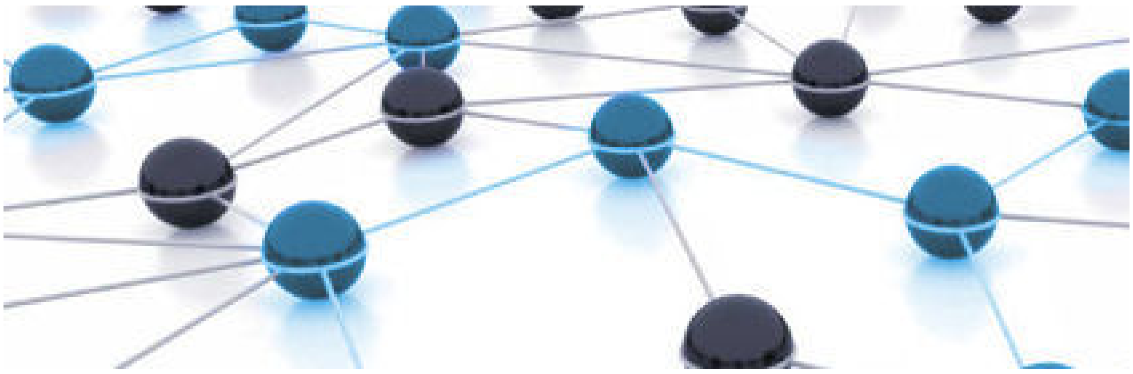


Predict political orientation of Twitter users

Text analysis and classification using machine learning

André Gonçalves da Silva Lince de Faria

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Rui Fuentecilla Neves

May 2018

Acknowledgments

First I would like to thank my thesis advisor, Prof. Rui Neves, for his weekly help and feedback during the course of this work where he consistently guided me in the right direction and provided exceptional and critical feedback in times of need.

Also, I would like to thank my colleagues and friends for their support and advice, not only during the development of the thesis, but during all of my academic path.

Finally, I must thank my family for providing me guidance and support throughout my academic and personal life and especially during the process of researching and writing this document.

Abstract

Social Network sites like Facebook or Twitter have been widely adopted by politicians for the disclosure of personal opinions and for advertising political campaigns with the objective of increasing the potential electorate. Nowadays the polls system is not the most reliable source of information regarding the arrangement of dependable data capable of allowing the political parties to switch strategies with confident expectation for better results. This thesis presents a algorithmic system capable of realizing a prediction regarding the political orientation of Twitter users by comparing the text publicized by each one to those of political parties in attempt to close this gap of unqualified information.

To realize the prediction of political orientation, a popular group of well known machine learning algorithms such as Neural Networks and Support Vector Machines were implemented and orchestrated together into a single algorithm so that in the end, the most accurate prediction could be accomplished. The final solution correctly classified 85% of political linked users as left or right wing supporters considering a training score of 70%, and considered based on a large sample of Portuguese Twitter users with unknown political side that 82% are possibly left wing oriented.

Keywords

Political orientation; Social Networks; Twitter; Machine Learning; Text classification.

Resumo

As Redes sociais como o Facebook ou Twitter foram amplamente adotadas por políticos e correspondentes partidos para divulgação de opiniões partidárias e propaganda de campanhas políticas com a finalidade de aumentar o potencial eleitorado. Para um partido político, compreender a população atempadamente é extremamente vantajoso pois permite que os mesmos dirijam a sua campanha política por forma a melhor satisfazer as necessidades da população.

Hoje em dia, o sistema de sondagens não é a fonte mais confiável de informação ṕrevia, uma vez que não permite que os partidos tomem ações decisivas com um grau de certeza devidamente justificado.

Esta tese apresenta um sistema algorítmico capaz de realizar uma previsão sobre a orientação política de utilizadores da plataforma Twitter, comparando o texto divulgado por cada utilizador com os correspondentes a partidos políticos na tentativa de encontrar uma fonte mais confiável de informação à parte das sondagens politicas.

Para realizar a previsão da orientação política, um grupo popular de algoritmos de inteligência artificial, como Redes Neurais e *Support Vector Machines*, foi implementado e orquestrado num único algoritmo capaz de prever com maior precisão a orientação politica dos utilizadores do Twitter. A solução final classificou corretamente cerca de 85% dos utilizadores politicos como sendo apoiantes das frentes de esquerda e direita considerando um resultado de 70% de precisão na fase de treino, e demonstrou baseado numa amostra de utilizadores representativos Portugueses com orientação politica desconhecida *à priori*, que cerca de 82% são apoiantes de partidos politicos esquerdistas.

Palavras Chave

Orientação politica; Redes sociais; Twitter; Inteligência artificial; Classificação de texto.

Contents

List of Figures

List of Tables

Acronyms

General

API	Application Programmable Interface
GUI	Graphical User Interface
IND	Undefined
IP	Internet Protocol
IPIP	International Personality Item Pool
IR	Information Retrieval

Political

BE	Left Block
CDS-PP	People's Party
PAN	People Animals Nature
PCP-PEV	Unitary Democratic Coalition
PS	Socialist Party
PSD	Social Democrat Party

Machine learning

KNN	K-Nearest Neighbors
L-BFGS	Limited memory Broyden–Fletcher–Goldfarb–Shanno
MLP	Multi-Layer Perceptron

NB	Naive Bayes
NLP	Natural Language Processing
OVA	One-Versus-All
RAKE	Rapid Automatic Keyword Extraction Algorithm
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
WEKA	Waikato Environment for Knowledge Analysis

1 Introduction

Contents

It's known that most people nowadays own a very rich online background, enabled by the appearance of social networks in the end of the 20th century [1], which holds useful latent information that many third parties would like to make use of.

Twitter, a popular micro-blogging website, is one of those networks trusted by a vast amount of users (reaching the 300 thousands [2]) that depend on it to share and provide first person impressions about current events while triggering discussions.

These kinds of social networks, such as Twitter and Facebook, are known to be used by politicians, which understood the potential of such websites for the advertising of political campaigns, an event that is most noticeable during election period. Political discussions on Twitter have been studied over the past years with the sole purpose of identifying ulterior characteristics of people in order to clarify their political position with higher degree of certainty than the antique polls methodology. The target of such studies wasn't always the same, where some attempted solely to gather textual properties of individual *tweets* [3], others endeavored to boldly predict the results of actual elections [4] [5] [6].

The techniques used behind the scenes to gather this kind of information rely heavily on machine learning algorithms which use the textual content of *tweets* in order to guess the political orientation of the corresponding user.

The approach taken by this thesis, motivated by previous history, was to implement such system in the Portuguese election system making use of social activity information from Twitter. The system provides direct classification output in regard to the political side of a user, given the provided *tweets*, while considering the motivations of 4 popular machine learning algorithms pointing their successes and shortfalls.

1.1 Motivation

The main motivation for the development of this thesis was the implementation and testing of an artificial intelligence algorithm capable of understanding Portuguese Twitter users in regard to their political positions.

It's well known that sometimes the polls system results are deviated from the truth which leads to the thinking of new possibilities to obtain more trusted and less subjective conclusions. Previous work demonstrated the potential of an algorithmic approach in alternative to traditional polls system, therefore the challenge of taking the first step towards implementing an artificial intelligence system in this context was the main motivational factor for pursuing this path.

1.2 Purpose

The main goal of this thesis is to create a system that uses machine learning and data mining techniques to predict the political orientation of Twitter users based on the textual content of their *tweets*.

To achieve this goal the system considers input from 4 different machine learning algorithms well regarded in the realm of text processing and classification. Each particular algorithm is trained with manually annotated data from political associated users and provides output together with the rest to classify a given user or group of users using solely the information from their *tweets*.

As each algorithm is trained using *tweets* with respect to the Portuguese lexicon it's expected to highlight defects and advantages to be explored further in future developments of this type.

1.3 Contributions

The main contributions of this work are:

- The application of a social mining system in the Portuguese national and political context.
- The unveiling of limitations along with possible resolutions and alternatives to take into account in works that follow this one in the same context.
- The joint application of multiple machine learning algorithms for social mining providing a comparative scenario between each one.

1.4 Outline

- Chapter 2 addresses the theory behind the developed work, such as the algorithmic concepts, techniques and most important terms, as well as the set of studies that motivated the one at hand.
- Chapter 3 illustrates and explains the methodology pursued together with the implemented architecture of the solution.
- Chapter 4 describes the metrics used to rate the solution and discusses the results obtained.
- Chapter 5 provides conclusions for this thesis, debating achievements and suggestions for future work.

2 Related Work

Contents

In this section some works and methodologies that inspired and stimulated the development of this thesis are referenced and evaluated considering its final objective: Predict the political alignment of Twitter users.

First, the social networking topic will be explored focusing on key aspects such as the impact that such platforms have in today's world and how it relates and provides a mean for politicians to explore and bring their campaigns forward.

Next the complete set of machine learning algorithms that compose the final solution developed in the scope of this work will are addressed, with focus on the key aspects of each algorithm functionality in relation to Natural Language Processing (NLP) applications.

For closing, a couple of research papers developed by fellow scholars who already addressed this topic in some depth will be referenced and analyzed altogether.

2.1 Social Networks

Commonly referred to as *virtual communities* or *profile sites*, a social network is a website that brings people together to talk, share ideas and interests and to ultimately make new friends.

The social networking movement was mainly pioneered by Friendster, which was considered the first social network of the masses, and today, virtual communities like Facebook, Twitter and LinkedIn are the dominant forces on the social internet having an estimate from 400 million to 1.2 billion registered users whereas Facebook leads the chart with roughly 1.2 billion users [2].

The instant adherence of people to social networking platforms proved to be an opportunity for businesses and for each professional to market their product and themselves while significantly reducing the effort to do so while increasing their target audience [7]. Politicians in particular took notice of the great potential yielded in social platforms as most users employ them for political discussions and communications [8]. As such, politicians started directing their political activities and campaigns towards social networks being that nowadays every politician and political party maintains an account on Facebook,Twitter or other social networking platform where they share their schedule with the expectancy to communicate to their electorate and appeal to supporters [9] [10].

To this end, politicians need to identify the potential electorate among different users on the network platform. The most active networks nowadays which employ political discussions are Facebook and Twitter each with subtle differences and limitations referenced next.

2.1.1 Facebook

Facebook is the largest and most recognized social network operating in the Internet today. The network was founded in 2004 by Mark Zuckerberg, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and

Chris Hughes [11]. When Facebook first appeared (named *thefacebook*) its initial purpose was to serve as Harvard's internal social network with restricted access from the outside.

The Facebook social anatomy differs slightly from Twitter's being that, in first instance, the networking object of it are people whereas Twitter focus is directed to the sharing of ideas instead of promoting people. The amount of information available on Twitter regarding a particular user pales in comparison to the amount of data accessible on Facebook, which hosts not only the user textual interventions but also his photos and matters of liking, which were at some point easily accessible by third parties making the platform a target for data mining applications.

The initial objective of this work was to explore this easily accessible gap on Facebook, in resemblance to previous works, where a consumer program would extract and consequently mine the information of each user assigning him or her to the corresponding political side. However, due to the consequences of recent events, namely the suspense regarding the association of Donald Trump and Hilary Clinton election campaigns to Facebook and the Cambridge Analytica case of user privacy abuse made it impossible to pursue this path as the Facebook developers Application Programmable Interface (API) restricted the access to such applications.

One powerful feature once exploited by previous scholars was the Facebook liking system, where a user leaves *likes* in matters of interest providing any consumer application insight regarding the user tastes and interests.

2.1.2 Twitter

Twitter is another popular social networking site where users share short messages called *tweets* to the public. A key point in regard to the working of this platform is that each user's information, being his *tweets* or his personal information (which is limited), is public by default. This fact, together with the large amount of users present on the platform made Twitter a worthy alternative for the pursue of this work of political data mining.

In contrast to Facebook, a Twitter post, known as *tweet*, is limited to 140 characters. This limitation on characters imposes the content of the *tweet* itself to be more concise and directed, inciting the user to sometimes make use a particular textual feature which is the *hashtag*. An *hashtag* is a short token started by an hashtag sign followed by some short text holding a particular meaning. For example the usage of the *hashtag #tbt* holds the meaningful expression *Throw Back Thursday* in fewer characters.

The developed system does not take particular attention to the hashtags written by users meaning no special consideration is given to this kind of information in comparison to the rest of the *tweet's* content. In the future this could be considered as a possible improvement point for this application.

Twitter also allows its users to reference someone in the content of a *tweet*, a feature which is called a *mention*, enabling a reference to a specific person by specifying the *at* sign followed by a given username as in *@donaldtrump*. This functionality in resemblance to the latter is not consciously explored, being only covered in the generated feature set.

2.2 Algorithms

In this section the main algorithms and techniques orchestrated together in the implemented system are described and analyzed, briefly exposing for each one the core functionalities and way of working.

2.2.1 RAKE - Rapid Automatic Keyword Extraction

Extracting keywords is one of the most important tasks when working with text. Readers benefit from keywords because they can judge more quickly whether the text is worth reading, websites gain from it because they can group similar texts by their contents, and algorithm programmers profit from it as the text dimensionality gets reduced to the most important features [12] [13] [14]. Keywords are applied frequently to improve the functionality of Information Retrieval (IR) systems. For example, for search mechanisms based on documents and corresponding content that make use of keywords in order to find similar papers. The *modus operandi* of nowadays search engines, like Google or Bing which rely heavily on keyword extraction and indexing, are the most notorious [15]. Keyword based indexing systems have even been applied in the field of biology to perform the annotation of the biological function of different protein sequences [16].

The Rapid Automatic Keyword Extraction Algorithm (RAKE) algorithm is a recently developed method used for keyword extraction released in 2015 [17] and it's based on the fact that keywords frequently contain multiple words but rarely contain standard punctuation or stop words or others with minimal lexical meaning. In order to understand the behavior of RAKE the following explanation will consider a text quote from a Facebook post by Mark Zuckerberg as below:

> Tonight, a Brazilian judge blocked WhatsApp for more than 100 million people who rely on it in her country. We are working hard to get this block reversed. Until then, Facebook Messenger is still active and you can use it to communicate instead. This is a sad day for Brazil. Until today, Brazil has been an ally in creating an open Internet. Brazilians have always been among the most passionate in sharing their voice online. I am stunned that our efforts to protect people's data would result in such an extreme decision by a single judge to punish every person in Brazil who uses WhatsApp. We hope the Brazilian courts quickly reverse course. If you're Brazilian, please make your voice heard.

The RAKE algorithm receives as input data a list of language-specific stop words, and set of phrase and word delimiters covering some punctuation terms:

- Stop words: *of, is, to, the, this, and, me, that, a*, etc...
- Phrase delimiters: *, ; . - ? !*
- Word delimiters: as an illustrative example the word *Wi-Fi* has the hyphen sign as a word delimiter.

Upon starting keyword extraction on a document, RAKE parses its content into a set of candidate keywords. First the document content is split into an array of words by the specified word delimiters. After its split into sequences of contiguous words at phrase delimiters and stop word positions. Words within a sequence are assigned the same position in the text and together are considered as a candidate keyword:

> Tonight – Brazilian judge blocked Whatsapp – 100 million people – rely – country – working hard – block reversed – Facebook Messenger – active – communicate instead – sad day – Brazil – Brazil – ally – creating – open internet Brazilians – always – most passionate – sharing – voice online – stunned – efforts – protect people's data – result – extreme decision – single judge – punish every person – Brazil – WhatsApp – hope – Brazilian courts quickly reverse course – Brazilian – please make – voice heard.

All stop words, word delimiters and phrase delimiters disappeared from the original post leaving only meaningful words and compositions. The next step of the RAKE algorithm procedure is to create a graph of co-occurrences based on the candidate set that was extracted, and assign to each term a score defined as the sum of its member word scores. Many evaluation metrics can be used for calculating word scores like Term Frequency (TF), Term Frequency–Inverse Document Frequency (TF-IDF) or even by using the *Wordscores* algorithm [18]. Still, this explanation will consider sum of co-occurrences method visible through a co-occurrence matrix as illustrated in table 2.1.

Considering the biggest numbered cluster, involving the words *brazillian, judge, blocked* and *WhatsApp*, all four words are extracted together as there are no stop words nor phrase or word delimiters in between words. The word *brazillian* occurs three times in the whole post and occurs once for each word in the sentence. The case of adjoining keywords can be problematic because RAKE breaks candidate keywords by stop words which means extracted keywords do not contain interior stop words and therefore expressions like *self–hosted* would be lose their meaning. To find these adjoining keywords RAKE looks for pairs of keywords that adjoin one another at least twice in the same document in the same order. A new candidate keyword is then created as a combination of those keywords and their interior stop words then the score is calculated as the sum of its members keyword scores. Finally, after the candidate words are scored, the top *T* scoring candidates (*T* is computed as one third of the number of words in the graph [17] [14]) are selected as representative keywords for the document or text:

- Number of words = 54

	tonight	brazilian	judge	blocked	WhatsApp	100	milion	people	rely	Facebook	Messenger	active	communicate	instead	sad	day	Brazil	ally	creating	open	internet	Brazilians	please	make	heard
tonight	1																								
brazilian		3	1	1	1																				
judge		1	2	1	1																				
blocked		1	1	1	1																				
WhatsApp		1	1	1	2																				
100						1	1																		
million						1	1																		
people						1	1	1																	
rely									1																
Facebook										1	1														
Messenger										1	1														
active												1													
communicate													1	1											
instead													1	1											
sad															1	1									
day															1	1									
Brazil																	3								
ally																		1							
creating																			1						
open																				1	1				
internet																				1	1				
brazilians																						1			
please																							1	1	
make																							1	1	
heard																									1

Table 2.1: Matrix of co-ocurrences for Mark Zuckenberg's post

- T = $\lceil 54 \div 3 \rceil$ = 18

- Document keywords = [*brazilian, brazil, judge, WhatsApp, voice, tonight, blocked, million, people, rely,working, hard, facebook, messenger, active, sad, day, internet*]

According to the study of Rose, Engel ,Cramer and Cowley [17], the creators of RAKE, the algorithm achieves higher precision and similar recall in comparison to existing techniques. In contrast to methods that depend on natural language processing techniques to achieve results, RAKE takes a simple set of inputs and automatically extracts keywords in a single pass, making it suitable for a wide range of documents. It's simplicity and efficiency relinquishes the need for high computing power and vast resources.

2.2.2 Cross Validation

Cross validation is a technique used for assessing the performance of machine learning models when they're still in the training stage. It helps in the understanding of how the model would generalize to and independent dataset.

When given a machine learning problem two types of datasets exist - known data or training dataset and unknown data or test dataset as in figure 2.1. By using cross validation the model would be tested in the training phase to check for *overfitting* or *underfitting* and to understand how the model would generalize to independent data, which is the reason for the existence of the test dataset given in the problem.

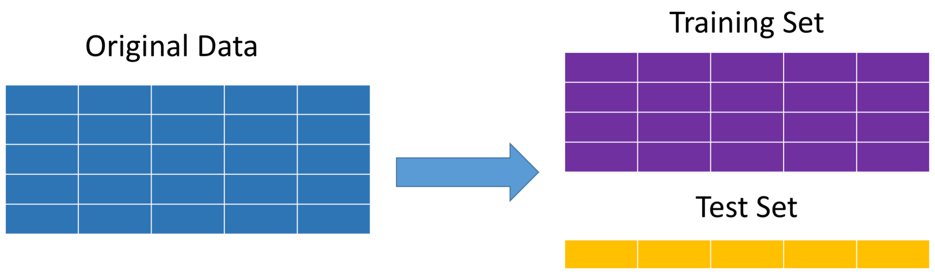


Figure 2.1: Train and test sets for machine learning models.

In a round of cross validation, the machine learning model is trained using the training set and the predictions would be verified against the test set. In the end, the results from all the rounds, in case of more than one, are averaged to estimate the accuracy of the model.

K-Fold Cross Validation

In K-Fold cross validation, as illustrated in figure 2.2, the data is divided into K equally balanced subsets, where each partition is called a fold. Then, for each iteration, a given fold is used as the validation set, whereas the remaining are joined to constitute the training dataset. For each iteration the model is trained using a slightly different dataset and validated using a completely different test fold in comparison to the previously used.

The usage of the K-Fold Cross Validation method enables the data scientist to understand the weak-

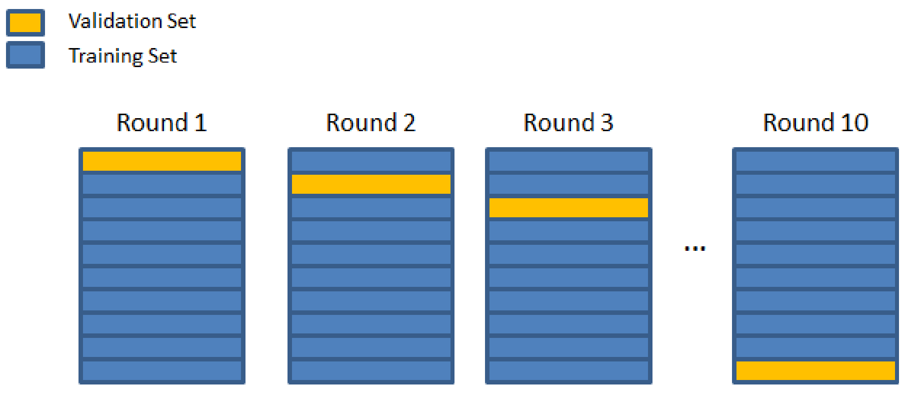


Figure 2.2: Train and test sets for each iteration considering $K = 10$.

nesses of the trained model be it in an *overfitting* situation, where the model grasps every detail of the training data being incapable of generalizing the learnings to new unknown information having low bias and high variance values, or in a *underfitting* situation, where model cannot capture the underlying trend of the data as it does not fit the data well enough having low variance and high bias values.

2.2.3 Support Vector Machine

The Support Vector Machine (SVM) is a state of the art supervised learning algorithm developed for pattern classification that has grown in popularity in recent times. The algorithm was proposed by Vladimir Vapnik in 1995 [19]. Although the SVM can be applied to various optimization problems such has regression, the classic problem is that of data classification. Figure 2.3 illustrates the result of applying the SVM algorithm to a group of data points. In the example, the data points are identified as being men or women, and the goal is to find a hyper-plane that separates each group of data points correctly by a maximal margin. Figure 2.3 only depicts the 2-dimensional case where the data points, by colors of blue and red, are linearly separable.

The optimal separating hyperplane is the one that is further away from the data points of each category/class, which has the highest margin towards each data cluster. Considering the case at hand, the hyperplane equation is the classical line equation: $y = ax + b$ or $y - ax - b = 0$.

Since more than 2 dimensions can be addressed by the SVM, the equation is normally $w^T x = 0$ which translates to equation 2.1 in vector notation.

$$w = \begin{bmatrix} -b \\ -a \\ 1 \end{bmatrix}, x = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{2.1}$$

Following this reasoning, the conclusion is that $w^T x = y - ax - b$. Figure 2.4 illustrates an hyperplane separating 2 groups of data.

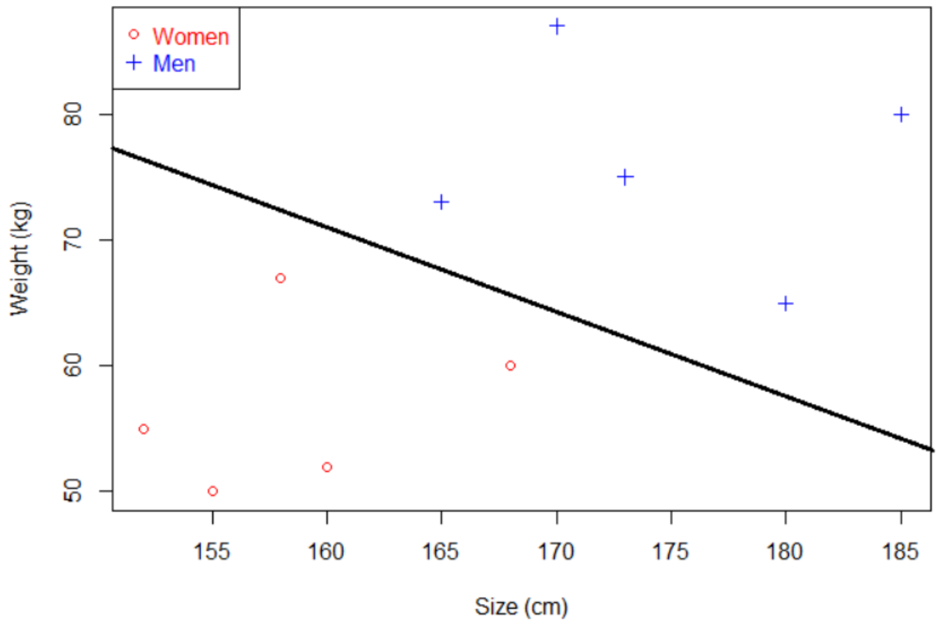


Figure 2.3: SVM hyperplane that classifies data as being man or women according to weight and height

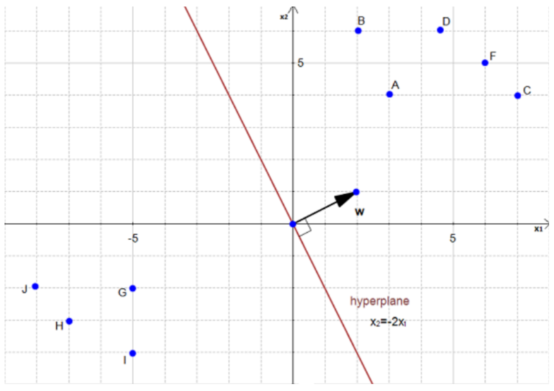


Figure 2.4: Hyperplane with b=0 with corresponding normal vector $\vec{w}$

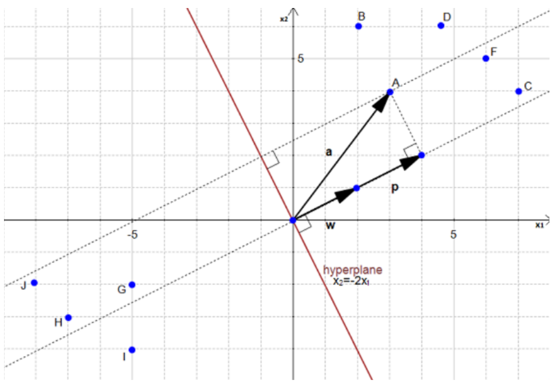


Figure 2.5: Projection of vector $\vec{a}$ onto the normal $\vec{w}$ resulting in $\vec{p}$

Considering $w_0 = 0$ the hyperplane $x_2 = -2x_1$ or $x_2 + 2x_1 = 0$ will cross the origin of the referential. In vector notation the previous equation would be translated to expression 2.2 in vector notation.

$$w = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.2}$$

The distance to the point $A(3,4)$ from the line would be the distance between A and its projection onto the plane, this would result in vector $\vec{p}$ which is the projection of A against the normal vector of the line, denoted as $\vec{w}$. The distance and result of this projection would be vector $\vec{p}$ in figure 2.5. Assuming the $\vec{u}$ as the direction of $\vec{w}$, the normal to the line would be as in equation 2.3.

$$\vec{u} = \begin{bmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix} \tag{2.3}$$

The magnitude or length of vector $\vec{p}$ is $\|\vec{p}\|$ and is given by the formula $(\vec{u} \cdot \vec{a})\vec{u}$ where $\vec{a}$ is a vector starting from the origin ending in point A.
Concluding this example, the value for $\|\vec{p}\|$ is $\sqrt{4^2 + 2^2}$ = $2\sqrt{5}$ and the margin to point A is computed as $2\,\|\vec{p}\|$ = $4\sqrt{5}$.

Recalling that the final objective of the SVM is to maximize the margin between the hyperplane and each data cluster, amongst all possible hyperplanes meeting the constraints, the chosen one will be the one with the smallest norm $\|\vec{w}\|$ ($\vec{w}$ refers to the properties of the hyperplane) equivalent to the biggest margin.
Therefore, the search for the optimal margin is an optimization problem focused on minimizing in (w, b) the $\|\vec{w}\|$, subject to $y_i(w \cdot x_i + b) \geqslant 0$ for any $i = 1, 2, ..., n$. Solving the equation will return the smallest value for $\|\vec{w}\|$.

For a more abstract example of the SVM consider plotting on a 2D space some data points where each one would represent a person with certain characteristics that would fit into a specific category known beforehand. After finding the desired line, the SVM would be able to receive an unknown data point representing a person and assign him or her to one of the two categories in either side of the hyperplane. The SVM considered in this thesis is not limited to a binary classification rather, it supports categorization in more than 2 classes. In such cases, a One-Versus-All (OVA) SVM approach is considered, where one SVM is fitted per class being that in the end the data point is assigned to the label corresponding to classifier with the highest score.

2.2.4 Naive-Bayes Classifier

The Naive-Bayes classifier is a technique based on the Bayes Theorem with an assumption of independence among predictors. The classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round and about two centimeters in radius. Even if these features depend on each other or upon the existence of other features, all of these properties independently contribute to the probability that the fruit in question is an apple and that is why it is known as *Naive*.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2.4}$$

Despite the naiveness, the algorithm tends to outperform more powerful alternatives [20] [21]. The applications of the Naive-Bayes classifier range from making decisions about treatment processes [22] to *spam* filtering of emails [23].

Email text	**Class**
people are buying the latest fashion items from home.	spam
yhe new fashion magazine for women is eligible for buying.	spam
are you coming tomorrow? bring beer.	non-spam
hi honey! tonight you better come in fashion.	non-spam

Table 2.2: Spam filtering examples.

Considering a spam-filtering scenario, the bayesian classifier would require a dataset for training as illustrated in table 2.2 where for each textual content is assigned a class of *spam* or *non-spam*. Before creating the Bayes model it is required to decide on a set of features. The most common method when dealing with textual data is to create features from words using word frequencies. This is similarly known as the *bag of words* method. For this example all stop words are removed from the emails text resulting in the contents of table 2.3

Email text	**Class**
people buying latest fashion items home.	spam
new fashion magazine women eligible buying.	spam
coming tomorrow bring beer.	non-spam
honey tonight better come fashion.	non-spam

Table 2.3: Spam filtering examples without stop-words with 18 distinct keywords.

In order to classify a new incoming email, the classifier would consider the word frequencies applying to them the equation of the Naive-Bayes theorem in 2.4. Consider the following email: *"Most people are buying this new dress. You better buy it too!"*. Following the removal of stop words the email would be reduced to: *people buying new dress better buy*.

In order to detect if the incoming email is most probable to be spam or not the next step is to find which of $P(spam \mid people\ buying\ new\ dress\ better\ buy)$ or $P(non{-}spam \mid people\ buying\ new\ dress\ better\ buy)$ has the higher score by applying the Bayes theorem considering the available information about already classified emails.

By comparing $P(people\ buying\ new\ dress\ better\ buy \mid spam) \times P(spam)$ and $P(people\ buying\ new\ dress\ better\ buy \mid non-spam) \times P(non-spam)$ together, after applying the Bayes theorem equation and removing the common denominator, all these probability values are achievable considering the data available in table 2.3. Assuming that every word in the incoming email is independent, the probability of the email being spam could be found by solving equation 2.5.

$$\begin{aligned}
&P(people\ buying\ new\ dress\ better\ buy \mid spam) = \\
&P(people \mid spam) \times P(buying \mid spam) \times P(new \mid spam) \\
&\times P(dress \mid spam) \times P(better \mid spam) \times P(buy \mid spam)
\end{aligned} \tag{2.5}$$

Considering $P(spam)$ and $P(non-spam)$ to be $2/4$, the remaining required values can be calculated accordingly as in table 2.4 being that $P(people \mid spam)$ means counting how many times the word *people* appears in *spam* samples divided by the total number of words in *spam*. In order to avoid a nullification scenario, in the multiplication of conditional probabilities, if the returned value is zero for example $P(dress|spam)$, then by applying the Laplace smoothing technique for balancing, the value is converted to 1 and the number of possible words are added to the divisor to prevent a resulting value greater than zero. For *spam* the number of possible words are 10 whereas for *non-spam* there are 9.

Word	$P(word \mid spam)$	$P(word \mid non-spam)$
people	$\frac{1+1}{12+18}$	$\frac{0+1}{9+18}$
buying	$\frac{2+1}{12+18}$	$\frac{0+1}{9+18}$
new	$\frac{1+1}{12+18}$	$\frac{0+1}{9+18}$
dress	$\frac{0+1}{12+18}$	$\frac{0+1}{9+18}$
better	$\frac{0+1}{12+18}$	$\frac{1+1}{9+18}$
buy	$\frac{0+1}{12+18}$	$\frac{0+1}{9+18}$

Table 2.4: Conditional probability matrix values

Solving both equations at 2.5 in regard to *spam* and *non-spam* situations returns the probabilistic values displayed in 2.6 indicating that the new incoming email is considered a spam email.

$$\begin{aligned}
P(people\ buying\ new\ dress\ better\ buy \mid spam) = 8,23045^{-9} \\
P(people\ buying\ new\ dress\ better\ buy \mid non-spam) = 2,58117^{-9}
\end{aligned} \tag{2.6}$$

2.2.5 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) is a non-parametric method used for classification and regression that has been used for estimation and pattern recognition since 1970 [24]. It can be considered as a supervised learning algorithm as it makes use of previous known data in order to classify new information. For this work, the KNN will be used as a mean for classification (not regression) on which the output of the classifier will be a class membership.

To understand the behavior of the K-Nearest-Neighbors classification algorithm a simple example illustrated in figure 2.6 is considered where the algorithm classifies an unknown data point using the *neighbors* perspective.

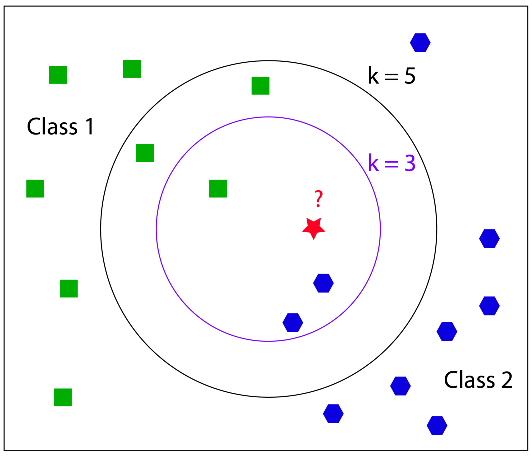


Figure 2.6: KNN algorithm for classification

The first step to build a KNN classifier is to train it with data for which the category is known beforehand. For the case of figure 2.6, the classifier was trained using the data represented by the green squares and blue hexagons being that the ultimate goal is to accurately classify unknown data, like the red star to be a green square or a blue hexagon.

As the algorithm name indicates, the classification is done through the nearest neighbors of the data point to be classified. In figure 2.6 the red star is plotted side by side with other data points for which the class is already known. To classify the new data point the KNN considers a group of points, more specifically *K* points which are the *K* points nearest to the one to be classified as a green square or blue hexagon. K is considered the hyper-parameter of the function justifying the non-parametric nuance of KNN.

In order to assess the selection of the nearest *K* points the algorithm considers one of the following similarity measures: Euclidean, Manhattan or Minkowski distances.

The Euclidean distance formula (2.7) states that the distance between 2 points is equal to the length of the line segment connecting them. The value of *n* is dependent of the number of space-dimensions. In this case $n = 2$, for the x and y dimensions.

$$d(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{n} (\vec{q_i} - \vec{p_i})^2} \tag{2.7}$$

The Manhattan function computes the distance that would be traveled to get from one data point to the other if a grid-like path was followed. The distance between two items is the sum of the differences of their corresponding components for x and y in case of a two-dimensional space (2.8). The value of *n* is variable according to the number of dimensions.

$$d = \sum_{i=1}^{n} |x_i - y_i| \tag{2.8}$$

Finally the Minkowski distance, a generalization of the Euclidean and Manhattan distance, defines the distance between two points in a normed vector space(2.9).

$$d(\vec{x}, \vec{y}) = (\sum_{i=0}^{n-1} |x_i - y_i|^p)^{1/p} \tag{2.9}$$

If p = 1 the distance given by the Minkowski function will be equivalent to the Manhattan distance and if p = 2 it would result in the Euclidean function. Considering the measures referenced before, if the variables are not in the continuous space, none of the three distance functions could be used. For such cases, the Hamming distance should be addressed.

Having a generic understanding of the distance measures its now possible to classify the red star data point, accounting the *K* neighbors nearest to the point. Regarding *K* with the values of 3 and 5, as displayed in table 2.5, for 3 neighbors the red star is classified as a blue hexagon and for *K*=5 is considered a green square.

K	**Class**
1	Blue
2	Blue
3	Blue
4	Blue or Green
5	Green

Table 2.5: *K-neighbors* and respective class assignment for the red star of figure 2.6.

The best choice of *K* depends upon the data. Generally, larger values of *K* reduce the effect of noise on the classification [25] therefore, the classification is always subjective to the number of neighbors that are considered when classifying.

2.2.6 Perceptron

The Perceptron algorithms are contained in the universe of neural networks. To understand the behavior of a Multi-Layer Perceptron which is considered in the final solution, its ancestor the Single-Layer Perceptron, will be explained enabling a better understanding of the multi-layered version.

Single-Layer Perceptron

The Single-Layer Perceptron, invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt [26], is an algorithm used for learning a linear binary classifier: a function that maps its input x to an output value $f(x)$ as represented in equation 2.10.

$$f(x) = \begin{cases} 1, & w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.10}$$

The main components and architecture of a Single-Layer Perceptron, inspired by the information processing of a single neural cell, are illustrated in figure 2.7, where the x value corresponding to each of the features x_1 and x_2 becomes input for the Perceptron. The input value 1 is the bias, a calibration parameter that allows the classifier to shift the decision boundary left or right with the hope of enhancing the performance for obtaining more accurate results.

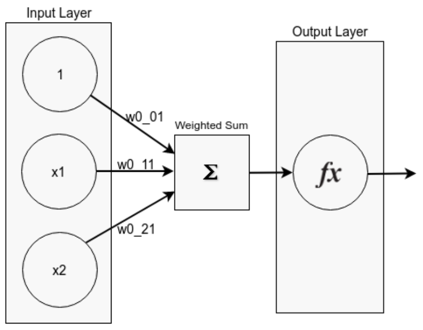


Figure 2.7: Architecture of a Single-Layer Perceptron.

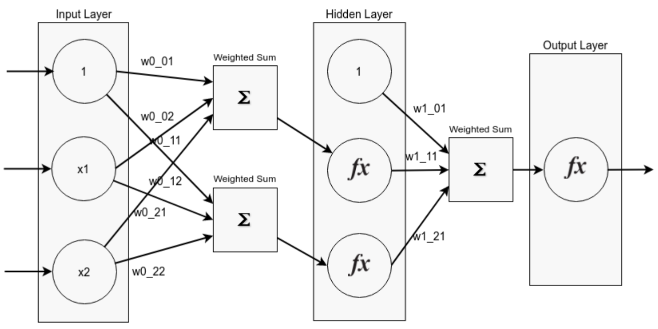


Figure 2.8: Architecture of a Multi-Layer Perceptron.

The weights, $w0_{01}, w0_{11}$ and $w0_{21}$ are values computed over time while training the model. Each weight starts with a random value which is adjusted over each training iteration according to the obtained training error. The weights and corresponding features are multiplied and summed accordingly as specified by the *weighted sum* function that returns a real number input to the last neuron.

Finally the output layer neuron computes the result by converting the input originated from the weighted summation into the final output value according to the activation function. This process is named forward pass. The activation function for the Perceptron is described in equation 2.10 with the addition of the remaining weight-input pairs corresponding to the input features.

The learning phase of the Perceptron occurs after each forward pass where each weight is adjusted according to equation 2.11.

$$w_{new} = w_{old} + \alpha \times (output_{expected} - output_{predicted}) \times x \tag{2.11}$$

.

If the output is predicted correctly, the weight value will be kept for the iteration as the result of the procedure $\alpha \times (output_{expected} - output_{predicted}) \times x$ is equal to zero forcing $w_{new} = w_{old}$. The parameter α is an hyper-parameter called learning rate which controls the weight adjustment quota.

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.12) \qquad\qquad f(x) = max(0, x) \qquad (2.13)$$

Multi-Layer Perceptron

In contrast to the Single-Layer Perceptron, the Multi-Layer Perceptron in figure 2.8 distinguishes itself by allowing the usage of multiple inner layers, known as hidden layers, and non-linear activation functions.

The hidden layer illustrated in figure 2.8 for the Multi-Layer Perceptron is composed of three nodes, one of which is the bias neuron. The output of the remaining neurons of the hidden layer depend on the output from the neurons in the input layer $_1, x_1, x_2$ as well as the weights associated with the connections $w0_{01}, w0_{02}, w0_{11}, w0_{12}, w0_{21}$ and $w0_{22}$. In resemblance to the output neuron of the Single-Layer Perceptron in 2.7, the neurons in the hidden layer, apart from the bias node, own an activation function which typically corresponds to the sigmoid function (2.12) or the rectified linear unit or ReLU (2.13).

The value returned to the output neuron is obtained through the weighted summation of the values returned by each activation function in each neuron plus the bias value in the hidden layer which is then transformed by the activation function of the output neuron, resolving the final prediction value for the given input x_1, x_2.

The learning process of the Multi-Layer Perceptron is done by back propagating the errors usually through the Backpropagation algorithm [27] by means of the gradient descent optimization used for minimizing the network cost function typically the Mean-Squared-Error equation [28].

Initially all the network weights are randomly assigned. For every input in the training dataset, the neural network is activated and its output is observed. This output is compared with the desired output which is already known, and the error is *propagated* back to the previous layer. This error is noted and the weights are adjusted accordingly. This process is repeated until the output error is below a predetermined threshold or the dataset is completely processed.

2.3 State of the Art

This section considers the most relevant and most recent works developed in the field of social data mining mostly oriented towards political engagement, which provided essential input in regard to the development of this work. As the initial objective of this thesis was to tackle the Facebook network, the first task performed was the studying of Facebook social mining papers which served as the foundation for this project, therefore a few works relative to this platform and to this context are primarily analyzed followed by a set of studies and applications realized on top of the Twitter social network which dictated the final course of action.

2.3.1 Sentiment analysis

Sentiment analysis, also known as opinion mining, is the computational process of categorizing textual or verbal input, through the usage of NLP techniques, in order to assess the sentiment underlying such content.

Nowadays people are increasingly progressing towards online communities such as Facebook or Twitter where they can share their ideas and discuss opinions. As such, social networks have become a gold mine for data mining applications particularly for those employing sentiment analysis techniques.

Facebook

Due to current events, in particular in the case of privacy violations carried out by the Facebook company together with Harvard Analytica, access to the Facebook API was restricted where it was once possible to obtain critical information about users of the platform. Previously to these events, a set of applications were developed, with the purpose of understanding the personality and emotion of users, which gained some success taking into account the results obtained based on the fact that social networks have become a fundamental pillar of human interaction.

A group of scholars found that a machine can be more accurate to predict a person's personality than a fellow human [29]. Motivated to understand the depths of personality judgment, an essential social-cognitive activity [30] [31], the group developed a linear regression model capable of understanding humans more than humans understand themselves, by comparing the results of the model to the output provided by a personality questionnaire covering around 86.000 volunteers. Each volunteer answered a 100-item International Personality Item Pool (IPIP) [32] Five-Factor Model of personality questionnaire measuring the traits of *openness*, *consciousness*, *extroversion*, *agreeableness* and *neuroticism*, which covered self and other judgments criteria in the process. The model was trained on a set of Facebook *likes* [33] obtained from 70.520 participants who answered the questionnaire. To obtain the model judgments, the algorithm created a user-likes matrix, where each entry is set to 1 if there exists an association between a participant and the given Like and 0 otherwise. This matrix together with the coefficient values corresponding to each trait score from the IPIP questionnaire of each participant, are used to fit five Lasso Linear regression models [34]. A 10-fold cross validation is then applied to prevent *overfitting*.

The model training was based on the IPIP questionnaire results for each user involved and its accuracy was obtained through the average testing results obtained during the cross validation process. Each user was also judged by his friends in order to provide the human viewpoint for comparison. In the end, by comparing the IPIP questionnaire results with those obtained by the model and by the user's friends it was found that the model surpassed the score of the friend's judgments having an accuracy of 56% in comparison to that of friends with 49%.

Another interesting work carried out by fellow scholars in the same context from the one before, was grounded on a model capable of predicting the user personality using the information publicly available in his/her Facebook profile [35].

In resemblance to the previous work, a Five-Factor personality questionnaire was assembled and made public throughout a Facebook application gathering results from around 280 distinct users building a trustworthy knowledge base for training. At the same time, the profile of each of the 280 users was scanned and all the relevant data was assembled and quantified using text analysis tools to obtain a set of features.

The information considered from the user profile was the depth of his network in regard to friends, his direct profile information such as birthday, religion, education history and his hometown, his activities and preferences, his language features and more.

Only 170 users, from the initial set of 280, were considered by the model due to the limitation of information in regard to each one. Of all aggregated information were extracted 74 features per user. To predict the score of a given personality trait, from the total of 5, a regression analysis was performed using Waikato Environment for Knowledge Analysis (WEKA) software [36] with a 10-fold cross validation with 10 iterations. The final algorithm suffered from an average error of only 11% in comparison to the actual personality trait value.

Twitter

An interest paper developed in the Twitter realm covered the possibility of classifying latent user attributes including gender, age, regional origin, and political orientation based on the user language of informal communication such his/her *tweets* [37].

The first step to accomplish the established goal was to build the algorithm knowledge base, namely the training dataset as there was none available at the time. For a given set of users, the authors manually acquired a given set of Twitter personalities with such attributes available in order to enable a supervised approach to be followed.

From the users aggregated for training, the algorithm extracted a set of relevant socio-linguistic features from their *tweets* where for each feature was assigned a set of textual expressions. For example, the feature *laugh* was activated by the usage of expressions such as *LOL*, *LMAO* and *haha*.

Another approach taken was the consideration for unigrams and bigrams in the feature set for comparison, normalized by their term frequency, assuring the preservation of emoticon expressions.

The final approach was to conglomerate a stack model which considered as features the ones found by the *n-grams* feature model with $1 \leqslant n \leqslant 2$ and the socio-linguistic model along with their prediction weights. A set of linear binary SVM classifiers was trained, one for each latent attribute [38], in each approach considered.

For the case of the political orientation latent attribute, the performance of the *n-gram* considerate algorithm was the best averaging around 83% in accuracy contrasting with the social linguistic unit with 63% and the stacked alternate model with an 80% score.

2.3.2 Political engagement

This section refers to a few works or papers which directly tackled political engagement scenarios on social networks. A few cases are referent to the Facebook social network for reasons previously explained while the remaining are subjective to Twitter.

Facebook

A recent study conducted in the Facebook social network used a set of Facebook pages linked to political parties as a training standpoint for a text categorization algorithm which in turn would predict their political orientation based on the textual content of their pages [39].

Text categorization as previously disclosed, includes a set of techniques which attempt to extract various kinds of informations from textual content. Up until today, these techniques have been used to discover user details such as gender [40], age [41], language [42], and personality [43]. This background of information motivated the authors to develop a machine learning classifier capable of deriving, with a considerable degree of certainty, the political orientation of a group of Facebook users.

This experiment was realized over a set of 450 Israeli users which provided essential access to their personal Facebook pages, containing texts written in Hebrew. Each user from the set of 450, was asked to fill an online questionnaire on which they anonymously disclosed their political association. In order to train the classifier, a group of texts corresponding to nine major political parties of Israel were obtained and labeled as left, center or right depending on the side of the political party. In total, 646 posts containing 550.000 words were considered for training.

The classifier considered a set of features based on word frequencies in regard to the content of the political pages. From then on, the classifier was trained and tested using the K-means cross validation technique with K=10. The algorithm training was diverse in a way that it was tested in different scenarios for classifying users as left or right wingers and left, center or right wing supporters, based not only on content from political pages but also on content by the same corpora. The classifier was also capable of detecting swing voters, which correspond to voters who intend to switch allegiances.

In the end it was proven that using political texts as a training foundation for the algorithm returns considerable good results being that when the algorithm is trained on content from the same corpora its accuracy is around 90%. From the results obtained by the classifier it was also possible to identify which textual terms were most used by each political side.

Twitter

Twitter has been a critical platform when it comes to social mining. With the intent of predicting the political alignment of Twitter users [44], a group of scholars motivated by diverse political factors such has the 2010 US election campaign expenses, developed a machine learning classifier, namely a Support Vector Machine, capable of predicting the political orientation of Twitter users with 90% accuracy.

To do so it was required first hand to manually annotate a group of 1000 users as left or right supporters in order to provide training data to the algorithm. One of the most interesting details in regard to this work was that the algorithm itself was trained on two different scenarios: Firstly one SVM training was fulfilled over the textual content of each user while the other, which outperformed the first, was simply trained on hashtag metadata.

The first Support Vector Machine was trained on a feature-user matrix corresponding to the TF-IDF [45] weighted terms (each a single word known as an unigram) contained in each user's *tweets*. For this case, information such as hastags, stop words, mentions and URLs were removed in order to ease comparisons. The algorithm then passed through the procedure of cross-validation with 10 iterations which provided the final scores for this case. The accuracy obtained for this method was around 80%, which means from the set of 1000 manually annotated users 20% were misclassified.

The second training scenario considered only the presence of hashtags on the *tweets* of each user which were afterwards converted to features based on their relative frequency. For this approach the results reached a maximum of 90% accuracy when considering a set of around 5000 features.

2.3.3 Conclusions

In this chapter were explained the main topics that motivated the development of this thesis as well as the functionalities and algorithms which were considered in the development of the final solution. Near the end of this chapter a group of works previously realized by fellow scholars were referenced directing the spotlight to the major achievements of each as well as the methodologies implemented by each group. Table 2.6 displays a summarized view of relevant studies studied while developing this thesis.

The next chapter explains the methodologies and implementation approach considered for this development which highly correlates to some of the previously referenced.

Ref.	Year	Objective	Platform	Training base	Methodology	Performance
[29]	2014	Compare computer and human personality judgments	Facebook	Facebook likes	Lasso linear regression comparing with IPIP questionnaire results	56% vs 49% accuracy for computer and human respectively
[35]	2011	Predict user personality	Facebook	Public profile attributes	Gaussian and Decision Trees regression comparing with IPIP questionnaire results	Mean absolute error of 11% for each personality factor in relation to survey results
[37]	2010	Classify latent user attributes	Twitter	user *tweets*	socio-linguistic and n-gram features SVM classifier	83% accuracy using n-gram model for political orientation
[39]	2015	Predict political orientation of users	Facebook	Political pages	Text categorization classifier (type undisclosed)	90% when classifying political users; 82% for regular users classification
[44]	2011	Predict political orientation of users	Twitter	Political user *tweets*	SVM classifiers based on *tweet* textual content and *hashtags* only	80% score using textual content; 90% accuracy based on *hashtag* metadata

Table 2.6: Important studies assessed while developing this thesis.

3 Implementation

Contents

In this chapter the implemented algorithm is described in detail. Firstly, the architecture of the system is described in a summarized manner followed by an explanation of the possible threads of execution the algorithm might follow according to the user instructions.

The main functionalities that distinguish this algorithm are portrayed using illustrations in attempt to clarify the purpose of the components behind them and to describe the expected results at each step of execution.

3.1 System Architecture

This thesis presents a system that uses a collection of machine learning algorithms combined with text analysis techniques that altogether are capable of predicting the political orientation of an individual from his texts. The knowledge base used for this solution was Twitter, as the content used by the algorithm were *tweets*.

As illustrated in figure 3.1, the system's architecture is structured like a traditional layered architecture composed by three distinct layers: presentation layer, business logic layer and data layer. The layered architecture provides flexibility to exchange code, as the changing or adding of new modules and components to the system can be done with minimal impact to the overall system behavior due to the clear interface between layers. This means that the system was designed with the objective to be extensible which is considered a best practice in software engineering.

Following the illustration of figure 3.1 the execution of the algorithm can be summarized in the following steps:

1. The user inputs a command in the Graphical User Interface (GUI) to start the algorithm, accompanied by a properly structured configuration file where it can be defined: the arguments for the Master algorithm and his subjects, the subject algorithms to run, the number of *tweets* to classify as well as to train upon, the classification parameters to consider among other definitions.

2. After the execution parameters are defined, the algorithm starts by applying the specified configurations to the different modules of the system such as the Master Algorithm, the Data Manager and the Pre-processing components subsequently triggering the main execution flow.

3. After properly starting the execution, the algorithm either fetches the *tweets* from the Tweets Storage module, from the Twitter API directly through the Twitter API Clients component or from both, and inputs the result *tweets* to the Pre-processing module.

4. The Pre-processing component then realizes the filtering of the *tweets* and inputs the properly parsed result *tweets* to the Master Algorithm and his subjects for training and possible prediction.

5. Finally the Master Algorithm outputs the training and prediction results to the GUI and to log files for later analysis.

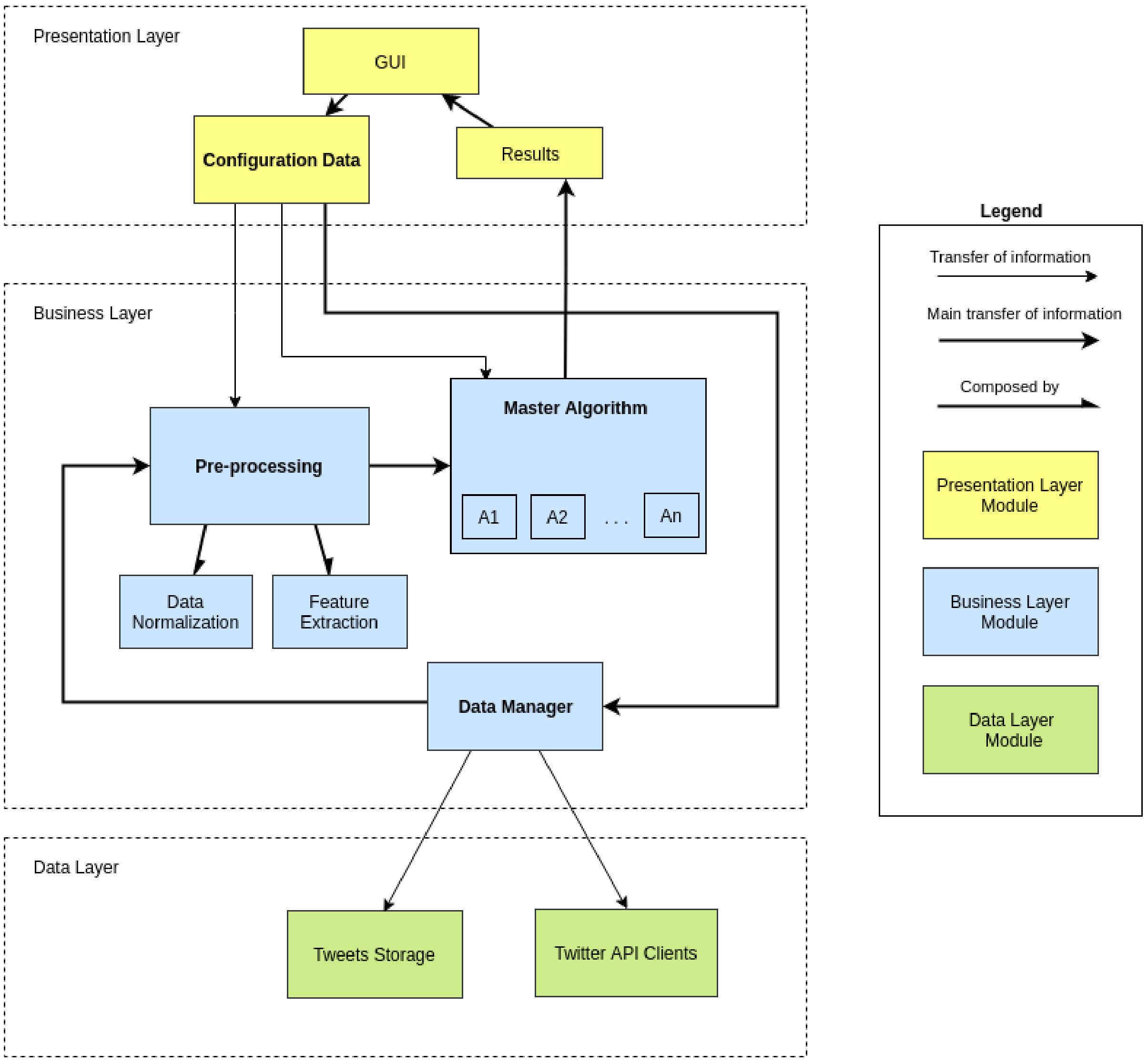


Figure 3.1: Algorithm architecture and general execution flow.

The main execution flow of the algorithm, that follows the initial configuration or setup, gathers 4 distinct phases which in order produce the final output: Data Collection, Data Pre-processing, Data Classification and Result Presentation.

As the algorithm is running through each phase, different modules, depicted in figure 3.1, interact providing each other with input and output data enabling the execution to proceed to the next stage until it reaches completion.

Each step referenced in figure 3.2 is colored according to the color scheme of the architecture diagram in figure 3.1 whereas the color of each phase is given in relation to the main architecture modules that realize it. For example, the first step Data Collection is comprised of three modules: Data Manager, Tweets Storage and Twitter API Clients therefore it's colored green in order to match the majority of the modules which belong to the Data Layer of the architecture.

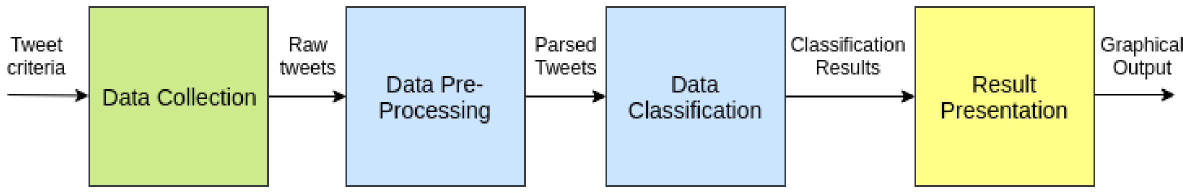


Figure 3.2: Algorithm execution flow.

In the next sections of this chapter the algorithm flow steps alongside its peer components will be explained in order to provide a clear understanding of the solution, starting by understanding the first input configurations passed through the GUI and the configuration file.

3.1.1 User Input

The first stage necessary to provide logic to the algorithm execution is the initial configuration data. The configuration data is present in a single file of the program which is requested from every entity belonging to the system. This file contains, as an example, the following data:

- Tuning parameters for each algorithm belonging to the system.
- Database selection criteria like number of *tweets* to retrieve from the database and time interval to consider when fetching them.
- Location of important files and directories relevant to the system such as the training files of each algorithm and class definitions.
- Execution path to be considered by the system.

The items referenced above are regarded as the most important but still don't cover the entire list of possible parameters and definitions which may be passed to the components of the system. Throughout the next sections, the specifications present in this file will become clearer as each component of the system illustrated in figure 3.1 uses one or more parameters specified in this configuration.

3.1.2 Data Collection

In the previous step the user provided a set of configurations to be applied to different components of the system before the execution is properly initiated. Among those configuration details the user determined a set of inputs which are applied in this step.

Introduction

In the Data Collection step there are three main stakeholder modules which are present in the architecture diagram of figure 3.1:

- Data Manager
- Tweets Storage
- Twitter API Clients

At this stage each module provides useful input that in the end enables the completion of a successful workflow as exemplified in pseudo-code excerpt C.1:

1. The Data Manager module receives a set of criteria denominated as *tweet criteria* where it's specified the parameters to consider when fetching for *tweets* such as: the number of *tweets* to retrieve, the criteria each *tweet* must obey such as the *username* or *timestamp*, and the origin of the *tweets* which may already be in the database.
2. The Data Manager then proceeds and attempts to fetch the *tweets*from the specified location. In case the user intends to use *tweets* already present in the database, the Data Manager redirects its request to the Tweets Storage module returning the *tweets* from the database accordingly. If on the other hand, the user seeks new *tweets*, the Data Manager redirects the request to the Twitter API Clients module which sends an HTTP request to the Twitter API for new *tweets* according to the arguments passed. The *tweets* resulting from the request are stored in the database and returned.
3. Finally, after the *tweets* that meet the criteria are successfully retrieved from the database, the Data Manager provides the resulting set of *tweets* as input to the next stage.

The modules highlighted in the summarized workflow play an important role in the system as they are responsible for providing the data for the algorithms to train and to execute upon. Next the Data Manager, Tweets Storage and Twitter API Clients modules will be described altogether.

Components

To better understand the modules that compose this step and how they interact with each other the illustration provided by figure 3.3 will serve serve as foundation for the explanation.

Data Manager

The main component that orchestrates this step is the Data Manager. This module is responsible for redirecting the user input to the different modules of this stage. As depicted in figure 3.3, the Data

Manager receives as input a group of parameters denominated as *tweets criteria* which consist mainly of the following data:

- Twitter *usernames* linked to political parties
- Twitter *usernames* of common users
- Time interval for *tweet* selection
- Database query parameters
- Desired execution flow

The collection of parameters specified above are encoded in a configuration file which adds some logic to the selection of *tweets* and grouping of users into a political party if desired. For training purposes in the Data Classification stage, the Twitter *usernames* are grouped into political parties which represent the classes used for the training of the algorithms. The remaining users, denoted as *common users* will be the users considered for prediction.

As depicted in figure 3.3 there are two possible execution flows for this stage which can occur exclusively or in sequence:

1. Download new *tweets* from the Twitter API and store the resulting set in the database
2. Fetch *tweets* from the database and return the selected results

The most common case of execution is the simple selection of *tweets* from the database according to a time interval and group of users which may or not be linked to a political party. This case typically means the user wishes to realize the training of the algorithms. If the algorithms are already trained, and the user intends to realize a prediction, then the set of *tweets* returned contains only *tweets* belonging to common users which will be classified later. In both cases, the Data Manager sends a request to the Data Storage component which replies with a set of *tweets* according to the query parameters passed as input. The Data Manager then returns the selected set as input to the next stage. To this scenario corresponds the arrows highlighted in red in figure 3.3. The resulting set might be a list where each entry corresponds to a single text per user (meaning that all the retrieved *tweets* of a user will be concatenated into a single text and then added to the set) in case of testing or a list of *tweets* directly fetched from the database in case of training.

The other scenario occurs when the user wishes to download new *tweets* and update the database with newer ones from the Twitter timeline within a time interval. These *tweets* may belong to a specified set of users or any if none are defined in the input. In this case the execution flow starts with the Data Manager which dispatches the request to the Twitter API Clients subsequently sending the response to the Data Storage module which will save the new *tweets* in the database. This scenario corresponds the arrows highlighted in green in figure 3.3.

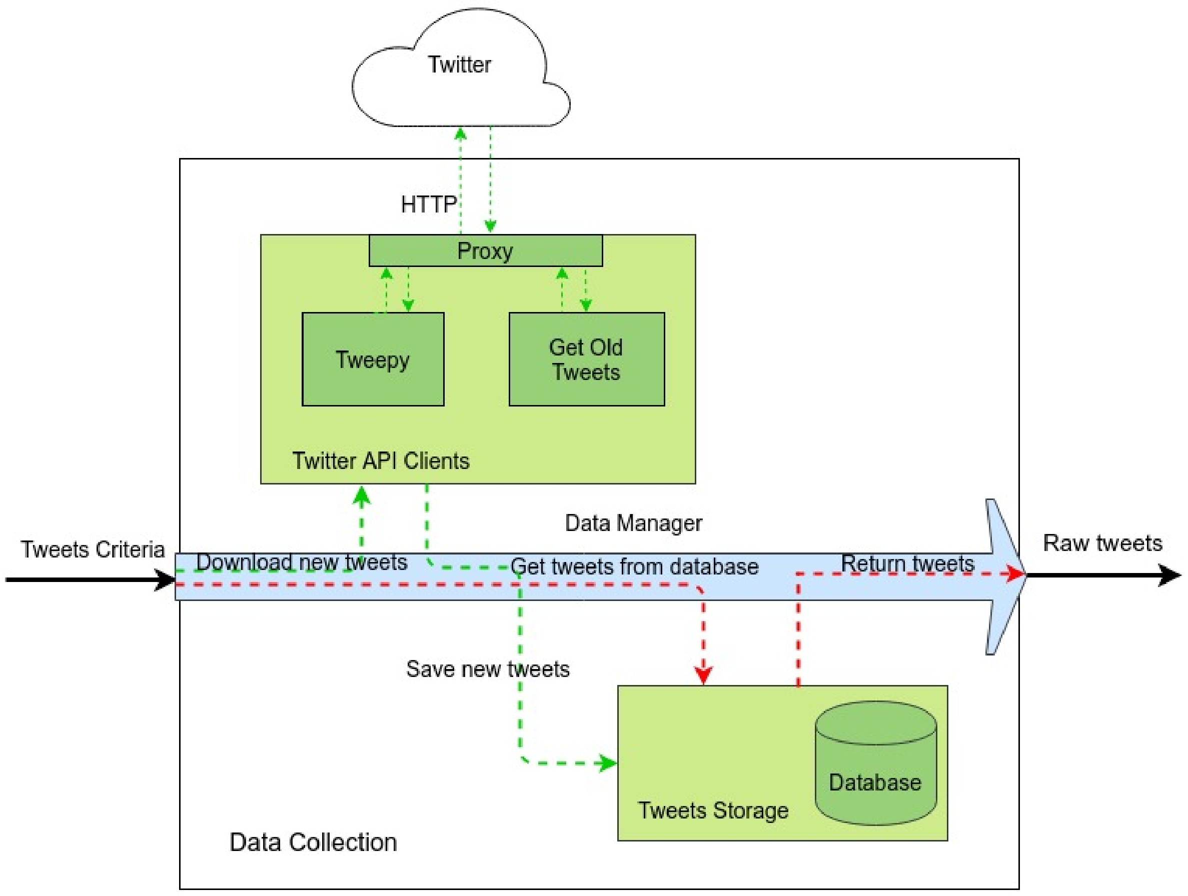


Figure 3.3: Execution flow in the Data Collection step.

Twitter API Clients

The Twitter API Clients module is responsible for the retrieval of *tweets* from Twitter using the Twitter API and other methods mentioned in this chapter. As described in the previous section, the input to the Twitter API Clients is received from the Data Manager which triggers the execution of each client. As represented in figure 3.3 the Twitter API Clients module is composed of two parts:

- Tweepy
- Get-Old-Tweets

Both elements are Twitter clients which have distinct approaches regarding the consumption of information from Twitter.

The first client, Tweepy, is a Python-based library which allows the consumption of the Twitter API remote endpoint using a client key as means of authentication. When using Tweepy it's possible to specify a set of parameters for a finer selection of *tweets* from the API. For example, it's possible to specify, considering the scope of this thesis, parameters such as:

- The username from whom the *tweets* should belong to.
- Information regarding the origin of the *tweets* using the location parameter.
- The time interval considered when retrieving the *tweets*.

Besides the parameters specified in the list above, others exist, which allow a finer tuning of the selection. One limitation that exists in Tweepy due to Twitter API permissions is that a direct client, such as this one, can only access *tweets* from the Twitter timeline up to two weeks prior to the request, and if the client wished only to retrieve a list of ordered *tweets* from the user profile, only a limited set of *tweets* would be available (in this case without time restrictions). Given that this work aimed at predicting the user political orientation given the textual content of his *tweets*, having a limited window of two weeks of *tweets* and a limited count of *tweets* per user would narrow the amount of data available, which in the end would compromise the quality of the results.

The Get-Old-Tweets module solves both problems pointed above having himself a trade-off factor to compensate. This module is also a Python-based client which behaves as a crawler when it comes to the consumption of information from Twitter. The user can provide the client with the following information:

- A time span with no limitations.
- The username or usernames from whom the *tweets* should belong to.

As the Get-Old-Tweets module behaves like a web crawler, this means that it parses web pages, in this case web pages from Twitter. It simulates a endless scroll in a Twitter page and retrieves specific information from *html* tags which conform to the parameters passed to the module.

The first trade-off this client has, is that it's only designed to support one endpoint corresponding to the Twitter timeline page, meaning no other Twitter page out of this format is supported taunting the inability to retrieve more detailed information if necessary.

The other limitation relates to the fact of Get-Old-Tweets being a web crawler. Crawling a page, using an endless scroll method, is very time consuming. For example, when retrieving for *tweets* while developing this algorithm, for a time interval of 30 days the client took approximately 1 week to retrieve all *tweets* from the entire country of Portugal considering a single thread of execution.

The final implementation encoded in the Twitter API Clients module was a Web Proxy. The necessity for the implementation of a proxy was induced by a blockage scenario where Twitter forbade the access to its endpoints by adding the Internet Protocol (IP) address of the machine making the requests to its blocked IP list. Given the intensity of requests for *tweets* made during the initial stages of this project, for testing purposes, this reaction is justified.

In order to work around this issue a *Tor* web proxy was implemented. Using the *Tor* onion router capabilities together with *PhantomJS* ensured a constant change of IP addresses when requesting *tweets* anonymously, making the root machine untraceable.

Data Storage

The final component of this stage is the Data Storage module which is responsible for managing all the interactions made with the system's database. This *sqlite* database contains all the *tweets* that were retrieved through the Twitter API Clients module and consists mainly of two main tables critical to all the execution workflow:

- *Tweets*
- *TweetsParty*

The *tweets* used for algorithm training will be stored in the *TweetsParty* table as they correspond to users who have a known connection to a political party and therefore it's likely that their *tweets* content relates to other users from the same party and to the party itself. The users belonging to a party known beforehand are grouped together in a configuration file which is passed as input in the beginning of the execution.

The *Tweets* table sole purpose is to store *tweets* from users with unknown political affiliation. The *tweets* stored in this table will be used for prediction.

In case of a user being linked to both tables the Data Storage module is programmed to prioritize the selection of users from the *TweetsParty* table and to ignore the entry in the *Tweets* table. This scenario might occur in case the user was found in the Twitter timeline and was priorly defined as an associate of a political party in the input configuration file.

3.1.3 Data Pre-Processing

The next stage after Data Collection is the Data Pre-Processing which receives as input a set of raw *tweets* or texts resulting from the execution of the previous stage and outputs a matrix significant of the *tweets* to be consumed by the classification algorithms. The execution flow and components for this step are illustrated in figure 3.4.

Introduction

In order to step through this course of execution the *tweets* or texts returned from the Data Collection stage need to be properly treated in order to be understood by the classification algorithms. As such, they need go through two distinct phases:

- Data Normalization
- Feature Extraction

The summarized execution workflow, simplified in pseudo-code excerpt C.2, is as follows:

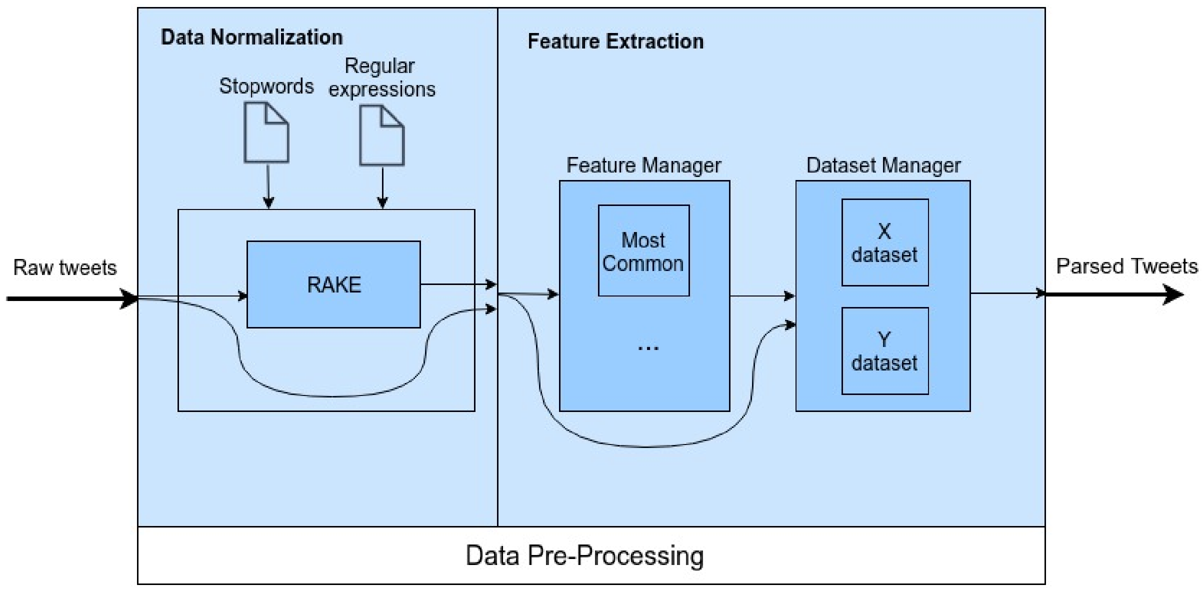


Figure 3.4: Execution flow in the Data Pre-Processing step.

1. The raw *tweets* returned by the Data Manager in the previous step are inputed directly into the Data Normalization module in order to be treated according to the configuration file definitions.

2. If configured, the *tweets* or texts will pass through the RAKE module which will first remove unwanted stop words and terms that match the syntax of the regular expressions present in configuration files. After the removal of such terms is complete the RAKE algorithm will proceed to extract all the relevant keywords and expressions found on each entry of the list. The RAKE module will return an array to the Feature Extraction phase where each entry (corresponding to a *tweet* or text) is a list of keywords.

3. If the RAKE module pass is not active, the *tweets* or texts will be sent directly to the next module without keyword extraction assuring the removal of stop words and regular expressions matches, being applied a classic split by space character to each entry in the list.

4. The resulting array of lists (an array containing a list per entry) will then be read by the Feature Manager in order to extract the most relevant features according to the *most common words* method in case the features were not provided from the configuration file.

5. Finally each list in the array will pass through the Dataset Manager module which in turn will transform each entry into a machine readable list of numerical values that mirrors the features found by the Feature Manager for each text or *tweet*, and return the corresponding transformed array to the classification algorithms in the next stage.

Components

The modules that compose this execution step are split in two execution scenarios. Each component present in one of the scenarios is explained below starting from the components present in the Data Normalization step.

RAKE

The first module present in the Data Normalization iteration depicted in figure 3.4 is RAKE. This component is responsible for realizing the tuning and removal of unnecessary words or terms that take part in the *tweets* text and return the fine grained *tweets*.

As illustrated, this module receives as input two files with distinct purposes:

- *Stopwords* file
- Regular expressions file

The first file, *Stopwords* contains a well formated list of stop words. A stop word, as explained in previous chapters, is a word that conveys no meaning. Typically these words belong in the universe of prepositions, adjectives or pronouns which are commonly used for referencing something and don't bring context or meaning to the text. As an example, using the following sentence:

- "*Real Madrid is a good club. They don't beat around the bush when it comes to scoring goals. Watch them live at http://www.somesite.com*".

The typical stop words which may be referred in the Stopwords file for this kind of text would be: *is, a, good, they, don't, around, the, when, it, comes, to, them, at*.
With the removal of this words the remainder of this text would be: *Real Madrid good club beat bush comes scoring goals Watch live http://www.somesite.com*.

The next file, *Regular expressions*, contains similarly to the first file, a list of regular expressions to remove if present in a sentence or in the list. This file contains mostly regular expressions to catch *URLs* or *hyperlinks*. Maintaining the original sentence from the latter example, the resulting text after removing the regular expressions matches would be: *Real Madrid good club. beat around bush comes scoring goals. Watch live*. The URL present in the original sentence is no longer present.

The RAKE algorithm encompasses both steps referred above. The order of consideration of each file can be specified in the initial configuration of the algorithm. When integrating both steps in the execution flow instead of completely removing stop words with no exception, RAKE attempts to find expressions that could contain stop words but that in a sense convey some meaning. For example, in the original sentence, the algorithm could keep the expression *beat around the bush* that in a way expresses meaning.

After a complete RAKE execution the algorithm would output to the next stage a list of words and expressions such as: *Real, Madrid, good, club, beat around the bush, comes, scoring, goals, Watch, live*.

The RAKE step in the Data Normalization stage is not obligatory. Should the execution of RAKE be skipped, this stage would output a list of keywords which would not contain any stop words, and words or sentences caught by regular expressions returning the resulting set using a split by space approach.

Feature Manager

This component exists in the scope of the execution of Feature Extraction. Its purpose is to, given resulting list of keywords and expressions corresponding to the Data Pre-processing input, return using specific search criteria, a list of words which would be representative of the *tweets* or texts. As mentioned in the summarized steps for this stage, the Feature Extraction stage is only necessary to occur in case a set of features is not provided initially.

In the scope of this project the only search criteria implemented to provide the representative list of words was the Most Common method as depicted in figure 3.4 which is a simple bag-of-words methodology considering word frequencies.

This method is based on the frequency of words which are contained in the input vector. The implementation of this approach receives as input a parameter n representing the number of features to consider and an array of *tweets* returning a ordered list of size n containing the most common n words which are present in the original text or group of *tweets*.

In short the Feature Manager component returns a smaller vector than the original containing words or expressions which obey to a given criteria and that represent the entire group of *tweets* or texts given as input.

Dataset Manager

The final module of the Feature Extraction stage is the Dataset Manager. This component returns different information considering the type of input it receives, as it can receive input with two distinct purposes.

The first possible input is comprised of an array of texts with *tweet* information, the list of features found by the Feature Manager and a group of labels in a list the same size of the array of texts which has the purpose of classifying each text entry in the array. In this case the Dataset Manager is expected to return a training dataset with X and Y values where, for each group of values corresponding to a text in X exists a corresponding label in Y.

The other possible input the Dataset Manager can receive is only the array of texts, resulting from the concatenation of *tweets* for each user, and the list of features. In this case, the Dataset Manager only returns the X array meaning that the algorithm is realizing a prediction as no labels are given to classify such texts.

As depicted in figure 3.5, the Dataset Manager contains two modules: dataset X and dataset Y. The reason for the existence of both modules was explained before.

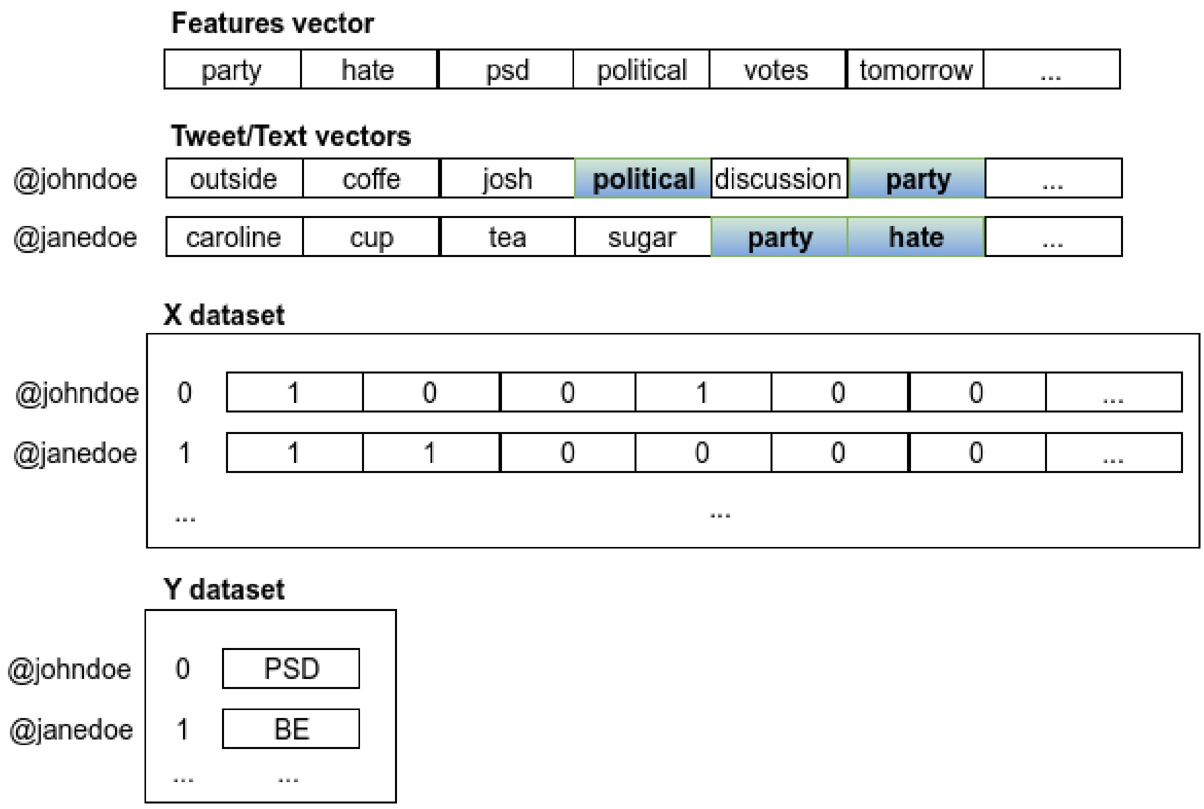


Figure 3.5: Dataset Manager input and output results in training.

Given the features found by the Feature Manager to be representative of the group of *tweets*, the X dataset will contain for each text or *tweet* present in the input group, a vector the size of the number of features with integer values representing a count for the number of times the word in the Feature vector at index *i* is present in the text or *tweet* vector. The value is zero in case the word or expression is never present.

Following the example, the text vector corresponding to *@johndoe*'s *tweets* contains two words, *party* and *political* which are present in the Features vector at indexes 0 and 3. Therefore, an entry in the X dataset corresponding to that *tweet* will exist, which will contain the value of 1 at indexes 0 and 3 and the value of 0 in all other entries of the vector. The value is not restricted to 1 as it must represent the term frequency meaning if the word *party* occurred a total of 4 times in the *tweet* vector, then the value in the X dataset vector at index 0 should be 4. Then, for each vector present in the X dataset array there will exist an entry in the Y array which classifies such vector (in case of training). The corresponding entry in the Y array is present in the initial configuration file and points to a specific class which, in the scope of this thesis, will be a political party linked to that particular *tweet* or group of *tweets* for the user in question. Regarding the example at hand, to the text from *@johndoe* already represented in the X dataset at index 0 corresponds an entry at the same index in the Y dataset with the classification value. The *tweet* from *@johndoe* is therefore labeled as *PSD* as predefined in the configuration file.

As stated before, the algorithm execution could take two possible paths according to configuration:

1. Training and Testing: In this case a X and Y dataset should be considered.

2. Prediction: In this case only a X dataset should be considered.

In the first case both datasets X and Y will be returned by the Dataset Manager. This corresponds to the training and testing of the algorithms.

The other scenario which only returns the X dataset, corresponds to the execution of the algorithms oriented to the prediction of a particular user or users given their texts or *tweets* meaning it's expected from algorithm to realize a classification based on its training. This assumes that the algorithm already realized the training and testing procedures beforehand.

3.1.4 Data Classification

The Data Classification stage in figure 3.2 contains the classification algorithms such as the Master Algorithm and all of his Subject Algorithms as depicted in figure 3.1. The detailed execution flow of this stage is present in figure 3.6.

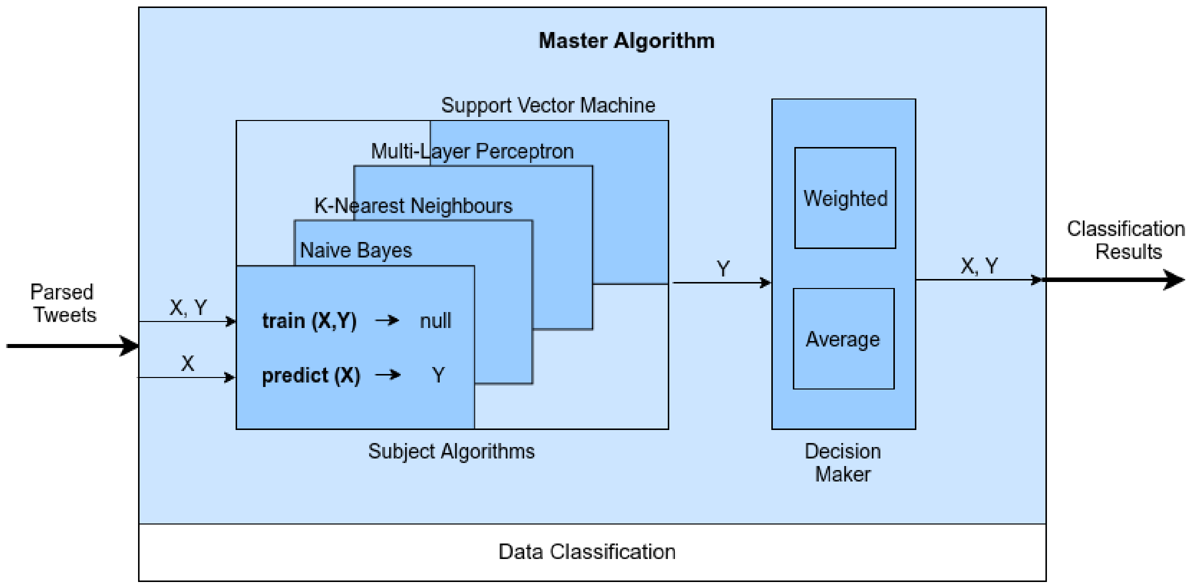


Figure 3.6: Data Classification execution flow.

Introduction

The Data Classification stage is were all the the texts or *tweets* fetched from the database and properly parsed are classified based on a set of algorithms that compose the Master Algorithm. The execution flow of this stage, illustrated in figure 3.6 and exemplified in pseudo-code excerpt C.3, can be summarized as follows:

1. The *tweets* or texts returned by the Dataset Manager in the Data Pre-processing step are inputed to the Data Classification module and directed to the entry point of the Master Algorithm.
2. Depending on the input content, the Master Algorithm realizes the training and testing or prediction for each subject algorithm by invoking the *train* or *predict* methods.
3. In case the *train* function is invoked, each subject algorithm trains on the input data, outputs the execution results in real time to log files and saves the final trained configuration to a file.
   If, on the other hand, the *predict* function is invoked, the Master Algorithm loads for each subject algorithm, the latest training configuration and realizes the prediction of the input vector X. Again, the execution results are saved to log files.
4. In both cases, the output results regarding the classification of the input vector X, go through the Decision Maker filter which makes a final decision regarding the classification results for each algorithm.
5. After the classification of each user is complete, in case the algorithm only realized a prediction the classification results together with the input data are output to the next stage. If the user only realized training, the training parameters such as *tweets*, features and testing scores are sent to the next layer as well.

Components

The modules that instate this execution stage all take part in the Master Algorithm. Each of the main components is explained in detail next.

Master Algorithm

The focus of the Data Classification step is the Master Algorithm. The role of this component is to orchestrate and dictate the execution of the Subject Algorithms and the remaining components. The Master Algorithm configuration is specified in the configuration file provided at the beginning of the execution.

As stated in the execution summary for this stage, the first duty of the Master Algorithm is to dictate the execution flow of the Subject Algorithms considering the context of the input data by invoking the

train or *predict* methods.
After each subject provided a response regarding the invoked method being *predict*, the Master Algorithm redirects the classification results to the Decision Maker component which makes a decision regarding the classification of each text or *tweet* which in short represent a single user. The decision method is specified in the initial configuration file.
For example, if the aggregate classification results of each algorithm in the Subject Algorithm were for a given user text: *[PSD, PSD, PSD, BE]*, and the selected decision method was *Average*, then the Master Algorithm final classification output for such user would be *PSD*.

Subject Algorithms

This algorithm grouping component is a critical component of the whole project. The subject algorithms are a set of well known classification algorithms such as *Neural Networks*, the *Naive Bayes* classifier and more. Each subject algorithm belonging to the group has two main functions which are:

- train
- predict

The *train* function receives as input arguments the X and Y arrays inputed by the Dataset Manager and returns no content, producing only the training and testing results file and the final training configuration file to be used in the future for prediction. It is expected for this method, as illustrated in figure 3.6, to not return any Y output array as its sole purpose is to realize algorithm training.

The *predict* function receives as input the X array from the previous stage and realizes a prediction of the text depicted in each vector of the array. This function considers the training configuration file for the subject algorithm where all the learnings are saved. The output of this function is a Y array where each composing vector contains the classification result for each entry in the X dataset.

For the current implementation of this thesis, the only configured classification algorithms are: *Multi-Layer Perceptron, K-Nearest Neighbours, Support Vector Machine* and the *Naive Bayes* classifier. Each algorithm in this module is configured differently and therefore it is required to specify in the initial configuration file the parameter values particular to each one. If none are specified the algorithm will assume the defaults which enable no control over tuning.

It is also possible to add new classification algorithms to the Subject Algorithms module without much work, needing only to ensure that the new algorithms implement the *train* and *predict* methods accordingly.

Decision Maker

The final module that composes the Master Algorithm is the Decision Maker. This component is only part of the execution in case the execution goal is to realize a prediction. In case this component belongs to the execution its behavior is depicted in in figure 3.7.

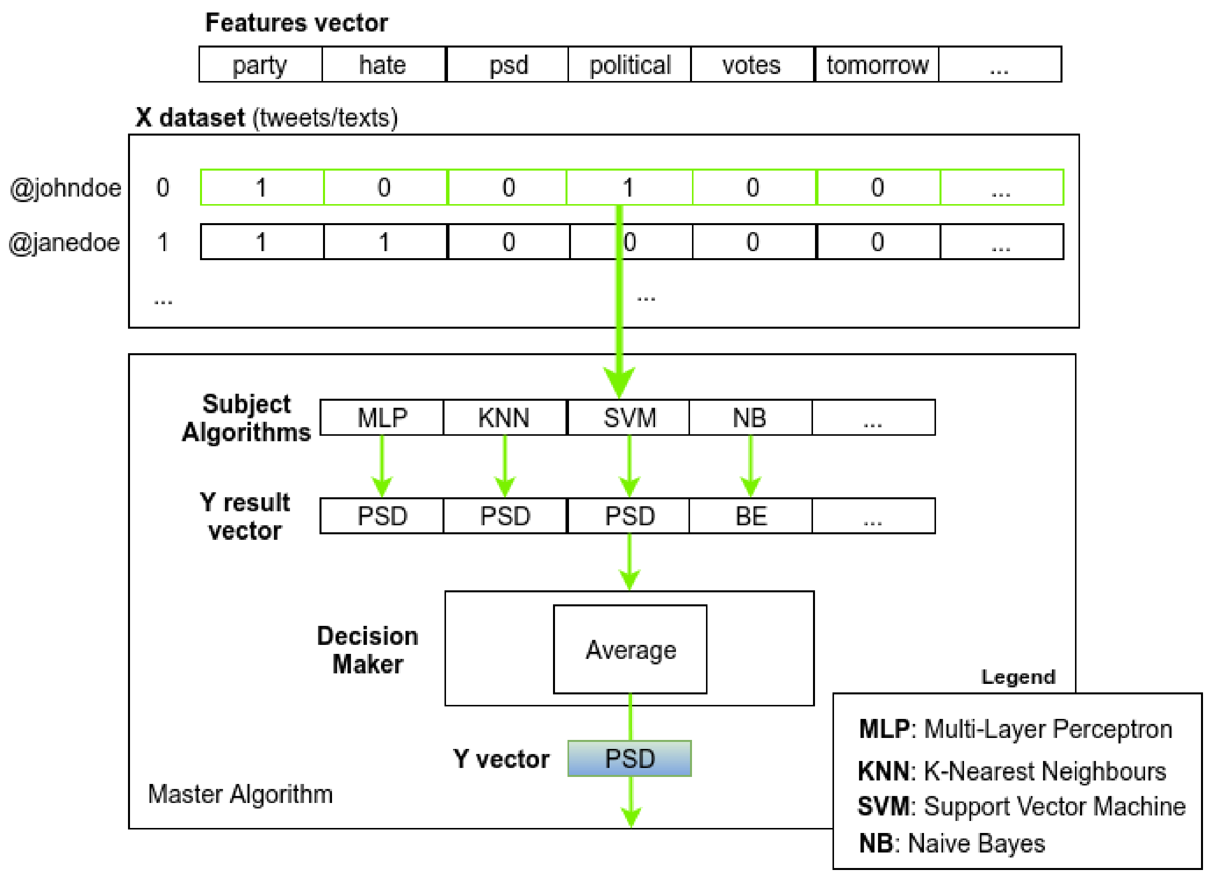


Figure 3.7: Text prediction scenario using *Average* decision method.

The Decision Maker component role is to provide a unique output vector Y taking into consideration the outputs given from each Subject Algorithm in regard to the classification of a text or *tweet*.

As illustrated in figure 3.7, a already parsed text in the form of a numerical input vector X is fed to each subject algorithm for classification. Each algorithm based on its particular training provides a class which they consider to best match the given input. For the example at hand, most classify the text as belonging to class *PSD*. At this point, the Decision Maker based on the *Average* decision method, which returns the most common class in the Y result vector, decides the final class the text should belong to is *PSD* leaving the input from NB unconsidered.

The other decision method also implemented in this solution corresponds to a *Weighted* deliberation. In this case the Decision Maker takes into account the prediction accuracy of each algorithm given in the training phase when assigning the final class to the text or *tweet*. For example, if the Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) algorithms obtained poor training score of only 10% accuracy but the Naive Bayes (NB) algorithm secured an accuracy score of 60% then, the Decision Maker would settle for the *BE* class. This competition would be 30% for the first three algorithms versus 60% for the latter. In short, the Decision Maker favors the class given by the algorithm or algorithms which have the highest aggregated accuracy score.

3.1.5 Result Presentation

The final execution step corresponds to the presentation of results regarding the overall execution of the algorithm. This execution stage has the purpose of presenting to the user a set of statistical information in graphical format concerning the performance of the algorithms. The graphics produced in this step only account for the training and testing of the Subject Algorithms. Throughout the following chapter the outputs of this execution stage will be discussed and detailed further.

3.2 Conclusion

In this chapter it was described the algorithm developed in this thesis in regard to its architecture and execution. The main modules and their roles were throughly explained with the hope of enabling a more clear understanding of the results described in the next chapter.

4

Results

Contents

In the previous chapter the algorithm architecture and execution details were explained in order to provide a clear system overview which will become clearer as the case studies are discussed. In this chapter the results derived from the system execution are analyzed.

Firstly, following the content of the previous chapter the initial configuration details are briefly disclosed in order to provide a clear understanding regarding the algorithm standpoint.

Next, the data details regarding the *tweets* used for training and prediction are reviewed and its limitations are clarified following an explanation of the evaluation metrics used to appreciate the developed algorithm.

Finally a set of three case studies are presented for review, where the most important aspects of each are highlighted pointing the advantages and shortcomings of the algorithm in each scenario.

4.1 Initial Configuration

All the experiments were conducted using a Toshiba Tecra laptop with a 256GB SSD, 8GB of RAM and a quad-core Intel core-i7 vPro 3.00GHz CPU with the exception of the *tweets* extraction stage that required a continuous running computer for a week's timespan. In this case an HP desktop with a 256GB SSD, with 16GB of memory and a server grade Intel Xeon 3.40GHz CPU was required.

In each case study presented in this section the average execution time was of 12 minutes using the Toshiba laptop. The most time consuming process was that of the *tweet* extraction which took 1 week until finishing considering a single thread of execution with room for improvement.

4.1.1 Configuration Files

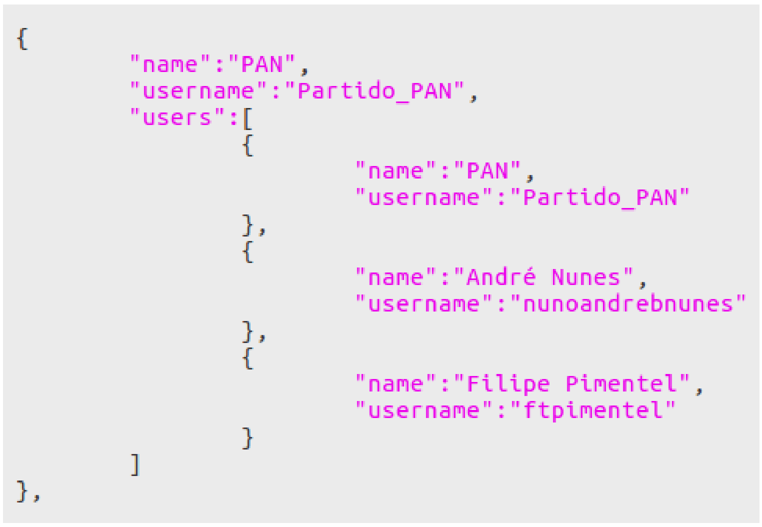
As referenced throughout chapter 3 regarding the algorithm implementation, it is required that a set of configuration files are properly configured for the algorithm to work as expected by the user who is executing it.

Filename	**Description**
settings.py	Hosts the configurations for the Master Algorithm and its subjects among others as well as log definitions.
parties-twitter-cfg.json	Hosts the configurations for the labels which will be considered on the classification step both for training and predicting.
user-input-cfg.json	Enables the user to insert texts for the algorithm to classify on the prediction step.

Table 4.1: Algorithm configuration files and corresponding description

In table 4.1 the required configuration files to be considered prior to execution are described. The most important configuration file is the *settings.py* file. As the description enlightens, this file is responsible for providing to all algorithms, such as the Master Algorithm, his subjects among others all of their configuration details.

The *parties-twitter-cfg.json* file defines the classes or labels which will be considered by the algorithm. It's in this file that all twitter party users are grouped into classes. For each label to be considered by the algorithm a group of usernames from twitter corresponding to a group of politicians is assigned. Below in figure 4.1 is an excerpt example of such configuration file for a given party with its users.



```
{
        "name":"PAN",
        "username":"Partido_PAN",
        "users":[
                {
                        "name":"PAN",
                        "username":"Partido_PAN"
                },
                {
                        "name":"André Nunes",
                        "username":"nunoandrebnunes"
                },
                {
                        "name":"Filipe Pimentel",
                        "username":"ftpimentel"
                }
        ]
},
```

Figure 4.1: Configuration file example considering labels used for classification.

The excerpt illustrated in figure 4.1 states that a class entitled *PAN*, which in this case defines a left wing political party, corresponds to the twitter username *Partido_PAN*. To this class or political party belong a group of users, namely the party itself (*Partido_PAN*), and 2 other users identified on twitter by the *nunoandrebnunes* and *ftpimentel* usernames.

The *tweets* of the users defined in figure 4.1 are stored in the database and will be fetched in the training stage according to this configuration file. These *tweets* will be used for training and testing purposes given that to these users and their *tweets* its assigned a label for classification which is the label *PAN*.

The remaining configuration file *user-input-cfg.json*, which is optional, is used for prediction. By default after training, the algorithm selects a random set of users and their *tweets* in order to classify them. If it's configured that the algorithm will realize a prediction using a custom user-defined text it's in this file that those texts will be hosted with respect to proper formatting.

4.1.2 Database

The database is the source of information for the algorithm which provides the necessary training, testing and prediction data to the system. The database weights around 500MB in size and holds a total count of 10630 distinct users considering both regular and political associate users.

	Number of Users	**Number of *tweets***
Party Users	42	251222
Regular Users	10584	2050028
Total	10626	2301250

Table 4.2: Database statistics.

The number of distinct *tweets* linked to political users and parties are 251222 while the number of *tweets* from regular users is 2050028. The number of distinct users linked to political parties are 46 while the number of regular users is 10584.

All the *tweets* from regular users, which are stored in the database, are referents to the time period ranging from 20th of August of 2017 to 30th September of the same year which corresponded to pre-election period completed in the 1st of October of 2017 with election day in Portugal. Table 4.2 summarizes the statistical details regarding the data stored in the database.

4.2 Evaluation Metrics

In order to evaluate the algorithm performance it is required to set a group of performance metrics. In this particular case the best performance metric would be to know beforehand the political orientation of twitter users which is not possible meaning that the only accurate results would be the ones output in the algorithm training stage which encompasses both training and testing procedures. Therefore, in order to evaluate the performance the first indicator will be the training results for accuracy of each subject algorithm.

Another approach to the evaluation of machine learning algorithms is to analyze their learning curve which plots the relation between the algorithm prediction in regard to the amount of training samples used. This plot is useful as it provides information about the limits of the algorithm and about the quality of the training data. Depending on the type of graphic output by the learning curve a *overfit* or *underfit* scenario may be inferred.

4.3 Case Studies

As described in the introductory excerpt of this chapter this section main objective is to reflect on the results provided by the algorithm execution. Therefore, a set of 3 case studies were realized each with a slightly different setup than the rest. On common ground, all case studies make use of 4 machine learning algorithms namely a (OVA) Support Vector Machine, a Naive Bayes classifier, the K-Nearest

Neighbor classifier and finally a Multi-Layer Perceptron neural network. These algorithms are referred to as *subject algorithms*.

In each scenario the subject algorithms are trained in a different set of label configurations, the first being a set of the most known political parties in Portugal, the second trains on Left and Right wings using a feature extraction algorithm known as RAKE and finally the last which is similar to the latter excluding the usage of the RAKE algorithm for feature extraction. In the end of training all algorithms realize a real prediction in order to obtain direct training performance from political users and to classify unknown users to each pre-configured class. In order to further understand the algorithms performance the predictions realized in each case study are made on the political users which were used before for training purposes.

Class or Label		**Twitter usernames**
Right	CDS-PP	_CDSPP, ribeiroecastro, helderamaralcds, diogo_feio, Telmo_Correia, CristasAssuncao
	PPD-PSD	DuarteMarques, ppdpsd, ppimpao, PSantanaLopes, CarlosCoelhoPE, Leitao_Amaro, aguiarbranco, passoscoelho, JBacelarGouveia
Left	PS	Joaogalamba, tbribeiro, AnaGomesMEP, psocialista, aapbatista, antoniocostampm, JSLisboa, FMedina_PCML, pnsantos_seap
	BE	EsquedaNet, catarina_mart, mmatias_, PedroFgSoares, JoanaMortagua, MRMortagua
	PAN	Partido_PAN, nunoandrebnunes, ftpimentel
	PCP-PEV	abrildenovo, migueltiago, OsVerdes, AntonioFilipe, PCP_Aveiro, CDUPCPPEV, JMRGoncalves1, brunoramosdias, _pcp_

Table 4.3: Classes or labels as defined in the configuration file for each case study.

Table 4.3 displays the configuration file setup for each case study scenario. In this table are assigned to a class the usernames of politicians who have an account on Twitter. For example, the user *_CDSPP* in the first case study belongs to the class *CDS-PP* and in the last 2 case studies scenarios belongs to the class *Right* being that in the first case study its label is the party itself.

The first steps taken towards the training of the algorithms were to gather a set of features that could represent the entire dataset of *tweets* present in the database. The generation of features as explained in 3 may or may not consider the usage of the RAKE algorithm, a scenario which is analyzed in case study 2. The usage of RAKE enables the feature array to consider features with more than a single word which would be considered an expression.

The generation of features for every case study considered a set of 350 random *tweets* per political user and party which resulted in around 13300 *tweets* parsed for the generation of the features list. This list is comprised of 250 words or expressions, depending on the scenario, which represent the 250 most common words or expressions found in the sample subset of 13300 *tweets*.

For the training of each subject subject algorithm which includes up to 4 machine learning algorithms a set of 600 random *tweets* per political user were considered in resemblance to the feature extraction step which resulted in around 22100 *tweets* in total. These *tweets* were used for the training of each algorithm.

Each of the subject algorithms was briefly described in chapter 2, explaining the behavior and characteristics of each one. The configuration and tuning of each algorithm followed the approach recommended by the *scikit-learn* machine learning library used in this project for text analysis cases. Still the configuration details are summarized next for each algorithm.

Support Vector Machine

The implemented Support Vector Machine follows a One-Versus-All or OVA approach meaning a linear binary SVM is trained per class which discriminates the rest. In the end, the one returning the most reliable result is considered to provide the final output. All the remaining configurations are the defaults provided by the *scikit-learn* library.

Naive-Bayes Classifier

The implemented Naive Bayes classifier is similar to the one described in chapter 2. In terms of the definitions over the *scikit-learn* library, the implemented algorithm is a multinomial Naive Bayes classifier, appropriate for text classification scenarios requiring word vector counts inputs as used in the context of this work.

K-Nearest Neighbors

TheK-Nearest Neighbors algorithm is configured to account for the information of the closest 10 neighbors using the *Minkowski* metric for calculating the distance between points. The KNN algorithm and corresponding behavior is explained in chapter 2.

Multi-Layer Perceptron

In the context of this work, the neural network is configured with a single input and output neuron and possesses 2 hidden layers, the first with 5 neurons and the second with 3. Amongst experiments this configuration of layers obtained the best results.

The network regards the sigmoid activation function in the neurons of the hidden layers and the output neuron using an improvement over the gradient descent algorithm entitled Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [46] for weight adjustment.

4.3.1 Case Study I

The first case study covers the classification of users in 6 distinct labels corresponding to each major political party in Portugal. The labels considered for this case study and corresponding users are referenced in table 4.3. These include the People's Party (CDS-PP), Social Democrat Party (PSD), Socialist Party (PS), Left Block (BE), People Animals Nature (PAN) and Unitary Democratic Coalition (PCP-PEV).

The first step to start the training and testing procedure is to obtain a set of features. These features are derived from a set of 13300 *tweets* containing pieces from all political entities present in table 4.3. The resulting set of 250 features are illustrated in table B.1. All the features in table B.1 are comprised of single words only.

Iteration	SVM	NB	KNN	MLP
0	.449	.454	.402	.457
1	.376	.465	.416	.47
2	.456	.462	.413	.469
3	.389	.467	.425	.465
4	.457	.458	.414	.464
Avg	.425	.46	.41	.47

Table 4.4: Case Study I - Cross-validation train-test results.

Afterwards, the algorithms realize the training and testing procedures iteratively by the means of cross-validation, a total of 5 train-test iterations are conducted where for each iteration the whole dataset is partitioned 80% for training and 20% for testing. The results of the cross-validation training are present in table 4.4. The cross-validation procedure is explained in chapter 2.

As illustrated in table 4.4 the average accuracy of each algorithm and the Master Algorithm in total was between 0.4 to 0.5 which means that in the train-test iterations the Master Algorithm correctly classified between 40-50% of the 46 political users. In order to understand the misclassification made by the algorithm, after the train-test validation, the algorithm was programmed to classify the same political users used for its training. The classification results for the prediction of political users are displayed in table 4.5 where the second column holds the users who were classified, the following 5 represent the classifications made by each subject algorithm and the Master Algorithm and the final column has the actual label of the political user.

Considering table 4.5 it's possible to observe that each party and main representative were correctly classified which is a positive indicator, meaning the algorithm grasped the core features that are typical for each party still risking being *overfit*. For example, the user *passoscoelho* is actually the former leader of Social Democrat Party or PSD named Pedro Passos Coelho. He was assigned to its actual party PSD by all algorithms except one. Also, the user *antoniocostapm* an alias to António Costa the current prime minister of Portugal was classified accordingly as well to his party PS.

On the other hand, the user *DuarteMarques* which is a partner of Pedro Passos Coelho on the Social Democrat Party was misclassified as supporter of the Left Block party a left wing group. The user *CristasAssuncao* was also misclassified as a Socialist Party or PS party supporter being she a supporter and leader of the right wing party CDS-PP.

Index	**Username**	**SVM**	**NB**	**KNN**	**MLP**	**Final**	**Actual**
0	AnaGomesMEP	PS	PS	PS	PS	PS	PS
1	AntonioFilipe	PCP-PEV	BE	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
2	CDUPCPPEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
3	CarlosCoelhoPE	PSD	PS	PSD	PAN	PSD	PSD
4	CristasAssuncao	PS	PS	PS	CDS-PP	PS	CDS-PP
5	DuarteLevy	PSD	PSD	PSD	PSD	PSD	IND
6	DuarteMarques	BE	BE	PSD	PS	BE	PSD
7	EsquerdaNet	BE	BE	PSD	BE	BE	BE
8	FMedina_PCML	PS	PS	PS	PS	PS	PS
9	JBacelarGouveia	PSD	PSD	PSD	CDS-PP	PSD	PSD
10	JMRGoncalves1	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
11	JSLisboa	PS	PS	PS	PS	PS	PS
12	JoanaMortagua	BE	BE	PCP-PEV	BE	BE	BE
13	Joaogalamba	PS	BE	PSD	PS	PS	PS
14	Leitao_Amaro	PSD	PSD	PSD	CDS-PP	PSD	PSD
15	MRMortagua	BE	BE	PSD	BE	BE	BE
16	OsVerdes	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
17	PCP_Aveiro	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
18	PSantanaLopes	PSD	PSD	PSD	PS	PSD	PSD
19	Partido_PAN	PAN	PAN	PAN	PAN	PAN	PAN
20	PedroFgSoares	BE	BE	PSD	PAN	BE	BE
21	Telmo_Correia	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP
22	_CDSPP	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP
23	_pcp_	PCP-PEV	PCP-PEV	PCP-PEV	PAN	PCP-PEV	PCP-PEV
24	aapbatista	PSD	PS	PSD	PS	PS	PS
25	abrildenovo	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
26	aguiarbranco	PSD	PSD	PSD	CDS-PP	PSD	PSD

Index	Username	SVM	NB	KNN	MLP	Final	Actual
27	antoniocostapm	PS	PS	PS	PS	PS	PS
28	brunoramosdias	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
29	catarina_mart	BE	BE	CDS-PP	BE	BE	BE
30	diogo_feio	CDS-PP	CDS-PP	CDS-PP	PAN	CDS-PP	CDS-PP
31	ezraklein	PSD	PSD	PSD	PSD	PSD	IND
32	fteconomics	PSD	PSD	PSD	PSD	PSD	IND
33	ftpimentel	PAN	PAN	PAN	PAN	PAN	PAN
34	helderamaralcds	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP	CDS-PP
35	migueltiago	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV	PCP-PEV
36	mmatias_	BE	BE	BE	BE	BE	BE
37	nunoandrebnunes	PAN	PAN	PAN	PAN	PAN	PAN
38	passoscoelho	PSD	PSD	PSD	PS	PSD	PSD
39	pnsantos_seap	PS	PS	PS	PS	PS	PS
40	ppdpsd	PSD	PSD	PSD	PSD	PSD	PSD
41	ppimpao	PSD	PSD	PSD	PS	PSD	PSD
42	psocialista	PS	PS	PS	PS	PS	PS
43	ribeiroecastro	PCP-PEV	PS	PS	PS	PS	CDS-PP
44	rortybomb	PSD	PSD	PSD	PSD	PSD	IND
45	tbribeiro	PS	PS	PSD	PS	PS	PS

Table 4.5: Case Study I - Political users classification.

In table 4.5 a few entries in the far right column named *Actual* are painted yellow with the label IND. The usernames whose entries resulted in the IND entry or Undefined correspond to database errors originated by the process of *tweets* storage. A subset of only 7 users, from the total of 46, were misclassified being that from those 7 a subset of 4 are identified by the label IND. Accounting for the removal of undefined users present in table 4.5 it's possible to derive an achieved accuracy of 92% when classifying users from the same corpora used for training.

As referenced in section 4.2 in order to validate the algorithm performance it's suggested the analysis of the learning curve plot which illustrates the relation between the algorithm accuracy score and the training data. By means this plot it's possible to infer if the algorithms suffer from *overfitting* and/or *underfitting*.

The classification results for this case study present in table 4.5 might suggest a slight *overfitting* scenario meaning the algorithm grasped to some extent the characteristics of each user as it predicted correctly most of them leaving aside undefined users but by doing so compromised the correct classification of new users which are outside the political scope. In this case it's possible the algorithm is lightly *overfit* in contrast to the training accuracy which averaged 50%. An above average *overfitting* scenario would possess training scores with high accuracy values for each iteration an low cross-validation test scores.

Figure A.1 displays the learning curve plots for each subject algorithm in this case study. Analyzing the plots it's possible to draw the following conclusions:

- The Support Vector Machine algorithm has low variance but medium to high bias meaning the algorithm might suffer from *underfitting*.
- The Naive Bayes algorithm identically to the latter may suffer from *underfitting*.
- The K-Nearest Neighbors algorithm is possibly *underfit* as it displays a medium to high bias score and has high variance meaning he might improve with the addition training samples in order to close the variance gap.
- The Multi-Layer Perceptron identically to the first two may suffer from *underfitting* despite having a better bias value.

In order to increase the algorithms bias value its indicated to proceed towards the tuning of the model to be more sophisticated while the decrease of variance is only achieved by the addition of training data. The learning curve plots in contradiction to the prediction results of political users indicate a *underfitting* scenario instead of *overfitting*. These contradicting scenarios may occur due to the complexity of the training data and by the lack of it, meaning that by increasing the amount of data used for training and by providing a more clean and polished dataset the algorithms should be capable of achieving better results in regard to preventing keen *overfitting* and/or *underfitting* scenarios.

4.3.2 Case Study II

This case study covers the classification of users in 2 distinct labels, similarly to the last case study, where each label corresponds to a political wing which may be for this case *Left* or *Right* being the first an aggregate for leftists and the second for right wingers. The labels considered for this case study and corresponding users are referenced in table 4.3 which were parsed from a configuration file where the classes configurations are hosted.

The first step to start the training and testing procedure, as before, is to extract a set of relevant features from the political dataset. The features were retrieved in the same conditions as the first case study and are present in table B.2 being 250 in total.

The generation of features for this case study was made using the RAKE algorithm. RAKE attempts to find relevant expressions in the sample dataset of *tweets* that convey some meaning. This way a

paulo portas	publiquei fotos	pedro passos coelho	sic noticias
camara municipal	passos coelho	antonio costa	js lisboa
new photo	partido comunista portugues	pedro santana lopes	new blog post
heloisa apolonia	santa casa		

Table 4.6: Case Study II - Subset of features which are RAKE expressions.

feature is not restricted to a single word meaning a feature may be an expression or group of words altogether.

From the set of 250 features referenced in table B.2 a subset of 14 are expressions which are displayed in table 4.6. Considering the features present in table 4.6, a few are indisputably oriented towards political figures or themes such as *paulo portas*, *pedro passos coelho*, *antonio costa*, *pedro santana lopes* and *heloisa apolonia* which make reference to some nowadays politicians.

Following the generation of features is the training and testing step. This step is done using cross-validation by splitting the dataset iteratively into a train and test sets with 80% and 20% measures respectively. The results of the train-test phase are present in table 4.7

Iteration	SVM	NB	KNN	MLP
0	.665	.675	.546	.63
1	.667	.675	.557	.63
2	.675	.682	.552	.63
3	.669	.68	.555	.63
4	.664	.671	.552	.63
Avg	.668	.677	.552	.63

Table 4.7: Case Study II - Cross-validation train-test results.

Analyzing the results in table 4.7 in comparison to the results obtained in the previous case study covering 6 labels, the performance increased. This behavior is expected due to the shortage of labels given the classification is now binary. The Multi-Layer Perceptron is the algorithm with the stablest performance of all, which by each training iteration maintained its score. The analysis of the algorithm learning curve present in figure A.2 displays a stagnation of the algorithm validation curve confirming the cross-validation scores obtained in table 4.7. This means following the explanation of the system in chapter 3, that the classification made by the neural network will have a smaller impact in comparison to the rest, discarding in this comparison the K-Nearest Neighbors classifier which exhibited the poorest performance of all which topped at best a score of 55%.

Index	Username	SVM	NB	KNN	MLP	Final	Actual
0	AnaGomesMEP	Right	Left	Left	Left	Left	Left
1	AntonioFilipe	Right	Left	Left	Left	Left	Left
2	CDUPCPPEV	Right	Left	Left	Left	Left	Left
3	CarlosCoelhoPE	Right	Left	Left	Left	Left	Right
4	CristasAssuncao	Right	Left	Right	Left	Left	Right
5	DuarteLevy	Left	Left	Right	Left	Left	IND
6	DuarteMarques	Right	Left	Left	Left	Left	Right
7	LeftNet	Right	Left	Left	Left	Left	Left
8	FMedina_PCML	Right	Left	Left	Left	Left	Left
9	JBacelarGouveia	Right	Right	Right	Left	Right	Right
10	JMRGoncalves1	Right	Left	Left	Left	Left	Left
11	JSLisboa	Right	Left	Right	Left	Left	Left
12	JoanaMortagua	Right	Left	Left	Left	Left	Left
13	Joaogalamba	Right	Left	Left	Left	Left	Left
14	Leitao_Amaro	Right	Right	Right	Left	Right	Right
15	MRMortagua	Right	Left	Left	Left	Left	Left
16	OsVerdes	Right	Left	Left	Left	Left	Left
17	PCP_Aveiro	Right	Left	Left	Left	Left	Left
18	PSantanaLopes	Right	Right	Left	Left	Right	Right
19	Partido_PAN	Right	Left	Left	Left	Left	Left
20	PedroFgSoares	Right	Left	Left	Left	Left	Left
21	Telmo_Correia	Right	Right	Right	Left	Right	Right
22	_CDSPP	Right	Left	Left	Left	Left	Right
23	_pcp_	Left	Left	Left	Left	Left	Left
24	aapbatista	Right	Left	Left	Left	Left	Left
25	abrildenovo	Right	Left	Left	Left	Left	Left
26	aguiarbranco	Right	Right	Left	Left	Right	Right
27	antoniocostapm	Right	Left	Left	Left	Left	Left
28	brunoramosdias	Right	Left	Left	Left	Left	Left
29	catarina_mart	Right	Left	Left	Left	Left	Left
30	diogo_feio	Right	Right	Left	Left	Right	Right
31	ezraklein	Left	Left	Right	Left	Left	IND
32	fteconomics	Left	Left	Right	Left	Left	IND

Index	Username	SVM	NB	KNN	MLP	Final	Actual
33	ftpimentel	Right	Left	Left	Left	Left	Left
34	helderamaralcds	Right	Left	Left	Left	Left	Right
35	migueltiago	Right	Left	Left	Left	Left	Left
36	mmatias_	Right	Left	Left	Left	Left	Left
37	nunoandrebnunes	Right	Left	Left	Left	Left	Left
38	passoscoelho	Right	Left	Right	Left	Left	Right
39	pnsantos_seap	Right	Left	Right	Left	Left	Left
40	ppdpsd	Right	Left	Left	Left	Left	Right
41	ppimpao	Right	Left	Right	Left	Left	Right
42	psocialista	Right	Left	Left	Left	Left	Left
43	ribeiroecastro	Right	Left	Left	Left	Left	Right
44	rortybomb	Left	Left	Right	Left	Left	IND
45	tbribeiro	Right	Left	Left	Left	Left	Left

Table 4.8: Case Study II - Political users classification.

Analyzing the classification or prediction results in regard to the political users displayed in table 4.8 it is possible to detect a few classification anomalies. Discarding the undefined users on error which are identified by the yellow color and label IND in the last column, a set of 9 users from the total of 42 political users were classified in error. Discarding the undefined users this results in a mislead of around 22% when classifying users from the same corpora as training. This results in 78% obtained accuracy. The next case study attempts a binary classification alike this one discarding the usage of RAKE. Comparing the results of each case its proven that the usage of RAKE worsened the overall results both in the cross-validation step in terms of score as well as in the final classification of the political users.

By analysis of the learning curve plot for each algorithm, in resemblance to the first case study, the K-Nearest Neighbors classifier still suffers from high variance as the gap between curves is considerable large indicating that it would benefit from the increase of training samples. The remaining 3 algorithms closed the variance gap, increasing in accuracy while decreasing the bias value, with the exception of the Multi-Layer Perceptron which stagnated midway with a high bias value equal to a possible low prediction score. For this algorithm in particular, which may be suffering from *underfitting*, increasing the complexity of the model could yield better and more precise results in resemblance to the previous case study.

4.3.3 Case Study III

The final case study is ultimately similar to the previous. In this scenario, the algorithms attempt to classify users as supporters of the *Left* or Right political wings using a pre-generated set of features representative of the training set. This features are referenced in table B.3 which in this scenario were not generated using the RAKE algorithm but were obtained through the *bag of words* frequency method using the space separator, meaning each feature is composed solely of a single word obtained by splitting a sentence using the space character.

Iteration	SVM	NB	KNN	MLP
0	.692	.687	.604	.683
1	.689	.693	.607	.694
2	.68	.686	.61	.69
3	.682	.689	.597	.686
4	.69	.689	.6	.686
Avg	.687	.689	.604	.688

Table 4.9: Case Study III - Cross-validation train-test results.

After the generation of features, each subject algorithm realized the cross-validation train-test iterations in resemblance to the previous case studies which yielded the results present on table 4.9.

The average accuracy for all algorithms was close to 67% as illustrated in table 4.9. Comparing the results of the previous case study, where the usage of RAKE was considered achieving 63% accuracy in cross-validation, the resulting differential was close to 4%. From this brief analysis it can be concluded in draft that the usage of the RAKE keyword extraction method might be inadequate or counterproductive in the Twitter network context. The most positively affected algorithm from the non-usage of RAKE was the Multi-Layer Perceptron neural network which increased around 5% in accuracy comparing to the previous case study results visible in table 4.7. The remaining algorithms increased in accuracy slightly while maintaining the aspect of their learning curves in regard to the previous case study. The learning curves for each algorithm for this scenario are displayed in figure A.3.

Index	**Username**	**SVM**	**NB**	**KNN**	**MLP**	**Final**	**Actual**
0	AnaGomesMEP	Right	Left	Right	Left	Left	Left
1	AntonioFilipe	Right	Left	Right	Left	Left	Left
2	CDUPCPPEV	Left	Left	Left	Left	Left	Left
3	CarlosCoelhoPE	Right	Left	Right	Right	Right	Right

Index	Username	SVM	NB	KNN	MLP	Final	Actual
4	CristasAssuncao	Right	Left	Left	Left	Left	Right
5	DuarteLevy	Left	Left	Right	Left	Left	IND
6	DuarteMarques	Right	Left	Right	Right	Right	Right
7	LeftNet	Right	Left	Left	Left	Left	Left
8	FMedina_PCML	Right	Left	Left	Left	Left	Left
9	JBacelarGouveia	Right	Right	Right	Right	Right	Right
10	JMRGoncalves1	Left	Left	Left	Left	Left	Left
11	JSLisboa	Right	Left	Left	Left	Left	Left
12	JoanaMortagua	Right	Left	Right	Left	Left	Left
13	Joaogalamba	Right	Left	Right	Right	Right	Left
14	Leitao_Amaro	Right	Right	Right	Right	Right	Right
15	MRMortagua	Right	Left	Right	Left	Left	Left
16	OsVerdes	Left	Left	Left	Left	Left	Left
17	PCP_Aveiro	Left	Left	Left	Left	Left	Left
18	PSantanaLopes	Right	Right	Right	Right	Right	Right
19	Partido_PAN	Left	Left	Left	Left	Left	Left
20	PedroFgSoares	Right	Left	Right	Right	Right	Left
21	Telmo_Correia	Right	Right	Right	Right	Right	Right
22	_CDSPP	Right	Right	Right	Right	Right	Right
23	_pcp_	Left	Left	Left	Left	Left	Left
24	aapbatista	Right	Right	Right	Right	Right	Left
25	abrildenovo	Right	Left	Left	Left	Left	Left
26	aguiarbranco	Right	Right	Right	Left	Right	Right
27	antoniocostapm	Right	Left	Left	Left	Left	Left
28	brunoramosdias	Right	Left	Left	Left	Left	Left
29	catarina_mart	Right	Left	Left	Left	Left	Left
30	diogo_feio	Right	Right	Right	Right	Right	Right
31	ezraklein	Left	Left	Right	Left	Left	IND
32	fteconomics	Left	Left	Right	Left	Left	IND
33	ftpimentel	Right	Left	Left	Left	Left	Left
34	helderamaralcds	Right	Right	Right	Right	Right	Right
35	migueltiago	Right	Left	Left	Left	Left	Left
36	mmatias_	Left	Left	Left	Left	Left	Left

Index	Username	SVM	NB	KNN	MLP	Final	Actual
37	nunoandrebnunes	Right	Left	Left	Left	Left	Left
38	passoscoelho	Right	Right	Right	Left	Right	Right
39	pnsantos_seap	Right	Left	Left	Right	Right	Left
40	ppdpsd	Right	Right	Right	Right	Right	Right
41	ppimpao	Right	Right	Right	Right	Right	Right
42	psocialista	Right	Left	Left	Left	Left	Left
43	ribeiroecastro	Right	Left	Left	Right	Right	Right
44	rortybomb	Left	Left	Right	Left	Left	IND
45	tbribeiro	Right	Left	Right	Right	Right	Left

Table 4.10: Case Study III - Political users classification.

Table 4.10 holds the classification results of political users for this case study. It's apparent the increase of accuracy in regard to the previous case study which misplaced 36% of political linked users while for this case the failures covered only 14% of all 42 users discarding the users on error identified by the yellow IND label. For this case study the prediction accuracy on the same corpora achieved near 85% classification success.

To finish the case study analysis the algorithms were directed to perform an overall classification of users unlinked to any political party known beforehand. The purpose of this prediction is to obtain a statistical perspective based on the setup of the Master Algorithm for this case study in regard to the political orientation of the Portuguese Twitter users.

Label / Prediction	Left	Right	**Total**
Absolute	8705	1879	10584
Percentage (%)	82%	18%	100%

Table 4.11: Case Study III - Classification of regular users.

According to the results in table 4.11, complemented by figures 4.2 and 4.3, 82% of the Portuguese population are *Left* wing supporters while the remaining 18% are supporters of the *Right*. Figures 4.2 and 4.3 illustrate the classification results in a graphical manner discriminating the overall classifications given by each algorithm.

It's possible to conclude according to figure 4.2 that the Naive-Bayes classifier and the Multi-Layer Perceptron were the algorithms most considerate to the Left while the remaining algorithms leaned

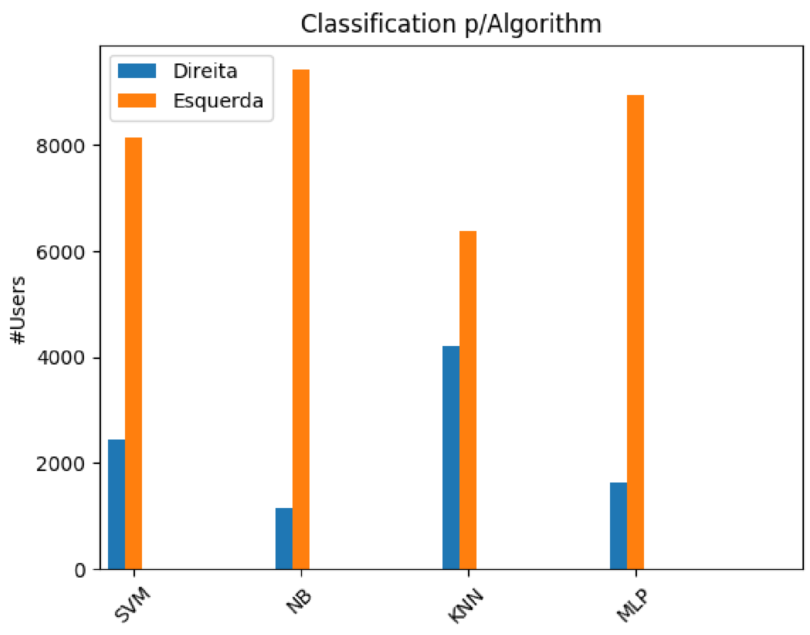


Figure 4.2: Case Study III - Overall prediction grouped by algorithm.

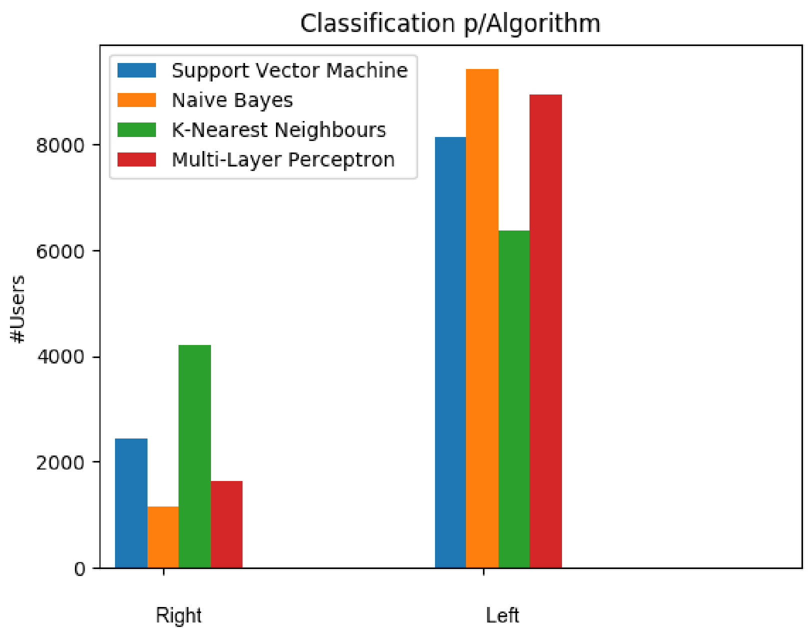


Figure 4.3: Case Study III - Overall prediction grouped by label.

towards the right more deliberately. This information can also be withdrawn from figure 4.3 where it's visible from the density of the histogram bars that most users from the regular users dataset are *Left* wing oriented.

5

Conclusions and Future Work

Contents

This thesis presents a system capable of realizing classification of regular Twitter users based on the content of their *tweets*. The algorithm uses a set Natural Language Processing (NLP) techniques to transform the textual content of *tweets* in to a machine readable array to be consumed by a set of machine learning algorithms.

On a first iteration the algorithm is trained on political *tweets* in order to obtain relevant input in regard to what identifies a determined political side. The algorithm is then tested by facing a set of unclassified users and their *tweets* which are assigned to a political side or party they are most likely to support based on their textual content.

The implemented system was tested in three case studies against slightly different scenarios using different configurations in order to verify the robustness and overall performance of it and each algorithm that composes it. The results obtained were promising. The next sections focus on the key findings and limitations that were discovered during the development of this work where a set of different paths are proposed for further development on this theme.

5.1 Conclusions

In this work 4 different machine learning algorithms namely a Support Vector Machine, a Naive Bayes classifier, the K-Nearest Neighbors algorithm and a Multi-Layer Perceptron neural network are tested to verify their ability to predict the political orientation of a predetermined set of Twitter users. A group of 3 case studies were considered for analysis where the performance of the system in all 3 scenarios yielded promising results.

In the first case study each algorithm is tested to assign Twitter users to one of 6 labels representing 6 major political parties in Portugal. The second case study, which considered the usage of the RAKE keyword extraction algorithm used for the generation of features, attempted to classify a user as a Left or Right wing supporter in resemblance to the third and final scenario where without the consideration of RAKE, the same algorithms classified the same set of users with a more promising output.

From the results obtained it can be concluded firsthand that the usage of the RAKE algorithm in the social network context is limited according to the results obtained in case study 2 in comparison to the following. The poor performance of the RAKE algorithm may be due to the textual quality of the *tweets* as the length of a *tweet* is limited to around 28 characters, as disclosed by Twitter employee Isaac Hepworth [47], meaning the occurrence of relevant textual expressions occurs with low probability.

In regard to the classification results, the lower the number of classes the highest the score as it was expected. Still, the classification results in terms of accuracy did not correspond to what was initially hoped. The shortage of libraries that provide support for the Portuguese lexicon limited the performance of each algorithm in a sense that the data preprocessing was not the best. For example, the usage of stemming techniques as well as synonym consideration could have improved the overall performance results. Another apparent defect the developed system did not account was the social network writing style that is syntactically poor, being subject to unfortunate textual errors which in the end compromised the results.

The most apparent pitfall encountered in the development of the system was the lack of available qualified training data. As it was already explained each subject algorithm requires a set of *tweets* for training which are produced by political linked users existing on Twitter. In comparison to the American politicians, the Portuguese politicians do not use Twitter for campaigning as much as the ones in the North America being their social activity very low and with low quality text. The short amount of qualified data surely limited the algorithm performance and was one of the greatest constraints found during development.

Still the overall results obtained in each case study proved to be interesting enough to pursue this path of application of machine learning techniques in a social networking context.

5.2 Future Work

As a follow-up to this work, several directions can be explored to improve the capability and results of the system described in this thesis. Some of those directions are:

- The quality of the training data was the most responsible for the lack of results. As such, improving the knowledge base, even outside the Twitter social network, might be a good approach to better the performance of the algorithm.

- If using the Portuguese lexicon, the best route to take might be to develop or search for a solid natural language processing library to serve as a foundation able to interpret the syntax and semantic of the Portuguese language. The support for stemming, synonym analysis, and string similarity techniques may prove to be a good approach.

- It could be interesting to switch the context of this thesis to tackle the social activities of other countries such as the United States of America or the United Kingdom as the underlying architecture of this system allows a swift transition of context, needing only to extract new *tweets* and make minor adjustments. The Twitter social activity in such countries may be greater.

- The system in its current state does not allow a multi-threaded environment neither any fine tuning in regard to the amount of *tweets* used for training of each algorithm in particular. As seen, the K-Nearest Neighbors algorithm in particular could benefit from the addition of training data which is not supported by the system as the amount of *tweets* is equally distributed. Adding this kind of configuration definitions could increase overall performance.

- The final suggestion regards the tuning of the machine learning algorithms which can be developed further always accounting for the balance of the bias-variance trade-off in regard to *underfitting* and *overfitting*.

Bibliography

[1] S. Edosomwan, S. K. Prakasan, D. Kouame, J. Watson, and T. Seymour, "The history of social media and its impact on business," *Journal of Applied Management and entrepreneurship*, vol. 16, no. 3, p. 79, 2011.

[2] Statista, "Global social networks ranked by number of users." [Online]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[3] M. Choy, M. L. Cheong, M. N. Laik, and K. P. Shung, "A sentiment analysis of singapore presidential election 2011 using twitter data with census correction," *arXiv preprint arXiv:1108.5520*, 2011.

[4] M. Marchetti-Bowick and N. Chambers, "Learning for microblogs with distant supervision: Political forecasting with twitter," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2012, pp. 603–612.

[5] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment." *Icwsm*, vol. 10, no. 1, pp. 178–185, 2010.

[6] B. O'Connor, R. Balasubramanyan, B. R. Routledge, N. A. Smith *et al.*, "From tweets to polls: Linking text sentiment to public opinion time series." *Icwsm*, vol. 11, no. 122-129, pp. 1–2, 2010.

[7] T. Weinberg, *The new community rules: Marketing on the social web*. O'Reilly Media, 2009.

[8] S. Stieglitz and L. Dang-Xuan, "Social media and political communication: a social media analytics framework," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1277–1291, 2013.

[9] C. B. Williams and G. J. Gulati, "Social networks in political campaigns: Facebook and the congressional elections of 2006 and 2008," *New Media & Society*, vol. 15, no. 1, pp. 52–71, 2013.

[10] D. Kreiss, "Seizing the moment: The presidential campaigns' use of twitter during the 2012 electoral cycle," *New Media & Society*, vol. 18, no. 8, pp. 1473–1490, 2016.

[11] S. Phillips, “A brief history of facebook,” *The Guardian*, 7 2007. [Online]. Available: https://www.theguardian.com/technology/2007/jul/25/media.newmedia

[12] S. Jones and G. W. Paynter, “Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications,” *Journal of the Association for Information Science and Technology*, vol. 53, no. 8, pp. 653–677, 2002.

[13] A. Hulth, “Combining machine learning and natural language processing for automatic keyword extraction,” 2004.

[14] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[15] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank, “Improving browsing in digital libraries with keyphrase indexes,” *Decision Support Systems*, vol. 27, no. 1-2, pp. 81–104, 1999.

[16] M. A. Andrade and A. Valencia, “Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families.” *Bioinformatics (Oxford, England)*, vol. 14, no. 7, pp. 600–607, 1998.

[17] S. Rose, D. Engel, N. Cramer, and W. Cowley, “Automatic keyword extraction from individual documents,” *Text Mining: Applications and Theory*, pp. 1–20, 2010.

[18] M. Laver, K. Benoit, and J. Garry, “Extracting policy positions from political texts using words as data,” *American Political Science Review*, vol. 97, no. 2, pp. 311–331, 2003.

[19] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] I. Rish, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM, 2001, pp. 41–46.

[21] P. Domingos and M. Pazzani, “On the optimality of the simple bayesian classifier under zero-one loss,” *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[22] J. Kazmierska and J. Malicki, “Application of the naïve bayesian classifier to optimize treatment decisions,” *Radiotherapy and Oncology*, vol. 86, no. 2, pp. 211–216, 2008.

[23] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A bayesian approach to filtering junk e-mail,” in *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62, 1998, pp. 98–105.

[24] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[25] S. L. M. S. D. Everitt, B. S.; Landau, *Miscellaneous clustering methods. Cluster Analysis, 5th Edition*, 2011.

[26] R. Frank, “The perceptron a perceiving and recognizing automaton,” *Cornell Aeronautical Laboratory, Buffalo, NY, USA, Tech. Rep*, pp. 85–460, 1957.

[27] H. Leung and S. Haykin, “The complex backpropagation algorithm,” *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101–2104, 1991.

[28] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[29] W. Youyou, M. Kosinski, and D. Stillwell, “Computer-based personality judgments are more accurate than those made by humans,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 4, pp. 1036–1040, 2015.

[30] D. C. Funder, “Accurate personality judgment,” *Current Directions in Psychological Science*, vol. 21, no. 3, pp. 177–182, 2012.

[31] T. D. Letzring, “The good judge of personality: Characteristics, behaviors, and observer accuracy,” *Journal of research in personality*, vol. 42, no. 4, pp. 914–932, 2008.

[32] L. R. Goldberg, J. A. Johnson, H. W. Eber, R. Hogan, M. C. Ashton, C. R. Cloninger, and H. G. Gough, “The international personality item pool and the future of public-domain personality measures,” *Journal of Research in personality*, vol. 40, no. 1, pp. 84–96, 2006.

[33] M. Kosinski, D. Stillwell, and T. Graepel, “Private traits and attributes are predictable from digital records of human behavior,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 15, pp. 5802–5805, 2013.

[34] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[35] J. Golbeck, C. Robles, and K. Turner, “Predicting personality with social media,” in *CHI’11 extended abstracts on human factors in computing systems*. ACM, 2011, pp. 253–262.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[37] D. Rao, D. Yarowsky, A. Shreevats, and M. Gupta, “Classifying latent user attributes in twitter,” in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*. ACM, 2010, pp. 37–44.

[38] T. Joachims, *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers Norwell, 2002, vol. 186.

[39] E. David, M. Zhitomirsky-Geffet, M. Koppel, and H. Uzan, "Utilizing facebook pages of the political parties to automatically predict the political orientation of facebook users," *Online Information Review*, vol. 40, no. 5, pp. 610–623, 2016.

[40] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni, "Gender, genre, and writing style in formal written texts," *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-*, vol. 23, no. 3, pp. 321–346, 2003.

[41] J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker, "Effects of age and gender on blogging." in *AAAI spring symposium: Computational approaches to analyzing weblogs*, vol. 6, 2006, pp. 199–205.

[42] M. Koppel, J. Schler, and K. Zigdon, "Determining an author's native language by mining a text for errors," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 624–628.

[43] J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker, "Effects of age and gender on blogging." in *AAAI spring symposium: Computational approaches to analyzing weblogs*, vol. 6, 2006, pp. 199–205.

[44] M. D. Conover, B. Gonçalves, J. Ratkiewicz, A. Flammini, and F. Menczer, "Predicting the political alignment of twitter users," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 192–199.

[45] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.

[46] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[47] S. M. Knowledge, "The average tweet length is 28 characters long, and other interesting facts." [Online]. Available: https://smk.co/article/the-average-tweet-length-is-28-characters-long-and-other-interesting-facts

A

Figures

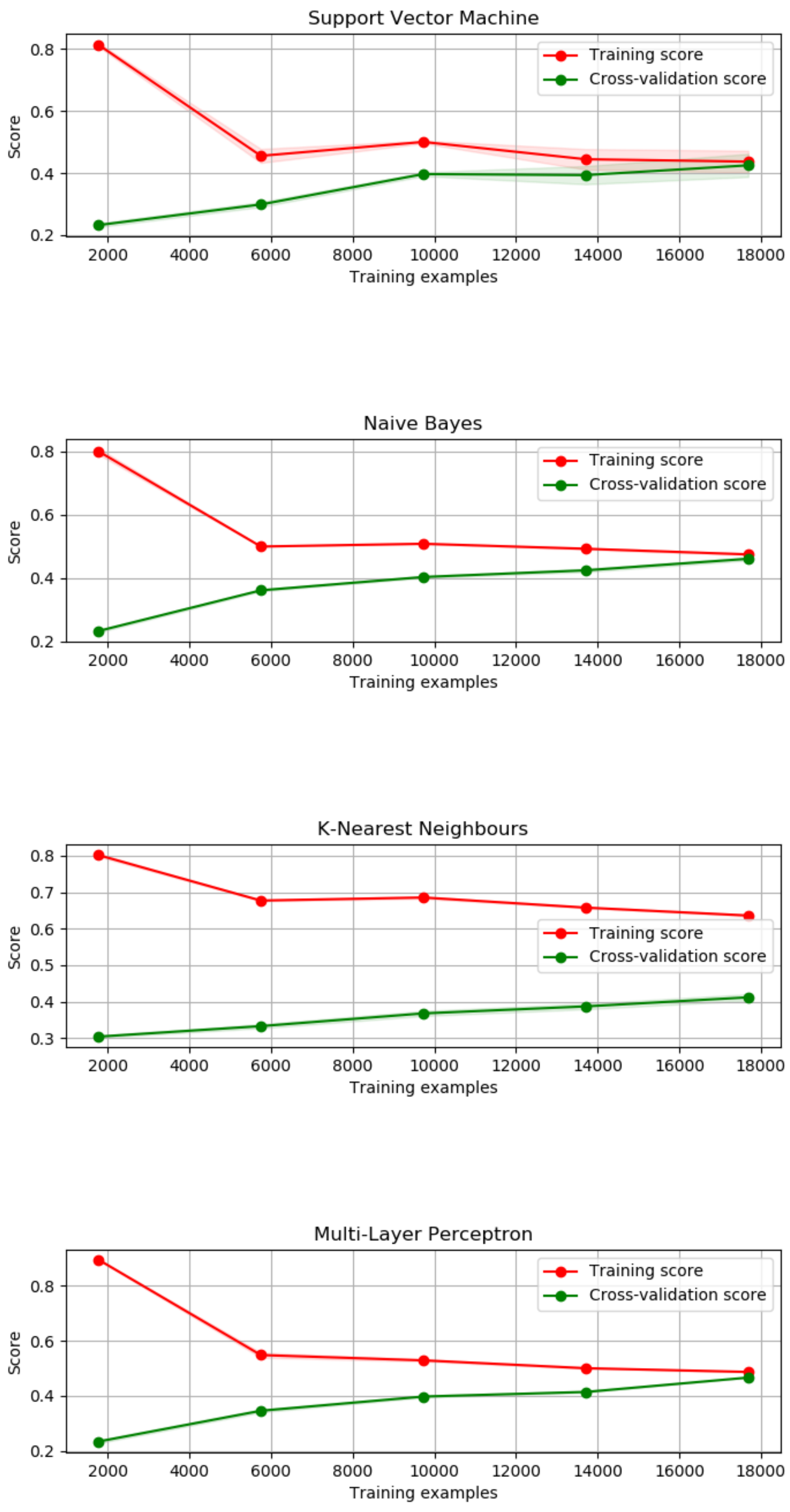


Figure A.1: Case Study I - Learning curves for each Subject Algorithm.

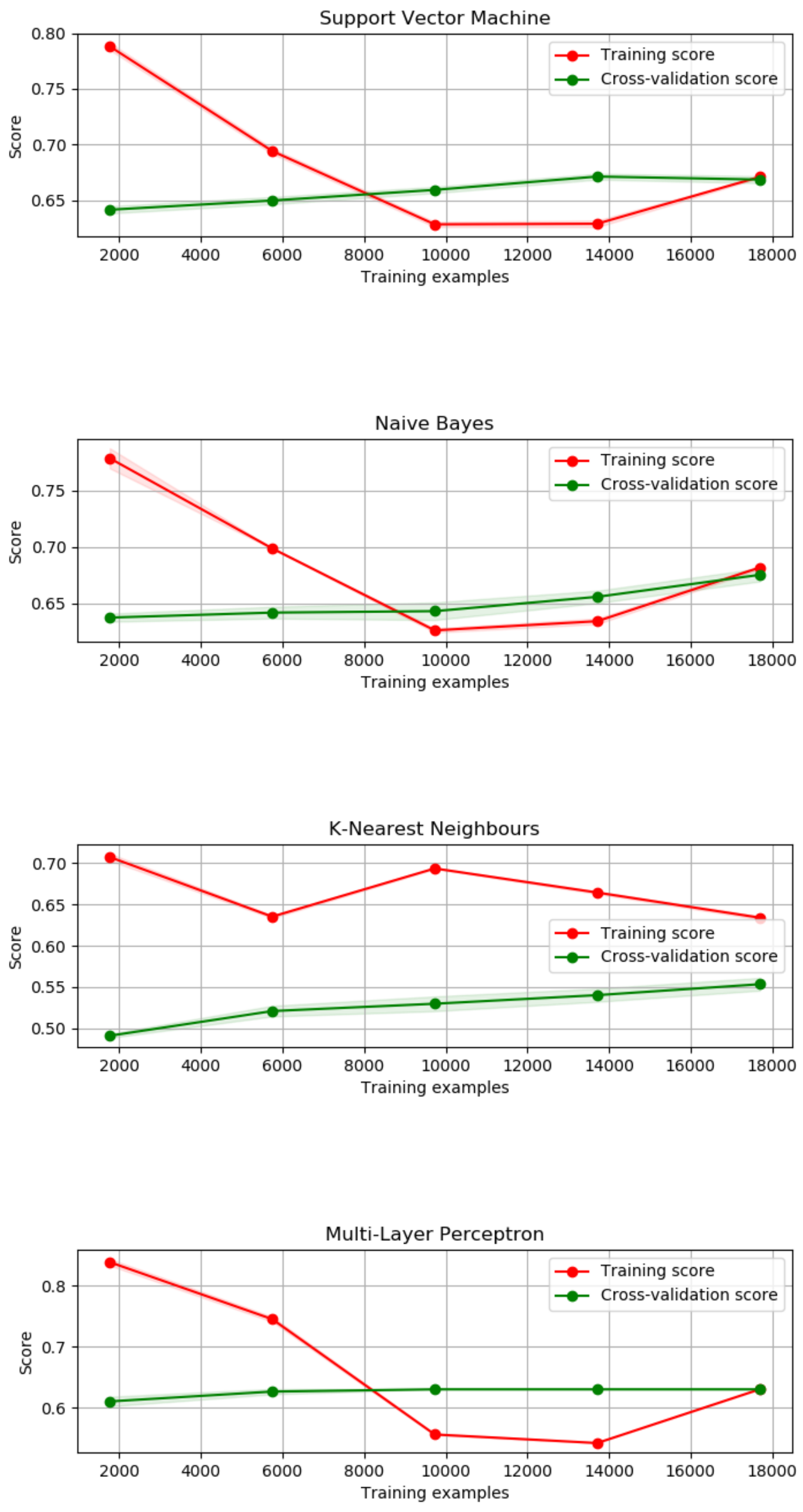


Figure A.2: Case Study II - Learning curves for each Subject Algorithm.

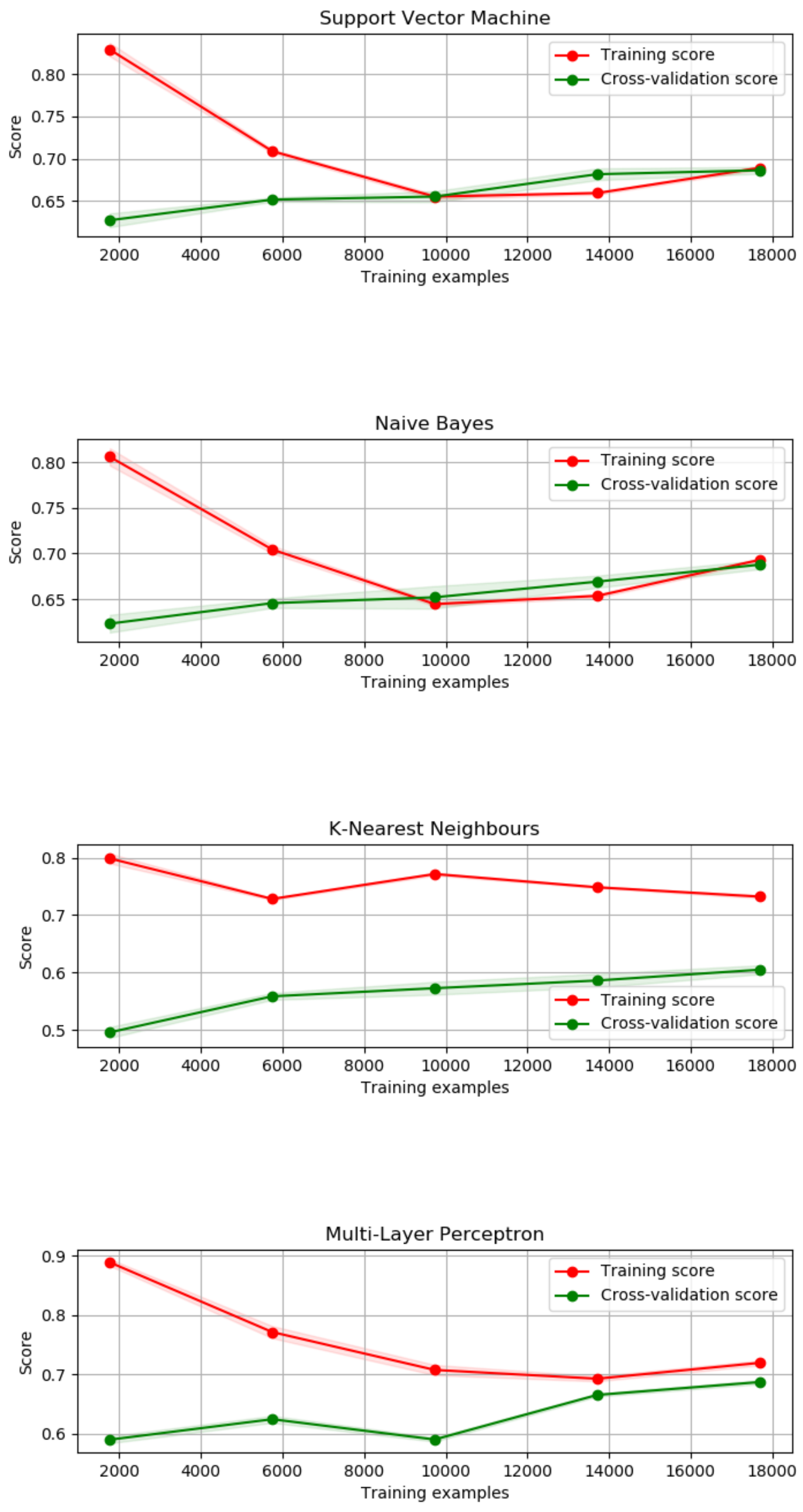


Figure A.3: Case Study III - Learning curves for each Subject Algorithm.

B

Tables

governo	portugal	lisboa	psd	ps	facebook
passos	pcp	pais	#cdu	debate	publiquei
coelho	politica	cds	nacional	pedro	portas
esquerda	presidente	paulo	cdu	antonio	foto
#pcppev	noticias	social	video	ar	assembleia
europa	porto	parlamento	comissao	portugueses	programa
amanha	costa	partido	jose	almada	intervencao
manha	#legislativas	campanha	imprensa	youtube	verdes
eleicoes	ministro	municipal	trabalhadores	republica	publico
#jeronimodesousa	twitter	camara	casa	melhor	reuniao
futuro	entrevista	socrates	mundo	fotos	europeu
visita	luis	>>	economia	publica	pan
#psd	parabens	album	artigo	joao	direitos
defesa	portugues	milhoes	divida	dias	apresentacao
vida	pev	sic	europeia	situacao	falar
saude	orcamento	lei	sousa	conferencia	desemprego
jovens	semana	democracia	#genteseria	santa	crise
portuguesa	new	silva	bloco	sessao	luta
#presidenciais	empresas	#pcp	seguranca	miguel	acordo
direito	marco	forca	candidatura	abril	crescimento
lista	preciso	animais	tvi	tc	congresso
educacao	lido	politicas	vamos	maria	voto
investimento	deputados	escola	amaral	lopes	solidariedade
emprego	propostas	cidade	helder	feira	mudar
soares	jornal	medidas	#portugal	economico	#pev
precisa	proposta	fala	distrito	problema	centro
confianca	alguem	caso	banco	aumento	vou
eleitoral	liberdade	austeridade	rtp	servico	heloisa
<<	impostos	audicao	ferreira	apolonia	iniciativa
continua	photo	socialista	juventude	cultura	historia
posted	causa	js	aveiro	jeronimo	encontro
jantar	ninguem	viseu	candidatos	gente	cavaco
homenagem	ue	opiniao	jorge	apresenta	parlamentar
antena	projeto	disse	salarios	carlos	claro

primeiro-ministro	volta	uniao	#marisa	financas	edicao
partidos	rua	ppdpsd	projecto	candidato	deputado
internacional	sabado	pena	comeca	espaco	fica
relatorio	defende	reforma	grecia	maio	conta
ambiente	euros	frente	estarei	comicio	agua
importante	pagar	net	addthis	paises	fiscal
ideia	contas	nuno	oe	troika	tema
accao	noticia	seguro	mulheres		

Table B.1: Case Study I - Set of 250 features.

governo	portugal	facebook	lisboa	psd
ps	debate	pais	esquerda	presidente
pcp	cds	foto	publiquei	video
europa	porto	amanha	imprensa	manha
republica	reuniao	twitter	politica	ar
album	cdu	almada	entrevista	intervencao
campanha	paulo portas	noticias	trabalhadores	#jeronimodesousa
futuro	parlamento	verdes	programa	publiquei fotos
comissao	pedro passos coelho	portugueses	#cdu	youtube
apresentacao	lei	abril	vida	saude
economia	ministro	semana	tc	mundo
visita	defesa	assembleia	dias	lido
artigo	lista	confianca	conferencia	liberdade
pev	luta	milhoes	falar	acordo
tvi	parabens	deputados	eleicoes	sic noticias
crise	distrito	pan	melhor	jornal
voto	solidariedade	orcamento	aveiro	sousa
cidade	animais	camara	audicao	historia
addthis	camara municipal	candidatura	direito	democracia
maio	candidatos	austeridade	troika	ambiente
passos coelho	aumento	jeronimo	euros	bloco
casa	jovens	feira	propostas	ue
oe	divida	direitos	partido	rtp
proposta	educacao	correio	centro	viseu
publico	investimento	rua	opiniao	situacao
caso	sabado	encerramento	abertura	daqui
braga	marco	socrates	#agenda	passos
frente	vale	sessao	problema	discussao
agua	justica	comecar	jantar	saiba
combate	empresas	emprego	antonio costa	forca
vamos	primeiro-ministro	servico	importante	cultura
impostos	privatizacao	desemprego	antena	posted
qualidade	relatorio	crescimento	causa	seguranca
mulheres	congresso	reforma	inauguracao	claro

vou	#googlealerts	conta	$<<$	gente
noticia	concelho	constituicao	js lisboa	new photo
cara	jsd	partido comunista portugues	edicao	bruxelas
homenagem	grecia	ler	fica	iniciativa
nacional	greve	candidato	tema	dinheiro
financas	android	pedro santana lopes	negocios	aniversario
chegar	comicio	serio	discurso	bes
regresso	festa	pe	new blog post	mudar
participacao	tatus	vitimas	encontro	costa
pagar	participar	ultimos	fala	atus
escola	heloisa apolonia	paises	alguem	adicionei
madeira	politico	razao	santa casa	feito
gestao	preciso	tap	ministerio	coimbra
saida	deputado	leiria	pena	pt
us	perder	politicas	problemas	seccao
accao	projecto	reproducao	universidade	familia

Table B.2: Case Study II - Set of 250 features.

governo	portugal	lisboa	psd	ps	pcp
facebook	passos	debate	#cdu	pais	coelho
publiquei	politica	cds	esquerda	pedro	portas
paulo	presidente	noticias	nacional	#pcppev	social
cdu	video	programa	porto	antonio	foto
assembleia	partido	jose	comissao	ar	youtube
amanha	campanha	almada	parlamento	manha	melhor
eleicoes	europa	costa	reuniao	republica	trabalhadores
publico	portugueses	#legislativas	verdes	casa	europeu
twitter	municipal	imprensa	#jeronimodesousa	portugues	pan
mundo	futuro	intervencao	parabens	camara	entrevista
ministro	artigo	europeia	luis	#psd	visita
publica	socrates	economia	milhoes	escola	seguranca
orcamento	dias	ferreira	sic	silva	apresentacao
vida	defesa	<<	divida	fotos	abril
lista	voto	falar	saude	semana	lei
crise	joao	#pcp	new	direitos	cidade
proposta	democracia	pev	acordo	confianca	sousa
#presidenciais	lido	album	investimento	fala	candidatura
vamos	portuguesa	congresso	tc	juventude	bloco
jornal	santa	soares	tvi	#genteseria	conferencia
vou	claro	miguel	medidas	deputados	direito
caso	rtp	luta	preciso	situacao	historia
#pev	encontro	defende	forca	centro	maria
audicao	js	solidariedade	propostas	empresas	apolonia
jorge	apresenta	sessao	desemprego	vale	#portugal
crescimento	marco	liberdade	pena	iniciativa	continua
economico	contas	animais	distrito	educacao	alguem
photo	projecto	precisa	austeridade	emprego	ppdpsd
problema	eleitoral	heloisa	madeira	internacional	socialista
parlamentar	uniao	addthis	sociais	politicas	aveiro
causa	feira	pt	amaral	candidato	financas
ue	troika	viseu	opiniao	comunista	euro
problemas	helder	gente	oe	cultura	lopes

excelente	paises	edicao	defice	partidos	pm
santos	fica	importante	justica	seguro	estarei
jovens	status	nuno	questiona	diario	posted
combate	$<<$	fcancio	minuto	impostos	agenda
vitoria	aumento	daqui	homenagem	#marisa	mudar
dinheiro	noticia	presenca	sede	servico	frente
reforma	cavaco	candidatos	ouvir	tatus	eur
jantar	mulheres	discurso	politico		

Table B.3: Case Study III - Set of 250 features.

C

Code

Algorithm C.1: Data Collection - download or select tweets based on a set of parameters.

Input : Tweet criteria to be considered on the retrieval of tweets.
Output: Set of raw tweets matching the criteria or none.

```
CollectData ($tweetCriteria$)
    if $tweetCriteria.downloadCriteria \neq null$ then
        $tweets \leftarrow \emptyset$
        foreach $client \in TwitterAPIClients$ do
            $tweets' \leftarrow client.download(tweetCriteria.downloadCriteria)$
            $tweets \leftarrow tweets + tweets'$
        end
        $saveTweetsToDatabase(tweets)$
    end
    if $tweetCriteria.selectCriteria \neq null$ then
        $rawTweets \leftarrow getTweetsFromDatabase(tweetCriteria.selectCriteria)$
        return $rawTweets$
    end
```

Algorithm C.2: Data Pre-Processing - generate features and process tweets for training or prediction.

Input : Set of raw tweets fetched from the Database matching given criteria.
Output: Set of processed/parsed tweets.

```
ProcessTweets ($tweets$)
    $features \leftarrow \emptyset$
    $featureTweets \leftarrow \emptyset$
    if $config.features == null$ then
        /* generate features if not defined in the configuration file */
        $featureTweets' \leftarrow getTweetsFromDatabase(config.featureTweetsCount)$
        $featureTweets' \leftarrow removeStopWordsAndExpressions(featureTweets')$
        if $useRake$ then
            $featureTweets' \leftarrow RAKE(featureTweets')$
        end
        $features \leftarrow mostCommon(featureTweets')$
    end
    else
        $features \leftarrow config.features$
    end
    /* parse tweets which will be used for training or prediction */
    $tweets' \leftarrow \emptyset$
    $tweets' \leftarrow removeStopWordsAndExpressions(tweets)$
    if $useRake$ then
        $tweets' \leftarrow RAKE(tweets')$
    end
    if $training == true$ then
        $parsedTweets \leftarrow getDatasetsXY(tweets', features)$
    end
    else
        $parsedTweets \leftarrow getDatasetX(tweets', features)$
    end
    return $parsedTweets$
```

Algorithm C.3: Data Classification - classification of users from their tweets in training or prediction scenarios.

```
Input   : Set of processed tweets.
Output: No direct output apart from logfiles.

Classify ($parsedTweets$)
    if config.train then
        /* training scenario given X and Y datasets                    */
        $X \leftarrow parsetTweets.getX()$
        $Y \leftarrow parsetTweets.getY()$
        foreach $subject \in MasterAlgorithm.subjectAlgorithms$ do
            $subject.train(X, Y)$
        end
    end
    else
        /* prediction scenario given X dataset only                    */
        $X \leftarrow parsedTweets.getX()$
        $Y \leftarrow []$
        foreach $subject \in MasterAlgorithm.subjectAlgorithms$ do
            $subject.load(config.training)$
            $y \leftarrow subject.predict(X)$
            $Y'.append(y)$
        end
        if config.decisionAverage then
            $Y \leftarrow getAverageClass(Y')$
        end
        else
            $Y \leftarrow getWeightClass(Y')$
        end
    end
```