *are:*

$$\begin{aligned} \exists\, \mathbf{x}^* \in \text{relint}(D) : \forall\, k \in \{1, \cdots, n_g\},\ g_k(\mathbf{x}^*) < 0 \\ \forall\, k \in \{1, \cdots, n_h\},\ h_k(\mathbf{x}^*) = 0 \end{aligned}$$

If there exists $n_g^{(0)} \in \mathbb{N}$ such that for all $k \leq n_g^{(0)}$ the function g_k is linear, then the conditions become:

$$\begin{aligned} \exists\, \mathbf{x}^* \in \text{relint}(D) : \forall\, k \in \{1, \cdots, n_g^{(0)}\},\ g_k(\mathbf{x}^*) \leq 0 \\ \forall\, k \in \{n_g^{(0)} + 1, \cdots, n_g\},\ g_k(\mathbf{x}^*) < 0 \\ \forall\, k \in \{1, \cdots, n_h\},\ h_k(\mathbf{x}^*) = 0 \end{aligned}$$

Now consider the SVM problem. There are no equality constraints h_1, ..., h_{n_h} and all inequality constraints $g_1, ..., g_{n_g}$ are linear, hence Slater's conditions simplify to the existence of $\mathbf{x}^* \in \text{relint}(D)$ such that $\mathbf{x}^*$ is feasible. The optimization variable for the SVM is $\mathbf{x} = (\mathbf{w}, b, \boldsymbol{\xi})$; it is rather clear that the SVM objective function as well as the two inequality constraints are defined over $\mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^n$ – where we remind that p designates the number of attributes and n the number of observations – hence: $D = \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^n$.

Now, letting $q \in \mathbb{N}$, it is also clear that: $\text{aff}(\mathbb{R}^q) = \mathbb{R}^q$ – from the definition of an affine hull, it suffices to take $m = 1$ and $\lambda \equiv \lambda_1 = 1$, then every element $\mathbf{x}$ of $\mathbb{R}^q$ is equal to $\lambda\mathbf{x}$. Subsequently, from the definition of the relative interior:

$$\text{relint}(\mathbb{R}^q) = \left\{ \mathbf{x} : \mathbf{x} \in \mathbb{R}^q,\ \exists\, \epsilon \in \mathbb{R}_+^*,\ N_\epsilon(\mathbf{x}) \cap \mathbb{R}^q \subseteq \mathbb{R}^q \right\}$$

Given that for any $\mathbf{x} \in \mathbb{R}^q$ any ball $N_\epsilon(\mathbf{x})$, $\epsilon \in \mathbb{R}_+^*$, is in $\mathbb{R}^q$, we can conclude that: $\text{relint}(\mathbb{R}^q) = \mathbb{R}^q$. Finally, letting S_1, S_2 be two sets, relative interiors have the following property:

Proposition 1 *Let S_1 and S_2 be two sets, then:*

$$\text{relint}(S_1 \times S_2) = \text{relint}(S_1) \times \text{relint}(S_2)$$

From this reasoning we derive a simplified technical condition (Boyd and Vandenberghe, 2009):

Definition 25 (simplified Slater condition for SVM) *Let $(\mathcal{P}_C)$ be a convex problem. Assume the following:*

- *The problem's inequality functions $g_1, ..., g_n$ are all linear;*
- *The domain of definition of the objective function* $\text{dom}(f)$ *is open.*

Then the Slater condition *simplifies to feasibility:*

$$O \neq \emptyset$$

With the above notions established, the below theorem gives a justification for dealing with the dual representation of a convex optimization program.

Theorem 6 (Strong duality of a convex problem) *Strong duality holds for a convex program $(\mathcal{P}_C)$ if Slater's conditions holds.*

Appendix B

Mathematical analysis of the SVM

This appendix constitutes a thorough mathematical analysis of the properties of the Support Vector Machine. We will deal with 3 topics:

- The SVM optimization program, using the results expounded in appendix A, and representation of the SVM optimal solution;
- Characterisation of unique solutions to the SVM problem; and
- The Vapnik-Chervonenkis dimension of the SVM

B.1 Study of the SVM optimization program

We start this section by formalising the SVM problem, stating the KKT conditions and going through the main properties of the SVM solution. In the following two subsections we derive the simplified dual formulation and the SVM decision function respectively.

B.1.1 Fundamental properties of the SVM problem

The Support Vector Machine optimization problem with soft margins and no feature mapping is the following – in the case where we introduce some feature mapping ϕ, the derivations are the same replacing $\mathbf{x}_i$ by $\phi(\mathbf{x}_i)$ and inner products $\mathbf{x}_i^T\mathbf{x}_j$ by kernel values $K(\mathbf{x}_i, \mathbf{x}_j)$:

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \right) \\ &\text{s.t. } \forall i \in \{1,...,n\} \ : \ y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i \\ &\qquad\qquad\qquad\qquad \xi_i \geq 0 \end{aligned}$$

Let $\mathcal{L}_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}} \equiv \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, such that $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are the Lagrangian coefficients for the SVM optimization. The corresponding Lagrangian function is:

$$\mathcal{L}_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) - \sum_{i=1}^{n}\beta_i\xi_i$$

The primal formulation of the optimization program is:

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \right) \\ &\text{s.t. } \forall i \in \{1,...,n\} \ : \ \alpha_i \geq 0 \\ &\qquad\qquad\qquad\qquad \beta_i \geq 0 \end{aligned}$$

Alternatively, by defining the function $\theta_P(\mathbf{w}, b, \boldsymbol{\xi})$ as follows

$$\theta_P(\mathbf{w}, b, \boldsymbol{\xi}) \equiv \max_{\boldsymbol{\alpha},\boldsymbol{\beta}\,:\,\forall\, i,\, \alpha_i,\, \beta_i \geq 0} \mathcal{L}\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right)$$

We can give a cleaner version of the primal problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \theta_P(\mathbf{w}, b, \boldsymbol{\xi})$$

The SVM problem can also be restated in its dual formulation:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left(\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{L}\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right) \right)$$
$$\text{s.t. } \forall i \in \{1, ..., n\} \;:\; \alpha_i \geq 0$$
$$\beta_i \geq 0$$

Again, we can define a function $\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$:

$$\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) \equiv \min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{L}\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right)$$

$\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the objective function for the dual formulation of the SVM program, which consists on maximizing $\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$$
$$\text{s.t. } \forall i \in \{1, ..., n\} \;:\; \alpha_i \geq 0$$
$$\beta_i \geq 0$$

Let $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$ be a solution to the SVM primal problem and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ a solution to the SVM dual problem. The KKT conditions for the primal and dual formulations are:

1. *Primal feasibility*:

$$\forall i \in \{1, ..., n\} :\; y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^* \right) \geq 1 - \xi_i^*$$
$$\forall i \in \{1, ..., n\} :\; \xi_i^* \geq 0$$

2. *Dual feasibility*:

$$\forall i \in \{1, ..., n\} :\; \alpha_i^* \geq 0$$
$$\forall i \in \{1, ..., n\} :\; \beta_i^* \geq 0$$

3. *Complementary slackness*:

$$\forall i \in \{1, ..., n\} :\; \alpha_i^* \left(1 - \xi_i^* - y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^* \right) \right) = 0$$
$$\forall i \in \{1, ..., n\} :\; \beta_i^* \xi_i^* = 0$$

4. *Lagrangian stationarity*:

$$\nabla_{\mathbf{w}} \mathcal{L}\left(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\right) = 0$$
$$\nabla_b \mathcal{L}\left(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\right) = 0$$
$$\nabla_{\boldsymbol{\xi}} \mathcal{L}\left(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\right) = 0$$

Now that we have fully expounded the SVM optimization problem, we can formulate its most critical properties – the interested reader can consult (Lin, 2001a) for additional details and results for the SVM problem:

- **Globality of the solution**: as it has already been discussed, the original objective function of the SVM is convex in all its variables. Letting $\mathbf{e}_i \in \mathbb{R}^n$ be the vector where the i^{th} element is equal to 1 and all other elements equal to 0, the original inequality constraints can be rewritten as follows:

$$g_k(\mathbf{w}, b, \boldsymbol{\xi}) \leq 0, \ k \in \{1, 2\}$$

  where:

$$g_1(\mathbf{w}, b, \boldsymbol{\xi}) = 1 - \left(\boldsymbol{\xi}^T \mathbf{e}_i + y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right)\right)$$
$$g_2(\mathbf{w}, b, \boldsymbol{\xi}) = -\boldsymbol{\xi}^T \mathbf{e}_i$$

  Functions g_k, $k \in \{1, 2\}$, are linear in all variables, hence convex. As a consequence, any solution to the SVM which is a local minimum is also a global minimum: any solution is a global one.

- **Existence of the solution**: For a simple SVM with $C = 0$, the condition for existence is obviously that the data is linearly separable. Furthermore, theorem 2.2 from (Lin, 2001a) establishes that for a soft-margin SVM such that $C > 0$ the SVM program is always solvable, i.e. there is at least one solution.

- **Necessity and sufficiency of KKT conditions**: the original objective function of the SVM is convex and continuously differentiable with respect to all variables. The original inequality constraints are linear and continuously differentiable with respect to all variables. Therefore a quintuplet $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is a solution to the SVM program if and only if it satisfies the KKT conditions.

- **Strong duality**: we have mentioned that the SVM objective function is convex. Furthermore, the SVM's inequality constraints are all linear with respect to all variables and the domain of definition D of the primal's objective function is open because: $D = \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^n$, as we showed earlier. Under these conditions, Slater's constraint qualifications conditions are trivially satisfied, which ensures that the SVM program is strongly dual: the solution of the primal problem and the dual problem are equal, hence we can solve either to determine the optimal hyperplane – the reader can consult for example (Erästö, 2001) for more details.

B.1.2 Derivation of the simplified dual formulation

We derive a simplified version of the dual formulation of the SVM optimization program. The goal is to obtain an expression of $\mathcal{L}$ which only depends on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. To achieve this, we use the necessity and sufficiency property of the KKT conditions on any solution to the SVM problem to rewrite the Lagrangian function.

Lagrangian stationarity allow us to obtain an expression for $\mathbf{w}$:

$$\begin{aligned} &\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \\ &\Leftrightarrow \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \\ &\Leftrightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \end{aligned} \tag{B.1}$$

Furthermore, the rest of the KKT stationarity conditions imply:

$$\begin{aligned}
&\nabla_b \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \\
&\Leftrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0
\end{aligned} \tag{B.2}$$

$$\begin{aligned}
&\nabla_{\boldsymbol{\xi}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \\
&\Leftrightarrow C - \alpha_i - \beta_i = 0 \\
&\Leftrightarrow \alpha_i + \beta_i = C
\end{aligned} \tag{B.3}$$

These conditions help us to rewrite the dual objective – we inject sequentially equations B.2, B.3 and B.1:

$$\begin{aligned}
\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(\mathbf{w}^T\mathbf{x}_i\right)\right) - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i\right)\right) \\
&= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\alpha_i - \|\mathbf{w}\|^2 \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j
\end{aligned}$$

We have derived an expression of the Lagrangian function which does not depend upon $\mathbf{w}$, b or $\boldsymbol{\xi}$. As a consequence, given that $\mathbf{w}$ can be written in terms of $\alpha_1, ..., \alpha_n$, b and $\boldsymbol{\xi}$ can be setted such that the KKT conditions are respected. The remaining constraints that depend on $\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_n$ are:

$$\begin{aligned}
&\forall i \in \{1, ..., n\} : \ \alpha_i \geq 0 \\
&\forall i \in \{1, ..., n\} : \ \beta_i \geq 0 \\
&\forall i \in \{1, ..., n\} : \ \alpha_i + \beta_i = C \\
&\sum_{i=1}^{n}\alpha_i y_i = 0
\end{aligned}$$

However the third line can be rewritten: $\beta_i = C - \alpha_i$. Then by enlarging the first constraint to:

$$\forall i \in \{1, ..., n\} : \ 0 \leq \alpha_i \leq C$$

We will be sure that for all i, $\beta_i \geq 0$: once all the α's have been determined, it suffices to set β_i equal to $C - \alpha_i$ for all $i \in \{1, ..., n\}$.

We can now state the dual formulation of the SVM program:

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j {\mathbf{x}_i}^T \mathbf{x}_j \right)$$

$$\text{s.t. } \forall i \in \{1, ..., n\} \ : \ 0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

B.1.3 Derivation of the decision function and the slack variables

Let $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ be a solution to the SVM. In the previous subsection we already computed the solution vector $\mathbf{w}^*$ in equation A.1:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

Our optimal decision function f^* can be expressed as follows:

$$\forall\, \mathbf{x}_{\text{new}} \in \mathbb{R}^p,\ f^*(\mathbf{x}_{\text{new}}) = \sum_{i=1}^{n} \alpha_i^* y_i {\mathbf{x}_i}^T \mathbf{x}_{\text{new}} + b^*$$

The function can be made even simpler. Let consider again the complementarity slackness conditions:

$$\forall i \in \{1, ..., n\} : \ \alpha_i^* \left(1 - \xi_i^* - y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^*\right)\right) = 0$$

$$\forall i \in \{1, ..., n\} : \ \beta_i^* \xi_i^* = 0$$

If $\alpha_i^* = C$ then, given that $\alpha_i^* + \beta_i^* = C$, we must have $\beta_i^* = 0$, hence:

$$\alpha_i^* = C \ \Rightarrow \ \xi_i^* \geq 0$$

Using the first KKT complementarity condition we derive the following property:

$$\begin{aligned} \alpha_i^* = C \ &\Rightarrow \ y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^*\right) = 1 - \xi_i^* \\ &\Rightarrow \ y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^*\right) \leq 1 \\ &\Rightarrow \ y_i f^*(\mathbf{x}_i) \leq 1 \end{aligned}$$

Similarly, if $0 < \alpha_i^* < C$, it must be the case that $\xi_i^* = 0$. From this we conclude that:

$$\begin{aligned} 0 < \alpha_i^* < C \ &\Rightarrow \ y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^*\right) = 1 \\ &\Rightarrow \ y_i f^*(\mathbf{x}_i) = 1 \end{aligned}$$

However, in the final case where $\alpha_i^* = 0$, we must also have $\xi_i^* = 0$ because $\beta_i^* = C$. By primal feasibility:

$$\begin{aligned} & y_i \left({\mathbf{w}^*}^T \mathbf{x}_i + b^* \right) \geq 1 \\ \Leftrightarrow\ & y_i f^*(\mathbf{x}_i) \geq 1 \\ \Leftrightarrow\ & \frac{y_i f^*(\mathbf{x}_i)}{\|\mathbf{w}^*\|} \geq \frac{1}{\|\mathbf{w}^*\|} \\ \Leftrightarrow\ & \frac{y_i f^*(\mathbf{x}_i)}{\|\mathbf{w}^*\|} \geq m^* \end{aligned}$$

Thus in case $\alpha_i^* = 0$ the scaled *functional margin* $y_i f^*(\mathbf{x}_i)$ of vector $\mathbf{x}_i$ is higher than or equal to the SVM margin – hence the data point is correctly classified.

The classification function of the SVM only depends on the instances from the training set for which $0 < \alpha_i^* \leq C$ – i.e. the points which are either on the margin or on the wrong side of the margin, either correctly classified or misclassified, which we have called the *support vectors*. Let $\{\mathbf{x}_1^{(SV)}, ..., \mathbf{x}_{N_{SV}}^{(SV)}\}$ be the set of support vectors with N_{SV} the number of support vectors and let the values y_i, α_i associated to them be $y_i^{(SV)}, \alpha_i^{(SV)}$. Then for all $\mathbf{x}_{\text{new}} \in \mathbb{R}^p$:

$$f(\mathbf{x}_{\text{new}}) = \sum_{i=1}^{N_{SV}} \alpha_i^{(SV)} y_i^{(SV)} {\mathbf{x}_i^{(SV)}}^T \mathbf{x}_{\text{new}} + b^*$$

We still have to compute the optimal value of b. From the complementarity slackness condition again, we get for some i_0 for which α_{i_0} is equal to 0:

$$\begin{aligned} & \alpha_{i_0}^* = 0 \Rightarrow 1 - \xi_{i_0}^* - y_{i_0} \left({\mathbf{w}^*}^T \mathbf{x}_{i_0} + b^* \right) = 0 \\ \Leftrightarrow\ & {\mathbf{w}^*}^T \mathbf{x}_{i_0} + b^* = y_{i_0} \left(1 - \xi_{i_0}^* \right) \\ \Leftrightarrow\ & b^* = y_{i_0} \left(1 - \xi_{i_0}^* \right) - {\mathbf{w}^*}^T \mathbf{x}_{i_0} \\ \Leftrightarrow\ & b^* = y_{i_0} \left(1 - \xi_{i_0}^* \right) - \sum_{j=1}^{n} \alpha_j^* y_j \mathbf{x}_j^T \mathbf{x}_{i_0} \end{aligned}$$

Finally, given $\alpha_{i_0}^*$ is null, we must have $\xi_{i_0}^* = 0$. The optimal intercept simplifies to:

$$b^* = y_{i_0} - \sum_{j=1}^{n} \alpha_j^* y_j \mathbf{x}_j^T \mathbf{x}_{i_0}$$

Let b_{i_0} be the above value for coefficient b^* and let $I_0 = \{i : \alpha_i^* = 0\}$ be the set of subscripts for which a value b_i can be computed as above. (Burges, 1998) suggests to take the following average value for b^*:

$$b^* = \frac{1}{|I_0|} \sum_{i \in I_0} b_i$$

To end our discussion on the SVM, we give the formula for determining the values of the slack variables – recall that they do not appear on the dual problem, and notice that the above value b^*

does not depend on them either. Using the two primal feasibility constraints:

$$\begin{aligned}\forall i \in \{1, ..., n\}\,, y_i f^*(\mathbf{x}_i) \geq 1 - \xi_i^* \\ \Leftrightarrow\ \xi_i^* \geq 1 - y_i f^*(\mathbf{x}_i) \\ \Leftrightarrow\ \xi_i^* = \max\left[0, 1 - y_i f^*(\mathbf{x}_i)\right]\end{aligned}$$

The reader might notice that the optimal value for ξ_i corresponds to the hinge loss which allows to state the SVM optimization program in its regularized version.

B.1.4 Characterisations of SVM solutions

Building on our previous findings, we now give a simplified characterisation of the SVM solution. These characterizations are convenient from an algorithmic perspective and there exist SVM solvers which explicitly utilize them.

We recall all KKT conditions for all $i \in \{1, ..., n\}$ – stationarity conditions have been rewritten:

$$y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - (1 - \xi_i) \geq 0 \tag{B.4}$$

$$\xi_i \geq 0 \tag{B.5}$$

$$\alpha_i \geq 0 \tag{B.6}$$

$$\beta_i \geq 0 \tag{B.7}$$

$$\alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - (1 - \xi_i)\right) = 0 \tag{B.8}$$

$$\beta_i \xi_i = 0 \tag{B.9}$$

$$\alpha_i + \beta_i = C \tag{B.10}$$

$$\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j = \mathbf{w} \tag{B.11}$$

$$\sum_{j=1}^{n} \alpha_j y_j = 0 \tag{B.12}$$

We show that the verification of these conditions might be simplified:

- The optimal weight vector $\mathbf{w}$ is determined through equation A.11, hence it suffices to define the weight vector that way to have condition A.11 verified for some $\boldsymbol{\alpha}^*$.
- As we already explained, A.6, A.7 and A.10 can be combined in the following concise form:

  $$0 \leq \alpha_i \leq C$$

  It then suffices to define β_i as $C - \alpha_i$ to have the 3 conditions automatically fulfilled.
- If $0 \leq \alpha_i < C$, then by A.9 and the 2^{nd} condition above we must have A.4 with $\xi_i = 0$.
- Equivalently, if $\alpha_i = C$, then by A.9 and the 2^{nd} condition above we must have A.4 with $\xi_i \leq 0$.
- Condition A.12 must be verified.

Hence to verify KKT conditions we must only verify the following 4 conditions for all $i \in \{1, ..., n\}$:

$$0 \leq \alpha_i \leq C \tag{B.13}$$

$$0 \leq \alpha_i < C \Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1 \geq 0, \ \xi_i = 0 \tag{B.14}$$

$$\alpha_i = C \Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - (1 - \xi_i) \geq 0, \ \xi_i \geq 0 \tag{B.15}$$

$$\sum_{j=1}^{n} \alpha_j y_j = 0 \tag{B.16}$$

For any vector $\boldsymbol{\alpha}^* = (\alpha_1^*, ..., \alpha_n^*)$ that verifies them, $\mathbf{w}$ and $\boldsymbol{\beta}$ are determined as explained above; b and $\boldsymbol{\xi}$ are constructed such that conditions B.13 through B.16 are satisfied.

Given that the KKT conditions are necessary and sufficient for a solution, any vector $\boldsymbol{\alpha}^*$ satisfying them is a solution to the SVM. Hence a SVM solver algorithm can be designed that seeks to find vectors that fulfil the conditions above.

B.2 Uniqueness of the SVM solution

As has been extensively discussed both in appendix A and in the 1st part of this appendix, we are guaranteed that the solution of the SVM is global. However, there is no optimization theorem that guarantees the *uniqueness* of this solution. We will now briefly discuss this matter both in terms of primal and dual solution.

B.2.1 Uniqueness of the primal solution

First and foremost, we are interested in the uniqueness of the optimal weights vector $\mathbf{w}^*$. To be unique, the objective function must be strictly convex, which would be true in the separable case as the term $C \sum_{i=1}^{n} \xi_i$ would be equal to 0 and the function $1/2\|\mathbf{w}\|^2$ is strictly convex. However, note that loose convexity – i.e. the function is not strictly convex – does not imply that the solution is not unique, but that there must be a case by case analysis.

Burges and Crisp (2000) derive a necessary and sufficient condition for the solution of a soft margin SVM problem to be unique, independently of the strict or loose convexity of the objective function. The unique solution is characterized by the cardinality of sets of support vectors. Let define the following sets of support vectors:

$$\mathcal{N}_1 = \{i : y_i = 1 \ , \ f(\mathbf{x}_i) < 1\}$$

$$\mathcal{N}_2 = \{i : y_i = -1 \ , \ f(\mathbf{x}_i) > -1\}$$

$$\mathcal{N}_3 = \{i : y_i = 1 \ , \ f(\mathbf{x}_i) = 1\}$$

$$\mathcal{N}_4 = \{i : y_i = -1 \ , \ f(\mathbf{x}_i) = -1\}$$

Vectors from $SV_{\mathbf{x}_i \, : \, i \in \mathcal{N}_1 \cup \mathcal{N}_2}$ lie on the margin or are sharply misclassified. Vectors from $SV_{\mathbf{x}_i \, : \, i \in \mathcal{N}_3 \cup \mathcal{N}_4}$ lie on the boundary of the margin.

The set of all support vectors is simply the union $SV_{\mathbf{x}_i \, : \, i \in \mathcal{N}_1 \cup \mathcal{N}_2 \cup \mathcal{N}_3 \cup \mathcal{N}_4}$. We have the following theorem:

Theorem 7 (Uniqueness of soft margin SVM solution) *The solution to the soft margin SVM problem is not unique if and only if at least one of the following two conditions hold:*

$$|\mathcal{N}_2 \cup \mathcal{N}_4| = |\mathcal{N}_1|$$

$$|\mathcal{N}_1 \cup \mathcal{N}_3| = |\mathcal{N}_2|$$

Furthermore, all support vectors have their Lagrange multiplier satisfying: $\alpha_i = C$. Finally:

- *If the 1^{st} condition holds, then: $\mathcal{N}_3 = \emptyset$.*
- *If the 2^{nd} condition holds, then: $\mathcal{N}_4 = \emptyset$.*

For example the first condition states that the total number of negatively-labelled support vectors must be equal to the number of positively-labelled bounded support vectors to have more than one solution. Hence, intuitively we might think that if we are dealing with a very large dataset which is not separable, chances are that the solution will be unique.

The following corollary can be derived from the theorem by noticing that it implies:

$$|\mathcal{N}_2 \cup \mathcal{N}_4| = |\mathcal{N}_1| \ \Rightarrow \mathcal{N}_3 = \emptyset$$

$$|\mathcal{N}_1 \cup \mathcal{N}_3| = |\mathcal{N}_2| \ \Rightarrow \mathcal{N}_4 = \emptyset$$

Corollary 1 *For any solution which is not unique, letting I_{SV} denote the set of indices of the corresponding set of support vectors, then we must have:*

$$\sum_{i \in I_{SV}} y_i = 0$$

This corollary gives us a necessary condition for a solution to the SVM problem to be non unique: the number of positively-labelled and negatively-labelled support vectors is equal.

B.2.2 Uniqueness of the dual solution

The other side of the uniqueness question relates to the representation of the optimal solution $\mathbf{w}^*$ in terms of the Lagrangian coefficients $\alpha_1^*, ..., \alpha_n^*$:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

It turns out that the above expansion of $\mathbf{w}^*$ might not be unique, even if $\mathbf{w}^*$ is. Consider the following toy example (Burges, 1998):

- We have four separable points $\mathbf{x}_1 = (1,1)$, $\mathbf{x}_2 = (-1,1)$, $\mathbf{x}_3 = (-1,-1)$ and $\mathbf{x}_4 = (1,-1)$ such that $y_1 = 1$, $y_2 = -1$, $y_3 = -1$ and $y_4 = 1$. The SVM solution to this problem is $\mathbf{w}^* = (1,0)$, $b^* = 0$ so that we only consider the first element of vectors $\mathbf{x}$.
- However, the expansion of $\mathbf{w}^*$ might be written with different vectors, for example $\boldsymbol{\alpha}^* = (0.25, 0.25, 0.25, 0.25)$, $\boldsymbol{\alpha}^* = (0.5, 0.5, 0, 0)$ or $\boldsymbol{\alpha}^* = (0, 0, 0.5, 0.5)$.

Burges (1998) gives a characterization of the case for which there are multiple expansions for the optimal solution $\mathbf{w}^*$: let $\boldsymbol{\alpha}^*$ be some optimal solution for the dual formulation; let $\boldsymbol{\alpha}' \in \mathbb{R}^n$ be some vector with the following characteristics:

1. $\forall\,(i,j) \in \{1,...,n\} \times \{1,...,n\}$, $\boldsymbol{\alpha}'^T \left(y_i y_j \mathbf{x}_i^T \mathbf{x}_j\right) = 0$

2. $\forall\, i \in \{1, ..., n\}\ ,\ 0 \leq \alpha_i^* + \alpha_i' \leq C$
3. $\sum_{l=1}^{n} \alpha_l'\, y_l = 0$
4. $\sum_{l=1}^{n} \alpha_l' = 0$

Then $\boldsymbol{\alpha}^* + \boldsymbol{\alpha}'$ is also a solution to the SVM problem. Hence, even if the solution $\mathbf{w}^*$ is unique it might have multiple representations in terms of Lagrangian coefficients $\boldsymbol{\alpha}^{*(1)}$, $\boldsymbol{\alpha}^{*(2)}$, ...

B.3 The VC dimension of Support Vector Machines

To finish off appendix B, we present learning theory related results for the support vector machine. Precisely, we focus on the Vapnik-Chervonenkis dimension of different types of SVM: we recall that the VC dimension of a model is interpreted as an indication of its complexity and hence of its generalization ability. These theorems should serve as a basis to discuss the following topics:

- The relationship between the data structure and the model's complexity;
- The limitations of the VC dimension as an indicator of models with good predictive power.

We will not describe or go through any of the proof for the theorems below. The interested reader might consult Burges (1998), where a proof is given for each one of these theorems.

The 1st theorem is a general result – in the sense that it is not only applicable to support vector machines but also to other linear models – concerning separating hyperplanes.

Theorem 8 (VC dimension of separating hyperplanes) *The VC dimension of the set of separating hyperplanes in $\mathbb{R}^p$ is $p + 1$.*

Alternatively, given that a linear SVM is simply a separating hyperplane defined by equation $\mathbf{w}^T\mathbf{x} + b$ with $\mathbf{w} \in \mathbb{R}^p$, we can reformulate the theorem as follows: "*the VC dimension of a linear SVM in $\mathbb{R}^p$ is $p + 1$*". Hence the VC dimension of a linear SVM is determined by the structure of the data: the greater the number of attributes the data has, the greater the dimension of the weight vector $\mathbf{w}$ and hence the greater the VC dimension of the SVM.

There is something undesirable about this property: the data structure seems to influence the generalization ability of the model, hence the more complex a problem is the worse the learning capacity of the model is supposed to be. Although it is understandable that more complex problems might be harder to generalize than simpler ones, some problems might involve data that has naturally more dimensions – i.e. attributes – and this should not have any direct relation to our ability to construct a linear model that accurately predicts labels outside the training data set. Going further, the theorem might seem even odder if we think as follows: attributes inform us about the nature of the data $\mathbf{x}_i$ so the more we have the better our model should be at predicting the labels y_i and hence generalize – or at least more attributes should not deteriorate its performance. It might seem like the VC dimension resembles more a metric to measure the complexity of a class of models $\mathcal{F}$ than a true representation of the generalization ability of $\mathcal{F}$.

Now that we have stated the theorem about linear SVM, we will look at a more general result that is a consequence of the previous one. For that, let first make some definitions.

We consider a general SVM equipped with a valid kernel K – we recall that a valid kernel is one for which the associated matrix $(K_{i,j})_{0 \leq i,j \leq n}$ is positive semi-definite – where K can be the linear kernel. The data at our disposal $\mathbf{x}_1, ..., \mathbf{x}_n$ comes from a space $\mathcal{L}$. We now define the following set:

$$\Phi(K) = \{\phi : K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle,\ (\mathbf{x}, \mathbf{y}) \in \mathcal{L} \times \mathcal{L},\ \phi : \mathcal{L} \to \mathcal{H}\}$$

As we illustrated before for $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2$, a given kernel K can have multiple mappings associated to it. The set $\Phi(K)$ is simply the set of all mappings implicitly defined by K. Letting $\Phi_K \equiv \Phi(K)$, we now define:

$$H(\Phi_K) = \{\mathcal{H} : \phi \in \Phi_K\}$$

$H(\Phi_K)$ is the set of higher-dimensional spaces $\mathcal{H}$ "generated" by Φ_K – each mapping ϕ is associated to a space $\mathcal{H}$; $H(\Phi_K)$ is simply the set of all these spaces. Finally, we define the following quantity:

$$d_K^{\min} = \min_{\mathcal{H} \in H(\Phi_K)} dim(\mathcal{H})$$

$d_K^{\min}$ is the dimension of the space $\mathcal{H} \in H(\Phi_K)$ with minimal dimension.

Given that a kernelized SVM can be viewed as a separating hyperplane in a high dimensional space $\mathcal{H}$, we can state the following theorem. Following Burges and for the sake of clarity, we will call elements from $H(\Phi_K)$ "*embedding spaces*" and "*minimal embedding space*" the embedding space associated to $d_K^{\min}$.

Theorem 9 (VC dimension of a SVM) *Let K be a positive semi-definite kernel which corresponds to a minimal embedding space $\mathcal{H}^{\min}$. Assuming the error penalty C in the SVM optimization program is allowed to take all possible values in $\mathbb{R}$, the VC dimension of the soft-margin SVM equipped with K is $d_K^{\min} + 1$.*

Although of little practical use due to the potential complexity of embedding spaces, this theorem shows again the relationship between the VC dimension of the SVM and the number of variables used for predicting the label: embedding spaces correspond to feature spaces into which we have projected the data through the mapping ϕ, hence the VC dimension of the SVM is now determined by the number of features – instead of the number of attributes. As we already explained, the structure of the learning problem, the available data and the mappings we apply to it seem to all have a decisive impact on the complexity of our model in terms of VC dimension.

We now state a final theorem – which we have already mentioned in our discussion about kernels – which should help us to continue the discussion on the relevance of VC dimension.

Theorem 10 (VC dimension of a SVM with Gaussian kernel) *Let K_G be the Gaussian kernel. Assuming the error penalty C in the SVM optimization program is allowed to take all possible values in $\mathbb{R}$, the soft-margin SVM equipped with K_G has infinite VC dimension.*

More precisely, a SVM equipped with a Gaussian kernel can correctly classify all given points provided the bandwidth σ is small enough (Burges, 1998; Keerthi and Lin, 2003). This property poses an interesting question: how is it possible that, despite having an infinite VC dimension, support vector machines with Gaussian kernels have a track record of excellent performance (Burges, 1998), as we already mentioned earlier? This ties back to our discussion: it seems the VC dimension metric has shortcomings in measuring the generalization capacity of a model or a family of models and is more appropriate as a number to precisely quantity the complexity of the model class.

Although the above theorems on the VC dimension of certain support vector machines are not particularly encouraging, researchers have been trying to derive stronger results, particularly in the

form of generalization bounds. We will not present them here as most of them require to define a complex class of classifiers named *Gap Tolerant Classifiers*, to which the SVM can be tied, but the interested reader can consult Burges (1998) and Erästö (2001) for an overview of some of these bounds.

Overall, the above discussion shows that, although SVM have been showed to perform well in many real learning problems, there does not seem to be a strong theoretical foundation from VC theory that can explain the SVM's strong empirical results.

Appendix C

The Sequential Minimal Optimization algorithm

C.1 Derivation of the updated values of the working set

We recall the dual objective function, assuming we are working with a kernel K:

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Maximizing this function with respect to $\boldsymbol{\alpha}$ is equivalent to minimizing the function:

$$\mathcal{L}^{-}(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i$$

Let define $K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. This last function might be rewritten in terms of $(\alpha_1, \alpha_2, \alpha_3^{\text{old}}, ..., \alpha_n^{\text{old}})$, keeping alphas from 3 to n fixed with their old values:

$$\begin{aligned}
\mathcal{L}^{-}(\alpha_1, \alpha_2) = & \frac{1}{2}\alpha_1^2 K_{1,1} + \frac{1}{2}\alpha_2^2 K_{2,2} + \alpha_1 \alpha_2 y_1 y_2 K_{1,2} \\
& + \alpha_1 y_1 \sum_{i=3}^{n} \alpha_i^{\text{old}} y_i K_{1,i} + \alpha_2 y_2 \sum_{i=3}^{n} \alpha_i^{\text{old}} y_i K_{2,i} \\
& - \alpha_1 - \alpha_2 + \text{constant}
\end{aligned}$$

Now recall that we had the following constraint:

$$\alpha_1 = y_1(\zeta - \alpha_2 y_2)$$

Let define $\nu_j = \sum_{i=3}^{n} \alpha_i^{\text{old}} y_i K_{j,i}$. The objective function can be again reformulated as a one-variable function – note that for all i, $y_i^2 = 1$:

$$\begin{aligned}
\mathcal{L}^{-}(\alpha_2) = & \frac{1}{2}(\zeta - \alpha_2 y_2)^2 K_{1,1} + \frac{1}{2}\alpha_2^2 K_{2,2} + \alpha_2 y_2 (\zeta - \alpha_2 y_2) K_{1,2} \\
& + (\zeta - \alpha_2 y_2)\nu_1 + \alpha_2 y_2 \nu_2 \\
& - y_1(\zeta - \alpha_2 y_2) - \alpha_2 + \text{constant}
\end{aligned}$$

Differentiating with respect to α_2 yields:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}^-(\alpha_2)}{\mathrm{d}\alpha_2} =& -y_2(\zeta - \alpha_2 y_2)K_{1,1} + \alpha_2 K_{2,2} + y_2\zeta K_{1,2} - 2\alpha_2 K_{1,2} \\ & - y_2\nu_1 + y_2\nu_2 \\ & + y_1 y_2 - 1\end{aligned}$$

We differentiate a 2^{nd} time to check for convexity:

$$\frac{\mathrm{d}^2\mathcal{L}^-(\alpha_2)}{\mathrm{d}{\alpha_2}^2} = K_{1,1} + K_{2,2} - 2K_{1,2}$$

Considering that kernels express similarity between vectors, the above expression should be non-negative under normal circumstances, which implies convexity. Hence the minimum of $\mathcal{L}^-$ is reached when equating to 0 the first derivative:

$$\alpha_2\left(K_{1,1} + K_{2,2} - 2K_{1,2}\right) = y_2\zeta\left(K_{1,1} - K_{1,2}\right) + y_2(\nu_1 - \nu_2) + 1 - y_1 y_2$$

Now, by construction we must have:

$$\zeta = \alpha_1^{\text{old}} y_1 + \alpha_2^{\text{old}} y_2$$

Moreover ν_j can be expanded as follows:

$$\nu_j = f(\mathbf{x}_j) - b^{\text{old}} - \alpha_1^{\text{old}} y_1 K_{1,j} - \alpha_2^{\text{old}} y_2 K_{2,j}$$

Hence by replacing ζ, ν_1 and ν_2 by their values and using the fact that kernels are symmetric:

$$\begin{aligned}\alpha_2\left(K_{1,1} + K_{2,2} - 2K_{1,2}\right) &= y_2\zeta\left(K_{1,1} - K_{1,2}\right) + y_2(\nu_1 - \nu_2) + 1 - y_1 y_2 \\ &= y_2\left(\alpha_1^{\text{old}} y_1 + \alpha_2^{\text{old}} y_2\right)\left(K_{1,1} - K_{1,2}\right) \\ &\quad y_2(f(\mathbf{x}_1) - \alpha_1^{\text{old}} y_1 K_{1,1} - \alpha_2^{\text{old}} y_2 K_{2,1} \\ &\quad - f(\mathbf{x}_2) + \alpha_1^{\text{old}} y_1 K_{1,2} + \alpha_2^{\text{old}} y_2 K_{2,2}) \\ &\quad + 1 - y_1 y_2 \\ &= \alpha_2^{\text{old}}\left(K_{1,1} + K_{2,2} - 2K_{1,2}\right) \\ &\quad + y_2\left(f(\mathbf{x}_1) - f(\mathbf{x}_2) + y_2 - y_1\right)\end{aligned}$$

We end up obtaining:

$$\begin{aligned}\alpha_2^{\text{new}} &= \alpha_2^{\text{old}} + y_2\frac{f(\mathbf{x}_1) - f(\mathbf{x}_2) + y_2 - y_1}{K_{1,1} + K_{2,2} - 2K_{1,2}} \\ &= \alpha^{\text{old}} + y_2\frac{(f(\mathbf{x}_1) - y_1) - (f(\mathbf{x}_2) - y_2)}{\eta}\end{aligned}$$

This last expression is the one we have presented in the main text.

To obtain the value of $\alpha_1^{\text{new, clipped}}$, we just use the fact that:

$$\zeta = \alpha_1^{\text{new, clipped}} y_1 + \alpha_2^{\text{new, clipped}} y_2$$

Hence:

$$\alpha_1^{\text{new, clipped}} = \alpha_1^{\text{old}} + y_1 y_2 \left(\alpha_2^{\text{old}} - \alpha_2^{\text{new, clipped}} \right)$$

C.2 Derivation of the bounds L and H

Let assume that $y_1 = y_2$. Then by construction we have:

$$\begin{aligned} \zeta &= \alpha_1^{\text{old}} + \alpha_2^{\text{old}} \\ &= \alpha_1^{\text{new, clipped}} + \alpha_2^{\text{new, clipped}} \end{aligned}$$

Box constraints imply that:

$$0 \leq \alpha_1^{\text{new, clipped}} \leq C$$

It comes:

$$\begin{array}{rccccc} & 0 & \leq & \alpha_1^{\text{new, clipped}} & \leq & C \\ \Leftrightarrow & 0 & \leq & \zeta - \alpha_2^{\text{new, clipped}} & \leq & C \\ \Rightarrow & \min(\zeta, C) & \geq & \alpha_2^{\text{new, clipped}} & \geq & \max(\zeta - C, 0) \\ \Leftrightarrow & H & \geq & \alpha_2^{\text{new, clipped}} & \geq & L \end{array}$$

Assuming now that $y_1 = -y_2$, we get by multiplying by y_1:

$$\begin{aligned} \zeta &= \alpha_1^{\text{old}} - \alpha_2^{\text{old}} \\ &= \alpha_1^{\text{new, clipped}} - \alpha_2^{\text{new, clipped}} \end{aligned}$$

Hence:

$$\begin{array}{rccccc} & 0 & \leq & \alpha_1^{\text{new, clipped}} & \leq & C \\ \Leftrightarrow & 0 & \leq & \zeta + \alpha_2^{\text{new, clipped}} & \leq & C \\ \Rightarrow & \max(-\zeta, 0) & \leq & \alpha_2^{\text{new, clipped}} & \leq & \min(C - \zeta, C) \\ \Leftrightarrow & H & \leq & \alpha_2^{\text{new, clipped}} & \leq & L \end{array}$$

Appendix D

Overview of the Random Forests algorithm

In this last appendix, we give a very brief overview of the theory of Random Forests. We assume the reader has a certain familiarity with decision trees methods as well as statistical bootstrapping.

D.1 Decision trees

Decision trees form a very popular class of Machine Learning algorithms. Some of the main exponents of this category are the Classification And Regression Tree (CART), the C4.5 algorithm or the Chi-squared Automatic Interaction Detector (CHAID). Here we expose the working principles of this family of models.

Let us assume that we have a data sample $S_n = \{(\mathbf{x}_1, y_1),...,(\mathbf{x}_n, y_n)\}$ such that for all i, $\mathbf{x}_i = (x_{i,1}, x_{i,2},..., x_{i,p})$ is in $\mathcal{X}$ and y_i is in $\mathcal{Y}$, where $\mathcal{X}$ is a p-dimensional space and $\mathcal{Y}$ a 1-dimensional space. Decision trees algorithms generate binary decision rules that allow to sequentially reach a predicted response $T(\mathbf{x}_i)$ based on data $\mathbf{x}_i$. For example, let us assume that all variables in $\mathbf{x}_i$ are quantitative and that we want to predict the response of a data point $\mathbf{x}$ – this can be a class or a quantity. Let k be an integer and v_1, v_2 and v_3 values in the domain of the first, second and third variables of $\mathbf{x}_i$. A possible decision tree would work as follows:

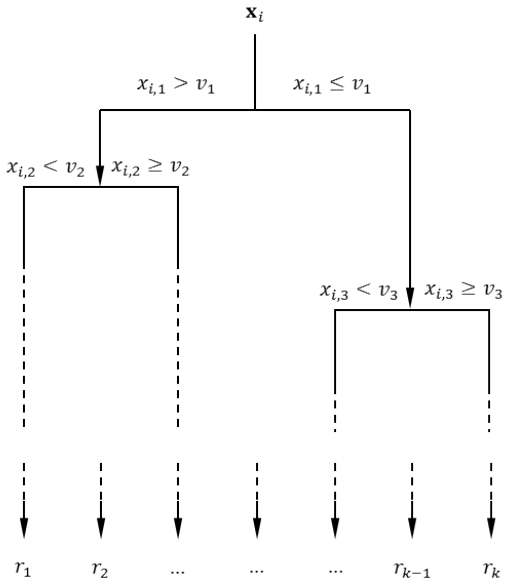

Figure D.1: *Decision tree algorithm*

At each tree node we look at the value of the node's *splitting variable* for observation $\mathbf{x}$ and we

determine whether the value is above or below the node's *split value* – v_1, v_2, v_3, etc.. By navigating through the tree guided by these binary rules, we end up at a terminal node and the predicted response corresponds to the value associated to that node. Let L be the number of terminal nodes, $\{l_1,...,l_L\}$ the terminal nodes and r_t the response if $\mathbf{x}$ ends up in node l_t – for a regression task $r_t \in \mathbb{R}$ and for a classification task $r_t \in \mathbb{Z}$. For both tasks, the decision function of the tree can be written as:

$$T(\mathbf{x}) = \sum_{t=1}^{L} r_t \mathbb{1}_{\{\mathbf{x} \in l_t\}}$$

To grow the tree, an algorithm is required: it should determine recursively the best splits by selecting both the splitting variable and the split value that optimize a certain criterion C, also known as *impurity measure*: for example this criterion can be the mean squared error for regression problems or the error rate for classification problems. We will not discuss further impurity measures as they are not necessary for our exposition of the Random Forest algorithm.

D.2 From decision trees to Random Forests

Random Forests belong to the class of ensemble learning algorithms; ensemble learners work by averaging the predictions of many weak learners, a mechanism which has been showed to have very good performance in cases where the weak learners have low bias but high variance (Hastie, Tibshirani and Friedman, 2009). As their name indicates, Random Forests are constructed by generating multiple trees – which constitute a case of low bias, high variance learners – than are subsequently averaged to make a prediction, let it be a class label or a regression quantity.

We will be working of the data sample S_n again. Let us also define $F_p = \{f_1^{\mathbf{x}}, ...,f_p^{\mathbf{x}}\}$ as the set of attributes or features of vectors $\mathbf{x}_i$. In order to generate good predictions, Random Forests have two built-in mechanisms to control the correlation between the trees in the forest:

- **Sample bootstrapping**: each tree in the forest is fitted to a bootstrapped sample of size B, $S'_B \in S_n$, such that $1 \leq B \leq n$.
- **Feature bootstrapping**: each node in a tree is determined from a bootstrapped set of features of size Q, $F'_Q \in F_p$, such that $1 \leq Q \leq p$.

We assume there are M trees in the forest. The Random Forest training algorithm works as follows:

Algorithm 3 *Random Forest training algorithm*

1: Input $\{(\mathbf{x}_i, y_i)\}_i$, M, B and Q
2: For $m = 1$ to M:
- Generate a bootstrapped sample S'_B of size B
- While the minimum number of samples $S_{\min}$ is not reached, grow a tree T_m recursively to the bootstrapped sample by repeating the following steps for each terminal node:
  - Generating a bootstrapped set of features F'_P of size P;
  - Determining the optimal splitting variable/split point according to criterion C
  - Splitting the node into two children nodes

3: Output the ensemble of trees $\{T_m\}_{1 \leq m \leq M}$

Note that the bootstrapped features set is drawn *at each node splitting*, instead of for each tree as it is the case for the bootstrapped data.

The Random Forest predicts a response to a new observation by either averaging the responses of each tree for a regression task or by majority voting in the case of a classification problem.

To understand the underlying mechanism of the Random Forest algorithm, note that the trees of the forest T_1,..., T_M are random variables depending on the sub sample S'_B and the random feature sets $\{F'_P\}$. Then the variance of the forest F is equal to:

$$\mathbb{V}\text{ar}(F) = \mathbb{V}\text{ar}\left(\sum_{m=1}^{M} T_m\right) = \sum_{m=1}^{M} \mathbb{V}\text{ar}(T_m) + \sum_{m_1=1}^{M} \sum_{\substack{m_2=1 \\ m_2 \neq m_1}}^{M} \mathbb{C}\text{ov}(T_{m_1}, T_{m_2})$$

Now, by using bootstrapped data samples to grow each tree and bootstrapped features to split each node, we are uncorrelating the trees thus decreasing their pairwise covariance and the variance of the overall Random Forest.