# Machine Learning Applications for Small and Medium Enterprises

Chatzikyrou, P.A.

# Machine Learning Applications for Small and Medium Enterprises

Master thesis

Pantelis Aristotelis Chatzikyrou

Supervisors:
prof.dr. F.M. Spieksma
prof.dr. E.A. Verbitskiy

February 17, 2022
Mathematisch Instituut, Universiteit Leiden

# Executive summary

Despite the fact that Machine Learning (ML) has been proven to be a value generating technology for businesses, small and medium enterprises have not yet captured its potential. One can assume that the advanced knowledge, equipment, and software required might not be worth the resources and the financial risk, but is such an assumption justifiable? What Machine Learning has to offer to the companies of the domain? And, first of all, is such an approach feasible? If so, under what conditions?

To answer these questions, we have seen that ML can facilitate growth adding elements in the value chain of an organisation. Examples of such cases can be found in key areas such as marketing, operations management, finance and more. It is not a secret that ML is commonly used by companies for advanced customer segmentation. To do so, a data driven behavioural approach based on customers' interaction with the firm is usually utilised. Such an approach can result in the finest segmentation imaginable; for instance, it is possible to classify customers based on the likelihood to buy in the near future, the possibility to churn, or their personal preferences. As we mentioned above, ML is also used in totally different areas, such as manufacturing. Businesses utilised the technology to check product quality, maintain machinery, or even improve productivity. The areas and the examples are numerous and it is highly probable that one or many of those apply to most small and medium companies.

Looking at feasibility and conditions, things seem to be improving lately. Recent advances in computers software and hardware have widely decreased the cost of acquisition making ML easily accessible to the public. Open source software freely available on the web has been a great contributor to this trend. In addition, academic institutes are showing more and more interest in subjects related to Artificial Intelligence. Consequently, it is also justifiable that the

number of graduates that possess the “know-how” on ML tools and methods is gradually increasing. Now, given that the external factors tend to support such an approach for Small-Medium Enterprises, we will present at a glance the internal requirements. To begin with, a clear definition of the problem and the project objectives are of primary importance. This will further indicate the data needed and the methods to be tested and finally applied. Hence, the availability of data is also crucial. Not only the size and the quality matters but also their relevance to the given problem; this actually largely determines the outcome of a project. Finally, a close co-operation between analysts and managers is key in order for the former to gain insights into the company’s operations and functions.

To test in practice the procedure we claimed above, we have conducted an experiment in a real-world small business. We decided to apply a framework for customer segmentation with respect to churn. Substantially, we studied whether detecting customers that are going to leave the company and its services is possible in advance. In this case, managerial actions could prevent churn increasing company’s profitability. Despite the small amount of data and the scattered information available in the dataset, the ML algorithms tested performed well achieving good results. Precisely, the outcome showed that 80% of the churners can be actually detected in advance, thus supporting the possibility of proactively preventing churn.

To sum up, we have seen that Machine Learning shows potential for small and medium enterprises. Given that, we would like to invite companies in the domain to take the time and effort to look into this emerging technology. The areas of application, and the applications themselves, are numerous and we constantly see an upwards trend. With low entry boundaries and low financial risk, it is suggested that Small and Medium Enterprise should grasp the opportunity and find what ML has to offer to their organisations.

# Contents

# Acronyms

**AUC** area under the ROC curve. 19, 31, 32, 34

**B2B** Business-to-Business. 31

**B2C** Business-to-Customer. 31

**CLV** Customer Lifetime Value. 27, 28, 30

**CNC** Computer Numerical Control. 39

**DT** Decision Tree. 14, 29, 43, 62

**EGM** Electronic Gaming Machines. 43

**ERM** Empirical Risk Minimisation. 2, 4, 5, 7

**ESVM** Extended Support Vector Machine. 29

**FN** False Negative. 17

**FP** False Positive. 17

**IIoT** Industrial Internet of Things. 37

**ML** Machine Learning. 1, 2, 8, 19, 25–27, 67

**MP** Maximum Profit. 31

**PdM** Predictive Maintenance. 37–40

**PvM** Preventive Maintenance. 37, 38

**Ra** roughness average. 39, 40

**RBF** Radial Basis Function. 11, 12

**ROC** receiver operating characteristic. 18

**Rq** root mean square roughness. 39

**Rz** mean roughness depth. 39

**SME** Small-Medium Enterprise. 1, 8, 25, 26, 39, 53, 67, 68

**SVM** Support Vector Machine. 1, 5–9, 11, 13, 14, 29–31, 33, 43, 53, 54, 58–61, 63, 64, 66

**TN** True Negative. 17

**TP** True Positive. 17

# Chapter 1

# Introduction

Machine learning has seen a great growth in terms of popularity, applicability and potential during the past couple of years. To a large extent, the availability of big data and the strong computational power present in modern computers have facilitated this trend. Researchers utilise such models in a broad range of problems, primarily to classify data, make predictions, discover patterns and/or visualise data. The applications are also numerous and nowadays are widely met in business environments. Large companies are already exploiting machine learning in order to generate value in areas such as marketing, operations management, finance and many more.

On the other hand, it seems that SMEs lack behind in the adaptation of the new tools, leaving unexplored the potential benefits of the newly introduced technology. This can be easily justified for the past years when machine learning algorithms were not easily accessible, required expensive hardware, could only be implemented by field experts and in general, had not yet been proven reliable and capable of exceptional performance. However, things seem to have changed with the corresponding barriers decreasing substantially resulting in a rapid growth of machine learning popularity, while new applications arise one after the other.

Having that said, we believe that today ML can not only create value for large corporations, but to a large extent also for smaller companies. Companies in the SME sector should reposition themselves against the emerging technology and become up-to-date with the possibilities the field has to offer. Given that, we aim to investigate the SMEs landscape and investigate whether machine learning is a suitable technology for them and their needs.

With that in mind, and in order to form a broad overview of the topic so that we can draw accurate conclusions for our hypothesis, we have studied the literature in combination with empirical research. More precisely, case studies have been utilised in the former case. For the latter, an experiment where we tested a predictive ML framework on a real-world small business was conducted.

The rest of the text is organised as follows. In Chapter 2, the fundamentals of supervised machine learning are presented and two such models, the Support Vector Machine and the Decision Trees, are strictly defined. In Chapter 3, a similar discussion around reinforcement learning is held and Q-learning is introduced. An in-depth analysis of examples of business problems where ML solutions show a large potential is then performed in Chapter 4. In Chapter 5, an experiment with real-world data from an SME is presented in order to validate the theoretical findings. The conclusion of the research is available at the end of the thesis in Chapter 6.

# Chapter 2

# Basics of Supervised Machine Learning

## 2.1 Introduction

Imagine that we encounter the following problem: one needs to classify two types of fruits, apples and bananas. According to our human experience, there are a couple of features based on which we can distinguish those fruits, e.g., colour and softness. Commonly, apples are red and hard, while bananas are yellow and soft. That is just a simple example of a classification task that humans preform and based on this, or on similar information as an input, one can classify objects. However, in an equivalent way, machines can learn to perform analogous tasks. Various such algorithms have been developed in the field of Machine Learning.

To begin, we will intuitively present some basic ML algorithms as well as give insights in the mathematical background of the discussed methods. With respect to that, we will first need to illustrate the framework of a learning algorithm, the so-called statistical learning framework. Furthermore, we introduce the concept of Empirical Risk Minimisation, which is fundamental in the evaluation of the models. Then a family of linear predictors is defined so we can lastly provide the reader with insights in the hypothesis class for binary classification problems, the class of halfspaces.

In this chapter, the books *Understanding machine learning: From theory to algorithms* [43] and, *The elements of statistical learning: data mining, inference, and* *prediction* [21] have been used as primary sources for the discussion and the mathematical background of the methods. In addition, at this point, it is crucial to highlight that the following notation will be used from now on in the text:

| | |
|---|---|
| $[m]$ | the set $\{1, \dots, m\}$ |
| $\|c\|$ | absolute value of the scalar c |
| $\|\mathcal{A}\|$ | cardinality of the set $\mathcal{A}$ |
| $\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{w}$ | vectors |
| $\langle \boldsymbol{x}, \boldsymbol{w} \rangle$ | $= \sum_{i=1}^{m} x_i w_i$ (inner product) |
| $\|\|\boldsymbol{w}\|\|$ | $= \sqrt{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}$ (the $l_2$ norm of $\boldsymbol{w}$) |

### 2.1.1 The Statistical Learning Framework

In order to facilitate an algorithm that learns based on previous experience gained, a determinate learning framework is required. In an elementary such learning framework, there are three key components that must be available to the learning algorithm: the domain set, the label set, and the training data. The *domain set* can be any arbitrary set, which is usually denoted by $\mathcal{X}$. It consists of *points*, or *instances*, that could be labelled. These domain points are commonly vectors of features or attribute values. The *label set*, $\mathcal{Y}$, is a fixed set of the available labels. For instance, in a typical binary classification, the label set is $\{0, 1\}$ or $\{-1, 1\}$. Finally, we denote by $\mathcal{S}$ the *training data*. That is pairs of a finite collection of domain points together with their corresponding labels, i.e., $\mathcal{S} = ((x_1, y_1), \dots, (x_m, y_m))$, $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i \in [m]$. We regularly refer to the training data as *training set* as well.

On a top level, the task of a learning algorithm is to utilise knowledge retrieved from the training data in order to generate a *prediction rule*, $h : \mathcal{X} \rightarrow \mathcal{Y}$, based on which labels are assigned to new domain points. With "new" we refer to domain points that do not belong to the training data, i.e., $x_i \in \mathcal{X} \setminus \mathcal{S}$. We commonly mention $h$ as the *predictor*, or *hypothesis*, and since $h$ is dependent on the training set, $\mathcal{S}$, we denote such a predictor by $h_{\mathcal{S}}$. Finally, in the case of a classification task, a predictor is also widely called *classifier*.

As it has already been discussed, the training data is a collection of pairs in $\mathcal{X} \times \mathcal{Y}$. For the selection of such a training set, we assume that domain points have been drawn independently and identically from a probability distribution $\mathcal{D}$. In the given setting the learner has no prior knowledge of the distribution

$\mathcal{D}$. Assuming that a *labeling function* $f : \mathcal{X} \to \mathcal{Y}$, such that $y_i = f(x_i)$, for all $i \in [m]$, exists, the collected data points are then labelled. Overall, it can be said that the objective of the learning algorithm is to approximate the function $f$.

To measure the success of the approximation, an evaluation criterion is required. Given a random instance sampled from the probability distribution $\mathcal{D}$, the *error of a classifier* is defined to be the probability that the instance is not correctly labelled by the predictor $h$. Mathematically,

$$L_{\mathcal{D},f}(h) := \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \ .$$

Alternatively, we call $L_{\mathcal{D},f}(h)$ a *generalisation error*, *risk*, or *true error* of $h$.

### 2.1.2 Empirical Risk Minimisation

As we have seen above, a learning framework is summarised as follows: a training set $\mathcal{S}$ is sampled from a certain distribution $\mathcal{D}$ on $\mathcal{X}$, and it is then labelled according to some function $f$ with labels from $\mathcal{Y}$. Given that $\mathcal{D}$ and $f$ are unknown, the learning algorithm strives to find a classifier $h_{\mathcal{S}}$ that minimises the error of the labeling procedure.

Naturally it follows that in order to evaluate and compare predictors the estimate of that error becomes crucial. Since the learner has no prior knowledge about $\mathcal{D}$ and $f$, the computation of $L_{\mathcal{D},f}(h)$ is not feasible. With that in mind, we need to determine an estimate of the error that can be useful to the learner. That turns out to be the error of the predictor over the training data, usually called *training error* or *empirical risk*. Formally, it is defined as:

$$L_{\mathcal{S}}(h) := \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m} \ .$$

That is a plausible approach since the training set is a window to the world available to the learning algorithm. Given that, we can summarise that we try to find a predictor $h$ that minimises the training error $L_{\mathcal{S}}(h)$ and from now on we will refer to this technique as Empirical Risk Minimisation (ERM).

Despite the direct representation and the expected efficiency of the ERM learning rule, difficulties might arise. In many cases, even if we manage to minimise the empirical risk, it is highly probable that we end up with a large true error due to the inherent overfitting. By *overfitting* we refer to the situation where the learner tends to perfectly interpret the training data, however, it fails badly to generalise accordingly in the given domain set. In order to solve this problem, we generally apply the ERM technique over a limited space of predictors. To clearly illustrate this, a selection of a hypothesis class $\mathcal{H}$ precedes the generation of the training sample $\mathcal{S}$. This approach helps the learner to avoid overfitting, nevertheless, it biases it towards a certain set of prediction rules. That should be carefully examined, when it comes to the selection of $\mathcal{H}$.

### 2.1.3 Linear Predictors

We can now introduce the family of linear predictors, a family of hypothesis classes widely used by learning algorithms. What is more, this class plays an important role in the algorithm called Support Vector Machine, which is discussed in the following section. To this end, we introduce the definition of the class of affine functions, $L_d \in \mathcal{H}$:

$$L_d = \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \ ,$$

where

$$\begin{aligned} h_{\mathbf{w},b} :\ & \mathbb{R}^d \to \mathbb{R} \\ & \mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b \end{aligned}$$

In some cases, it is handy to omit the *bias* term $b$. That can be simply done by inserting $b$ as an extra coordinate in $\mathbf{w} = (w_1, \dots, w_d)$ and, simultaneously, adding a coordinate with value 1 at all $\mathbf{x} \in \mathcal{X}$. That is $\mathbf{w}' = (b, w_1, \dots, w_d) \in \mathbb{R}^{d+1}$ and $\mathbf{x}' = (1, x_1, \dots, x_d) \in \mathbb{R}^{d+1}$. The hypothesis function $h_{\mathbf{w},b}$ can be then represented as

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle = h_{\mathbf{w}'}(\mathbf{x}') \ .$$

From now on, we will refer to that simplified form as the *homogeneous hypothesis function* or just as the homogeneous case.

#### Halfspaces

Compositions of a function $\phi : \mathbb{R} \to \mathcal{Y}$ with a function in $L_d$ can generate different predictors. In particular, in binary classification, the sign function is often used,

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{, if } x < 0 \\ +1 & \text{, if } x \geq 0 \end{cases} \ .$$

Consequently, we can define a hypothesis class for binary classification. Assuming that $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$, the *class of halfspaces* is defined to be

$$HS_d = \text{sgn} \circ L_d = \{\mathbf{x} \mapsto \text{sgn}(h_{\mathbf{w},b}(\mathbf{x})) : h_{\mathbf{w},b} \in L_d\} \ .$$

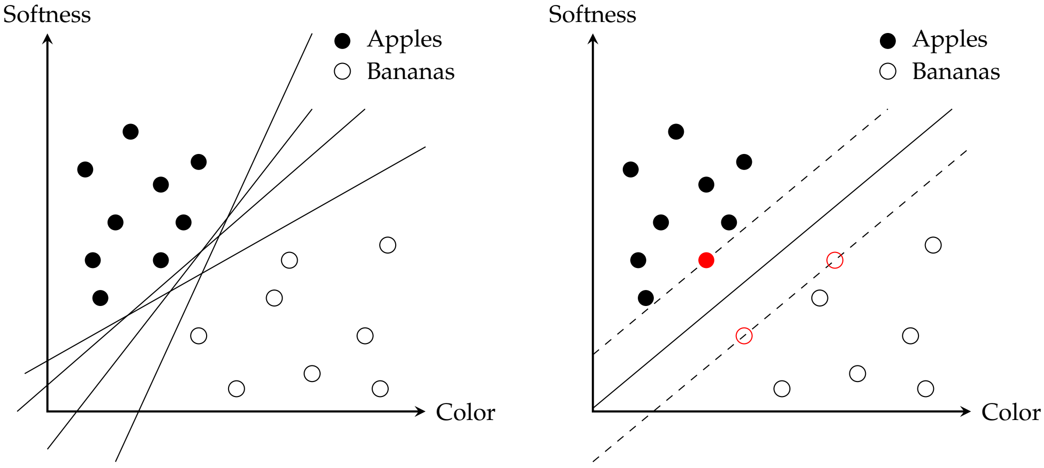
Therefore, a halfspace in the hypothesis class $HS_d$ receives a vector $\boldsymbol{x}$ and outputs a label given by $\text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$. As we will see in the following section, the class of halfspaces is necessary for the formulation of the Support Vector Machine model.

## 2.2 Support Vector Machine

### 2.2.1 Model Description

The Support Vector Machine (SVM) is a supervised machine learning technique, widely used for classification tasks. Its objective is to draw an optimal hyperplane that distinctly classify training points in an d-dimensional space.

Let us recall the previous example. We would like to categorise fruits, apples and bananas, based on two properties, colour and softness. Based on these features, we can plot the data points on a plane and try to draw a line such that to separate the two distinct classes. Obviously, there are many, actually infinite, lines that can separate the given points (Figure 2.1). However, the SVM algorithm strives to identify the line which yields the maximum margin, i.e., the line that sits at the maximum distance from data points of both classes. This thought is justifiable, since it is simply estimated that future instances will be classified with more confidence.



(a) Infinite lines can separate the two classes.

(b) The optimal line.

Figure 2.1: Apple and banana points on the plane based on the two properties, softness and colour.

### 2.2.2 Linearly Separable Case

So far, for simplicity, we assumed a dataset with only two features that can be easily depicted in a two-dimensional space, a plane. The general, d-dimensional case as well as the formal definition of the margin and the optimal hyperplane follow.

Let $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ be a training set of examples, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, $\forall i \in [m]$. This set is *linearly separable* if there exists affine function $h_{\mathbf{w},b}$ such that $y_i = \text{sgn}(\langle \mathbf{w}, \mathbf{x_i} \rangle + b)$ for all $i$. Alternatively, we can rewrite this condition as

$$\forall i \in [m],\ y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0\ .$$

All halfspaces, $h_{\mathbf{w},b}$, that satisfy this condition are ERM hypotheses. For any linearly separable training sample, there are infinitely many ERM halfspaces. Therefore, to make a clear decision rule we need to define additional conditions. To this end, we define the margin to be the minimal distance between a point in the training set and a given hyperplane. SVM is based on the heuristic idea of selecting a hyperplane with the maximal margin.

The following proposition is essential for the formal definition of the SVM algorithm.

**Proposition 2.1.** *The distance of a point* $\mathbf{x} \in \mathbb{R}^d$ *from a hyperplane* $\{\mathbf{v} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{v} \rangle + b = 0\}$*, where* $||\mathbf{w}|| = 1$*, is equal to* $|\langle \mathbf{w}, \mathbf{x} \rangle + b|$*.*

Given that, the minimal distance of points to the separating hyperplane, the margin, is equal to $M = min_{i \in [m]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$. These minimising training points, $\mathbf{x}_i$, are called *support vectors*. Finally, the SVM rule can be analytically defined by

$$\begin{aligned} \max_{\mathbf{w},b} \quad & M \qquad (2.1) \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq M \;, \quad i = 1, \ldots, m \\ & ||\mathbf{w}|| = 1 \;. \end{aligned}$$

Given that the points are linearly separable, there is always a solution to the aforementioned problem.

Now, we can drop the norm constraint on $\mathbf{w}$, and let $(\mathbf{w}, b)$ be an admissible pair for the above maximisation problem. We set $\mathbf{w}' = \mathbf{w}\frac{1}{M}$ and $b' = b\frac{1}{M}$. It directly derives that

$$y_i(\langle \mathbf{w}', \mathbf{x}_i \rangle + b') = \frac{y_i}{M}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \;.$$

In addition, we see that the maximisation problem of the margin, $\max_{\mathbf{w},b} M$, is equivalent to minimising the norm of $\mathbf{w}$, that is $\min_{\mathbf{w},b} ||\mathbf{w}||$. As a result, the maximisation problem in (2.1) can be transformed into the following equivalent quadratic minimisation problem.

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 \qquad (2.2) \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \;, \quad i = 1, \ldots, m \;. \end{aligned}$$

In the optimisation problem (2.2), the exponent and the scalar in the objective function can be easily omitted, however they have been intentionally added since they will significantly simplify future computations.

### 2.2.3 Non-linearly Separable Case

#### Soft-SVM

So far we have seen how to define the maximal margin in order for the SVM algorithm to be able to classify linearly separable instances. However, things

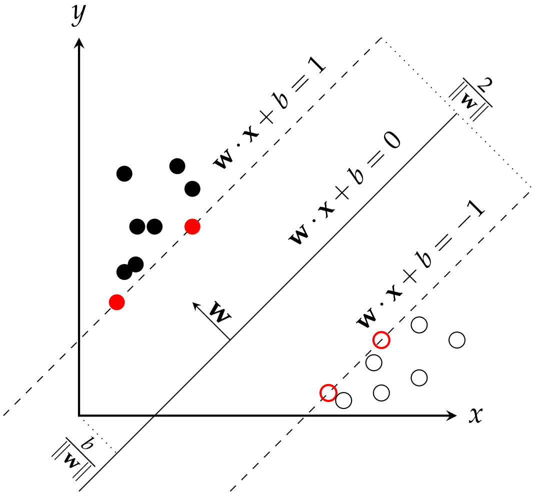

$y$
$x$
$\mathbf{w} \cdot \mathbf{x} + b = 1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$
$\mathbf{w}$
$\frac{2}{||\mathbf{w}||}$
$\frac{b}{||\mathbf{w}||}$

Figure 2.2: The support vector machines classifier. The support vectors are denoted with red.

are not always that convenient and we commonly need to use non-linearly separable sets. To this end, in 1995, C. Cortes and V. Vapnik introduced a "Soft Margin Hyperplane Algorithm" [13]. The so-called soft-SVM utilises a less strict rule in order to define the optimal hyperplane. To this end, nonnegative slack variables, $\xi_1, \ldots, \xi_m$, are introduced and the strict constraints $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ are substituted by the soft constraints: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$. Hence, the aim of the soft-SVM is to minimise the norm $\mathbf{w}$ and the average of $\xi_i$. The first term is corresponding to the margin and the latter to the violations of the constraints. Finally, to control the tradeoff between the aforementioned terms a parameter $C$ is introduced and analytically, the optimisation problem takes the following form:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & ||\mathbf{w}||^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \;, \quad i = 1, \ldots, m \;. \end{aligned} \tag{2.3}$$

### Lagrange Duality

In order to solve the problem (2.2), a Lagrangian approach can be implemented. The primal form of the Lagrange function is

$$\begin{aligned}&\mathcal{L}_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})\\ &\quad = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y_i\sum_{j=1}^{d}x_{i,j}w_j + b) - 1 + \xi_i) - \sum_{i=1}^{m}\beta_i\xi_i\ , \qquad (2.4)\end{aligned}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$ for all $i$.

The $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the nonnegative Lagrange multipliers. In order to find the minimum in (2.4), we apply the Karush–Kuhn–Tucker conditions [27, 31], namely

$$\frac{\partial \mathcal{L}_\mathcal{P}}{\partial w_j} = 0\ , \quad j = 1, \ldots, d \qquad (2.5)$$

$$\frac{\partial \mathcal{L}_\mathcal{P}}{\partial b} = 0 \qquad (2.6)$$

$$\frac{\partial \mathcal{L}_\mathcal{P}}{\partial \xi_i} = 0\ , \quad i = 1, \ldots, m\ , \qquad (2.7)$$

and

$$\alpha_i(y_i\langle(\mathbf{x}_i, \mathbf{w}\rangle + b) - 1 + \xi_i) = 0\ , \quad i = 1, \ldots, m \qquad (2.8)$$

$$\beta_i\xi_i = 0\ , \quad i = 1, \ldots, m \qquad (2.9)$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0\ , \quad i = 1, \ldots, m\ . \qquad (2.10)$$

From Equations (2.5), (2.6) and (2.7) one has

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i \qquad (2.11)$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0\ , \quad i = 1, \ldots, m \qquad (2.12)$$

$$\alpha_i + \beta_i = C\ , \quad i = 1, \ldots, m\ . \qquad (2.13)$$

Substituting (2.11), (2.12) and (2.13) into $\mathcal{L}_P$, yields the corresponding dual problem:

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{m} \alpha_i \qquad (2.14) \\
\text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\
& ||\boldsymbol{\alpha}|| \leq ||\mathbf{C}|| \, ,
\end{aligned}
$$

where $\mathbf{C} = (C, \dots, C) \in \mathbb{R}^m$.

### Kernel Method

In some cases, non-linearly separable data points might have more favourable properties when mapped into a different space. The Kernel Method is a strategy that facilitates this approach. To this end, the data points are mapped into a higher-dimensional feature space where the classification is then performed. Consider a "feature" map $\psi : \mathcal{X} \to \mathbb{R}^D$ and define a kernel function on $\mathcal{X} \times \mathcal{X}$ as:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}') \rangle \, .$$

By replacing the inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ in (2.14) by a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we can apply the SVM algorithm in a space of higher dimension. This enables us to find a non-linear hyperplane, in the transformed feature space, which can better classify points. However, the inverse image of the hyperplane to the initial space will form a non-linear decision boundary (Figure 2.3). In Table 2.1, we present three widely used kernel functions: linear, polynomial of degree r, and the Radial Basis Function (RBF).

**Table 2.1.** Common SVM kernel functions.

| Kernel function | Expression |
|---|---|
| Linear | $K(x, x') = \langle x, x' \rangle$ |
| Polynomial | $K(x, x') = (\gamma \langle x, x' \rangle + c)^r$ |
| Radial basis | $K(x, x') = \exp(-\gamma ||x - x'||^2)$ |

Here, it is important to highlight that $\psi$ is only used in order to define the kernels. Effectively, we will only work with the various kernel functions and the feature maps are left aside once the former has been formed.

Commonly, the soft-SVM and the kernel method are combined, with the objective function:

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i,j}^{m}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) - \sum_{i=1}^{m}\alpha_i \qquad (2.15)\\
\text{subject to} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0\\
& ||\boldsymbol{\alpha}|| \leq ||\mathbf{C}||\ ,
\end{aligned}
$$

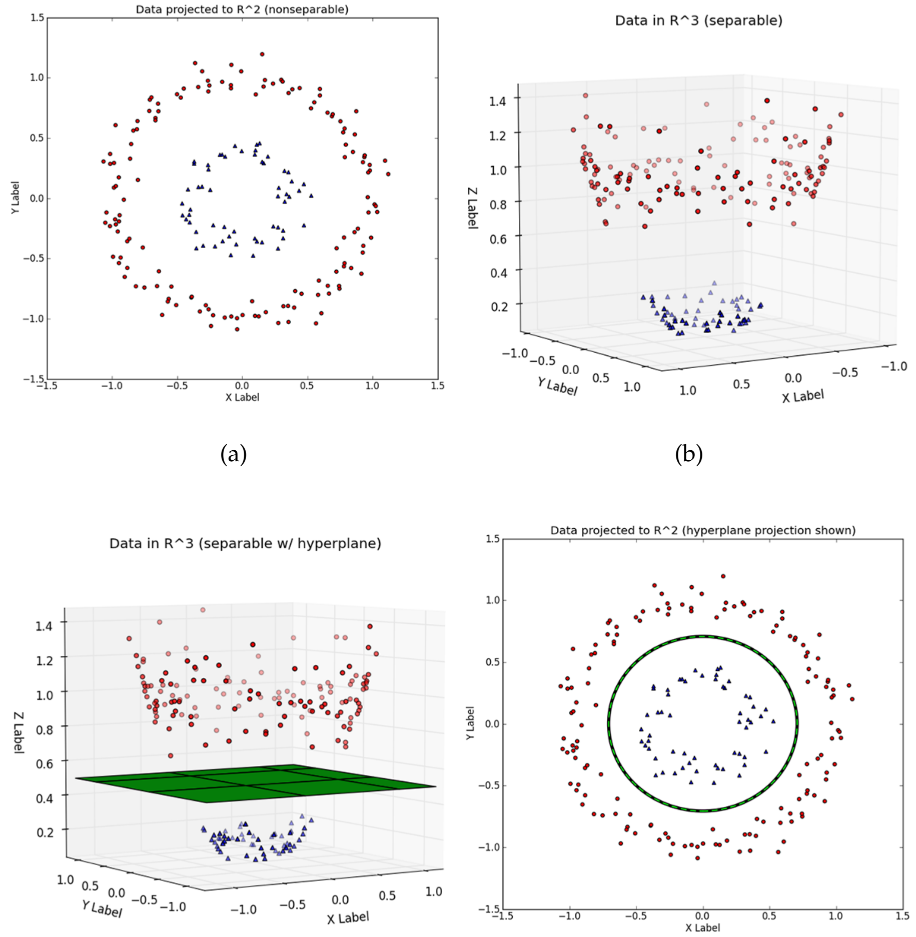
where $\mathbf{C} = (C, \ldots, C) \in \mathbb{R}^m$.

### The role of $C$ and $\gamma$ hyperparameters

In order to define the optimal hyperplane, two parameters play an important role, the regularisation term $C$, which was first met in the minimisation problem (2.2), and the scalar $\gamma$ in the polynomial and radial basis (RBF) kernel functions.

The regularisation term $C$, or simply the penalty, by definition controls the effect of the slack variables $\xi_i$ on the minimisation problem. For values $C$ close to zero, the optimal hyperplane algorithm tends to ignore misclassified instances. This probably results in poor classification outcomes, usually referred to as *sample underfitting*. On the other hand, large values of $C$ lead to an increased significance of the misclassified points, which might lead to sample overfitting. Values closer to 1 balance the norm corresponding to the margin and the violation of the constrains. However, the right choice of $C$ is strongly dependent on the problem formulation and the given dataset.

Both the RBF and the polynomial kernel functions include the parameter $\gamma$. We will discuss the radial basis function since it is more intuitive. In this case, the squared Euclidean distance among the support vectors and the rest of the instances controls the effect of the latter on the decision boundary. Vectors close to a given support vector lead to kernel values close to one, while for points further away the term goes to zero. The parameter $\gamma$ determines the speed of the aforementioned dissipation. Intuitively, low values of $\gamma$ lead to the fact that vectors further from the support vectors affect the decision boundary. On the other hand, given high values of the parameter, a more detailed boundary is drawn based only on the support vectors of the model. In such a case overfitting typically occurs leading to low robustness of the classifier.



(a)

(b)

(c)

(d)

Figure 2.3: The Kernel Method. (a) Non-linearly separable instances; (b) instances mapped to a higher dimension space; (c) instances are linearly separable by a hyperplane; (d) the inverse image of the hyperplane to the initial space is non-linear [28].

The right choice of $C$ and $\gamma$ is a major issue of the SVM model, which is usually addressed by two techniques that we will later present in Section 2.5, the so-called *grid search* and *k-fold cross-validation*.

## 2.3 Decision Trees

### 2.3.1 Model Description

As we have seen, SVM is a powerful technique. However, it is clear that the applied decision rule is not intuitive, and on top of that, the interpretation of the model for a large number of features is rather impossible. When it comes to problem solving, intuition and interpretability are commonly required. To this end, we present an algorithm which is well-known for its simplicity and its interpretation power, the Decision Tree (DT).

A Decision Tree is a predictor resembling a tree; precisely, a common representation of the algorithm is a reverse tree with roots on top and leaves at the bottom. To simply illustrate the DT algorithm we start with the by now familiar fruit classification example.

We intend to classify apples and bananas using the decision tree shown in Figure 2.4. At first, imagine that all the data points are concentrated in the node at the top. The classification process starts by asking a simple question, such as "What is the colour of the fruit?". Two answers are available here, Red and Yellow. If the answer is Red, a point reaches a terminal node labelled as Apple. In case the answer is Yellow, we require more information since there are both apples and bananas with yellow colour. We then pose another question to learn more about the fruits available, e.g. regarding softness. "Is the fruit soft or hard?". By answering Soft a point can be finally classified as a Banana or an Apple otherwise.

Formally, a decision tree model is a tree structured predictor, $h : \mathcal{X} \to \mathcal{Y}$. The label associated with a data point $\mathbf{x}$ is predicted by traveling from the so-called *root node* to a terminal node or a *leaf*. Internal nodes represent attribute tests or predefined splitting rules. The possible outcomes form the *branches* of the tree. For simplicity reasons, we illustrated the binary classification case, i.e., $Y = \{0, 1\}$. However, by analogy, the model can be applied for other prediction tasks too [43], suck as multiclass classification and regression tasks.

Now, looking back at the example, the presented tree can be interpreted as an $h : \{0, 1\}^2 \to \{0, 1\}$ classifier. Let $\mathbf{x} = (x_1, x_2) \in \{0, 1\}^2$ be a vector of features, where in our case $x_1$ and $x_2$ are the colour the the softness respectively. For the former we set 0 for Yellow and 1 for Red, while for the latter 0 for Hard and 1 for Soft. To start the classification procedure, the binary test $\mathbb{1}_{[x_1=1]}$ is

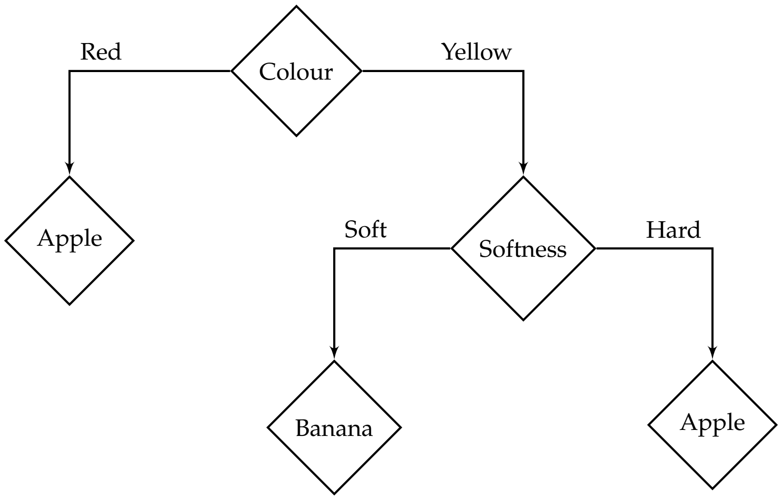


Figure 2.4: Classification of fruits with the decision tree algorithm.

applied for all the data in the root node. Observations with $x_1 = 1$ follow the left branch and directly reach a leaf marked as $A_1$ (e.g. Apple). The rest of the points follow the right branch and end up in an intermediate node where a test on the second feature is applied, that is $\mathbb{1}_{[x_2=1]}$. Based on this rule, the remaining points are split into two leaves which are labelled as $A_1$ and $A_2$. This way each instance follows a root-to-leaf path resulting to a classification outcome.

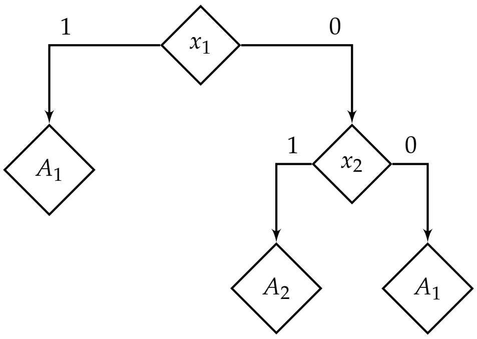


Figure 2.5: Decision tree model.

The most valuable properties of the decision trees are now becoming clear. These are their simplicity and their interpretation power. Furthermore, another advantage is the ability to generate straightforward decision rules, easily applicable in every domain.

### 2.3.2 Growing Decision Trees

We can now give a general framework of how to structure a decision tree predictor. Assume $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$, a set of already classified points, as the training set. A label and a $d$-dimensional vector of features are associated with every training point $s_i$. Let $k$ be the available labels, $\mathbf{x} = (x_1, x_2, \dots, x_d)$ the vector of features, and $m_j, j \in \mathbb{N}$, nodes. Then, $k(m_j)$ will denote the assigned label to node $m_j$, and $\mathcal{R}_{m_j}$ the set of points in this node.

We start the process from the root node $m_1$ where the full dataset is available, i.e., $\mathcal{R}_{m_1} = \mathcal{S}$. We compute the proportion of the given classes in $\mathcal{R}_{m_1}$, and label the node according to the majority vote $k(m_1)$. Formally, the proportion of class $k$ in node $m$ is defined as

$$p_{mk} = \frac{1}{|\mathcal{R}_m|} \sum_{x_i \in \mathcal{R}_m} \mathbb{1}_{[y_i = k]} \ .$$

As a result of the previous notation, the class with the majority vote in node $m$ is equal to

$$k(m) = \operatorname*{argmax}_k p_{mk} \ .$$

In the next steps, a sequence of iterations is performed. Gradually, we apply a splitting procedure testing attributes and attribute values. At every step we measure the performance of the action taken and, finally, we opt for the one resulted in the maximum gain. The data are then divided based on the selected rule, the nodes are labelled, and the process continues. Commonly, we force this procedure to terminate in order to avoid overfitting the given dataset. Usually, criteria such as the depth of the tree, the maximum number of observations in a node, or the number of features tested are applied.

According to the aforementioned procedure, we need to define a "gain" measure based on which we assess the performance of a single leaf split. A variety of splitting criteria has been proposed, including training error, Gini index minimisation and information gain maximisation. Overall, the gain measures are based on the same principle, that is the gain in node purity. As *purity* we describe the homogeneity of a set with regard to the labels of its members. We briefly present the most common such measures; for more details on each algorithm we refer to its respective reference in column 2 of Table 2.2.

**Misclassification error**:

$$E = 1 - p_{mk(m)} = \frac{1}{|R_m|} \sum_{x_i \in R_m} \mathbb{1}_{[y_i \neq k(m)]}$$

**Entropy**:

$$G = \sum_{k=1}^{K} p_{mk} \log p_{mk}$$

**Gini intex**:

$$D = -\sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

The misclassification error is the simplest purity measure available. However, due to the following reasons it is rarely used in practice. Both the entropy and the Gini-index are differentiable, which facilitates more approaches when it comes to numerical optimisation. In addition, these two measures are more sensitive to changes in node probabilities than the misclassification error.

**Table 2.2.** Decision tree algorithms.

| Algorithm | Author | Data type | Branches per node | Impurity criteria |
|---|---|---|---|---|
| CART | Breiman et al. (1984) | Discrete and continuous | Two | Gini index |
| ID3 | Quinlan (1986) | Discrete | Two or more | Entropy |

## 2.4 Evaluation Methods and Metrics

In order to assess a model and benchmark it against others, a range of evaluation methods and metrics are used based on different criteria. As a first step in every method, we need to identify the number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) instances. In general, it holds that

$$TP + FN = P \; ,$$

$$TN + FP = N \; .$$

A model classifies $TP + FP = P'$ points to the positive and $TN + FN = N'$ points to the negative class [6]. Some widely used evaluation methods for classification tasks are the following.

**Confusion matrix**: this is a matrix that summarises the prediction results of a classification task. Usually, we find the actual values, $x$, in rows and the predicted values, $x'$, in columns. Then, the matrix elements are count values of the $(x, x')$ pairs. For instance, for a binary classification task, the confusion matrix is:

|  |  | **Prediction outcome** |  |  |
|---|---|---|---|---|
|  |  | **n′** | **p′** | **Total** |
| **Actual value** | **n** | True negative | False positive | N |
|  | **p** | False negative | True positive | P |
|  | **Total** | N′ | P′ |  |

**Accuracy**: this is defined as the fraction of correctly classified instances, i.e., the sum of true positives and true negatives over the sample size. In case of highly imbalanced datasets, this needs extra attention; the model accuracy can be high even if the minority class is totally misclassified.

$$Accuracy = \frac{\text{\# Correct Predictions}}{\text{Total \# Predictions}} = \frac{TP + TN}{TP + FP + TN + FN} .$$

**Sensitivity or Recall**: the percentage of correctly classified observations in the positive class.

$$Recall = \frac{TP}{TP + FN} .$$

**Specificity**: the percentage of correctly classified observations in the negative class.

$$Specificity = \frac{TN}{TN + FP} .$$

**Precision**: the percentage of correctly classified as positive over the positive classified instances.

$$Precision = \frac{TP}{TP + FP} .$$

**F1-score**: the harmonic mean of precision and recall. It is widely used for imbalanced datasets. A perfect classifier has F1-score equal to 1.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} .$$

**AUC**: the area under the receiver operating characteristic curve (AUC). For binary classifiers, the receiver operating characteristic (ROC) curve is defined as the plot of true positive rate (sensitivity) against the false positive rate (1-specificity), with sensitivity on the $y$-axis and 1-specificity on the $x$-axis [17]. A random classification has AUC equal to 0.5 whilst a perfect classifier, to 1 [6].

$$AUC = \int_0^1 \frac{TP}{P} d\frac{FP}{N} = \int_0^1 Sensitivity\ d(1 - Specificity)\ .$$

**Top-decile lift**: a metric that compares the classification achieved with the random classifier. It is widely used for imbalanced datasets. In order to calculate the metric, the instances are sorted in a descending order based on their probability to belong in the minority class. The ratio of true positive in the top 10% of the list ($\beta_{10\%}$) over the percentage of positive instances ($\beta_0$) is defined as the top-decile lift. A top-decile lift of 5 means that the classifier detects 5 times more instances than a random one [51].

$$Top - decile\ lift = \beta_{10\%}/\beta_0\ .$$

## 2.5 Hyperparameter Tuning Using Grid Search and k-fold Cross-validation

In Machine Learning, the hyperparameters of the algorithms are crucial for the final predictive power of the models. In order to obtain those values that optimise model's performance on the given dataset, two techniques known as *grid search* and *k-fold cross validation* are utilised.

Grid search is a method according to which a model is trained and evaluated with numerous combinations of hyperparameter values. At first, these values are derived from a $n$-dimensional grid, where $n$ denotes the number of hyperparameters tuned (Table 2.3). Once the grid is formed, the model is trained with every single entry of the grid searching for the optimal set of configurations.

**Table 2.3**. $(C, \gamma)$ hyperparameter grid.

| $\gamma \backslash C$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|---|
| $\gamma_1$ | $(C_1, \gamma_1)$ | $(C_2, \gamma_1)$ | $(C_3, \gamma_1)$ | $(C_4, \gamma_1)$ |
| $\gamma_2$ | $(C_1, \gamma_2)$ | $(C_2, \gamma_2)$ | $(C_3, \gamma_2)$ | $(C_4, \gamma_2)$ |

Traditionally, ML engineers test such entries on the so-called validation set. The validation set is formed by further dividing the training points, however, due to the fact that this split can significantly limit the data available for model training (Figure 2.6) alternatives are required. *k*-fold cross-validation is a process that facilitates hyperparameter tuning utilising the existing training data. The process can be summarised as follows: (1) the training set is divided into *k* sets (Figure 2.7); (2) the model is trained successively *k* times on $k-1$ folds and tested on the remaining one (Figure 2.8); (3) the average performance of the model, in *k* rounds, is calculated. Commonly, the two techniques, grid search and *k*-fold cross-validation, are used together in order to form sets of hyperparameters and determine the optimal such configurations for the final model.

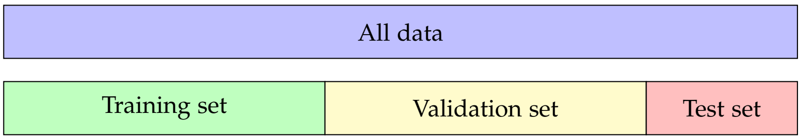


Figure 2.6: Traditional approach: Dataset split in training, validation and test set.

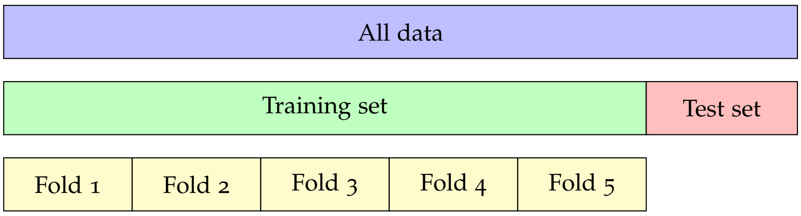


Figure 2.7: Dataset split in training and test set. The training set is further divided into 5-folds in order to apply cross-validation.

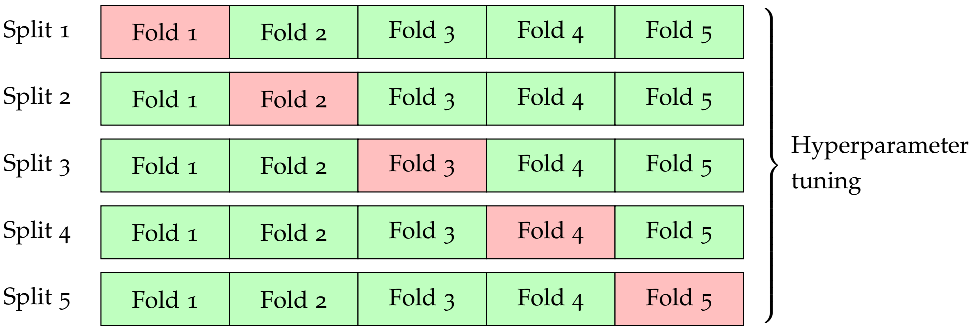


Figure 2.8: The training set is divided into k-folds and the model is trained successively *k* times on the $k-1$ folds (green) and tested on the remaining one (red). The performance of the model is calculated as the average of *k* rounds.

# Chapter 3

# Basics of Reinforcement Learning

## 3.1 Introduction

As discussed above, in supervised learning an external supervisor that provides knowledge to the model is required. Contrary to that, in reinforcement learning no such supervisor is required, but the model learns with respect to its own experience. Precisely, the key element in reinforcement learning is the learning process followed by an algorithm based on the interaction with a given environment. Algorithms in this class are usually called *learning agents* and they are characterised by their ability to perform tasks, analyse results and search for possible ways to improve performance, just by themselves. To further interpret, RL comprises processes in which the learning agent determines decision making rules for specific situations in order to maximise the resulting reward (or minimise the penalty). Given that external input is not available, the agent needs to explore and learn by making various attempts with the reward, or penalty, to regulate the success of every step taken.

Given that the agent has no prior knowledge of the environment, it needs to explore the options available. On the other hand, when more and more experience on the environment is gained, it makes sense for the agent to follow decisions that have been proven profitable in the past. Directly, a strong characteristic of reinforcement learning arises, that is a trade-off between the exploration and exploitation. Consequently, a firm balance between these two needs to be maintained in order to create an effective model.

We can now deepen and describe a RL system which, as we discussed, is comprised of two major elements, the learning agent and the environment. The latter consists of *states* and sets of *actions* that are available to the former per given state. The agent interacts with that environment and strives to take the optimal actions in order to optimise a long-term incentive. In addition, the following components are part of the system: (a) Policy: it defines the set of actions that the agent takes in every state of the given environment. (b) Reward: in every state, the reward determines the yield for the action taken. In a RL system, the objective of the agent is to maximise the long-term overall return (or minimise loss). (c) Value function: it determines the expected long-term yield. The value function maps every state to an accumulated future reward. Finally, (d) Model of the environment: it describes a possible underlying pattern of the learning environment. [48]

## 3.2 Q-learning

Q-learning is a model-free reinforcement learning method. As *model free* are defined the models where no explicit knowledge on the system dynamics (environment and consequences) is available [36]. In Q-learning, the general idea is that the *agent* learns the optimal policy by taking actions in a discrete, finite world. A reward (or penalty) is directly connected to every action taken. Actions are performed repeatedly and evaluated based on a long-term reward. In 1989, C. J. Watkins and P. Dayan proved that Q-learning converges, and hence the agent can identify the best policy by forming a look-up table [49].

According to Watkins and Dayan, the problem is formalised as follows. We assume a discrete, finite world $\mathcal{X}$ where an agent is taken actions at every time step. The actions available belong in a finite set $\mathcal{A}$. Assuming that the agent takes the $n$-th step, he is at a state $x \in \mathcal{X}$ and he chooses an action $\alpha \in \mathcal{A}$. Based on the action selected he receives a reward, $r_n$, which has a mean value of $\mathcal{R}_{x_n}(\alpha_n)$ depended only on the state and the action. Given that, the agent follows a stationary policy, i.e., a non-random decision process in which the action at time $t$ is selected only based on the state at time $t$ [42].

**Definition 1.** *We call* ***stationary policy*** *a function $\pi$ from the state space $\mathcal{X}$ to the set of actions $\mathcal{A}$, that is*

$$\pi : \mathcal{X} \to \mathcal{A} .$$

When such a policy occurs, the sequence of states forms a Markov chain with transition probabilities:

$$\mathcal{P}_{x_n y}[\alpha_n] = Prob[x_{n+1} = y | x_n, \alpha_n] \ .$$

Now, we need to define a metric so the learning agent can evaluate the actions taken and, finally, determine an optimal policy. With that in mind, we introduce the concept of discounted rewards; rewards that are expected to receive $s$ steps later worth less than rewards received now. A discount factor $\gamma^s$, where $0 < \gamma < 1$, reduces their value. Hence the agent expects to receive $\mathcal{R}_x(\pi(x))$ immediately after performing an action according to the policy $\pi$, and moves to a state $y$ with probability $P_{xy}[\pi(x)]$. The new state has a value of $V^\pi(y)$. Analytically, for a policy $\pi$, the value of a state is calculated by the *total discounted expected reward*:

$$V^\pi(x) := \mathcal{R}_x(\pi(x)) + \gamma \sum_y P_{xy}[\pi(x)] V^\pi(y) \ .$$

Therefore, it makes sense to look for a policy such that maximises $V^{\pi^*}(x)$.

**Definition 2.** *A policy $\pi^*$ is said to be $\gamma$-discount optimal if*

$$V^{\pi^*}(x) = \sup_\pi V^\pi(x), \forall x \in X \ .$$

Based on the following theorem of Stochastic Dynamic Programming [42], there is at least one optimal stationary policy for every state in the system.

**Theorem 3.2.1.** *For every state $x \in X$, there is at least one $\gamma$-discount optimal stationary policy, $\pi^* : \mathcal{X} \to \mathcal{A}$, such that,*

$$V^{\pi^*}(x) = \max_\alpha \left\{ \mathcal{R}_x(\alpha) + \gamma \sum_y P_{xy}[\alpha] V^{\pi^*}(y) \right\} \ .$$

Given that $\mathcal{R}_x(\alpha)$ and $P_{xy}[\alpha]$ are known, there are a few dynamic programming methods that can compute $V^*$ and $\pi^*$. However, this is not the case, so we need to find a work-around. We define the Q-values of a state $x$ as the expected discounted reward for taking action $\alpha$ and from then on following policy $\pi$. Analytically,

$$\mathcal{Q}^\pi(x, \alpha) = \mathcal{R}_x(\alpha) + \gamma \sum_y P_{xy}[\pi(x)] V^\pi(y) \ .$$

By estimating the Q-values we can draw an optimal policy. For convenience, we denote the Q-values obtained by the optimal policy, $\mathcal{Q}^{\pi^*}(x,\alpha)$, by $\mathcal{Q}^*(x,\alpha)$, $\forall x,\alpha$. It follows that $V^{\pi^*}(x) = \max_\alpha Q^*(x,\alpha)$. The objective of the Q-learning is to learn those unique Q-values that maximise the expected discounted reward. In order to learn, the following steps are repeatedly taken by the agent, defining an episode (time-slot or epoch):

- observes its current state $x_n$,
- selects and performs an action $\alpha_n$,
- observes the subsequent state $x_{n+1}$,
- receives an immediate payoff $r_n$, and
- adjusts its $Q_{n-1}$ values using a learning factor $\beta_n$, according to:

$$\mathcal{Q}_n(x,\alpha) =$$
$$= \begin{cases} (1-\beta_n)\mathcal{Q}_{n-1}(x,\alpha) + \beta_n[r_n + \gamma V_{n-1}(x_{n+1})] & \text{, if } x = x_n \text{ and } \alpha = \alpha_n \\ \mathcal{Q}_{n-1}(x,\alpha) & \text{, otherwise} \end{cases}$$

where

$$V_{n-1}(y) := \max_b\{\mathcal{Q}_{n-1}(y,b)\}$$

is the best estimate for agent at state $y$. We also assume that initial Q-values, $Q_0(x,\alpha)$ are given.

The following theorem ensures convergence of the aforementioned setup. However, without the condition that for each starting state and action, the sequence of episodes that facilitate the learning process consists of an infinite number of episodes, convergence can not be guaranteed.

**Theorem 3.2.2.** *Let $n^i(x,\alpha)$ be the index of the i-th time that action $\alpha$ is tried in state $x$. Given bounded rewards $|r_n| \leq \mathcal{R}$, learning rates $0 \leq \beta_n < 1$, with*

$$\sum_{i=1}^{\infty} \beta_{n^i(x,\alpha)} = \infty, \sum_{i=1}^{\infty}[\beta_{n^i(x,\alpha)}]^2 < \infty, \forall x,\alpha,$$

*then $\mathcal{Q}_n(x,\alpha) \to \mathcal{Q}^*(x,\alpha)$ as $n \to \infty, \forall x,\alpha$, with probability* 1.

# Chapter 4

# ML Applications in Business Domain

In this chapter, we discuss key problems in business and novel ML solutions that can be applied to analyse these. Our research and analysis are conducted with respect to the Small-Medium Enterprises and the special needs of businesses in this domain. As we will soon see, in comparison with large scale organisations, multiple restrictions are present in smaller businesses when it comes to the implementation of such novel techniques.

A condensed overview of the key issues that commonly arise when one studies such an approach can be summarised as follows: (a) Availability of data: small and even moderate sized companies do not always collect, store and analyse data relevant to their operations and customers, or other types of data. (b) Quality of the data available: even when data are partially available, the lack of systematic and strictly defined record processes can significantly decrease reliability. Missing values and wrong information are widely present in such datasets. (c) Budget: to a large extent, SMEs are precautious regarding investments in novel, unfamiliar approaches. Given the hardware and software infrastructure commonly required in an AI framework, cost is a major threat in the domain. (d) Willingness to adopt recent technological developments: SMEs' management is often driven by traditional approaches and tactics. Even when all the aforementioned components are available, the organisation needs to have the required foundations and resilience in order to draw and apply information-driven strategies.

On the other hand, the broad range of ML applications in business, e.g. in manufacturing, marketing, customer relations, e-commerce, human resources, and the largely unexplored potential by SMEs offers tremendous opportunities. According to Jabłońska, M. R., and Pólkowski, Z. (2017) [24], SMEs striving to acquire a competitive advantage can benefit from AI technologies to "improve organisational performance, lower costs, raise sells, automate customer management, advanced data collection and processing, save time and limit flaws". Besides, more and more AI-based approaches are becoming widely available and often for more affordable rates than in the past. Low-cost hardware and open source software, in combination with the increased simplicity of the implementation, have strongly supported this direction during the previous years. This eventually, establishes ML-based solutions as an emerging trend in SMEs. What is more, on occasion, the restricted size of small-medium businesses, including their functions and people, encloses a form of agility that, in many cases, bigger organisations are lacking, which further leverages their potential in the domain.

Finally, to gain more insight in the processes, the needs but also the challenges of machine learning applications in SMEs, we present a number of successful case studies in the rest of the chapter. Precisely, three problems are introduced in the areas of customer loyalty, equipment maintenance and pricing. To the extent possible, we focussed on industries and business cases that are widely alike to the aforementioned SME format. However, since that was not always feasible due to the limited literature related to SMEs, an easy adaptation of the applied framework by firms in the SME field was the second criterion utilised for the selection of the presented cases.

## 4.1 Customer Churn

### 4.1.1 Problem Description

**Churn and retention rate**

It is well-known from the literature that the cost to obtain a new customer is often five times higher than the cost to retain an already existing one [12]. In business, customer churn describes the phenomenon that a customer leaves a company for good, usually shifting to a competitor or stop using its products or services entirely. Churn rate is defined as the ratio of the customers who churn during a period, usually monthly or annually, over the total number of customers in the beginning of that period. Similarly, the retention rate is defined based on the loyal customers of that period. Analytically, it holds that:

$$\text{Churn rate} = \frac{\text{Churned customers}}{\text{Customers at the beginning of a period}} = 1 - \text{retention rate} \quad .$$

As it can be seen, decreasing the churn rate yields an increase in the retention rate. This further results in an increase in Customer Lifetime Value (CLV) (Figure 4.1), which is defined for a company to be the total worth of a customer. CLV is one of the most crucial business indicators as it has a direct impact on the firm's profitability and overall performance. It simultaneously underlines in an efficient and effective way the firm's relationship with its customers [9]. That makes the need for companies to closely monitor and improve retention rate clear by restricting as much as possible churn. As a matter of fact, it has been observed that only a 5% improvement in retention rate can result in approximately 25% increase in business profitability [10].

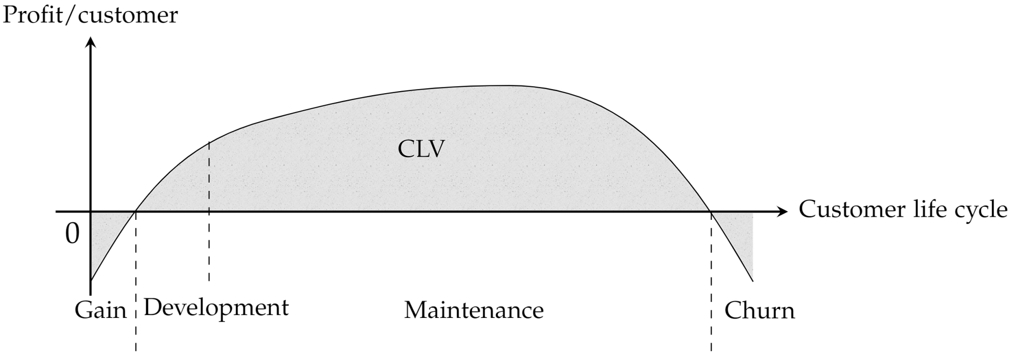


Figure 4.1: Customer life cycle and customer lifetime value [50, 3].

**Customer segmentation based on CLV**

Today companies have a variety of advanced tools to detect and analyze customer behaviour. By applying ML algorithms, firms can obtain more in-depth and accurate customer segmentation results. Traditional segmentation approaches are based on descriptive factors, such as demographic and geographic information. However, modern approaches, as the one proposed in "Analyzing the applications of customer lifetime value (CLV) based on benefit segmentation for the banking sector" [26], utilise causal factors targeting the estimated CLV of a customer. With respect to that, by detecting the group of customers prone to churn, i.e., the ones with zero or low expected value, companies can draw more effective retention strategies.

Having said that, personalisation is one of the most powerful tactics to increase customer loyalty and reduce churn. Personalisation builds strong bonds between the firm and its customers and makes it difficult for them to abandon it for a new brand or another relative service. Following the trend of our era towards one-to-one marketing, companies are called to better understand customers and their preferences. Among marketers it is clear that different customer segments imply the need for different marketing mixes; deeper customer segmentation requires an individual approach tailor made for the end customer (customised content, favorite platforms, etc). In addition, nowadays, firms have the tools and the infrastructure to customise products and services according to individual preferences and taste.

Secondly, by early detecting the segment which is not willing to pursue further collaboration with the brand, companies can better allocate their budget and focus on the potentially profitable future customers. There is always a proportion of customers for instance, that is not interested in further purchases or, in other cases, is definitely going to cancel its subscription for a variety of reasons. It is a waste of budget and time for the firm to try to convince them with discounts, special offers or loyalty programs. In contrast, there are always potential churners who are looking for better deals. By focussing on them, firms can boost their retention rates and have a sufficient impact on their businesses.

Finally, to conclude, the retention rate is an intangible asset of the firm. High retention rate, hence low churn, implies a healthy and successful business model. Customer satisfaction leads to loyalty, therefore, higher CLV. Customer equity, the sum of CLVs, forms an interesting business indicator which holds the following properties: transforms customers value into a countable metric and it has the main impact on the cash flow generated. It is clear now that CLV is immediately related to brand equity. To strengthen this point, high brand equity attracts customers and creates loyalty. Following the same pattern, the firm increases its brand equity and eventually its market value.

### 4.1.2 Customer churn detection with SVM

Given the potential of the application, analysts across the industries are called to perform the churn detection task. The extensive research on the topic has resulted in a sufficient body of literature for analysts to refer to. By studying customer churn in different corporate environments, we end up with specific properties repeatedly appearing in the prediction task (regarding the e-commerce industry we refer the reader to [50]). Its major characteristics can be described as follows: (1) the distinction of churn and loyal customers is a typical binary classification problem; (2) the datasets are usually highly imbalanced; i.e., in a typical corporate environment the number of the actual churners is only a minority of the population (usually 2-10%, depending on the industry); (3) both relative and irrelevant to churn data are included in the datasets, that makes the integration procedure intense; (4) identifying causal churn factors is complicated.

There has been a wide variety of binary classifiers proposed for the customer churn prediction problem. Based on the problem's nature and the high accuracy of the model across the literature, we opt for and present cases utilising the SVM algorithm. First, the SVM classifier can work with linear and nonlinear datasets. The Kernel Method is usually applied when the complexity of the data cannot warranty linear discrimination whereas variations of the model, like Extended Support Vector Machine (ESVM) [50], can deal with the strong class imbalance usually present in the churn data.

The ESVM model is based on the soft-SVM algorithm with kernels and an embedded margin calibration for improved classification results in imbalanced sets. To briefly describe the set-up, the parameter $C$ is replaced by $C^+$ and $C^-$ in order to treat loyal customers and churners respectively. To deal with the nonlinearity of the data, similarly to the Soft-SVM model, the nonnegative slack variables $\xi_1, \dots, \xi_m$ are introduced. Finally, parameters $\rho_1$ and $\rho_2$ are used for margin calibration. To underline the success of the model, empirical results for the churn detection problem have shown that Extended Support Vector Machine outperforms Artificial Neural Networks, Decision Tree and SVM algorithms [50].

### 4.1.3 Case Studies

**Churn prediction in manufacturing industry**

In 2012, Chen, Fan & Sun [10] studied in depth churn prediction with SVM techniques. According to the authors, firms are in possession of a profuse amount of both static and transactional data and with respect to that they discussed extensively data preprocessing techniques involved in churn detection. As our main focus is on classification methods, and the final results of the research for the SVM classifier are very similar, in our text we will only consider the most simplistic and easily implementable approach in their work.

To briefly describe the experimental design, demographic information and transactional data of individual customers are used as an input for the learning algorithm. Little preprocessing is required regarding the demographic part of the data, whereas transactional records need to be converted in order to form multivariate time series of fixed length. Following the aforementioned simplistic approach, the time series are transformed into static data (usually by applying weight averages) in order to reduce the dimensions and eventually form the vectors of features. Concerning the target variable, churn is defined based on CLV in a six-month interval at the end of a studying period. Analytically, CLV equal to zero indicates a churner (denoted by 1) and positive value a loyal customer (denoted by $-1$).

Regarding the empirical analysis, among others, the Adventure dataset retrieved from AdventureWorksDW, was used. The dataset contained 701 customers and more than 60,000 transactional records. After data cleansing, the final set consisted of 633 customers and their purchase history. In the given set, the churn rate found to be quite high, however churners still shape a minority class, with the corresponding figure reaching 24%. For the implementation of the classification, only three variables relating to customers' transactional behaviour finally utilised, i.e., volume (amount of spending), frequency (number of purchases) and variety (number of products purchased). A three-year time frame used in the duration of which the corresponding observations recorded in a monthly base. In addition, seven static variables holding customer personal information included in the formation of the final set as well.

Next, by applying a sampling procedure the dataset was divided in three equal sets in order to form the training, the validation and the test set accordingly. Due to the temporal nature of the data, the examined periods of each set differ. For model training and hyperparameter optimisation the period between July 2002 and June 2004 was selected, while a testing period in the future was used, January 2003 to December 2004.

Given the tendency of the SVM algorithm to perform better on balanced datasets and the imbalanced nature of the churn data, the authors conducted the experiment in both balanced and imbalanced sets. A technique, widely known as undersampling, was applied in order to balance the observations in the training and validation sets. Substantially, with undersampling the size of a class is limited to the desired one by sampling instances. For example, to yield balance among two classes the sample of the majority class can be limited to the minority class size.

Subsequently, a grid search applied in the validation set. The resulted optimal hyperparameters were used to train the classification algorithm before the model is tested on the test data. To assess the performance, the authors recorded numerous metrics, that is: Accuracy, Sensitivity, Specificity, AUC, top-decile lift), the Maximum Profit (MP) and the H-measure (H).

Despite the relatively small number of observations used for model training, the results for the SVM classifier on the test set showed robustness across balanced and imbalanced data. For the balanced data, accuracy of 95.83% and sensitivity of 93.33% were achieved. Further, the AUC is constantly higher than 90% for all the dataset indicating good generalisation properties on imbalanced sets. The top-decile lift remained stable at 8.22 across the experiment (Table 4.1). Lastly, it is worth mentioning that a conventional laptop computer with an Intel Core i3 processor and 2GB of RAM was utilised which highlights the low technical requirements for such an experiment.

**Table 4.1**. Results on balanced and imbalanced data. The parameter $\theta$ determines the non-churners over churners ratio [10].

| Metric | $\theta = 1$ | $\theta = 2$ | $\theta = 5$ | $\theta = 15$ |
|---|---|---|---|---|
| PCC | 95.83 | | | |
| Sensitivity | 93.33 | | | |
| Specificity | 98.33 | | | |
| AUC | 99.03 | 92.81 | 91.19 | 91.44 |
| Lift | 8.22 | 8.22 | 8.22 | 8.22 |
| MP | 5.30 | | | |
| H | 95.72 | | | |
| Time | 1.20 | | | |

**Churn prediction in B2B e-commerce industry**

To address the research gap on churn prevention in the Business-to-Business (B2B) area, N. Gordini and V. Veglio [19] proposed a churn prediction model tailored for the e-commerce industry. Compared to Business-to-Customer (B2C), B2B churn has an even more severe impact on the business. Precisely, the customers are limited and the transactions that they generate are more frequent and of higher value. With respect to that, the authors utilised a dataset of a major Italian online fast-moving consumer goods company and studied the transactional records of over 80,000 business customers during one year.

In order to apply a churn detection framework, two equal samples, of 20,000 observations each, have been selected and used as training and test set. Based on the fact that churners represent the minority class, about 10% of the data, the authors applied sampling techniques so that balance results between loyal and churn customers within the training set. In contrast, the test set was sampled with respect to the original situation, i.e., 10% churn.

As remarked by the researchers, the small number of predictors selected for the experiment makes the research robust across enterprises. Besides, the predictors used are assumed to be widely available at every company in the e-commerce domain. In detail, the predictive variables are categorised within four groups, that is: Customer Data, Login, Transaction and Web log. The variables in the groups contain both socio-demographic and behavioural attributes (Table 4.2). Precisely, Customer Data include personal information of the customer, i.e., ID number, name, gender, address, profession, email, mobile phone number and date of registration. Login holds information of the customer's registration in the system like customer id, login id, login date and login page. Transaction summarises key information about transactional records, namely frequency, recency, length, monetary indicator, product categories and failure. At last, Web log records behavioural data about customer's web visit such as date, remote host, method, page and request status. Again, the dependent variable is churn, a binary variable based on the customers' annual transactional history. Customers with no purchase within the year are defined as churners and denoted by 1. Respectively, active customers (at least one purchase during the year) are defined as non-churners and denoted by 0.

Regarding the $C$ and $\gamma$ parameter, the authors compared different optimisation approaches. At first, the traditional approach utilising accuracy as the performance metric in the k-fold cross-validation (SVMacc) applied, whilst the AUC metric was selected in the second round of experiments (SVMauc). The optimal $C$ and $\gamma$ pair generated by the later resulted in the highest cross-validated accuracy in the training set. The results were similar for the test set; it was shown that the real churners identification is highly probable, with SVMauc outperforming SVMacc achieving 89.98% of accuracy, 88.61 AUC and 5.26 top-decile lift (89.12, 87.95 and 4.98 for SVMacc respectively).

In addition, an evaluation of the most critical churn prediction variables was carried out. This step provides crucial insights for further managerial actions that can further leverage personalised content. Results inline with literature showed that higher number of purchases (frequency), longer time since last order (recency), length of relationship as well as predictors related to product category and failure attribution are of the highest importance. Contrary to that, monetary value and descriptive data were proven to be of lower significance. Lastly, on the technical part, the LIBSVM SVM toolbox for MATLAB, available by Chang and Lin [8] has been utilised for the analysis.

**Table 4.2.** Predictor variables for churn prediction in B2B e-commerce industry [19].

| Variables | Attributions |
|---|---|
| Customer data | ID, Name, Gender, Age, Address, Profession, Email, Mobile phone, Register date |
| Login | Customer ID, Login ID, Login date, Login page |
| Transaction | Frequency (the number of transaction observed in a period) |
| | Recency (the time of the last transaction) |
| | Length (number of days since first log in) |
| | Monetary indicator (total spending of a customer and total spending per each category) |
| | Product categories (the number of product purchased from each category) |
| | Failure (missing or damaged items, poor quality, return of products, refunds) |
| Web log | Log date, Remote host, Method, Page, Request status |

**Churn prediction in subscription services**

In "Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques" K. Coussement and D. Van den Poel [14] studied early churn detection for subscription services. With respect to that an experiment conducted utilising a dataset that includes customers of a Belgian newspaper publishing company. Among other classification algorithms, SVM applied to classify subscribers into potential churners and loyal customers.

The studied company implements a tailor made subscription plan, based on the customer's desired subscription period and current available offers. The customers are obliged to pay a fixed amount and cannot quit before the maturity date. After that, for an additional period of four weeks the company delivers its services free of charge giving the opportunity to the subscribers to renew their plans. Heuristically, churn is defined as whether a customer will renew, or not, the subscription prior the end of that additional period. In general, data from January 2002 to September 2005 has been utilised whilst to define churn only a time interval between July 2004 and July 2005 has been considered. The predictive variables are based on a 30-month period with information derived from customers' individual activities. Information is collected in two levels, i.e., subscription-level and subscriber-level. At the former, details about the current plan is included while the latter is mainly focussed on customer's personal data. The predictors are defined and divided into four variable category groups, client/company-interactions, renewal-related information, socio-demographics and subscription-describing information (Table 4.3).

As it generally holds, here again churn customers comprise the minority class of the data (11% churn). To boost algorithm performance, an undersampling technique applied in order to form a balanced set of 45,000 instances which was used for model training. A test set of same size was drawn with respect to the normal distribution (11% churn) so it reflects the actual situation. As in the previously discussed case, to define the optimal $C$ and $\gamma$ parameters, a k-fold cross-validation with AUC as performance metric was applied. To evaluate the performance of the classifiers three metrics has been calculated, i.e., accuracy, AUC and the top-decile lift. The latter has a significant role in customer retention management due to the inherent cost in customer targeting. Again, the SVMauc model outperformed SVMacc, and the results showed accuracy of 88.63%, AUC equal to 85% and top-decile lift 4.492.

**Table 4.3**. Explanatory variables for churn prediction in subscription services [14].

| Category | Variables |
|---|---|
| Client/ company-interaction variables | The number of complaints |
| | Elapsed time since the last complaint |
| | The average cost of a complaint (in terms of compensation newspapers) |
| | The average positioning of the complaints in the current subscription |
| | The purchase motivator of the subscription |
| | How the newspaper is delivered |
| | The conversions made in distribution channel, payment method & edition |
| | Elapsed time since last conversion in distribution channel, payment method & edition |
| | The number of responses on direct marketing actions |
| | The number of suspensions |
| | The average suspension length (in number of days) |
| | Elapsed time since last suspension |
| | Elapsed time since last response on a direct marketing action |
| | The number of free newspapers |
| Renewal-related variables | Whether the previous subscription was renewed before the expiry date |
| | How many days before the expiry date, the previous subscription was renewed |
| | The average number of days the previous subscriptions are renewed before expiry date |
| | The variance in the number of days the previous subscriptions are renewed before expiry date |
| | Elapsed time since last step in renewal procedure |
| | The number of times the churner did not renew a subscription |
| Socio-demographic variables | Age |
| | Whether the age is known |
| | Gender |
| | Physical person (is the subscriber a company or a physical person) |
| | Whether contact information (telephone, mobile number, email) is available |
| Subscription-describing variables | Elapsed time since last renewal |
| | Monetary value |
| | The number of renewal points |
| | The length of the current subscription |
| | The number of days a week the newspaper is delivered (intensity indication) |
| | What product the subscriber has |
| | The month of contract expiration |

## 4.2 Predictive Maintenance

### 4.2.1 Problem Description

In production and operations management, maintenance is a common practice across all industries and factories. The costs and the implications associated with it will be discussed in this section. To briefly give an overview, it has been found that, depending on the industry, 15% to more than 40% of the production cost is associated with the cost of maintenance [20]. All in all, maintenance management is a complicated and costly function, the key objective of which is to optimise equipment application and overall performance.

To clearly illustrate the impact of the maintenance management in an organisation, we can focus on industrial businesses where machinery equipment is used to a large extent. As in every company, the operating profit must capture and exceed the corresponding cost to the maximum extent possible. Now, to define operational cost, factors such as -but not limited to- equipment ownership, cost of failure and production losses are always involved. In detail, equipment ownership expenses are based on three significant figures, the purchase price, the time of use and, lastly, the costs of maintenance.

As mentioned before, maintenance is a complicated yet critical component of every company. Successful management keeps the operations running smoothly and avoids direct cost and indirect implications associated with equipment failure spreading across the organisation. Immediate costs include spare parts, labor, downtime, overhead expenses, tools and consumables. Spare parts and expertise availability is an issue which further increases complexity and cost. When it comes to the associated implications, there is a plethora of details to consider with production loss and safety coming first. Besides, customer service and firm reputation are almost always vulnerable in such critical situations.

During the past years, two maintenance concepts were mainly met around the industrial world. Due to their importance and extensive use, we briefly present them and describe the key points so we can later fully illustrate the inherent potential in the novel approach.

**Traditional perspective**
Firstly, we refer to the run-to-failure or reactive maintenance. This type of maintenance is based on the reactive approach, i.e., maintenance is only carried

out after failure of the equipment. Despite the high cost and inefficiency of this practice, it is the most frequent one due to its simplicity. With this approach managers are usually forced to high expenses and extended fixing time which leads to extra production costs and lost sales during the downtime period. Apart from that, safety issues are also present since the equipment is running without inspection and maintained under pressure.

Secondly, a more sophisticated method, the Preventive Maintenance (PvM), follows. Here, maintenance is scheduled in advance based on run time or process iterations. Often, problems and failures are prevented but, on the other hand, unnecessary service costs, including labor and spare parts, lead to inefficient resources allocation inflating operational expenses.

**Novel approach**

Technological developments, especially in areas such as automation and data exchange, have contributed to the fourth industrial revolution. Industry 3.0 began when first computers were used in the manufacturing world to ease and increase operations and production. With the rapid raise of technology Industry 4.0 became the new era in which further innovation and optimisation is possible. Now, computers and machines are linked with advanced software which performs analysis for complex decision-making tasks. The concept of Industrial Internet of Things (IIoT) supports this approach by connecting machines with people and sharing data. Smart sensors installed in machinery and maintenance software communicate through network and monitor performance thanks to large amount of data generated. IIoT centralise control and information in one place making it easily accessible and ready for further processing. Given that, innovative tools and technology introduced by Industry 4.0 have facilitated a novel approach in maintenance management, the so-called Predictive Maintenance (PdM).

PdM is the latest approach and a breakthrough in operations management. It is the state-of-the-art technique which makes monitoring equipment health feasible. What is more, PdM introduces a condition-based framework which empowers optimised maintenance budget allocation. Early functional anomalies prediction is possible by utilising historical and real time data. Smart sensors are used to monitor functionality and provide a large amount of variant data, such as vibration signals, temperature and rotation speed, in which useful information about equipment and parts of it is inherent. The potential can be easily seen. Not only because system breakdowns can be predicted and fixed in time but also because periodical unnecessary checks, present in the PvM approach, can be avoided. Lastly, this practice can also be beneficial for equipment performance and its lifespan.

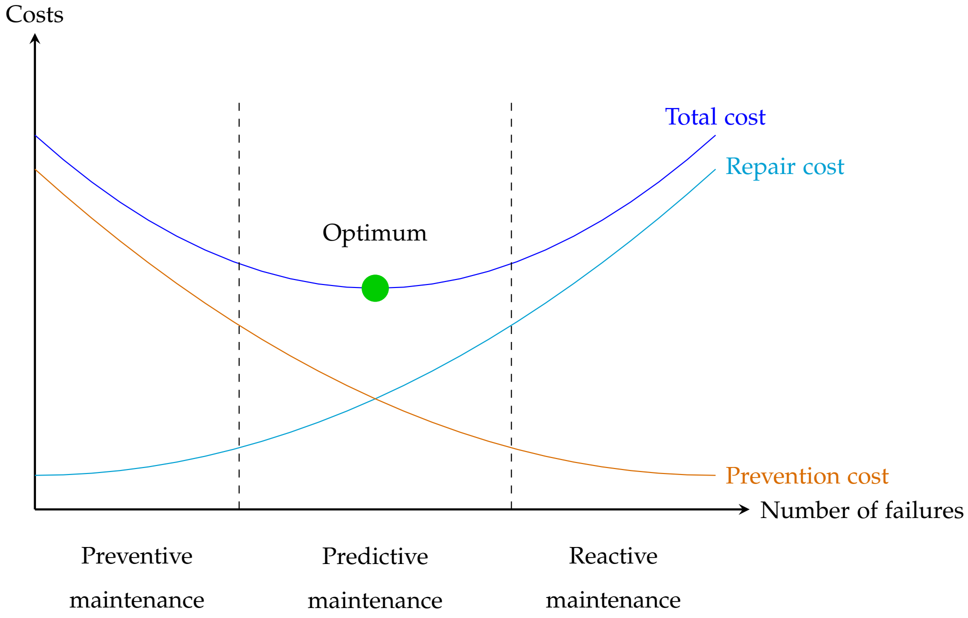


Figure 4.2: Cost of maintenance [30].

### 4.2.2 Predictive maintenance with the support of decision tree algorithms

Across literature, multiple machine learning models have been utilised to address the PdM challenge, however decision trees are often on top of the list. Except for their multiclass classification ability, that one might be in favour in a PdM framework, and their strong predictive power, decision trees have natural interpretation and explanatory properties. They can illustrate and justify predictions based on generated rules and feature selection. The tree structure and splitting procedure are easily understandable and naturally interpreted, making the model preferable when it comes to problem understanding and decision-making. In the predictive maintenance case, rule extraction is a desirable feature. It leads to a better understanding of the the current machinery condition without costly and time-consuming inspections by experts. Furthermore, decision trees can underline feature significance resulting in further information useful in maintenance management.

During the years, advanced tree algorithms have also been introduced for predictive maintenance. For example, the Fuzzy Min–Max-CART is a hybrid tree model which combines the predictive power of neural networks enhanced with a rule extraction mechanism based on decision trees. The Fuzzy Min–Max-CART model has been extensively studied and utilised for online motor fault detection and diagnosis by M. Seera, C. P. Lim and C. K. Loo [44]. In the study, bearings behavior was analysed by monitoring vibration signals. In order to evaluate the model in variant environments, noise signals of 10%, 20% and 30% were manually added. The predictions reached an impressive accuracy from 93.45 (for 30% noise) to 100 (without noise). Furthermore, a useful and easily interpretable tree has been extracted by the model to illustrate the classification procedure and support decision-making.

### 4.2.3 Case Studies

**A low-cost predictive maintenance framework**
As it has been so far discussed, due to increased complexity in manufacturing ecosystems, the predictive maintenance approach gains gradually more attention. Arguing that emerging technologies of industry 4.0 facilitate PdM, E. Sezer, D. Romero, F. Guedea, M. Macchi and C. Emmanouilidis introduced a low-cost easy to develop PdM system tailored for the SMEs domain [46].

The study conducted was based on a Computer Numerical Control (CNC) turning unit producing simple metal parts. For the firm, the high quality of the product is a requirement, therefore roughness is constantly measured based on three variables: roughness average (Ra), root mean square roughness (Rq) and mean roughness depth (Rz). To reduce defective products and machinery damage without unnecessary services, a PdM system was developed utilising low cost hardware and open source software. To this end, a single board computer (Raspberry Pi 3 model B) equipped with a compatible multi-sensor board (Sense HAT) was installed on the CNC's turning centre.

In order to control the production process, two are the main parameters which determine machine's function, speed and feed rate. The first one defines the relative rotational speed of the workpiece, while the second one the relative velocity of the cutting tool. The experiment was run in three cycles where three pairs of values were used for the control parameters, one with high values ({3600, 0.3}), one with medium values ({2600, 0.2}) and one with low values ({1600, 0.1}). For each cycle of the experiment, the flawless and the total products produced was counted until the cutting tool was damaged or broke. At the same time, vibration signals and temperature figures were monitored with the aforementioned setup and data stored in a cloud.

Subsequently, to build the desired predictive framework the following steps were applied: the raw dataset was imported in RStudio for preprocessing. Simple statistical features, like mean, standard deviation and maximum were computed. A uniform data frame among different experiments and feature types was developed, as well as data were cleansed and normalised. Data from the second experiment (medium values) would be utilised to train the model, whilst the decision tree model would be tested with data retrieved from the experiments one and three. Regarding the target variable, it is defined based on a targeted Ra threshold: 1 if Ra exceeds that threshold or 0 otherwise.

In spite of the fact that the framework was developed to illustrate the potential simplicity of the PdM approach and more sophisticated experimentation setups can be formed, results were remarkable: (1) A strong link between temperature, vibration (x-axis) and roughness (Ra variable) was observed. An increase of the former two affected negatively the products surface smoothness warning for the upcoming failure of the equipment. (2) The tree algorithm predicted the turning point from flawless to defective products with an average accuracy of 81%. Finally, the resulting decision tree is available for a root cause analysis by management. To briefly illustrate the decision process, the root node was formed by a split based on temperature mean value ($x1<0.687$ or $x1\geq0.687$). The rest of the nodes were created based on the same variable (with different decision boundary) or by utilising the x-axis vibration (x3) and temperature standard deviation (x2) variables. The tree graph is presented in the Figure 4.3.

**Online bearing fault detection**

According to J. S. L. Senanayaka, H. Van Khang and K. G. Robbersmyr [45], bearings are one of the most critical components of rotating machinery. Time-consuming and costly service interruptions in an industrial machine must be initiated due to a bearing failure. Hence, it is important to detect such faults in advance and perform condition-based maintenance in early damage stages. Given that, the authors introduced a tree classification model aiming to encounter the novel predictive maintenance approach. Furthermore, the framework was designed with respect to the online learning approach. With the term online learning we refer to a real-time learning, a process in which the

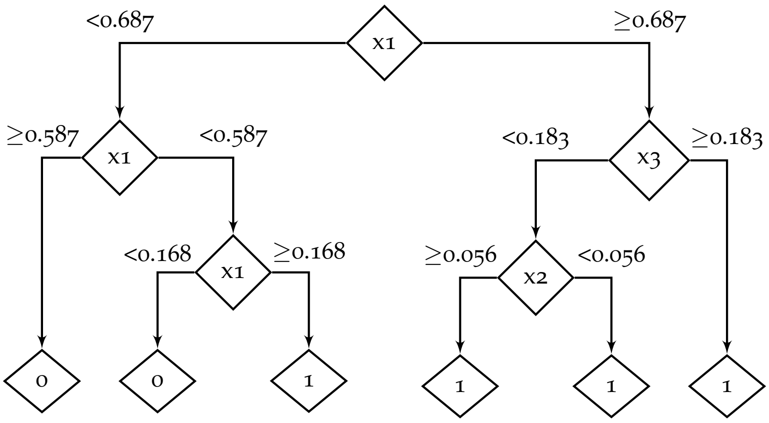


Figure 4.3: The decision tree formed for product defectiveness, where x1 variable represents the mean temperature, x2 the standard deviation of the temperature and x3 the mean X-axis vibration [45].

algorithm changes based on the available input from moment to moment. In such an online system, the main components are the following: the data acquisition system, the database to store data and the learning software installed on a server.

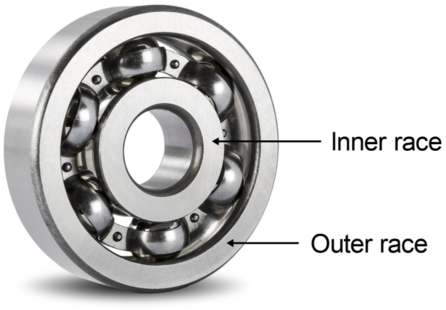


Figure 4.4: Inner and outer race of a bearing.

The main objective of the project was to develop a decision tree capable to classify bearing health statuses with the training input of vibration features. To this end, short duration vibration data was recorded and logged in the system in fixed time intervals. Five frequency and time related features calculated from the bearing physical characteristics, such as geometry and rotational speed, were selected for the model training (Table 4.4). Additionally, the raw vibration signal turned out not to be as useful as the envelope of the signal; as a result, the former was used in the proposed learning framework.

**Table 4.4**. Selected features utilised in the predictive maintenance framework [45].

| Feature | Description |
|---|---|
| RMS | Root mean square of the signal |
| FTF | Energy in fundamental train frequency |
| BSF | Energy in ball spinning frequency |
| BPFO | Energy in ball pass frequency outer race (Figure 4.4) |
| BPFI | Energy in ball pass frequency inner race (Figure 4.4) |

In order to generate input data for the algorithm, a run-to-failure test was conducted with the following set up: four bearings were installed and set to rotate by 2000 rpm while a radial load was applied. To measure vibration signals, eight high sensitivity accelerometers which recorded one-second data every ten minutes and with a sampling frequency of 20kHz were installed. Next, the predictive variables were calculated from the resulted measurements. The dataset has been divided into two equal parts to form the training and test sets for the model. Regarding the target variable, five classes have been formed in order to reflect useful information about the overall health of the spare parts, as well as specific upcoming problems. The five status classes that have been chosen based on bearings functionality can be seen in Table 4.5.

**Table 4.5**. Number of samples and class accuracy [45].

| Class | Description | No of training/<br>test samples | Recall/<br>accuracy |
|---|---|---|---|
| Class 1 | Healthy | 2377 | 98.7 |
| Class 2 | Inner-race degradation (IR_D) | 279 | 84.9 |
| Class 3 | Inner-race Failure (IR_F) | 36 | 91.7 |
| Class 4 | Outer-race degradation (OR_D) | 430 | 97.2 |
| Class 5 | Outer-race Failure (OR_F) | 17 | 94.1 |
| Overall | | 3139 | 97.1 |

The data was labeled accordingly and the model trained on the generated training set. To ensure robustness, the model was then tested on the unseen data. As a result, an impressive overall accuracy of 97.1% was achieved (Table 4.5). The healthy class was detected successfully with about 99% recall. The IR_F, OR_D and OR_F classes scored 91.7, 97.2 and 94.1 in the same figure respectively. Contrary to that, the IR_D class showed a low of 85%, where 41 out of 279 instances were classified wrong as healthy. Besides that, the results demonstrate the power of the framework and, by extension, the potential benefits of the predictive approach in maintenance management area.

**Proactive fault diagnostics of electronic gaming machines**

In gaming industry, defective electronic machines not only lead to lost income for a business but can harm the manufacturer's reputation too. With this in mind, M. Butler and V. Kešelj in "Data Mining Techniques for Proactive Fault Diagnostics of Electronic Gaming Machines" [7] introduced a framework for timely fault detection. The model aims to provide information about the machine's operation status and notify for upcoming abnormal function ensuring a sufficient time to react and schedule maintenance.

To model the problem, event updates from the Electronic Gaming Machines (EGM) were logged to a server. The raw data was collected from two non-sequential one-week periods (Table 4.6) and was processed for features extraction. During these periods, the reported states of the machines in fixed time intervals were counted in order to form the entries for the predictive variable. The authors argue that the health status of a machine is inherent in the number of the occurrence of the 104 available states during these intervals. For the studied classification task, four classes have been chosen: EGMs with no faults (Normal-0), EGMs which end up offline (Abnormal-1), EGMs with fault alarm other than bill acceptor (General False Alarm-98) and EGMs reported bill acceptor problem (Specific False Alarm-99). Two alternative classification methods, DT and SVM, were tested and evaluated in six rounds of the introduced experiment. In addition, 10-fold cross validation was utilised to determine the optimal configurations for the two algorithms and subsequently, the algorithms were trained and tested.

**Table 4.6**. Data used in the predictive maintenance framework [7].

| | Period 1 | Period 2 |
|---|---|---|
| Time | 1 week | 1 week |
| No of data entries | 3,135,508 | 3,107,201 |
| No of machines | 7867 | 7660 |
| No of potential states | 104 | 104 |

For the decision tree the results showed that fault identification is highly probable. A general accuracy of about 90% was achieved. Precisely, in five out of the six datasets about 95% of the instances was classified correctly and only for the one a low of 70% was achieved. However, within all sets the precision for the "Abnormal" class constantly scored above 94%. That underlines the effectiveness of the method while that is the most important metric to avoid the downtime of a machine. Lastly, despite the fact that SVM model showed similar results, as we have seen, the decision trees are easier to understand and visualise. Given that, the authors underline the importance of the interpretation ability of the decision tree algorithm and its capability of extracting simple business rules easy to approach by management teams.

## 4.3 Dynamic Pricing: A Promising Application

### 4.3.1 Problem Description

**Revenue Management**

In 1997, R. Cross referred to Revenue Management as "Revenue Management ensures that companies will sell the right product to the right customer at the right time for the right price" [15]. In general, revenue management aims to optimise firm's operations, including inventory management, pricing, and the cost of production, in order to maximise its generated revenue. However, similar approaches, focussed on turnover optimisation, was well known since the 80's, with the American Airlines being the first company launching such a pricing model to control and manage reservations inventory to increase company's profitability. As a result, the model contributed $1.4 billion over three years at the airline [47].

The success story of the airline undoubtedly raised the popularity of the model, however, traditionally, revenue management techniques have been only applied in airline, hotel, and car rental industries. According to S.E. Kimes common factors of industries which facilitate revenue management are: (1) relatively fixed capacity of the system, (2) demand can be segmented into clearly-identified partitions (price or service time sensitive demand), (3) inventory is perishable, (4) products are sold well in advance, (5) demand fluctuates substantially, and (6) low marginal sale costs and production costs, but high-capacity change costs [29].

Only recently, similar techniques have been adapted from other industries as well. The main reasons contributed in this are: (1) demand data are widely available, (2) prices can easily change with the adoption of new technologies, and (3) the development of advanced software for analyzing demand and dynamic pricing [16]. The most well-known attempt of a smart pricing model in another industry occurred in 2000 when Amazon.com was spotted fluctuating DVD prices based on customer's willingness-to-pay [18]. In 2012, the price of a microwave oven was recorded changing 11 times in single day, fluctuating between $744.46 and $871.49 [5]. By 2014, L2 Think Tank estimated that Amazon.com was adjusting prices 2,500,000 times per day [33].

**The potential of smart pricing**
Assuming a product or service, a decrease in price usually yields, more or less, an increase in demand. Therefore a downward-sloping demand curve in general holds (Figure 4.5). Since the sales generated revenue is the product of price and quantity sold, it is justifiable to try to maximise the revenue by adjusting the price. However, even when optimisation is feasible, the monetary potential of the market cannot be fully captured from a single offer. During the years it was obvious that a better, a smart pricing model needs to be defined. Nowadays, there are two fundamental approaches of Smart Pricing, that is differential and dynamic pricing.

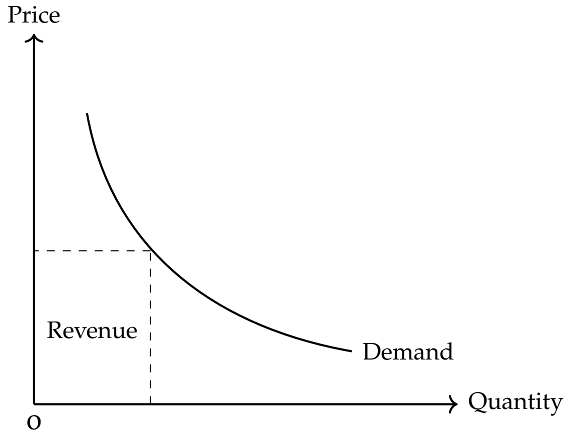


Figure 4.5: Demand curve [2].

Differential pricing aims to charge different prices to different customers. Companies tend to differentiate their products/services in order to meet the demand of different customer segments (Figure 4.6). A typical example of this strategy is met in the airline industry. Companies differentiate the ticket price based on customer type, leisure and business travellers. By providing additional flexibility and comfort, airlines differentiate their services and capture the high-end market resulting in extra profits. Pricing differentiation strategies are applied in other markets as well. Supermarkets and clothing stores use loyal cards so customers can collect points and benefit from future discounts. Frequently, only a segment of the market follows the procedure since it is time consuming and requires effort. This way firms sell the same product on different price capitalising customers' willingness-to-pay.

Now, it gets clearer that in order to apply such strategies a fine partition of the market is needed. Usually, price or service time sensitivity criteria are utilised.

Furthermore, for such an effective strategy, firms need to draw barriers among segments that customers cannot easily cross. The business class benefits and the point collection procedure are "fences" that companies build in order to discourage customers to migrate among segments.

Strategies widely applicable in the Differential Pricing scheme are the following: (1) group pricing (e.g., student discounts and ladies' nights), (2) channel pricing (e.g., online vs brick and mortgage price), (3) regional pricing (based on location), (4) time-based differentiation (e.g., express delivery), (5) product versioning (e.g., leisure and business tickets), and (6) coupons and rebates.

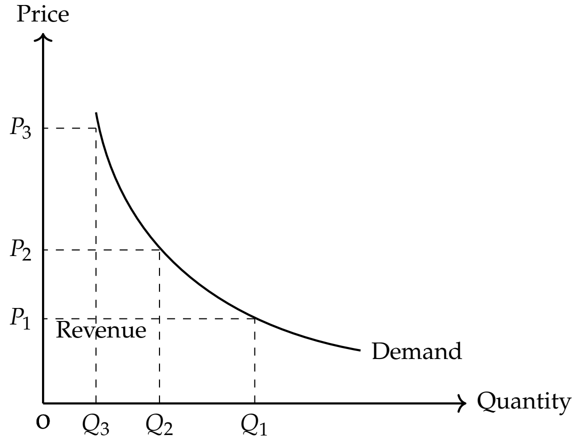


Figure 4.6: Price differentiation.

Dynamic pricing is a strategy where firms charge different prices over time. The demand rarely remains stable and adjustments in pricing are required to maximise revenue in new environments (Figure 4.7). The urge to apply usually lies on market's and industry's potential. Markets with arbitrary or seasonal demand variability, and industries with limited capacity or (short) defined planning horizon belong in the most promising cases. Even the optimal fixed pricing model cannot but capture a moment in time since they remain stationary based on a past state of the market. Dynamic pricing models take into consideration a large number of variables, such as demand, interdependent products, competitor's behavior, customer segment, and inventory in order to define the optimal pricing model in real time.

Internet enables this approach. E-commerce firms track a large number of data, including website traffic, consumer preferences, demographics, inventory, even information regarding their competitors. The ability for fast and, sometime, easy data analysis makes the real-time dynamic pricing feasible

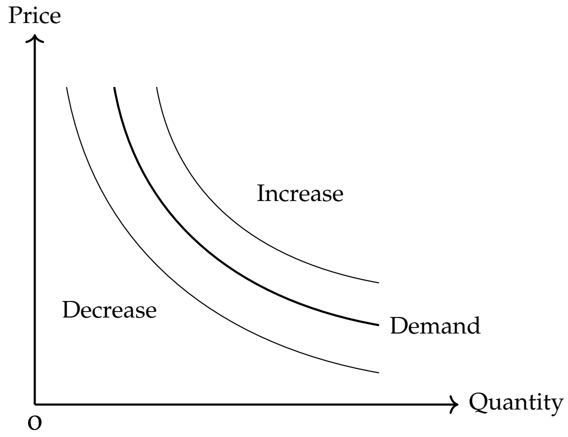


Figure 4.7: Demand fluctuation. An increase or decrease in demand results in suboptimal revenue generation. Prices need to be adjusted dynamically to follow the curve.

[25]. Additionally, the strategy is an emerging trend in brick-and-mortar stores as well [1]. Advanced analytics and technological innovations like electronic shelf labels and smart shelves can bring the opportunities of dynamic pricing to physical stores [4, 41].

According to W. Reinartz [40], there are two forms of dynamic pricing: the weak and the strong form. In the first prices are adjusted over time but remain consistent across customers. The second form describes a practice where prices change over time and among customers. Finally, personalised pricing is the ultimate goal of the strong form. In this case, firms try to define customers' willingness-to-pay and draw personal pricing plans.

### 4.3.2 Dynamic pricing enabled by Q-Learning

By closely studying the dynamic pricing problem and the available literature, we can derive that Q-learning shows the most reliable approach. The main property that leads to this outcome is that Q-learning is a model-free algorithm, hence no prior knowledge of the environment is needed. In the case of dynamic pricing, that is no assumption about the demand curve is required. The model can be trained online and learn the demand dynamically as well as the effect of the price on it. Additionally, the model shows a large possibility for adaptation in a given setting. Despite the long (theoretically infinite) and costly period the model needs to converge, Q-learning seems that detects the most reliable approximation of the optimal pricing policy.

During the years, many variations have been proposed with respect to different industries or markets, but two are the most recurrent assumptions in the Q-learning approach; monopolistic markets and perishable goods. However, there is a large number of articles that encounters more complex settings, e.g., oligopolies [11, 32], non-perishable goods [11, 34], and interdependent products [38].

### 4.3.3 Limitations in Small-Medium Enterprises

As we have discussed, dynamic pricing has a large potential and it is definitely the future for both e-commerce and physical stores, but when it comes to the implementation, managers have a considerable number of issues to consider before put hands on the project [22].

- Despite the fact that Q-learning algorithms are a very promising solution, they need an extended, and probably costly, training period. Such algorithms have to extensively explore the business landscape and experiment with price variations in order to find the optimal pricing pattern. This, requires long time and firm's tolerance in temporary profit loses.
- At the moment there are no empirical evidences of revenue increase by utilising machine learning dynamic pricing models. A large number of articles present ready (more or less) to implement models but does not evaluate their performance on a real market. Usually, simplified simulations with numerous market assumptions are constructed. Common assumptions authors make are:
  - The market consists of myopic (not strategic) consumers.
  - Weak dynamic pricing models are mostly utilised overlooking the potential for finer segmentation of the market.
  - Customers arrive to the store according to a homogeneous or inhomogeneous Poisson process. However, in real life assumptions in both cases are violated.
  - Customers buying behaviour is poorly formed in simplistic market environments.
- There is a lack of standard and universal evaluation metrics. In the literature, many times Q-learning approaches outperform simple static pricing models. However, a comparison study among machine learning models

is not available.

- Every single market has its own characteristics. Due to the multidisciplinary nature of the problem, the contribution of economists, computer scientists and marketers is crucial.
- The customers need to have a perception of fairness. Pricing policies transparency is a crucial issue while poor approaches can damage badly firm's reputation.

### 4.3.4 Case Studies

**Dynamic Pricing in E-commerce**

Online retail markets show a large potential for revenue management. In 2006, C. V. L. Raju, Y. Narahari, and K. Ravikumar [37] described a Reinforcement Learning setting for dynamic pricing with customer segmentation. Two customer types were studied based on their willingness-to-pay; captives and shoppers. As captives are defined the mature, loyal to the seller customers whereas shoppers are price sensitive and attracted by offers.

For simplicity, a monopolistic market with a single retailer with a unique product is examined. However, the seller has two offers available, i.e., a single product at the base price and a "buy three-pay two" promotion. The customers arriving at the store are categorised based on their preference in the aforementioned offers. Captives are interested in the first ignoring the quantity discount, while shoppers would opt for the latter.

Additionally, a limited inventory is assumed which is replenished based on a fixed policy. Since captives are of higher than the shoppers to the seller value, they experience a priority in the available stock as well as a lead time commitment in case of lack of items. In such a case, two virtual queues are formed with a maximum of $N$ backlogged requests to be allowed. The retailer can observe the waiting list but the customers have no information about it.

Finally, customers make a purchase decision only if they derive strictly positive utility. The utility function is defined differently for the two customer types, but, overall, it compares the seller's offer with customers' standards. Captive to the seller customers have the benefit to evaluate both price and lead time quotes provided by the retailer. Additionally, customers in this category are modeled to learn these benefits and tend to revisit the store for future purchases as well. In contrast, shoppers can only compare the price with their willingness-to-pay. In case of no inventory available, they need to place their order in the shopping cart and to revisit the store again later. Then, they check the status of their order. In the case of stock-out, they leave for good. Otherwise, they only buy the items if the price quote provided leads to a positive utility. Finally, they exit the system.

Online retailers with apparel, food, and other types of products could easily fit in the given, or similar, formulation. To test the model in a realistic environment, a simulation of the aforementioned system was designed. Two different evaluation metrics were utilised by the learning agent (the retailer) to define the optimal pricing and the optimal inventory replenishment policy.

The following modeling assumptions were used: the price set available to the seller is set to be $A = \{8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 12.0, 13.0, 14.0\}$ and the maximum number of 10 customers per queue is allowed. The maximum inventory capacity is fixed at 20 units and the reorder point at 10. The 40% of the customers is assumed to be captives, and customers arrive in the store based on a Poisson distribution with mean inter-arrival time 15 minutes. Captives afford to buy an item based on a uniform distribution $U(8, 14)$ and, respectively, $U(5, 11)$ holds for shoppers. Similarly, captives' acceptable lead time follows a $U(0, 12)$ distribution. The inventory gets replenished with an exponential lead time with a mean of 3 hours. Shoppers exit the system after waiting more than a time interval exponentially distributed with a mean of 1,5 hours. The inventory holding cost is 0.5 per unit per day and the backlog costs 0.5 per backlogged order per day. Finally, the merchandise costs a rate of 4 per item to the seller.

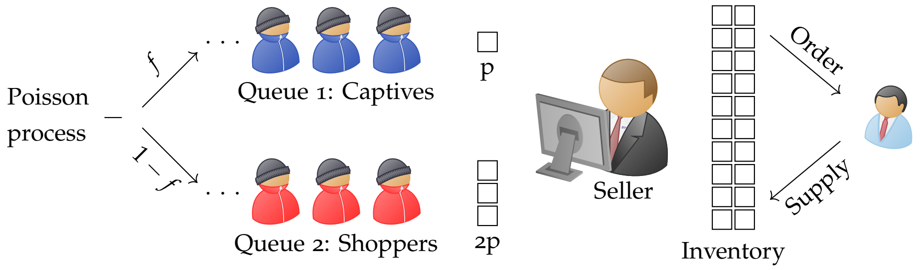


Figure 4.8: The model of the retail store with two customer segments and inventory replenishment [37].

The system consists of a finite number of states which are defined by three parameters: $X_i(.)$ the number of backlogged orders in queue $i = 1, 2$, and $I(.)$ the inventory level at time t. Hence a state is given by a vector $(X_1(t), X_2(t), I(t))$.

Given that setting, a partially observable Markov decision process is formed. An adjustment of the Q-learning algorithm, in order to fit in the continuous time setting, is utilised to learn the best strategy for every estate in the system.

The first experiment was conducted with respect to the long-term discounted profit. The outcome is plotted in Table 4.7 showing the optimal pricing strategy per state. By closely examining the table, three remarks can be made:

1. The price level is high when the inventory level is high as well. This decision seems straight forward since higher holding costs are present.
2. In states $(k, 0, 0), k > 0$ two price groups are observed. In case of low or high number of captives in queue 1, the price is higher compared to the modest case. That is reasonable because in a no inventory and low demand state, the seller can afford to wait for a customer with high willingness-to-pay. On the other hand, when high demand is observed it is reasonable to increase the price as well. However, given a modest situation, due to the inventory holding and backorder costs, the seller will try to increase the demand by lowering the price.
3. Similarly, in case of the states $(0, k, 0), k > 0$, low prices are observed when a low number of shoppers is waiting in the queue 2 and higher for $k > 5$. That is explained simply by assuming that in case of high demand there are shoppers who are willing to pay a higher price.

These trends are directly connected to the studied problem. By adjusting the parameters of the system, we can simulate another market and the opposite trends might be found. That is the power of the model which can adapt to the given environment and provide decision support in non-trivial settings.

For the second group of experiments, the same setting is retained except that now shoppers' willingness-to-pay fluctuates uniformly in $(8, 14]$ ($(5, 11]$ previously), and the long run average profit per unit time is used as the evaluation metric. The authors tend to examine different inventory policies in order to find the optimal reorder point. Based on the dynamics of the system, it was found that the seller should place an order of 18 items when his inventory drops at the level of 2. The new policy resulted in a long run average profit per unit time of 8.410. In contrast, the previous $(10, 10)$ policy showed a way lower rate of 5.748.

Furthermore, one more case, in which the promotion is also dynamically adjusted, was studied. That leads to a more complex, two-dimensional $A \times A_v$

**Table 4.7**. The optimal stationary pricing policy learned by the agent. We underline with blue, green and red the states which correspond to the remarks 1, 2 and 3 respectively [37].

| Best price | States of the system |
|---|---|
| 8.0 | (0,0,0), (0,1,0), (0,2,0), (0,3,0), (0,4,0), (0,5,0), (1,1,0), (1,3,0), (1,6,0), (1,7,0), (2,3.0), (2,5,0), (2,6,0), (2,7,0), (2,8,0), (2,10,0), (3,1,0), (3,3,0), (3,10,0), (4,2,0), (4,3,0), (4,4,0), (4,5,0), (4,8,0), (4,10,0), (5,0,0), (5,4,0), (6,4,0), (6,9,0), (7,2,0), (7,4,0), (8,3,0), (8,5,0), (9,3,0), (10,1,0), (0,5,1), (0,6,1), (0,7,1), (0,0,2), (0,1,2), (0,3,2), (0,4,2), (0,5,2), (0,10,2) |
| 8.5 | (1,4,0), (1,5,0), (1,9,0), (2,4,0), (2,9,0), (3,2,0), (3,7,0), (3,8,0), (4,7,0), (5,5,0), (5,7,0), (5,9,0), (6,0,0), (6,1,0), (6,2,0), (6,3,0), (6,6,0), (7,1,0), (7,7,0), (7,9,0), (8,6,0), (8,7,0), (10,4,0), (0,2,1), (0,3,1), (0,2,2), (10,6,2), (0,7,2), (0,9,2), (0,0,3) |
| 9.0 | (1,2,0), (1,8,0), (3,4,0), (3,5,0), (5,2,0), (5,3,0), (5,6,0), (6,5,0), (7,0,0), (7,6,0), (8,2,0), (9,4,0), (0,8,2) |
| 9.5 | (2,0,0), (2,1,0), (3,6,0), (4,1,0), (5,1,0), (6,8,0), (7,10,0), (8,1,0), (8,8,0), (9,1,0), (10,8,0) |
| 10.0 | (3,9,0), (4,0,0), (5,8,0), (9,2,0), (9,5,0), (0,1,1) |
| 10.5 | (2,2,0), (4,6,0), (6,10,0), (7,5,0), (9,6,0), (9,7,0), (9,8,0), (10,2,0), (0,10,1) |
| 11.0 | (1,7,0), (6,9,0), (8,4,0), (8,0,0) |
| 12.0 | (0,8,0), (5,10,0), (6,7,0), (8,9,0), (10,0,0) |
| 13.0 | (0,7,0), (7,3,0), (7,8,0), (9,10,0), (10,5,0), (0,0,6), (0,0,16) |
| 14.0 | (0,6,0), (0,9,0), (0,10,0), (1,0,0), (3,0,0), (4,9,0), (9,0,0), (8,10,0), (9,9,0), (10,6,0),<br>(10,7,0), (10,9,0), (0,0,1), (0,8,1), (0,9,2), (0,0,4), (0,0,5), (0,0,7), (0,0,8), (0,0,9), (0,0,10), (0,0,11), (0,0,12), (0,0,13), (0,0,14), (0,0,15), (0,0,17), (0,0,18), (0,0,19), (0,0,20) |

grid, where $A_v = \{2.0/3, 2.1/3, 2.2/3, 2.3/3, 2.4/3\}$ is the promotion offered. In every epoch, the seller chooses a pair $(p, d_v) \in A \times A_v$. Finally, in this scenario the inventory replenishment $(15, 5)$ policy is selected. The resulted long run average profit per unit time found to be 11.515.

# Chapter 5

# Experiment with SME Data

## 5.1 Introduction

To gain further insights regarding the potential, but also the challenges in SME domain, in this chapter, we apply a customer churn prediction framework with real world SME data. To this end, data from a small company in Greece producing handcrafted articles is utilised. The company holds no more than the trivial information on its customers and their transactional behaviour. Therefore, a small amount of data, concerning both entries and features, is available for processing. Mainly, customer orders stored in the firm's archive are used to form the input of the learning algorithms. Having that said, two models widely used in the literature for churn detection are utilised in the experiment, the SVM and the decision tree algorithms. Regarding the proposed experimental design, both the nature of the business and the data available have been taken into consideration. With the given setting, the results showed that detecting potential churners is highly probable with the algorithms performing overall similarly well.

## 5.2 Modeling Techniques

As it has been seen, the SVM algorithm has shown a large potential in binary customer classification tasks such as churn detection. However, more factors need to be considered in order to take full advantage of the model. Based on the discussion held in Section 2.2.3, the structure of the classes within the dataset varies, and commonly the two classes might overlap; therefore, there is a need for various kernel functions to be tested when a selection for the SVM algorithm is made. The Kernel is a crucial component that can drastically affect the classification outcome and hence model's reliability; given that, three different functions are tested for the SVM model, that is linear, radial basis and a third-degree polynomial function, with the corresponding models denoted as SVM$_{linear}$, SVM$_{rbf}$ and SVM$_{poly}$ respectively. In addition, since $C$ and $\gamma$ parameters also play a major role in the decision boundary formed, a 5-fold cross validation on the training set is utilised to tune and eventually optimise models' performance in the available dataset.

Furthermore, another model extensively discussed for its accuracy, simplicity and interpretability, the decision tree algorithm, will be benchmarked against the SVM classifiers. Also in this case, given the different gain criteria available, a 5-fold cross validation technique will be applied on the training set. During the process, the maximum depth of the generated tree is also examined. This is in order to avoid overfitting the training set that could result in poor generalisation power of the model.

## 5.3 Empirical Analysis

### 5.3.1 Data

In this experiment, real world data from a small manufacturing unit in Greece is utilised. The raw data available is stored in a Microsoft Access database which is mainly used by the firm to monitor customer orders. The database consists of 827 unique customers and about 3000 transactions. The company serves both individuals and business customers therefore information regarding both is present in the database. Given the scope of the project, only information regarding business customers was extracted and processed. In addition, only data from 2016 to 2019 has been retrieved for the experiment. Observations with missing information, and those with no valid orders in years 2016, 2017 and 2018 were deleted as well.

The final dataset includes 530 unique retailers, from 32 countries around the globe, and their transactional records from January 1, 2016 to December 31, 2018. During this three-year period, 1740 orders, including a couple of cancelled ones, had been placed.

Due to simplicity, the limited data available and the fact that the firm operates based on an annual pattern (participation in international trade shows with seasonal peaks and lows), we have decided to split the dataset based on years and consider the customers present in multiple ones as unique per year. By utilising this approach, the final dataset consists of $N = 869$ customers and their one-year purchases.

### 5.3.2 Experimental Design

**Mathematical framework:** For each customer $i$, $i \in [m]$, we define $\mathbf{x}_i$ to be a d-dimensional vector of features, and denote with $x_{i,j}$ the $j-th$ entry for the $i-th$ customer. This results in a two-dimensional matrix with customers in rows and predictive variables in columns. To define the prediction classes, we simply denote loyal clients as $y = 0$ and churners as $y = 1$. This leads to a typical binary classification problem from $\mathcal{X} \rightarrow \{0, 1\}$.

**Definition of churn:** Due to the nature of the products, i.e., towards high-end, long shelf life and particular designs, and firm's strategy, non-advertising and preference for local stores and boutiques, a low order frequency is typical across clientele. Additionally, the fact that the firm ensures competition-free exclusiveness in retailers' areas for a year, leaded to the decision not to define churn in a period shorter than this time interval. What is more, to formulate our churn prediction framework, transactional records from a single year were used for each customer and we defined churn based on their transactional behavior on the following year. Therefore, those with at least one transaction in the second consecutive year are defined as loyal customers, and those with no valid orders are considered as churners.

**Training and test set:** In this study, contrary to the typical case, the dataset seems to be balanced with respect to churn. With the aforementioned design, the churn rate has been calculated to be pretty high, close to 40%. Precisely, for the years 2017, 2018 and 2019 the actual numbers are 43%, 46%, and 35% respectively. To reflect the temporal nature of the data and test the robustness of the model, we have decided to train our models in the first two years and test them in the most recent one. Eventually, this leads to two sets with 588 training and 281 test points respectively (Table 5.1).

**Data preprocessing:** Both personal and transactional information is stored in company's data warehouse. Regarding personal information, almost no work needed to be done. However, transactional data needed extra attention in order to retrieve the required information and form the predictive variables. Subsequently, simple statistical features, like mean and maximum, have been computed to form predictive variables. Finally, the aforementioned variables are prepared according to algorithms' specific requirements to ensure a reliable outcome.

**Table 5.1**. Training and test set distribution with respect to churn variable.

| Class | Number of observations | Relative percentage |
|---|---|---|
| *Training set* | | |
| Churners | 262 | 45 |
| Loyal customers | 326 | 55 |
| Total | 588 | 100 |
| | | |
| *Test set* | | |
| Churners | 99 | 35 |
| Loyal customers | 182 | 65 |
| Total | 281 | 100 |

**Model selection:** In order to find the optimal hyperparameters for the models we apply a 5-fold cross-validation on the training set. For every model, a multidimensional grid that contains all the possible combinations of hyperparameter values is constructed. The algorithms are trained with all the available combinations, and the final configurations for the models are selected based on the maximum cross-validated accuracy.

**Model testing:** To benchmark the final classifications, four evaluation metrics are utilised: accuracy, f1-measure, recall and precision. Accuracy, the percentage of correctly classified observations is our main criterion. Since detection of the churn class is our primal concern, recall plays a significant role in model evaluation as well.

**Software and hardware:** Microsoft Access is used to retrieve the raw data from the company's database. The predictive variables are formed in Microsoft Excel and exported as csv format in order to import them in Python with the support of Pandas toolkit [39]. The Scikit-learn, a machine learning package for Python, is finally used for the implementation of the experiment [35]. For all the experiments, a gaming laptop with an Intel Core i7 processor operating at 2.60GHz and a 16GB RAM has been utilised.

### 5.3.3 Data Preprocessing and Variables Formation

In the company's data warehouse information regarding both customers and orders is stored. The customer related data available is the following: customer id, company name, country, city, address, post code, email, phone number and VAT id. Regarding their orders we have: order id, date of order, value, freight costs, execution date, way of order, products, quantities and cost per unit. That information is preprocessed for variable extraction that will be later utilised in the prediction model. Overall, mainly behavioral data is utilised, since the company does not hold more than the trivial personal information about its customers. Precisely, only the address, in country level, can provide extra insights given that EU customers experience lower freight and tax rates and no quotas. No other useful information can be leveraged from these figures, therefore, we mainly focussed on the behavioral data from orders available.

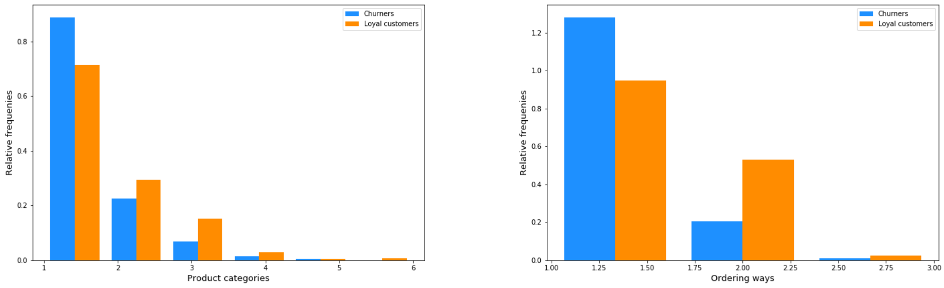
We divided the predictive variables into four categories as follows: personal information, orders-related, product-related and service-related variables (Table 5.2). In the first one, little information regarding the customer itself is present, only customer id, country and region. The second group includes all the features related to customers transactional behaviour during the studied time interval. That is: frequency, spending rate, annual spending and order cycle. The products-related variables are the following: product categories, quantity rate and quantity. Lastly, the service-related variables are service speed, maximum delay, freight rate, total freight, ordering ways and abandoned transactions.

**Table 5.2.** Variables used for the churn prediction framework

| Category | Variables |
|---|---|
| *Independent variables* | |
| Personal information | Customer id (the unique number associated with every customer in firm's database) |
| | Country (the country where a customer mainly operates at) |
| | Region (whether customer's country belong to EU or not) |
| Order -related variables | Frequency (number of valid transactions during a year) |
| | Spending rate (average value spent per transaction) |
| | Annual spending (total value of transactions within a year) |
| | Order cycle (average elapsed time between two consecutive transactions) |
| Products -related variables | Product categories (number of different categories products have been purchased from) |
| | Quantity rate (average number of products purchased per transaction) |
| | Quantity (number of products purchased in a year) |
| Service -related variables | Service speed (average elapsed time from order to dispatch) |
| | Maximum delay (maximum elapsed time from order to dispatch) |
| | Freight rate (average freight cost per dispatch) |
| | Total freight (total freight cost paid by the customer in the given period) |
| | Ordering ways (number of different ways used for placing an order) |
| | Abandoned transactions (number of transactions cancelled before execution) |
| *Dependent variables* | |
| | Churn (whether a customer leaves company for good or not) |

Both categorical and continuous variables are used in the study. In order to visualise the influence of the categorical features on the target classes, side-by-side bar charts of relative frequencies are plotted (Figure 5.1). Similarly, for the continuous variables, the density functions are benchmarked against churn (Figure 5.2). In both cases, extreme values have been removed for the better interpretation of the data points.

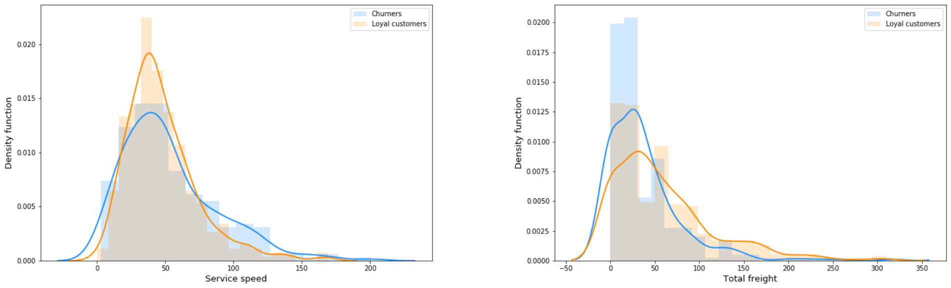
Finally, given that the SVM algorithm is distance sensitive, i.e., strives to maximise the distance between the decision boundary and the support vectors, it performs better on data that have the properties of a standard normal distribution with mean equal to 0 ($\mu = 0$) and, standard deviation to 1 ($\sigma = 1$). Given that this condition is not always met by default, a technique called standardisation is applied. According to it, the predictive variables $\mathbf{x}_j$ are mapped to $\mathbf{z}_j$ through the transformation $\mathbf{z}_j = \frac{\mathbf{x}_j - \mu_j}{\sigma_j}$. To improve performance, standardisation is applied on the dependent variables prior to feeding the SVM models (Figure 5.3).



(a) Product categories

(b) Ordering ways

Figure 5.1: Side-by-side bar charts of relative frequencies with respect to the target class.



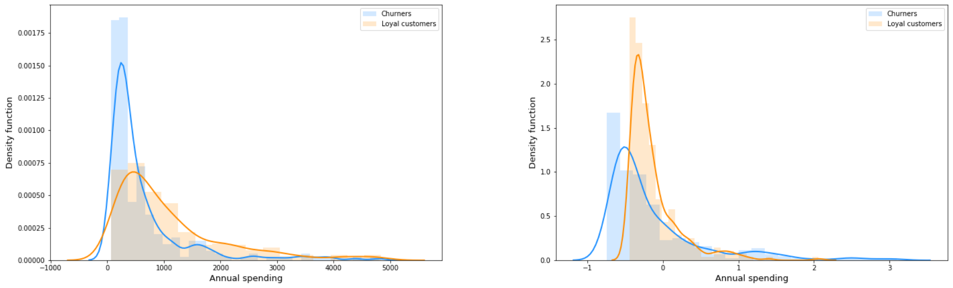
(a) Service speed

(b) Total freight

Figure 5.2: Side-by-side plots of density functions with respect to the target class.

### 5.3.4 Hyperparameter Tuning

Now, in order to find the optimal hyperparameters for the models, we apply a grid search and a 5-fold cross-validation on the training set. To this end, two grids were constructed for the SVM models. A grid search on $C$ and class weights utilised for the linear SVM, while for the radial and polynomial models the hyperparameter $\gamma$ was added in the grid. In both cases, we chose exponential sequences of $C$ and $\gamma$, i.e., $C = 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{13}$ and $\gamma = 2^{3}, 2^{1}, 2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}, 2^{-13}, 2^{-15}$. This is a typical choice in churn prediction frameworks [19][14] and a practical solution for SVM hyperparameter tuning in general [23]. All the $(C, \gamma)$ pairs were tested for different class weights to find the optimal configurations for each model.



(a) Initial distribution

(b) Standardised distribution

Figure 5.3: Density function of annual spending variable with respect to target class, before and after standardisation of the data.

Respectively, for the decision tree algorithm we tested various values for the following hyperparameters: gain criterion, max tree-depth and class weights. The sets which resulted in the highest cross-validated accuracy were used to finally train and test the models.

## 5.4 Results

### 5.4.1 Cross-validation Results

From the hyperparameter tuning for the linear SVM, it yields that $2^{-5}$ is the optimal value for C, with a cross-validated accuracy of 70.91%. Additionally, no class weight should be used to train the model. Table 5.3 shows the results of the grid search on C where the cross-validated accuracy used as the evaluation metric.

**Table 5.3**. Cross-validated accuracy for SVM$_{\text{linear}}$ per $C$ (no class weight).

| $C$ | $2^{-5}$ | $2^{-3}$ | $2^{-1}$ | $2^{1}$ | $2^{3}$ | $2^{5}$ | $2^{7}$ | $2^{9}$ | $2^{13}$ |
|---|---|---|---|---|---|---|---|---|---|
| | **70.91** | 70.73 | 70.56 | 70.40 | 58.67 | 50.33 | 52.71 | 55.43 | 52.89 |

Regarding $SVM_{\text{rbf}}$, no class weights should be used according to the results. In addition, the optimal $(C, \gamma)$ pair appears to be $(2, 2^{-7})$, with the maximum accuracy observed to be 70.91%. The cross-validated accuracy for all the pairs can be found in Table 5.4. Similarly, the selected hyperparameters for the $SVM_{\text{poly}}$ are $(2^{-1}, 2^{-3})$, which resulted in a 70.05% score of the same metric (Table 5.5). However, in this case, the model seems to be in favor of balanced classes, so that configuration was utilised during the model training. Finally, grid search for the tree algorithm showed that entropy should be used as a gain criterion, and the maximum depth of the tree should be 3.

**Table 5.4**. Cross-validated accuracy for $\text{SVM}_{\text{rbf}}$ per $(C, \gamma)$.

| $\gamma \setminus C$ | $2^{-5}$ | $2^{-3}$ | $2^{-1}$ | $2^{1}$ | $2^{3}$ | $2^{5}$ | $2^{7}$ | $2^{9}$ | $2^{13}$ |
|---|---|---|---|---|---|---|---|---|---|
| $2^{3}$ | 55.47 | 55.13 | 60.56 | 62.25 | 61.74 | 61.06 | 61.75 | 61.75 | 54.42 |
| $2^{1}$ | 55.47 | 59.21 | 62.59 | 63.44 | 63.10 | 62.59 | 61.73 | 61.74 | 43.85 |
| $2^{-1}$ | 55.47 | 62.26 | 66.32 | 64.28 | 62.25 | 61.05 | 61.22 | 61.56 | 55.78 |
| $2^{-3}$ | 54.96 | 66.83 | 68.02 | 68.02 | 64.62 | 61.73 | 60.02 | 56.95 | 57.82 |
| $2^{-5}$ | 55.47 | 66.16 | 69.72 | 69.72 | 69.04 | 68.52 | 66.48 | 64.44 | 50.87 |
| $2^{-7}$ | 55.47 | 55.13 | 69.89 | **70.91** | 70.56 | 69.55 | 68.87 | 68.36 | 50.49 |
| $2^{-9}$ | 55.47 | 55.47 | 56.66 | 70.06 | 70.57 | 70.57 | 69.89 | 69.20 | 53.39 |
| $2^{-11}$ | 55.47 | 55.47 | 55.47 | 56.32 | 69.89 | 70.91 | 70.56 | 70.40 | 63.43 |
| $2^{-13}$ | 55.47 | 55.47 | 55.47 | 55.47 | 56.32 | 69.89 | 70.91 | 70.73 | 70.40 |
| $2^{-15}$ | 55.47 | 55.47 | 55.47 | 55.47 | 55.47 | 56.49 | 69.89 | 70.91 | 70.56 |

**Table 5.5**. Cross-validated accuracy for $\text{SVM}_{\text{poly}}$ per $(C, \gamma)$.

| $\gamma \setminus C$ | $2^{-5}$ | $2^{-3}$ | $2^{-1}$ | $2^{1}$ | $2^{3}$ | $2^{5}$ | $2^{7}$ | $2^{9}$ | $2^{13}$ |
|---|---|---|---|---|---|---|---|---|---|
| $2^{3}$ | 48.27 | 51.02 | 52.38 | 53.90 | 53.90 | 53.90 | 53.90 | 53.90 | 52.89 |
| $2^{1}$ | 55.59 | 47.95 | 48.62 | 48.27 | 51.02 | 52.38 | 53.90 | 53.90 | 53.90 |
| $2^{-1}$ | 68.18 | 64.95 | 59.16 | 55.59 | 47.95 | 48.62 | 48.27 | 51.02 | 53.90 |
| $2^{-3}$ | 58.49 | 69.03 | **70.05** | 68.18 | 64.95 | 59.16 | 55.59 | 47.95 | 48.27 |
| $2^{-5}$ | 45.89 | 46.74 | 52.36 | 58.49 | 69.03 | 70.05 | 68.18 | 64.95 | 55.59 |
| $2^{-7}$ | 43.34 | 43.51 | 45.38 | 45.89 | 46.74 | 52.36 | 58.49 | 69.03 | 68.18 |
| $2^{-9}$ | 48.80 | 48.80 | 48.80 | 43.34 | 43.51 | 45.38 | 45.89 | 46.74 | 58.49 |
| $2^{-11}$ | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 43.34 | 43.51 | 45.89 |
| $2^{-13}$ | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 43.34 |
| $2^{-15}$ | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 | 48.80 |

In Table 5.6, all the cross-validation results, i.e., the capability of the learning algorithm to fit the training data, can be found. We see that the linear and radial SVM performed similarly, with 70.91% accuracy, while they slightly outperformed the polynomial model which barely reached 70%. Finally, the decision trees showed same accuracy with the former SVM models (70.90%). All the classifiers, except $\text{SVM}_{\text{poly}}$, seem to fit the target class relatively well with the scored recall equal to 80.20%.

**Table 5.6**. Cross-validation results based on accuracy as evaluation metric.

| Model | Configurations | Recall | Precision | F1 score | Accuracy |
|---|---|---|---|---|---|
| $\text{SVM}_{\text{linear}}$ | $(2^{-5}, -)$, None | **80.20** | 63.05 | 70.39 | **70.91** |
| $\text{SVM}_{\text{rbf}}$ | $(2, 2^{-7})$, None | **80.20** | 63.05 | 70.39 | **70.91** |
| $\text{SVM}_{\text{poly}}$ | $(2^{-1}, 2^{-3})$, Balanced | 75.44 | **63.40** | 68.57 | 70.05 |
| DT | entropy, 3, None | **80.20** | 63.09 | **70.40** | **70.90** |

As we have extensively discussed, analysts are commonly in favor of decision tree models due to their advantage, which is no other than the clear interpretation of the classification procedure. With respect to that, the resulted decision tree, with the split criteria and the training instances per node, can be found in Figure 5.4. As it can be seen, order cycle, i.e., the average elapsed time between two consecutive transactions, is the most significant variable since it is placed at the root of the generated DT. In detail, the first question is formed as "Is the order cycle less or equal to 252.5 days?". For those cases that the answer is "True", they are moving to the next node where loyal customers outnumber churners. Whereas, the remaining instances are driven to another node mainly consisting of churners. Similarly, other criteria such as spending rate, frequency, and region of the customer follow in the splitting process. Finally, seven leaves are formed with three of them being labelled as Loyal, including 256 data points, and the rest four as Churner, consisted of 332 samples.

An additional useful insight that can be gained from the plot is the confidence of a classification based on the purity of the nodes. For instance, entropy close to 0 indicates homogeneous leaves and the corresponding points are more likely to be correctly classified. On the other hand, entropy figures closer to 1 correspond to leaves containing points with mixed labels and high uncertainty regarding the classifications outcome.

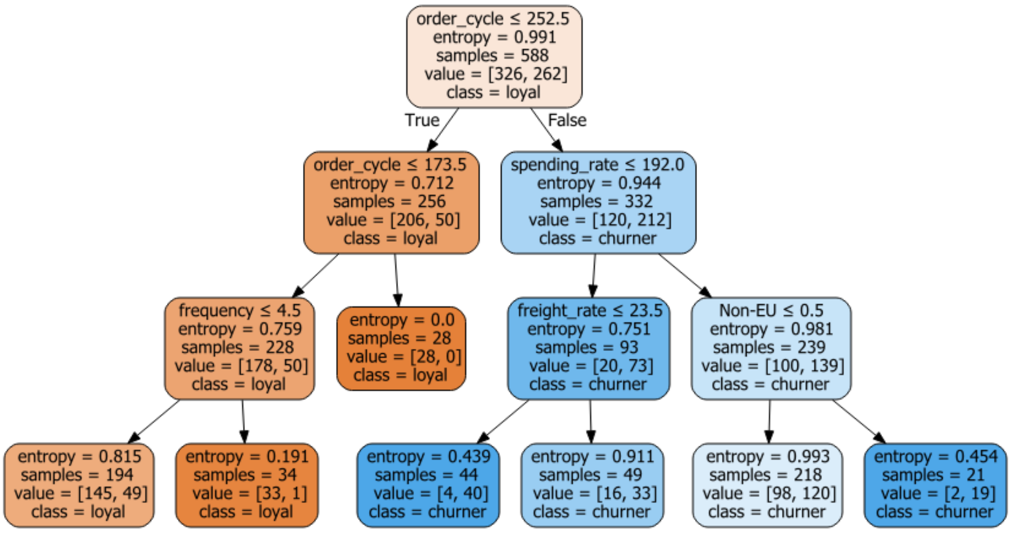


Figure 5.4: Resulted decision tree classifier.

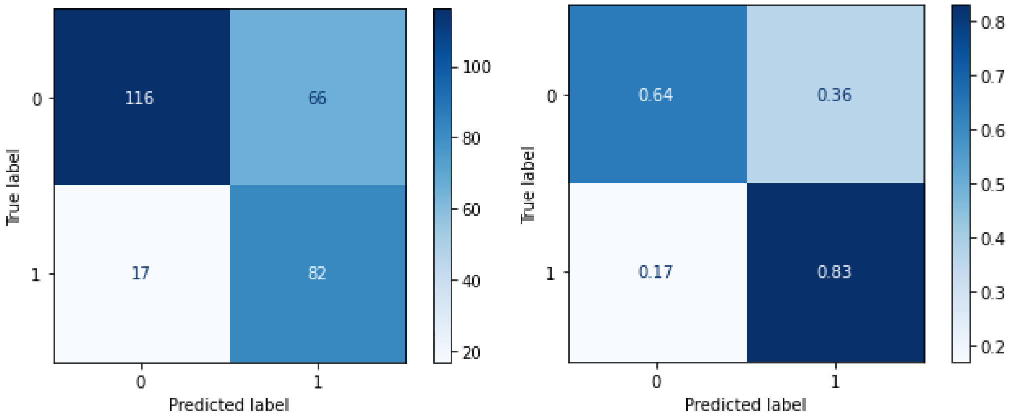
### 5.4.2 Test Results

All four models were tested on the unseen data of 2018. The linear SVM reached slightly lower accuracy than the cross-validation testing, whereas it outperformed the other methods in all four metrics. Precisely, the accuracy of the model resulted in being 70.46% while the recall reached 82.83%. The $\text{SVM}_{\text{rbf}}$ and the tree relatively underperformed the linear model with accuracy for both close to 70%. Nevertheless, $\text{SVM}_{\text{rbf}}$ seem to detect the churn class analogously well to the first model with recall at the same level. The DT model almost reached 82% at the same metric. Concerning $\text{SVM}_{\text{poly}}$, all the aforementioned models outperformed this algorithm. It barely missed 70% of accuracy and scored a low of 76.77% of recall. A summary of the precise figures per classifier is presented in Table 5.7.

**Table 5.7.** Test results.

| Model | Recall | Precision | F1 score | Accuracy |
|---|---|---|---|---|
| $\text{SVM}_{\text{linear}}$ | **82.83** | **55.41** | **66.40** | **70.46** |
| $SVM_{\text{rbf}}$ | **82.83** | 55.03 | 66.13 | 70.11 |
| $SVM_{\text{poly}}$ | 76.77 | 55.07 | 64.14 | 69.75 |
| DT | 81.82 | 55.10 | 65.85 | 70.11 |

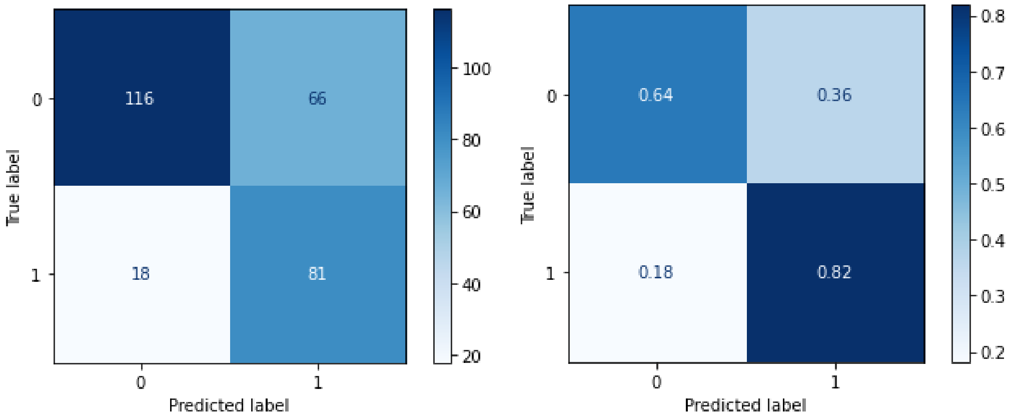
The actual classification achieved from the $\text{SVM}_{\text{linear}}$ is shown in a confusion matrix (Figure 5.5). 82 out of 99 churners were correctly classified whereas, 116 loyal customers (64%) was placed in the right class as well. The corresponding RBF model almost achieved identical results, with only one additional loyal customer being misclassified as a churner. Similarly, benchmarking the decision tree against the linear SVM, the classified instances overlapped with only one churner less being detected in the tree case. The corresponding confusion matrix is shown in Figure 5.6. Finally, when it comes to the SVM polynomial model, results varied significantly. Again, with respect to the linear model, 17 instances were reversely classified. In detail, SVM poly missed 6 churners and 4 loyal customers in addition. However, the classifier managed to detect 7 specific loyal clients that the rest of the models failed. Given the overall performance and the increased significance of the recall, the poly algorithm achieved the weakest classification.



(a) Number of customers

(b) Normalised values

Figure 5.5: Confusion matrix of linear SVM on unseen data.



(a) Number of customers

(b) Normalised values

Figure 5.6: Confusion matrix of decision trees on unseen data.

## 5.5 Conclusions and limitations in the experimental design

Overall, the selected classifiers showed robustness with the results being similar across the SVM and the tree algorithms. Given that the dataset was relatively balanced with respect to churn, and despite the limitations of the data, all models managed to cope reasonably well with the given problem. The most scored above 70% accuracy and about 82% recall. These results highlight the effectiveness of the methods applied and validate our initial assumption that the firm can potentially detect customers prone to churn in advance. For the experiment, any specific data collection routine prior to the implementation was absent, and only data that the company held at hand was utilised for feature extraction. To a large extent, this reflects the actual situation in MSEs, so the results confirm that even such an approach can potentially generate value for businesses in the domain.

To further comment on the data, limitations were faced regarding both quality and quantity, thus, we highlight the following points. The fact that the company only partially records personal information about its customers, excluding relevant features, such as gender and age, eliminates churn determinant insights that might potentially arise from the corresponding underling patterns. The company should take actions to improve this part since such information is easy to obtain during the creation of customer profile and could also directly benefit the firm in its corporate communication. In the same direction concerning behavioural data, in Section 4.1.3 we discussed the potential towards churn identification hidden in customers' online behaviour. However, in our set of data such information is not at all present. Nowadays, numerous low-cost or even free platforms and applications enable the approach of online behaviour tracking. The company should grasp that opportunity that will not only improve the discussed classification task but, will lead to a deeper customer understanding overall and support decision making as well.

What is more, the small size of data available significantly limited our alternatives in the experimental design. Based on our approach, we considered customer annual entries as independent from the previous years. As a result, past knowledge that could have been accumulated and utilised in the prediction task is eliminated due to the definition of the studied instances. However, the results showed that churn determinant information was present in the data. With respect to the literature, we have also confirmed the importance of the data and the formed features, which finally, end up being two of the most critical elements of a predictive framework.

The simplicity of the implementation of the application is another remarkable point. Once the objective and the desired approach was determined, the implementation of the predictive framework itself was a simple, straightforward and free of cost procedure. An analyst with relatively little study on the problem and the firm specifics can successfully complete such a task. To strengthen this point, the source code developed in Python for the radial SVM model can be found in Appendix B. The simplicity of the approach is also highlighted by the relatively small size of the script and the small number of steps followed to the final results.

# Chapter 6

# Conclusions

To conclude our long discussion on the machine learning applications in Small-Medium Enterprises, we can primarily remark on the large potential laying on the business domain. Precisely, we have seen that SME ML solutions have been only researched and applied to a small extent. This encloses the prospective of a competitive advantage against competition in key operation areas.

Looking at the literature, SMEs remain widely an unexplored area, however we can highlight the following points that show that the trend cannot but grow in the future. To begin, referring back to the case studies but also to the experiment, we saw that machine learning models have the power to learn how to perform complex -for the humans- tasks resulting in reliable outcomes valuable in businesses' decision-making. Furthermore, the rapid growth of technological equipment as well as open-source applications have made ML extensively accessible with low cost. Given that, small and medium businesses can approach the new for them technology with the minimum risk inherent in the investment. Finally, ML has been proven to be intuitive. The relatively easy interpretation of the algorithms and their outcome makes them suitable for the business management that often lacks deep scientific background.

Concerning the requirements, in a very high level, we can say that the proper problem identification, the set of a clear objective, and the availability of data are crucial elements for a successful implementation of ML techniques in SMEs. Once again, we would like to mention the importance of studying the problem's nature and highlight that different type of problems required different solution methods. Nowadays, a large range of ML algorithms is available to encounter an extensive number of problems. Now, it is worth mentioning that we should not trust results blindly, and analysts need to be aware of the problem, the objectives set, and the models' specifics in order to form a valuable for the firm framework. Of course, the maturity of a company regarding data collection is another critical factor that can determine the outcome of such an approach. However, we have seen that it is probable the right information to be present in the small and little data that SMEs hold. It is then analysts' job to dig in the datasets and leverage any useful information that the set can provide.

As a concluding reflection, we would like to encourage small and medium companies to approach machine learning technologies and discover themselves possibilities to improve operations and strategic decision-making. As an outcome of this thesis, we can confirm the positive perspective that businesses in the domain have with no other than their own willingness to discover the potential of this emerging technology being the most crucial factor.

# Bibliography

[1] (2020) How Dynamic Pricing Can Save Brick-and-Mortar Retail in 2020. https://www.365retail.co.uk/how-dynamic-pricing-can-save-brick-and-mortar-retail-in-2020/

[2] Berdine, G. (2014). Supply and Demand: Government Interference with the Unhampered Market in US Health Care. The Southwest Respiratory and Critical Care Chronicles, 2(7), 21-24.

[3] Berger, P. D., & Nasr, N. I. (1998). Customer lifetime value: Marketing models and applications. Journal of interactive marketing, 12(1), 17-30.

[4] Bringing Dynamic Pricing to Brick-and-Mortar. https://www.intel.com/content/www/us/en/retail/merchandising-inventory-management/dynamic-pricing.html

[5] Angwin, J. & Mattioli, D. (2012). Coming soon: Toilet paper priced like airline tickets. Retrieved from https://www.wsj.com/articles/SB10000872396390444914904577617333130724846 [Accessed: 2020-06-24].

[6] Burez, J., & Van den Poel, D. (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3), 4626-4636.

[7] Butler, M., & Kešelj, V. (2010, May). Data mining techniques for proactive fault diagnostics of electronic gaming machines. In *Canadian Conference on Artificial Intelligence* (pp. 366-369). Springer, Berlin, Heidelberg.

[8] Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3), 27. Software available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[9] Chang, W., Chang, C., & Li, Q. (2012). Customer lifetime value: A review. *Social Behavior and Personality: an international journal*, 40(7), 1057-1064.

[10] Chen, Z. Y., Fan, Z. P., & Sun, M. (2012). A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. *European Journal of operational research*, 223(2), 461-472.

[11] Chinthalapati, V. L. R., Yadati, N., & Karumanchi, R. (2006). Learning dynamic prices in multiseller electronic retail markets with price sensitive customers, stochastic demands, and inventory replenishments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(1), 92-106.

[12] Colgate, M. R., & Danaher, P. J. (2000). Implementing a customer relationship strategy: The asymmetric impact of poor versus excellent execution. *Journal of the Academy of Marketing Science*, 28(3), 375-387.

[13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273-297.

[14] Coussement, K., & Van den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with application*s, 34(1), 313-327.

[15] Cross, R. G. (2011). *Revenue management: Hard-core tactics for market domination.* Currency.

[16] Elmaghraby, W., & Keskinocak, P. (2003). Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management science*, 49(10), 1287-1309.

[17] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 861-874.

[18] Garbarino, E., & Lee, O. F. (2003). Dynamic pricing in internet retail: effects on consumer trust. *Psychology & Marketing*, 20(6), 495-513.

[19] Gordini, N., & Veglio, V. (2017). Customers churn prediction and marketing retention strategies. An application of support vector machines based on the AUC parameter-selection technique in B2B e-commerce industry. *Industrial Marketing Management*, 62, 100-107.

[20] Han, T., & Yang, B. S. (2006). Development of an e-maintenance system integrating advanced techniques. *Computers in Industry*, 57(6), 569-580.

[21] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

[22] Hilsen, H. O. Ø. (2016). Simulating Dynamic Pricing Algorithm Performance in Heterogeneous Markets (Master's thesis, NTNU).

[23] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. 2003.

[24] Jabłońska, M. R., & Pólkowski, Z. (2017). Artificial Intelligence-based Processes in SMEs. *Studies & Proceedings of Polish Association for Knowledge Management*, (86).

[25] Jayaraman, V., & Baker, T. (2003). The Internet as an enabler for dynamic pricing of goods. *IEEE Transactions on Engineering Manageme*nt, 50(4), 470-477.

[26] Kahreh, M. S., Tive, M., Babania, A., & Hesan, M. (2014). Analyzing the applications of customer lifetime value (CLV) based on benefit segmentation for the banking sector. *Procedia-Social and Behavioral Sciences*, 109, 590-594.

[27] Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago.

[28] Kim, E. (2013). Everything you wanted to know about the kernel trick. Url: `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`.

[29] Kimes, S. E. (1989). Yield management: a tool for capacity-considered service firms. *Journal of operations management*, 8(4), 348-363.

[30] Klanker, G., Oslakovic, I. S., & Palic, S. S. (2017). The impact of different maintenance policies on owners costs: Case Studies from Croatia and the Netherlands. In Proceedings of Joint COST TU1402–COST TU1406–IABSE WC1 Workshop: The value of SHM for the reliable Bridge Management, 1-3.

[31] Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear programming, in (j. neyman, ed.) proceedings of the second berkeley symposium on mathematical statistics and probability. University of California Press, Berkeley.

[32] Kutschinski, E., Uthmann, T., & Polani, D. (2003). Learning competitive pricing strategies by multi-agent reinforcement learning. *Journal of Economic Dynamics and Control*, 27(11-12), 2207-2218.

[33] McLean, M. (2014). Insights on amazon's dynamic pricing. `https://vimeopro.com/l2inc/amazon/video/103542251` [Accessed: 2020-06-24].

[34] Narahari, Y., Raju, C. V. L., Ravikumar, K., & Shah, S. (2005). Dynamic pricing models for electronic business. *Sadhana*, 30(2-3), 231-256.

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.

[36] Quentin, J. M., Huysa, B., Anthony, C., & Peggy, S. (2014). Reward-Based Learning, Model-Based and Model-Free. Encyclopedia of Computational Neuroscience.

[37] Raju, C. V. L., Narahari, Y., & Ravikumar, K. (2006). Learning dynamic prices in electronic retail markets with customer segmentation. *Annals of Operations Research*, 143(1), 59-75.

[38] Rana, R., & Oliveira, F. S. (2015). Dynamic pricing policies for interdependent perishable products or services using reinforcement learning. *Expert systems with applications*, 42(1), 426-436.

[39] Reback, J., McKinney, W., Den Van Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 3. Zenodo.

[40] Reinartz, W. (2002). Customizing prices in online markets. *Symphonya. Emerging Issues in Management*, (1), 55-65.

[41] Roose, S. (2017). Dynamic Pricing is Also Possible in Physical Stores: Here's How. `https://www.omniaretail.com/blog/pricing/dynamic-pricing-physical-stores`.

[42] Ross, S. M. (2014). *Introduction to stochastic dynamic programming*. Academic press.

[43] Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[44] Seera, M., Lim, C. P., & Loo, C. K. (2016). Motor fault detection and diagnosis using a hybrid FMM-CART model with online learning. *Journal of intelligent manufacturing*, 27(6), 1273-1285.

[45] Senanayaka, J. S. L., Van Khang, H., & Robbersmyr, K. G. (2017, August). Towards online bearing fault detection using envelope analysis of vibration signal and decision tree classification algorithm. In *2017 20th International Conference on Electrical Machines and Systems (ICEMS)*, 1-6.

[46] Sezer, E., Romero, D., Guedea, F., Macchi, M., & Emmanouilidis, C. (2018, June). An industry 4.0-enabled low cost predictive maintenance approach for smes. In 2018 *IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 1-8.

[47] Smith, B. C., Leimkuhler, J. F., & Darrow, R. M. (1992). Yield management at American airlines. *Interfaces*, 22(1), 8-31.

[48] Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. Cambridge: MIT press.

[49] Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4), 279-292.

[50] Yu, X., Guo, S., Guo, J., & Huang, X. (2011). An extended support vector machine forecasting framework for customer churn in e-commerce. *Expert Systems with Applications*, 38(3), 1425-1430.

[51] Zhu, B., Baesens, B., & vanden Broucke, S. K. (2017). An empirical comparison of techniques for the class imbalance problem in churn prediction. *Information sciences*, 408, 84-99.

# Appendix A

# Proofs of the theorems, propositions, and lemmas

### Proposition 2.1

*Proof.* The distance between the point $\mathbf{x}$ and the hyperplane is given by

$$\min\{||\mathbf{x}-\mathbf{v}|| : \langle \mathbf{w}, \mathbf{v}\rangle + b = 0\} \ .$$

By setting $\mathbf{v} = \mathbf{x} - (\langle \mathbf{w}, \mathbf{x}\rangle + b)\mathbf{w}$ we observe that

$$\langle \mathbf{w}, \mathbf{v}\rangle + b = \langle \mathbf{w}, \mathbf{x} - (\langle \mathbf{w}, \mathbf{x}\rangle + b)\mathbf{w}\rangle + b = \langle \mathbf{w}, \mathbf{x}\rangle - \langle \mathbf{w}, \mathbf{x}\rangle ||\mathbf{w}||^2 - b||\mathbf{w}||^2 + b = 0 \ .$$

Therefore $\mathbf{x} - (\langle \mathbf{w}, \mathbf{x}\rangle + b)\mathbf{w}$ is laying on the hyperplane and its distance from $\mathbf{x}$ is

$$||\mathbf{x}-\mathbf{v}|| = ||\mathbf{x} - \mathbf{x} - (\langle \mathbf{w}, \mathbf{x}\rangle + b)\mathbf{w}|| = |\langle \mathbf{w}, \mathbf{x}\rangle + b| ||\mathbf{w}|| = |\langle \mathbf{w}, \mathbf{x}\rangle + b| \ .$$

Now, we just need to show that this is the minimal distance. To this end, we assume any other point $\mathbf{u}$ on the hyperplane, hence $\langle \mathbf{w}, \mathbf{u}\rangle + b = 0$. It derives that

$$\begin{aligned}
||\mathbf{x}-\mathbf{u}||^2 &= ||\mathbf{x}-\mathbf{v}+\mathbf{v}-\mathbf{u}||^2 \\
&= ||\mathbf{x}-\mathbf{v}||^2 + ||\mathbf{v}-\mathbf{u}||^2 + 2\langle \mathbf{x}-\mathbf{v}, \mathbf{v}-\mathbf{u}\rangle \\
&\geq ||\mathbf{x}-\mathbf{v}||^2 + 2\langle \mathbf{x}-\mathbf{v}, \mathbf{v}-\mathbf{u}\rangle \\
&= ||\mathbf{x}-\mathbf{v}||^2 + 2(\langle \mathbf{w}, \mathbf{x}\rangle + b)\langle \mathbf{w}, \mathbf{v}-\mathbf{u}\rangle \\
&= ||\mathbf{x}-\mathbf{v}||^2 + 2(\langle \mathbf{w}, \mathbf{x}\rangle + b)(\langle \mathbf{w}, \mathbf{v}\rangle - \langle \mathbf{w}, \mathbf{u}\rangle) \\
&= ||\mathbf{x}-\mathbf{v}||^2 + 2(\langle \mathbf{w}, \mathbf{x}\rangle + b)(-b+b) \\
&= ||\mathbf{x}-\mathbf{v}||^2
\end{aligned}$$

□

# Appendix B

# Source code

```
#import packages
import numpy as np
import pandas as pd

#preprocessing
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler

#models selection
from sklearn.model_selection import ParameterGrid, KFold

#import classifiers
from sklearn import svm

#import metrics
from sklearn.metrics import accuracy_score, f1_score, recall_score,
    precision_score, confusion_matrix

#import train and test sets
train_set = pd.read_csv('train_set.csv')
test_set = pd.read_csv('test_set.csv')

#define predictive and target variables in the train and test sets
X = train_set.drop(['customer_id', 'country', 'churn'], axis=1)
y = train_set['churn']
X_test = test_set.drop(['customer_id', 'country', 'churn'], axis=1)
y_test = test_set['churn']
```

```
#define columns transformation function for standardisation and
    one-hot-encoding
svm_columns_trans = make_column_transformer(
                  (OneHotEncoder(), ['region']),
                  (StandardScaler(), X.columns[1:]),
                  remainder='passthrough')

#define k-fold cross-validation
nfolds = 5
kf = KFold(n_splits=nfolds, shuffle=True, random_state=1992)

#insert metrics for model evaluation
metrics = [recall_score, precision_score, f1_score, accuracy_score]

#apply columns transformation
X = svm_columns_trans.fit_transform(X)
X_test = svm_columns_trans.fit_transform(X_test)
y=y.to_numpy()
y_test=y_test.to_numpy()

#define hyperparameters grid
svm_param_grid = {
       'C': [ 2**(-5), 2**(-3), 2**(-1), 2**(1), 2**(3), 2**(5),
           2**(7), 2**(9), 2**(13)],
       'gamma': [2**(3), 2**(1), 2**(-1), 2**(-3), 2**(-5), 2**(-7),
           2**(-9), 2**(-11), 2**(-13), 2**(-15)],
       'class_weight' : [None, 'balanced', {0:1, 1:2}]}
hyperpar = ParameterGrid(svm_param_grid)

#k-fold cross-validation
res = np.zeros((nfolds, len(metrics), len(hyperpar)))

for fold, (train_index, val_index) in enumerate(kf.split(X), 1):
    for index in range(len(hyperpar)):

       print('Splitting the validation set....')
       X_train = X[train_index]
       y_train = y[train_index]
       X_val = X[val_index]
       y_val = y[val_index]

       print('Model training....')
```

```
        model = svm.SVC(C=hyperpar[index]['C'], kernel='rbf',
            gamma=hyperpar[index]['gamma'],
            class_weight=hyperpar[index]['class_weight'],
            random_state=1992, max_iter=1000)
        model.fit(X_train, y_train)

        print('Inference phase....\n')
        y_pred = model.predict(X_val)
        res[fold-1,:,index] = [metric(y_val, y_pred) for metric in
            metrics]

#average over k-folds to obtain cross-validation results
res = np.mean(res, axis=0)

#optimal results based on cross-validated accuracy
res[:,np.argmax(res[3,:])]

#optimal hyperparameters for SVM
opt_index = np.argmax(res[3,:])
C_opt = hyperpar[opt_index]['C'];
gamma_opt = hyperpar[opt_index]['gamma']
weight_opt = hyperpar[opt_index]['class_weight']
print(' C = ', C_opt, '\n',
      'gamma = ', gamma_opt, '\n',
      'class weight = ', weight_opt)

#train and test model with optimal hyperpameters
svm_rbf = svm.SVC(C=C_opt, kernel='rbf', gamma=gamma_opt,
    class_weight=weight_opt, random_state=1992, max_iter=1000)
svm_rbf.fit(X,y)
y_pred_test = svm_rbf.predict(X_test)

#test results
print(' Recall: ', recall_score(y_test, y_pred_test),'\n',
      'Precision: ', precision_score(y_test, y_pred_test),'\n',
      'F1 measure: ', f1_score(y_test, y_pred_test),'\n',
      'Accuracy: ', accuracy_score(y_test, y_pred_test),'\n',
      'Confusion matrix:\n', confusion_matrix(y_pred_test, y_test))
```