POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Politecnico
di Torino

Master's Degree Thesis

Mining user reviews on public transport systems using machine learning techniques

Supervisors

Prof. Silvia Anna CHIUSANO

Prof. Luca CAGLIERO

Dott. Elena DARAIO

Candidate

Daniela MARTORANA

April 2022

Summary

Nowadays, with the increasingly pervasive advent of the digital world, we are inundated with large amounts of data of every kind. An important category of such data is the text format. This data might be automatically generated by machines, but more frequently it is user-generated content and it is linked to many daily contexts.
An important part of this data, which will be dealt with in this thesis work, concerns user reviews of products and services. The challenges in this context are both (i) to analyse this large amount of information content to automatically obtain useful and concise information on what people think of a product or service and (ii) to ensure that, with this information, the companies and authorities involved make improvements to their current systems.
Machine learning techniques have been frequently used to analyse text data and extract useful insights. This type of text analysis belongs to the area of *Natural Language Processing (NLP)*.

In this thesis work, we will focus on the field of *Sentiment Analysis (SA)*, to build a classification model able to label a sentence with a positive or a negative sentiment. In this thesis work, two other NLP fields will also be discussed: *Topic Modeling* and *Text Mining*. In parallel, preprocessing techniques will also be addressed.

As a reference case study, a collection of user-provided reviews regarding mobility services and their sustainability in the city of Dubrovnik (Croatia) has been considered. The data corpus contains 2000 labelled reviews that were first retrieved from Trip Advisor and then selected, cleaned and collected into the so-called Gold Standard Corpus (GSC) in a previous research work. Indeed, it is thought to be more useful to directly analyse comments freely provided on social networks, rather than official statistics and traditional surveys carried out by public administrations, as the former are more reflective of the real opinion of users. This analysis is particularly useful because transport practices have an impact on sustainability from a social, economic and environmental perspective and politicians are looking

for ways to address these problems by increasing the efficiency and sustainability of the transport system. Sentiment Analysis, by analysing the opinions expressed by users, can help them to make more informed decisions.

The main objectives of this thesis work are the followings: (i) to define a new methodology to analyse textual data with respect to the traditional methods; (ii) to build consequently a sentiment analysis algorithm that classifies the instances correctly into positive and negative reviews; (iii) to find a classification model that satisfies the objective (ii), while providing a compact and interpretable representation of the model. The model should highlight terms and concepts related to the strengths and weaknesses of mobility system.

A new methodology for Sentiment Analysis has been designed to achieve good accuracy, but also provide an interpretable classification model. Specifically, the textitAssociative Classifier L^3 was chosen to be used. In L^3 the classification model is a set of classification rules. This model is very accurate in many different application domains. In addition this model is easily readable and interpretable.
Before creating the classification model, many other preliminary steps have been addressed in this thesis work. First, several textual pre-processing and data transformation techniques were investigated and applied in order to generate a clean data collection compatible with L^3 or other machine learning algorithms. More specifically, the following operations have been carried out: correction of some formatting errors in the text, expansion of contracted forms, removal of special characters, numbers, accents, extra spaces and stopwords from the text, subdivision of sentences into tokens, PoS tagging of tokens and finally the application of stemmatization and lemming to individual tokens.
Then, Topic Modeling techniques (such as *Latent Dirichlet Allocation (LDA)*) were used to derive additional information content from the data, to carry out both features selection and features extraction, and finally obtain a data representation suitable for the L^3 classification method.
Specifically, based on Topic Modeling five different solutions have been designed and implemented. In *Method 1* the n most important words in the k chosen topics were selected, and then the data were filtered based only on the set of selected words. A Bag of Words representation was created to obtain a configuration compatible with L^3, in which features are the selected words. The other methods map each instance to the corresponding topic representation of that instance (*Method 2A*) or to the corresponding topic representation of each word of that instance (*Methods 2B* and *2C*). Therefore in these methods the features are the k topics chosen. Finally, *Method 3* is very similar to *Method 1*: the n most important words in the k chosen topics were selected, and the data were filtered based only on the set of selected words. Furthermore, each word in the sentence will be mapped to the

corresponding topic. A Bag of Words representation was created, in which features are the selected words + the k topics.
In all these methods, a search in the parameter space was carried out, looking for the values of the parameters n and k (and some other metric depending on the method) that would maximise the value of the accuracy of the prediction of the L^3 classifier applied on the test set.
Different baselines were also considered without the use of topic modeling and/or with a different classifier, such as Support Vector Machines (SVM), in order to have a term of comparison.
The results obtained in this thesis work demonstrate the accuracy and the validity of our proposed approach, which can be extended to other datasets and contexts. The best results obtained are quite good. They are also better than the performance of the baselines with the SVM classifier, when models with the same number of features are compared.
More specifically, the best results of *Method 1* and *Method 3* emerge, which achieve the accuracies of 0.887 and 0.896, respectively. These are the highest results obtained in this work. They are also higher than the accuracies of the corresponding baselines with the SVM classifier applied to data transformed using *Method 1* and *Method 3* (0.882 and 0.891 of accuracies, respectively).
This is an important result even if the gap is not very high. Indeed, it must be considered that SVM is one of the best performing algorithms in machine learning and also specifically in Sentiment Analysis. So, the fact that our method produces a slightly better result than SVM, is a very valuable point.
More generally, as regards *Method 1*, when varying the number of features fed to the classifier, the following considerations can be made. The results with L^3 are significantly higher than the results of the baselines created without the use of Topic Modeling. The results of *Method 1* with L^3 are also slightly better than the results after applying SVM to the same method. Only in a few cases the results with the SVM classifier are slightly superior. This happens usually when the number of features fed to the SVM classifier is significantly higher than a number of features compatible with the L^3 classifier. Indeed, the downside of the method defined in this thesis work is that the process is very slow as the number of features increases. In particular, the computation time starts to become unsustainable when more than 23/24 features are used.
The other methods developed in this thesis work represent the data using topics only. They produced lower accuracies than *Methods 1* and *3*. Probably because mapping the data directly with topics leads to too synthetic representation and too much information is lost. When applying SVM to these methods, L^3 generally produces comparable or slightly inferior results. However, SVM produces a model that is not readable and not interpretable, unlike the rules produced by L^3. These rules are perfectly readable, both with the human eye, with algorithms, and with

graphs.

Deepening this last concept, through the use of these rules, we were able to obtain a readable and summarizing model. It provides the main words and concept associations, regarding positive and negative opinions, about sustainable transport. Through this model, a graphical representation has been created that allows an immediate visual analysis of these concepts. In particular, it allows to analyse the correlation and co-occurrence of concepts and terms together. The insights obtained from this representation can be used by the administrations to improve the service offered.
An alternative representation of the model using SHAP and applying it to SVM was also presented. This was done in order to possibly highlight the strengths of our representation and to assess the complementarity of the two representations and obtain more information.

Many of the things that have been done in this work and the methodology developed, in the future could be extended to other works of the same type. Some improvements that could be carried out are as follows. A way to parallelise the process can be studied or more powerful machines can be used. In this way, the L^3 classifier could also be used with a greater number of features. One could also try other Topic Modeling algorithms, which maybe work better for this dataset and task. For example, *Non-Negative Matrix Factorization (NMF)* algorithm has good performance even with a limited size dataset, like ours. Finally, the interpretability aspect of the model could be further investigated. The preliminary analysis made in this thesis work on SHAP can be deepened, and the complementarity of this method with the rules produced by L^3 can be studied.

Table of Contents

List of Figures

Acronyms

NLP
Neural Language Processing

SA
Sentiment Analysis

GSC
Gold Standard Corpus

AC
Associative Classifier

SVM
Support Vector Machine

VADER
Valence Aware Dictionary and sEntiment Reasoner

NLTK
Natural Language Toolkit

LDA
Latent Dirichlet allocation

PoS
Part of Speech

BoW
Bag of Words

TF-IDF

Term Frequency-Inverse Document Frequency

SHAP

SHapley Additive exPlanations

Chapter 1

Introduction

General context, problem and traditional solutions

Nowadays, with the increasingly pervasive advent of the digital world, we are inundated with large amounts of data of every kind. An important category of such data is the text format. This data might be automatically generated by machines, but more frequently it is user-generated content and it is linked to many daily contexts.
An important part of this data, which will be dealt with in this thesis work, concerns user reviews of products and services. The challenges in this context are both (i) to analyse this large amount of information content to automatically obtain useful and concise information on what people think of a product or service and (ii) to ensure that, with this information, the companies and authorities involved make improvements to their current systems.
Machine learning techniques have been frequently used to analyse text data and extract useful insights. This type of text analysis belongs to the area of *Natural Language Processing (NLP)*.

In this thesis work, we will focus on the field of Sentiment Analysis (SA), to build a classification model able to label a sentence with a positive or a negative sentiment. In this thesis work, two other NLP fields will also be discussed: *Topic Modeling* and *Text Mining*. In parallel, pre-processing techniques will also be addressed.

As far as Sentiment Analysis is concerned, there are many methods to carry out this kind of analysis in literature. Roughly speaking, there are mainly two ways to address the problem of Sentiment Analysis: rule or lexicon-based approaches and machine learning algorithms, which are then divided into classical machine learning algorithms (e.g. Support Vector Machines, Naïve Bayes, Linear Regression) and

neural networks (e.g. Convolutional Neural Network, Recurrent Neural Network) [1].

As a reference case study, a collection of user-provided reviews regarding mobility services and their sustainability in the city of Dubrovnik (Croatia), has been considered. A data analysis methodology based on NLP techniques has been defined to mine useful insights from this data collection.
The data corpus contains 2000 labelled reviews that were first retrieved from Trip Advisor and then selected, cleaned and collected into the so-called *Gold Standard Corpus (GSC)*, in a previous research work of Serna, Soroa, Agerri [2]. Indeed, it is thought to be more useful to directly analyse comments freely provided on social networks, rather than official statistics and traditional surveys carried out by public administrations, as the former are more reflective of the real opinion of users. This analysis is particularly useful because transport practices have an impact on sustainability from a social, economic and environmental perspective and politicians are looking for ways to address these problems by increasing the efficiency and sustainability of the transport system. Sentiment Analysis, by analysing the opinions expressed by users, can help them to make more informed decisions.

The main objectives of this thesis work are the followings: (i) to define a new methodology to analyse textual data with respect to the traditional methods; (ii) to build consequently a Sentiment Analysis algorithm that classifies the instances correctly into positive and negative reviews; (iii) to find a classification model that satisfies the objective (ii), while providing a compact and interpretable representation of the model. The model should highlight terms and concepts related to the strengths and weaknesses of mobility system.

A new methodology for Sentiment Analysis has been designed to achieve good accuracy, but also provide an interpretable classification model. Specifically, the *Associative Classifier (AC)* L^3 [3] was chosen to be used. In L^3 the classification model is a set of classification rules. This model is very accurate in many different application domains. In addition this model is easily readable and interpretable.
Before creating the classification model, many other preliminar steps have been addressed in this thesis work.
First, several textual pre-processing and data transformation techniques were investigated and applied in order to generate a clean data collection compatible with L^3 or other machine learning algorithms.
Then, Topic Modeling techniques (such as *Latent Dirichlet Allocation (LDA)*) were used to derive additional information content from the data, to carry out both features selection and features extraction, and finally obtain a data representation suitable for the L^3 classification method.

Specifically, based on Topic Modeling five different solutions have been designed and implemented.

To summarise, the main contributions and results of this work are to create a new Sentiment Analysis algorithm with high performance and, at the same time, provide an interpretable model, in contrast to other algorithms in literature.
In order to estimate the quality of the proposed approach, it has been compared with traditional solution methods, such as *Support Vector Machines (SVM)*.

The thesis work is organised as follows. Chapter 1 is a general introduction to the context, the problem and the methodologies adopted. Chapter 2 will analyse the theoretical background and literature underlying this thesis work. Chapter 3 will discuss the methodology that has been adopted in this thesis work to achieve the established objectives. Chapter 4 will present and comment on the practical results obtained with this methodology. It will also deal with interpretability aspects of this approach. Finally, Chapter 5 will draw the final and summary conclusions of this thesis work. After this, there will be a section dedicated to the Bibliography.

Chapter 2

Theoretical background

In this section the fundamental concept behind this thesis will be introduced, which is the *Natural Language Processing (NLP)*. Its main characteristics and some types of NLP tasks will be presented, focusing in particular on those that have been addressed in this work.

Natural Language Processing is a field of computer science and, more specifically, of Artificial Intelligence, which aims to program computers so that they understand natural language data (text and spoken words) in a similar way to humans, by processing large amounts of this kind of data.
NLP is the union of computational linguistics, a way to model human language with rules, with statistical, machine learning, and deep learning models. These techniques allow computers to process natural language data and to *understand* its meaning and the sentiment behind them.
In order to mention a few examples of the capability of NLP techniques, they can generate computer programs that can translate text [4, 5, 6], can answer to voice commands [7, 8], and can synthesise large volumes of text quickly [9, 10, 11]. NLP can be used also for voice-operated GPS systems, digital assistants, speech-to-text dictation software. It can also be adopted in company solutions that help speed up and simplify business operations and increase employee productivity [12].
Unfortunately, NLP is not an easy field, since human language is filled with ambiguities that make very difficult to make a program that determines in a correct way the true meaning of text or voice data. Also humans take years to learn some ambiguities of the language, but software and application must identify and understand them accurately from the start.
For this reason, several NLP tasks decompose human text and voice data in ways that help the software to grasp the meaning of what it's processing.
Some of the most common tasks of NLP are explained in the following:

- **Speech recognition**: is the task of transforming voice data into text [7], [13].
- Part of speech tagging: is the process of identify the part of speech of a word based on its usage and context.
- Word sense disambiguation: is the selection of the meaning of a word with multiple meanings in order to select the word that makes the most sense in that situation, in order to avoid ambiguity.
- Named entity recognition: identifies words or phrases as useful entities.
- Co-reference resolution: identifies if two words relate to the same entity.
- Sentiment Analysis: tries to retrieve subjective sentiments (such as attitudes, emotions, sarcasm, confusion, and so on) from the text, giving them a positive or negative or neutral connotation.
- Natural language generation: tries to put some structured information into human language.

In this thesis, in particular, it will be worked on and deepened three tasks of NLP: Text Mining, Sentiment Analysis and Topic Modelling.

2.1 Text Mining

Text Mining is the process of deriving high-quality and previously unknown information from text. More specifically, it is an Artificial Intelligence technique that uses Natural Language Processing to transform free and unstructured text from documents/databases, such as web pages, articles, emails, social media posts/comments etc., into structured and normalised data, in order to identify meaningful patterns and new insights, automatically extracting information from the different resources. High-quality information is typically obtained by devising patterns and trends by means such as statistical pattern learning.
In this way companies will be able to explore and discover hidden relationships within their unstructured data.
Common text mining challenges involve categorising text, clustering text, extracting concepts/entities, summarising documents and modelling relationships between entities.

Text is one of the most common data types in databases and documents. This kind of data can be broadly organised into three categories:

- *Structured data*: These data are organized into a tabular format with many rows and columns.

- *Unstructured data*: These data do not have a standardised format.
- *Semi-structured data*: These data are a mixture of structured and unstructured data. They have some structure, but do not have enough organization to satisfy the criteria of a structured database.

Since the majority of digital textual data resides in an unstructured format, text mining is a very valuable practice in organisations, since, by processing the data, making it structured and deriving information from it, it improves the decision-making process of the companies, leading to better business results.

2.2 Sentiment Analysis

Sentiment Analysis (SA) is a fields of Natural Language Processing, which is concerned with building programs for the exploration and extraction of subjective opinions or sentiments gathered from written sources on a certain topic.
The objectives of Sentiment Analysis may stop at exploratory analysis of users' opinions, but often the information and opinions extracted are used to the advantage of some business or public interest operation, such as some aspect of city management. The algorithms of SA study the text in depth and find the basic concepts that indicate the attitude towards a certain product or service. Thus, Sentiment Analysis is an occasion to study the psychology of the public, investigate the article from the point of view of interest and understanding the market.
For example, Sentiment Analysis is used for activities as market research, public relations, product reviews, reputation management, customer service, product feedback and so on [14, 15, 16, 17, 18, 19, 20].
From a more technical point of view, Sentiment Analysis is a classification algorithm aimed at finding an opinion and information of particular interest in the process. Taking a step back, the classification process aims to define a model of a collection of classes, called classifier, which is constructed from a set of labelled data, called training set. The classifier is subsequently employed to properly classify new data for which the class label is unknown. Sentiment Analysis, in particular, is a classification algorithm that aims to label a new review as belonging to the *positive*, *negative* or *neutral* class.
It is composed from the following tasks: (i) finding and extracting opinion data on a given platform; (ii) assessing polarity (positive or negative) of data; (iii) define the topic (what is being talked about); (iv) determine the owner of the opinion.

Moreover, this algorithm could be used at different levels:

- *Document level*: for the whole text;

- *Sentence level*: retrieves the sentiment of a phrase;
- *Sub-sentence level*: gets the sentiment of sub-expressions in a phrase.

Since the SA deals with opinions, they are subjective, so extracting them is often complicated. Indeed, opinions are diverse and some are more valuable than others. Opinions have therefore been divided into subcategories:

- Direct opinion: directly states something and express a legitimate argument.
- Comparative opinion: here two products are compared on the basis of certain parameters. It is both an opinion of the product and also serves as a competitive research.
- Explicit opinion: all is plainly stated.
- Implicit opinions: are implied but not distinctly declared.

2.2.1 Types of Sentiment Analysis

Sentiment Analysis is divided into several types:

- *Fine-grained Sentiment Analysis*: it means determining the polarity of opinion. This can be a simple binary differentiation of sentiment in positive and negative classes, or it can also be a multi class differentiation, becoming more specific (e.g. as in Amazon reviews with 5 levels) [21].
- *Emotion detection*: it is used to identify cues to certain emotional moods. Typically, there is a mixture of lexicons and machine learning algorithms that identify states and the motivations [22], [23].
- *Aspect-based Sentiment Analysis*: its objective is to detect an opinion on a particular piece of the product. It is generally used in product analytics to monitor how the product is felt and what its strengths and weaknesses are from the user's point of view [24], [25], [26].
- *Intent analysis*: its aim is to assess what type of intention and action is manifested in the text of a message. It is widely used in client support systems to optimise the job [27], [28].

2.2.2 Sentiment Analysis algorithms

After finding the dataset and pre-processing the text appropriately, the next step is to decide on the algorithm model to identify the sentiment underlying the text. The model chosen will depend on several factors, for example the amount of data

to process, the precision needed for a certain purpose, or whether one wants an interpretable model or not.
Speaking broadly, sentiment analysis algorithms can be classified into two categories: Rule-based or lexicon-based approach and Machine Learning approach.

2.2.3 Rule-based or lexicon-based approach

This approach is based on manually created rules for classifying data to identify sentiment. This method uses dictionaries of words with positive or negative values to denote their polarity and strength of feeling to compute a score. Further functionality can be provided by incorporating expressions. This kind of sentiment analysis algorithms can be tailored to the context, developing more intelligent rules.

Functioning: There are two lists of words. One contains only positive words, the other contains negative words. It counts the number of positive and negative words in the text. If the number of negative words is greater than the number of positive words, it will give a positive sentiment, or vice-versa. If the two numbers are equal, it will result in a neutral sentiment.

Disadvantages: The disadvantage of this approach is that it does not take into consideration how words are combined in a phrase and the context, it only looks at occurrences, it lacks in flexibility and precision.
It is quick to implement, but then comes at an additional cost in that it needs regular maintenance in order to achieve consistent and better results.
This approach can be used for non-specific purposes to determine the tone of texts, which can be useful for client support.
Furthermore, rule-based Sentiment Analysis is commonly used to lay the foundations for the subsequent implementation of machine learning approaches.

In the following, the rule-based Sentiment Analysis algorithms used in this work will be introduced and discussed more in detail.

Vader
VADER (Valence Aware Dictionary and sEntiment Reasoner) [29] is a lexicon and rule-based Sentiment Analysis instrument which works particularly well with sentiments expressed in social media.
This will be used as a comparison and baseline in this work, as in this thesis we will use an associative classifier which classifies by means of rules derived from data collected from reviews expressed on a platform.
Vader is a model that is sensible to both the polarity (positive/negative) and intensity (strength) of the feeling. It is provided in the *Natural Language Toolkit*

(NLTK) [30] package and can be applied directly to the text and also to unlabelled data.
VADER's Sentiment Analysis is based on a dictionary that maps lexical features of words to the intensities of feelings. These are called *sentiment scores* and the sentiment score of a complete text can be calculated by adding up the intensity of each word in the text.
In addition to assessing the sentiment of individual words, VADER is smart enough to understand the context of these words, such as the presence of a *not* that creates a negative affirmation. It also comprehends the significance of capitalisation and punctuation.
Vader's objectives are: (i) classify sentence polarity (i.e. whether the text expresses a positive, negative or neutral opinion), (ii) document-level scope (i.e. assess the overall opinion by aggregating all sentences in a document), (iii) coarse analysis (i.e. figure out if the review is negative or positive without trying to figure out the degree of positivity or negativity).

Let us take a closer look at how Vader's classification of sentence polarities works. Vader uses the library *SentimentIntensityAnalyzer* imported from NLTK in order to carry out the classification.
VADER's *SentimentIntensityAnalyzer()* takes in input a string and, by using the function *polarity_scores*, returns a dictionary of scores for four categories:

- *negative*: the score for this category is referred as *neg* and denotes the score for the negative category for that sentence, it ranges from 0 to 1.

- *neutral*: the score for this category is referred as *neu* and denotes the score for the neutral category for that sentence, it ranges from 0 to 1.

- *positive*: the score for this category is referred as *pos* and denotes the score for the positive category for that sentence, it ranges from 0 to 1.

- *compound*: this score is calculated by summing the valence scores of each word in the lexicon and then by normalizing to be between -1, that indicates the most negative, and +1, that indicates the most positive.
  This metric is very useful when it is required a unique measure of sentiment for a phrase.
  Moreover, it is used when thresholds are to be set to classify sentences as positive, neutral or negative.
  Common threshold levels are:

  - negative sentiment: compound ≤ -0.05
  - positive sentiment: compound ≥ 0.05

– neutral sentiment: (compound > -0.05) and (compound < 0.05)

The compound sentiment, in general, is the most commonly used.

Note that the pos, neu and neg scores are ratios for the proportions of text that fall into each category, i.e. they should all add up to be 1.
These three metrics are useful for investigating the context and the way in which sentiment is communicated in a sentence.
For example, some writing styles may reflect a penchant for rhetoric heavily skewed in positive and negative sentiments (at different points in the same text), while other styles may use a large amount of neutral text and ultimately both types of writing will deliver a similar overall compound sentiment. So, using only the compound metric, such differences can't be identified.

The application of the function *polarity_scores* produces results of the type shown in the following example of output:

VADER is smart, handsome, and funny.—— 'pos': 0.746, 'compound': 0.8316, 'neu': 0.254, 'neg': 0.0
VADER is smart, handsome, and funny!—— 'pos': 0.752, 'compound': 0.8439, 'neu': 0.248, 'neg': 0.0
VADER is VERY SMART, handsome, and FUNNY.—— 'pos': 0.754, 'compound': 0.9227, 'neu': 0.246, 'neg': 0.0
VADER is VERY SMART, handsome, and FUNNY!!! ——'pos': 0.767, 'compound': 0.9342, 'neu': 0.233, 'neg': 0.0
VADER is not smart, handsome, nor funny.—— 'pos': 0.0, 'compound': -0.7424, 'neu': 0.354, 'neg': 0.646
At least it isn't a horrible book.—— 'pos': 0.363, 'compound': 0.431, 'neu': 0.637, 'neg': 0.0

2.2.4 Machine Learning approach

These Sentiment Analysis models use Machine Learning to understand the latent concept of affirmations, instead of using defined rules. With this type of algorithms, the accuracy of the analysis generally improves. This approach involves mainly the use of supervised machine learning algorithms.
These algorithms are trained with a lot training data until they can accurately predict the sentiment of the text. At this point new data are given to the classifier, which predicts the sentiment of the new data generally as negative, neutral or positive.
Additionally, unsupervised machine learning algorithms are used to explore the

data.

Machine learning approach with traditional models

Traditional models are predominantly used to establish text polarity. They are simpler and faster than deep learning models, but are less accurate.
The traditional machine learning methods mainly used are *Naïve Bayes* [31], *Logistic Regression* [32] and *Support Vector Machines (SVM)* [33].
They are widely used for large-scale sentiment analysis because they are capable of scaling to varying numbers of data.

One of these approaches, the Support Vector Machine (SVM), will now be discussed in more detail, as it is one of the most powerful algorithms in traditional machine learning and thus also in the field of NLP. Furthermore, it will later be used in this work as a baseline and a term of comparison for the new models implemented in this thesis.
Subsequently, the L^3 classifier, the method mainly used in this thesis, will be discussed.

Support Vector Machines (SVM)

Support-vector machines are ones of the most robust and most important supervised learning models to analyse data.
Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other.
More in detail, support-vector machine constructs a hyperplane in a high - or infinite - dimensional space that is used to classify new data depending on which side of the hyperplane they fall. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since, in general, the larger the margin, the lower the generalization error of the classifier is. So, in plain words, SVM try to find the plane in such a way to maximize the width of the gap between the two categories.

For what concerns the linear SVM, we are given a training dataset of n points of the form: $(x_1, y_1), ..., (x_n, y_n)$ where the y_i are either 1 or -1 label and x_i is a real vector of data of size p.
In general, if our data can be perfectly separated using a hyperplane, then there will exist an infinite number of such hyperplanes. This is because a separating hyperplane can usually be rotated or shifted up or down, without coming into contact with any of the observations.
So, in this case, where the training data are linearly separable, we want to find the

hyperplane with maximum *margin* that separate the two classes of data, so that the distance between them is as large as possible. The *margin* is defined as the distance between the closer point of the training set and a given separating hyperplane.

By analysing the problem from a mathematical point of view, we can write the equation of a general hyperplane as: $< w, x > +b = 0$. Where x are the set of points in the space, w is the normal vector to the hyperplane and b is a costant.
After some reasoning and mathematical calculation, the final optimisation problem for the linear SVM with hard margin is reached. This is expressed with the following equation: $(w, b) = argmin_{(w,b)} \frac{1}{2}||w||^2$ s.t. $y_i(< w, x_i > +b) \geq 1$.
The final solution obtained is: $w = \sum_i \alpha_i y_i x_i$.
So the solution, that defines the hyperplane, is a weigthed linear combination of the points of the two classes that lie on the margins, because for other points $\alpha = 0$. Those vectors are called support vectors, because they are the only vectors that *support* and define the decision function.

It is important to point out that there are other formulations of the SVM algorithm. There are Soft-Margin SVM and the Kernel SVM.
The Soft-Margin SVM is applied when the data are not perfectly linearly separable. It is a relaxation of the hard-SVM rule, discussed above, that can be applied if the training set is not perfectly linearly separable.
A natural relaxation is to allow the constraint to be violated for some of the examples in the training set, by keeping margin as wide as possible at the same time, so that other points can still be classified correctly.
This can be modeled by introducing non-negative slack variables $\xi_i > 0$, and replacing each constraint $y_i(< w, x_i > +b) \geq 1$ by the constraint $y_i(< w, x_i > +b) \geq 1 - \xi_i$. It's obtained an optimization problem and a solution similar to the hard-SVM rule.

The Kernel-SVM is applied when the data are not linearly separable at all.
A solution is to create nonlinear classifiers by applying the *Kernel trick*.
Specifically, in order to separate the data not linearly separable with the standard SVM, and so with a linear hyperplane, the data are mapped into a new feature space F with higher dimension and here the samples result linearly separable.
The *Kernel trick* avoids this explicit mapping and computation to an high dimensional space, that could be computationally expensive. Indeed, applying a kernel, is like implement a mapping function Φ that move the inner product in a higher dimension space F. The kernel can be defined as $K(x, x') =< \Phi(x), \Phi(x') >$.
The kernel can be easily applied and is very useful since can be shown that both the Decision Rule and Optimization Problem depend only on the dot product of the samples. Then the resulting algorithm is similar to the standard one, except

that every dot product is replaced by this nonlinear kernel function.
This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. So, the classifier is a hyperplane in the transformed feature space, while it may be nonlinear in the original input space.

L^3 classifier

L^3 is an associative classifier, that is an algorithm that produces association rules, that are a valuable tool for classification tasks. In this context, the rule consequent is a class label, and the classifier is a set of association rules.
Since association rules represent the correlation among values of different attributes simultaneously, in general, associative classifiers yield better accuracy than other rule-based classifiers.
The generation of an associative classifier consists of two steps. First, classification rules are extracted from the training data. Then, pruning techniques are applied to select a small subset of high-quality rules.
Usually, a large rule set is mined to allow the generation of accurate classifiers. However, rule mining may yield a huge number of classification rules. Rule extraction becomes difficult (or at least time consuming), and it becomes hard to optimally exploit the generated rules. Hence, pruning techniques, are exploited to reduce the complexity of the extraction task.
Excessive rule pruning affects associative classifiers, by discarding also useful knowledge together with low-quality rules, so pruning should be limited to a minimum.
To address both an excessive rule set size and overpruning, L^3 algorithm is a new associative classifier that relies on a lazy pruning approach coupled with a compact representation of the rule set. L^3 stands for *Live and Let Live*, i.e. pruning only takes place when strictly necessary. The lazy pruning technique performs a reduced amount of pruning by eliminating only rules that only misclassify training data.
During classification, L^3 adopts a two-step approach in which high-quality rules (rules used in the classification of training data) are considered first, and *unchecked* rules (rules unused during the training phase) are used next to classify unlabeled data, when them cannot be classified by means of the first type of rules.
Moreover, the compact form proposed in the work of L^3 represents a rule set without information loss and allows the regeneration of the complete rule set. This form allows reaching very low support thresholds and mining large rule sets. Rule compression provides a space-effective representation of these large rule sets in the classifier.

In the L^3 associative classifiers, the data of interest for classification are represented as a single relation D, whose schema is given by k distinct attributes and

a class attribute C.
A training data instance is a tuple in D where the class label is known, whereas a test data instance is a tuple in D where the class label is unknown. Each tuple in D is a set of pairs *(attribute, integer value)* plus a class label, each pair is called *item*. Each tuple is a transaction in D, characterized by a transaction ID (tid).
Association rules are rules in the form $X \rightarrow Y$, where both X and Y are sets of items. A rule $X \rightarrow Y$ is said to match a tuple $d \in D$ when $X \subseteq d$. The quality of an association rule is usually measured by two indexes: its support, given by the number of tuples matching X $\cup$ Y over the number of tuples in D, and its confidence, given by the number of tuples matching X $\cup$ Y over the number of tuples matching X. When association rules are used for classification purposes, Y is a class label. In associative classification, the chi-square (χ^2) test has been proposed, to select rules whose antecedent and consequent are positively correlated.
An associative classifier is built by selecting a set of *high-quality* association rules for its model. The usual approach is the extraction of a large number of classification rules from which high-quality rules are selected during a post-processing phase called rule pruning. The approach proposed in the paper of L^3, is the lazy pruning, in which only a much reduced amount of pruning is performed.
Given a classification rule set R, by means of lazy pruning, rules in R are partitioned in three subsets:

1. **Used rules.** These are rules that have already correctly classified at least one training data instance.

2. **Spare rules.** These are rules that have not been used during the training phase to cover any training data, but may become useful to classify (test) data not classified by used rules.

3. **Harmful rules.** These are rules that only wrongly classify training data. These rules are pruned, since they do not contribute in increasing classification accuracy.

The used rule set provides a high-level model of the training data including a set of rules abstracting the most relevant characteristics of each class. Since used rules represent the most frequent characteristics of each class, some slight variations, possibly less frequent, might not be covered by this model. The spare rule set allows the classifier to cover a wider range of (test) data. This rule set may include a large number of rules, and appropriate compression techniques are proposed to store this information. The used and spare rule sets are combined to create a two-level classifier.
Classification occurs in two steps. Initially, used rules are considered. If no rule classifies the new data, then spare rules are considered for classification. The availability of a large number of rules is important to allow the selection of *high-quality*

rules for the generation of an accurate classifier. However, is necessary to limit the complexity of the extraction task and the size of the resulting rule set. In this work, is proposed a compact form to represent rule sets with similar characteristics.

Compact rule set representation
This work proposes a compact form that provides a concise and complete representation of a set of classification rules.
The compression performed is lossless and the original rule set can always be regenerated from the compact form. This important property was not guaranteed by the compact forms proposed previously.
The proposed compact form is based on the concept of a *compact rule*. A compact rule encodes a set of rules labelled by the same class label, and characterised by the same value of measures of interest such as support, confidence and χ^2.

Correlated items are items that are contained in the same set of transactions.
A set of correlated items is represented by the concept of *macroitems*. Replacing each set of correlated items in the data set D with the corresponding macroitem, we obtain an alternative representation of D, the macrodata set associated to D, which will be referred to as $\bar{D}$ in the following. D and $\bar{D}$ include the same set of transactions.

A *closed itemset* is the maximal set of items common to a set of transactions. Closed itemsets have been described by means of the Galois closure operator γ. The closure of an itemset X was defined by means of the operator γ as the itemset $\gamma(X)$. X and $\gamma(X)$ are in the same transactions and therefore have the same support. X is a closed itemset if $\gamma(X) = X$.
Among the itemset represented in a closed itemset X, the shortest are called generators of X, and are denoted as G.
A closed itemset can have several generators. Macrogenerator and macroclosed itemsets are obtained by considering macroitems in $\bar{D}$ instead of items in D. These concepts are exploited to define the structure of a compact rule.

Definition 1 (Compact rule). Let $\bar{S}$ be an arbitrary macroclosed itemset, $\bar{G}$ be an arbitrary macrogenerator itemset of $\bar{S}$, and c be an arbitrary class label in $\bar{D}$. $\bar{r} :< \bar{G}; \bar{S} >\rightarrow c$ is a compact rule.
A compact rule in $\bar{D}$ represents a set of classification rules in D.

Definition 2. Let $r : X \rightarrow c_i$ be an arbitrary rule in D and $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c_j$ be an arbitrary compact rule in $\bar{D}$. r is encoded in $\bar{r}$ if 1) $c_i = c_j$, 2)$\forall \alpha \in \bar{G}, \exists i \in X | i \in \alpha$, and 3) $\forall i \in X, \exists \alpha \in \bar{S} | i \in \alpha$.

The longest rule encoded in a compact rule $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ is the rule whose antecedent contains all the items in $\bar{S}$.
This rule will be denoted as r_l. Instead, the shortest rules represented in $\bar{r}$ are the rules whose antecedent contains one single item for each macroitem in $\bar{G}$.

Theorem 1. Let $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$ and $r : X \rightarrow c$ be an arbitrary rule in D, encoded in $\bar{r}$. Then, X, $\bar{G}$, and $\bar{S}$ have the same tidset.

To build the classification model, rules are selected by means of the support, confidence, and χ^2 quality indexes.
The next property states that rules encoded in the same compact rule have the same values for all these quality indexes. Furthermore, they cover the same training data.
Property 2. Let $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$ and $r_i : X \rightarrow c$ and $r_j : Y \rightarrow c$ be two arbitrary rules in D encoded in $\bar{r}$. Then, 1) $sup(X) = sup(Y)$, 2) $sup(r_i) = sup(r_j)$, and 3) r_i and r_j cover the same data in D.
Hence, two rules that satisfy Property 2 have the same contingency table and, thus, the same χ^2 value.

Property 3. Let $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$ and $r_i : X \rightarrow c$ and $r_j : Y \rightarrow c$ be two arbitrary rules in D encoded in $\bar{r}$. Then, $\gamma(X) = \gamma(Y)$.

Property 4. Let $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$ and $r : X \rightarrow c$ be an arbitrary rule in D, encoded in $\bar{r}$. Then, 1) X is a closed itemset iff it includes all items in $\bar{S}$, and 2) X is a generator itemset iff it includes only one item for each macroitem in $\bar{G}$.

Theorem 5. Let R be the rule set in D that satisfies the given support, confidence, and χ^2 constraints and $\bar{R}$ be the compact rule set in $\bar{D}$ that satisfies the same constraints. $\bar{R}$ is a complete representation of R.
So, with this representation, compact rules contain all information needed to generate all rules encoded in them. Hence, it is always possible to regenerate R from $\bar{R}$.

The L^3_{Gen} mining algorithm

The L^3_{Gen} algorithm mines the compact representation $\bar{R}$ of rule set R. It performs a recursive projection of macrodata set $\bar{D}$ with respect to its macroitems. We denote as *projection* sequence the set $\bar{G}$ of macroitems used for the recursive projection of $\bar{D}$ and *projected* data set the resulting data set $\bar{D}_{\bar{G}}$. This data set includes the

transactions in $\bar{D}$ that contain the macroitems in $\bar{G}$.
Macroitems in $\bar{D}$ are first sorted on support and lexicographically. Next, they are considered in increasing order.
At each recursion level, L^3_{Gen} analyzes the projected data set $\bar{D}_{\bar{G}}$ and performs five main operations, briefly described in the following.

Macroitem merging. The projection process preserves item correlation. Furthermore, new correlations between items may arise in $\bar{D}_{\bar{G}}$, because of the reduced set of transactions included in it. Hence, correlations are recomputed after projection, and newly correlated items are merged together into new macroitems. By exploiting this property, the solution set is further compressed and the complexity of the extraction process is reduced.

Set $\bar{S}$ updating. Set $\bar{S}$ contains macroitems in $\bar{G}$. It also contains macroitems included in all transactions in $\bar{D}_{\bar{G}}$, which are removed from $\bar{D}_{\bar{G}}$. When projecting $\bar{D}_{\bar{G}}$ with respect to one of these macroitems, the new projection sequence $\bar{G}' \supset \bar{G}$ has the same tidset as $\bar{G}$.

Compact rule mining. L^3_{Gen} analyzes $\bar{D}_{\bar{G}}$ to mine the compact rules that satisfy the support, confidence, and χ^2 constraints. Each rule has the pair $< \bar{G}, \bar{S} >$ as antecedent and is labeled by a class label in $\bar{D}_{\bar{G}}$.

Support-based pruning. A macroitem is pruned when, for all classes in $\bar{D}_{\bar{G}}$, its frequency is below the support threshold.

Data set projection. At each recursion step, data set $\bar{D}_{\bar{G}}$ is further projected with respect to its macroitems. Each macroitem, once used for projecting $\bar{D}_{\bar{G}}$, is removed from $\bar{D}_{\bar{G}}$.

L^3_{Gen} mines a complete and lossless representation of rule set R.
The compact rule set mined by L^3_{Gen} encodes all rules in R with generator itemsets as antecedents. To encode all other rules in R, L^3_{Gen} collects in set $\bar{S}$ the macroitems that belong to all transactions in $\bar{D}_{\bar{G}}$. By the properties of closed itemsets, $\bar{S}$ is the macroclosed itemset for set $\bar{G}$.
To avoid generating the same projection sequence multiple times, L^3_{Gen} discards macroitems used to project $\bar{D}_{\bar{G}}$. As a consequence, the classification rules encoded by the compact rules in $\bar{R}$ are disjoint sets.
To implement the algorithm, they adopted a prefix-based tree structure and an item-based projection algorithm similar to FP-growth.

Classification model generation
The generation of the classification model is performed in two steps:

1. **Classification rule extraction.** Since the support is not a good quality index for rule selection, in this work they exploit their compact representation to perform rule extraction with very low support thresholds. This yields a wide selection of rules from which high-quality rules are selected in the following step.

2. **Classification rule pruning.** Several techniques are applied to the extracted rule set to select the most appropriate rules to include in the classification model. A *rule-rich* classification model provides useful knowledge for the classification of a wide range of test data. Hence, in L^3, only a reduced amount of pruning is performed. In particular, a confidence threshold and a χ^2 test are enforced. Then, *lazy* pruning is performed.

Before applying lazy pruning, rules below a 50 percent confidence threshold are pruned. Also, rules with a χ^2 value lower than a fixed threshold (3.84 at 95 percent significance level) are pruned. Since by Property 2 all rules encoded in a compact rule have the same confidence and χ^2 values, both pruning techniques can be directly applied on compact rules.

Lazy pruning
The main idea behind lazy pruning is to perform a *mild* pruning of the extracted rule set. Only rules that exclusively misclassify training data, denoted as *harmful rules*, are discarded from the model. The remaining rules are partitioned in two sets: 1) a small set of rules, denoted as *used rules*, which correctly classify at least one training data, and 2) a large set of rules, denoted as *spare rules*, which have not been used to perform classification during the training phase.
During classification, used rules are considered first. If no used rule classifies a given unlabeled data, then spare rules are exploited for classification. Hence, spare rules provide additional knowledge, which allows the classification of more unlabeled data and thus improves the quality of the classifier. The used rule set, which is characterized by a small number of high-quality rules, provides a general class model.
Lazy pruning is based on the database coverage technique, a pruning technique that selects the minimal number of rules necessary to cover each training data. As the first step, a sort order based on confidence, support, and rule length is imposed on rule set R.
Definition 3 (Rule rank). Let r_i and r_j be two arbitrary rules in R. Then, r_i precedes r_j if

1. $conf(r_i) > conf(r_j)$, or

2. $conf(r_i) = conf(r_j)$, and $sup(r_i) > sup(r_j)$, or

3. $conf(r_i) = conf(r_j)$, and $sup(r_i) = sup(r_j)$, and $len(r_i) > len(r_j)$ or

4. $conf(r_i) = conf(r_j)$, and $sup(r_i) = sup(r_j)$, $len(r_i) = len(r_j)$, and $lex(r_i) > lex(r_j)$,

where $len(r)$ denotes the number of items in r, and $lex(r)$ denotes the position of r in the lexicographic order on items.
The longest one (specialistic rule) is preferred to the shortest one (general rule). A specialistic rule covers the same training data but includes more constraints. Hence, it is considered more accurate. Since general rules are not pruned, they will be considered on any data not matched by specialistic rules.
Lazy pruning considers rules in set R one by one in sort order. For each rule r, all training data in D covered by it are selected. Rule r is included in the used rule set if it correctly classifies at least one data. All data in D matched by r are not considered further. Instead, rule r is pruned if it only performs wrong classifications.
The process continues until either all training data has been covered or all rules in R have been analyzed. All rules in R that have not been considered in this phase are included in the spare rule set. The used and spare rule sets are exploited to generate a two-level classifier. In particular, used rules are assigned to Level I, and spare rules are included in Level II. Harmful rules are pruned and do not contribute to the classifier.
In L^3, set R is represented by its corresponding compact rule set $\bar{R}$. In the following, it is described how lazy pruning is efficiently performed directly on compact rules, without decompressing them.
Lazy Pruning and Compact Rules. It is first introduced the concept of compact rule matching. Matching occurs when any rule represented in the compact rule matches the considered data instance.
Lazy pruning is performed without extracting all rules included in a compact rule. Instead, only the longest rule r_l is considered. When r_l 1) correctly classifies at least one training data or 2) does not classify any training data, by considering only r_l, it is possible to assign all rules in $\bar{r}$ to the appropriate level. Hence, this property significantly increases the efficiency of the pruning step. The next theorem proves it.
Theorem 6. Let $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$. Let r_i denote an arbitrary rule in $\bar{r}$ and r_l be the longest rule in $\bar{r}$. Then,

1. If r_l correctly classifies at least one training data (that is, r_l is a used rule), then $\forall r_i \in \bar{r}, r_i \neq r_l, r_i$ does not match any training data (that is, r_i is a spare rule).

2. If r_l does not match any training data (that is, r_l is a spare rule), then $\forall r_i \in \bar{r}, r_i \neq r_l, r_i$ does not match any training data (that is, r_i is a spare rule).

Given a compact rule $\bar{r}$, by Theorem 6.1, when r_l is a used rule, it is included in Level I, and the compact rule encoding it is included in Level II. On the other hand, by Theorem 6.2, when r_l is a spare rule, the compact rule encoding it is included in Level II.
Consider now the case in which r_l only wrongly classifies training data, that is, it is a harmful rule. Lazy pruning should discard r_l without affecting the training set. By Property 2, all rules in $\bar{r}$ cover the same training data. Hence, all other rules in $\bar{r}$ are also potentially harmful, that is, candidates to be pruned. For the sake of efficiency, when r_l is a harmful rule, we consider all rules in $\bar{r}$ to be not accurate, and we prune $\bar{r}$. The pseudocode in the figure 2.1 shows how lazy pruning is implemented in L^3. The compact rule set $\bar{R}$ and the training data set D are input parameters. Since only the longest rule in each compact rule is considered, set $\bar{R}$ is sorted by considering these rules (line 1). The longest rule r_l is extracted from the first compact rule in the sort order (lines 3-4). For each training data d matched by r_l (lines 5- 11), its correct or wrong classification is recorded. If r_l covers correctly at least one data, it is included in Level I, and its compact rule $\bar{r}$ is included in Level II (Theorem 6.1). Then, all data (both correctly and wrongly) covered by r_l are removed from D (lines 12-16). Hence, each data is removed once covered by a rule that correctly classifies at least one data. If rule r_l does not match any data, by Theorem 6.2 $\bar{r}$ is included in Level II (lines 17-18). If r_l only misclassifies data, $\bar{r}$ is pruned. The outer loop (lines 2-20) is repeated until either $\bar{R}$ or D is empty. All the compact rules in $\bar{r}$ that were not analyzed because D became empty are included in Level II (line 21).

Classification
To classify an unlabeled data d, Level I is first considered. The first rule that matches d labels it. If no rule in Level I matches d, then Level II is considered. The first matching rule in Level II classifies d. If no rule matches d, d is not labeled. Since Level II includes compact rules, the search of the first matching rule is slightly more complex in Level II than in Level I.
Given an arbitrary compact rule $\bar{r}$, all the rules encoded in it cover the same training data but not necessarily the same test data. Hence, different from the training phase, during the classification phase, the longest rule in $\bar{r}$ matching d may be different from the longest rule r_l in $\bar{r}$. Thus, all rules encoded in $\bar{r}$ should be considered and not only rule r_l. However, it is possible to extract from $\bar{r}$ the longest rule matching d without extracting all rules.
Definition 5. Let d be an arbitrary data instance and $\bar{r} :< \bar{G}, \bar{S} >\rightarrow c$ be an arbitrary compact rule in $\bar{D}$ matching d. Let $r : X \rightarrow c$ be an arbitrary rule in $\bar{r}$. r is the longest rule matching d if $\forall \alpha \in \bar{S}$ and $\forall i \in \alpha$, when $i \in d$, then $i \in X$.

```
Procedure generateClassifier(R̄,D)
1.  sort R̄ by longest encoded rules r_l;
2.  while (R̄ is not empty and D is not empty)
3.  {  r̄ = first compact rule in R̄;
4.     r_l = longest rule in r̄;
5.     for each d in D
6.     {  if r_l matches d
7.        {  r_l.classifiedData = r_l.classifiedData ∪ d;
8.           if d.class==r_l.class
9.              r_l.right++;
10.          else
11.             r_l.wrong++; }  }
12.    if r_l.right>0
13.    {  LevelI = LevelI ∪ r_l;
14.       LevelII = LevelII ∪ r̄;
15.       delete r_l.classifiedData from D;
16.    }
17.    else if (r_l.wrong==0 and r_l.right==0)
18.          LevelII = LevelII ∪ r̄;
19.    delete r̄ from R̄;
20. }
21. LevelII = LevelII ∪ R̄;
```

Figure 2.1: L^3 classifier generation.

Compact rules in Level II are sorted by considering the longest encoded rule r_l and according to the sort order in Definition 3. Then, the first compact rule $\bar{r}$ matching d is selected. In particular, the longest rule $r_i \in \bar{r}$ matching d is considered. Two cases are given:

1. $r_i = r_l$. So, all compact rules $\bar{r}'$ that follow $\bar{r}$ encode rules r_j with a lower rank than r_i, Thus, the data instance d is labeled by r_i.

2. $r_i \neq r_l$. In this case, there might be a compact rule $\bar{r}'$ following $\bar{r}$ in the sort order, which has the same support and confidence and encodes a rule r_j longer than r_i and matching d. Thus, all the compact rules with the same confidence and support as $\bar{r}$ should be considered. The longest matching rules encoded in them are extracted. The rule with the highest rank labels d.

Compact rule matching can be performed efficiently. At most one scan of all the items included in a rule is performed to check which items are included in d. During the scan, matching items are stored. Hence, at the end of the matching process, the longest matching rule r_i encoded in the compact rule $\bar{r}$ is immediately available.

Machine Learning approach with Deep Learning models

In general deep learning models give more accurate results than traditional models. For NLP tasks, such the Sentiment Analysis, deep learning is able to learn patterns across lot of layers from unstructured and unlabelled or labelled data. However, deep learning tends to need more data and is less scalable than traditional algorithms.
Some of the most popular deep learning models are neural networks such as CNN (Convoluted Neural Network), RNN (Recurrent Neural Network) and DNN (Deep Neural Network).

In general, the most common models of machine learning used for Sentiment Analysis classification algorithms are Naïve Bayes, SVM and Deep Learning, in particular RNNs.

Finally, there are also hybrid models for Sentiment Analysis [34]. They are a combination of rule-based and automatic models and in some circumstances manage to achieve the advantages of both approaches described above. If they are well developed, they offer the power of machine learning combined with rule customisation. Hybrid models are the newest and most efficient models in Sentiment Analysis and are nowadays a very popular approach in this field.

2.3 Topic Modeling

Topic modeling is an unsupervised machine learning and Natural Language Processing technique that is able to scan a set of documents, discover patterns of words and phrases, and group together similar words that best represent a set of documents. In other words, a topic model is a type of statistical model for discovering abstract *topics* occurring in a set of documents, hidden semantic structures in a text and what the balance of topics is in each document.
The *topics* generated by topic modelling methods are sets of similar words.
Since a document deals with some particular topics, it is expected that specific and correlated words will occur in the document more frequently, indeed a document usually deals with multiple topics in different proportions.
These models are also called *probabilistic* topic models, as they use statistical

algorithms to discover the latent semantic structures of a text.
Topic models can help organise and can provide an intuition for comprehending huge collections of unstructured text data, that a human being could not process. By detecting patterns such as word frequency and distance between words, a topic model groups together similar feedback, and underlines the words that appear most often. With this knowledge, it is possible to rapidly infer what each collection of texts is about.
Topic models, in general, have been used as a text-mining tool and also in order to detect informative structures in data like genetic information, images and networks. So they also have applications in areas such as bioinformatics and computer vision [35, 36, 37].
As topic modeling needs no training, it is a fast and easy way to start analysing data. However, there is no guarantee that the outcome will be accurate.

General Functioning
In general, Topic Modeling is a process that separates a corpus of documents into two parts:

1. A list of the topics addressed by the data in the corpus;

2. Different sets of documents of the corpus clustered according to the topics covered.

Each document involves a statistical distribution of topics that can be obtained by combining all the distributions for all the topics. Topic modeling methods try to understand which topics are present in each document and how strong and probable this presence is.

In the next section, will be explored how one of the most popular methods of topic modeling works, namely *Latent Dirichlet Allocation (LDA)* [38], which will be used later in this thesis.

2.3.1 Latent Dirichlet Allocation (LDA)

The aim of topic modeling, and so of LDA, is to automatically discover the topics from a set of documents. LDA is based on the concept that the documents can be observed, while the topic composition and distribution are hidden. Thus, the goal is reached by using these observed documents to infer the hidden parameters.
LDA is a statistical and unsupervised model that is based on the intuition that documents have multiple topics.
An important premise to make is that in LDA documents are treated as Bags of Words, i.e. the order of documents and the order of words in documents do not

matter.
It is characterised by its generative process, i.e. the basic assumption from which the LDA process starts is that the way a document was generated was to choose a set of topics and then for each topic to choose a set of words. This is an idealistic random process from which the model assumes that documents are generated. LDA, in order to find the topics, reverse engineers this process.

Generative process
In order to describe at a high level the generative process underlying LDA, the following is assumed.
It is supposed that there are a certain number k of topics. Each document in the collection is represented as a mixture of topics and each topic β_k is represented as a probability distribution over the words in the vocabulary.
The generative process of LDA is described by the following pseudocode:

for each document d **do**
  Draw topic distribution $\theta \sim$ Dirichlet(α)
  for each word at position n **do**
    Sample topic $z_n \sim$ Categorical(θ)
    Sample word $w_n \sim$ Categorical(β_{z_n})
  end for
end for

Figure 2.2: LDA process pseudocode

The topic distributions for the d-th document are denoted as θ_d, where $\theta_{d,k}$ is the topic distribution for topic k in document d, that are the probabilities that a document d belongs to a certain topic k: documents are seen as a distribution over the latent topics. This distribution is described by the Dirichlet distribution.
Then, for each word, the topic assignments for the d-th document are denoted as z_d, where $z_{d,n}$ is the topic assignment for the n-th word in document d.
Finally, the observed words for document d are denoted as w_d, where $w_{d,n}$ is the n-th word in document d, which is an element from the fixed vocabulary and is extracted from a distribution that depends on the probability distribution of the words in the topic sampled.
This document generating process has to be repeated for all the documents belonging to the corpus.

Probabilistic Graphical model representation
Another way to describe LDA is with a graphic representation and in particular with a probabilistic graphical model, that is a technique that provides a graphical language for describing families of probability distributions.
In particular, with the *plate notation*, which is often used to represent probabilistic graphical models, dependencies between the variables can be caught in a concise manner.

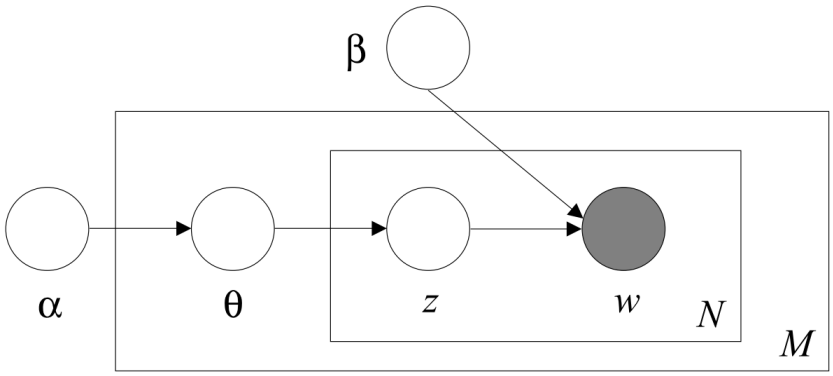

Figure 2.3: Graphical model representation of LDA

Each node in the figure 2.3 is a random variable and is labelled in relation to its role in the generative process. Rectangles denote the *plate* notation, which is used to encode the replication of variables.
There are three different levels in the schema, that are: the corpus, the documents and the terms.
The definitions of the variables are the following:

- M is the number of documents and the M plate denotes the collection of documents;
- N is the number of words in a given document and the N plate denotes the collection of words in the documents;
- α is the per-document topic distributions;
- β is the per-topic word distribution;
- θ is the topic distribution for document m;
- z is the topic for the n-th word in document m;
- w is the specific word.

It can be seen that the variable w is in grey. This is because it is the only observable variable in the system, while the others are hidden and are denoted in white.
Moreover, due to the presence of multiple levels, it can be noticed that z and w are sampled for each n: this means that the topic is sampled for each word in the document, and thus the same document can be characterized by several topics. Instead, the variables θ are sampled one time per document and the variables α and β are sampled once for the entire generation process.
It is precisely these last parameters that can be worked on by tuning them, in order to modify the model and improve it, if necessary.
In particular, α is a matrix where each row is a document and each column is a topic. A value in a cell represents the probability that document i contains topic j. A symmetrical distribution would mean that each topic is uniformly distributed in the document, vice versa for an asymmetrical distribution. This concept can be used when one has a priori knowledge of how the topics are distributed, and adjusts the α matrix accordingly, in order to achieve better performance.
Instead, β is a matrix where on the rows there are topics and on the columns there are words. A value in a cell represents the probability that topic i contains word j. Usually, the distribution of words throughout a topic is uniform. Therefore this parameter can be used to favour certain topics and words if necessary.

Chapter 3

Methodology

3.1 Dataset used

Regarding the dataset used in this work, a long search and exploration was carried out. The objective was to find information on sustainable mobility and in particular to find information derived from the content freely produced by users. This was done because it is believed that this content, which is derived from direct and voluntary feedback from users, is more reflective of reality than surveys, official statistics and traditional methods and provides an excellent and realistic source of information that transmits users' feelings about the mobility domain. This information could be used by public administrations at a later stage to make improvements taking into account both the sustainability of transport and the opinion of users.
The dataset created in the work [2] by Serna, Soroa and Agerri was chosen in this thesis to meet these aims. In the work [2], they exploited user-generated content to obtain a high-precision Sentiment Analysis model in the transport domain, in order to contribute to the analysis of transport sustainability. To develop such a model, they semi-automatically generated an annotated corpus of opinions on transport, which was then used to develop a classifier obtained through the fine-tuning of a pre-trained deep learning model based on the Transformer architecture, i.e. XLM-RoBERTa [39]. They also provide a transport classification score with respect to the sustainability of the transport types mentioned in that dataset.
More specifically, for what concerns the dataset, there is a scarcity of valid corpora for the field of transports. In order to overcome this lack, they provide a *Gold-standard Corpus (GSC)* dataset for this field, by hand annotating 2000 reviews from a *User Generated Content (UGC)* corpus of 117K comments written in English retrieved from TripAdvisor. These reviews cover a range of transportation modes, with varying degrees of sustainability, in a period between 2007 and 2020 and are

about travelling in Croatia.
The authors carried out a search through the different sections, capturing, downloading and selecting reviews and their original evaluations from 1 to 5. These sections cover public and private transport, such as electric trams, funiculars, ferries and taxi boats, shuttle, cycling, walking, etc.
They then did a cleaning and pre-processing only of the sentences evaluated with 1 and 5 stars to produce the Gold Standard Corpuss, as these extreme cases may be more reliable. They subsequently created the GSC with a manually labelled polarity and then manually checked each sentence to detect the true positive and negative polarity. They found many cases where the original TripAdvisor star rating did not correspond to reality. The sentences whose sentiment was not clear or which do not make sense after splitting the reviews into sentences were discarded. Additionally, those phrases considered to be incorrectly classified are discarded until a balanced dataset of 1000 positive and 1000 negative sentences was obtained.
In the paper, the authors also classify the types of transport in the dataset according to sustainability principles. Based on some indicators, they have created a transport classification in which the means of transport are sorted in order of sustainability and they listed the transportation modes found in UGC based on this classification.

3.2 Pre-processing phase

Before different machine learning techniques are applied, it is needed to start with text pre-processing, that is the process of cleaning and transforming text data into a usable format compatible with the algorithm that will be applied in the future. The pre-processing is an essential phase in the field of NLP as it substantially changes the results, since the quality of the data affects the capability of the model to learn, and it also serves to explore the data and understand some critical issues.

Specifically in this work, after importing the dataset from the github repository derived from the paper [2], it was read and collected in two lists: the first representing the original text of the reviews, called X; the second containing the labels, called Y.
At this point, some statistics on words and their frequency have been calculated and they will be presented in the next chapter.
From now on, a text pre-processing phase was implemented, in order to clean the data for the classification algorithms that will be used later.
This phase is carried out with the help of *Natural Language Toolkit (NLTK)* [30], that is a suite of libraries and programs for building Python programs to work with

natural language data.

The pre-processing steps involve the following techniques:

- *Remove accent and errors*:
  The first pre-processing operation applied is aimed at removing accents from words and errors in the text.
  In general, for many cases, it might be better to remove the wrong characters and accent. The wrong characters could be formatting errors or typoes. Therefore, there is often no single, predefined way to do it, but it depends on the specific situation and dataset.
  In this work, the accent from the characters is removed by first performing a normalisation through the *unicodedata* library, then an *ascii* encoding and finally a *utf-8* decoding.
  It has also been noted that there are characters in the reviews that generate formatting errors, producing meaningless characters in the dataset, that cannot be interpreted in a correct way.
  I tried to identify most of these characters manually and through visual evaluation. It was noticed that most of the time they were apostrophes (" ’"). So an attempt was made to locate them and replace them correctly with apostrophes and thus obtain correct sentences and words.

- *Remove special characters*:
  It is often better to remove special characters, such as brackets, punctuation symbols, numbers, etc., from the text as they do not provide information content.
  It was done by using regular expressions, with the help of the library called *re*.

- *Remove spaces*:
  often text data contain extra spaces that have to be removed. In this work, it was done thanks to a function that removes extra spaces between words in the text by using regular expressions, with the help of the library called *re*.

- *Expand contraction*:
  often, some words have an abbreviated form, especially in the English language. For example, the form *don't* stands for *do not*, *it's* stands for *it is*. Therefore, in order to carry out a more accurate and correct analysis, it is advisable to expand the contractions in the text data.
  In this work, it was done using the library *contraction*.

- *Tokenization*:
  is the procedure of breaking down the text provided in natural language into the smallest unit of a sentence, called *token*. Punctuation marks, single

words and numbers can be considered tokens. Thus, the body of text will be converted into a list of tokens.
It is used the module of NLTK called *tokenize*, and in particular the *word_tokenize*() method to split sentences into tokens of words, since punctuation, number and other special characters have already been removed.
There exists also the *sent_tokenize*() method to split a document or paragraph into sentences.

- *To Lower Case*:
  upper and lower case characters are interpreted differently by the machine and, consequently, so are words containing these characters. Therefore, it is preferable to render the words in the same case, since in this way the model interprets the words in the correct way.
  In this work, all letters were converted to lower case by uniforming the capitalisation.

- *Stop-words removal*:
  Stop-words are words that do not have a strong meaning and usually refer to the most common words in a language, e.g. *and, the, a, is, they*, etc. These are important when are used to communicate in human language, but for a program they are not very useful, since they do not carry specific and distinctive information to determine the polarity of a text, also because they are too common and evenly distributed in sentences, and are considered as noise in the text.
  Moreover, removing stopwords will save a huge amount of computing power and therefore time, by not giving to a model texts with so many useless and misleading words.
  In order to deal with stopwords can be used different methods, for example NLTK or *Scikit-learn* [40], which provide lists of stopwords. It is important to notice that there is no universal list of stopwords because it can change depending on the problem.
  Specifically, stopwords imported from the *Scikit-learn* library have been used. However, the word *not* has been removed from this list as it is considered to be vehicle of information regarding the determination of sentiment, since combined with other words in the sentence, it could overturn the final sentiment of that sentence.

- *Parts of Speech (PoS) tagging*:
  Part of speech (PoS) are the properties of words that define their function and usage in a sentence. Some examples of PoS tags are: *noun, verb, adjective*, and so on. The PoS is determined by the relationships of words in the phrase. This method tags each token in the list of tokenized words, with a corresponding

Parts of Speech identifier and thus generates tuples. After knowing the role of each word in the phrase, it's easier for the algorithm to understand what the sentence is talking about.
It is executed via the *pos_tag* method imported from the NLTK library and for each word, a tuple is produced as output, containing the word and its PoS tag.

- *Stemming*:
  is the process of reducing words to their roots. A word stem is not necessary the same root as a dictionary-based morphological root, it is just an identical or smaller form of the word.
  When stemming words, sometimes the result is incorrect, as stemming works on a rule-based basis, cutting suffixes in words following a given rule. This can lead to errors: overstemming and understemming.
  Overstemming happens when words are truncated excessively. In these cases, the sense of the word may be distorted. Understemming occurs when two different words are derived from the same root.
  The NLTK library provides several stemmers such as *PorterStemmer* [41], *SnowballStemmer* [42] and *LancasterStemmer*[43].
  In this work it was used the *PorterStemmer* from the NLTK library.

- *Lemmatization*:
  the last step of the pre-processing was the lemmatization. The purpose of lemmatization is to reduce inflectional forms to a common base form, i.e. finding the form of the correlated word in the dictionary. It is different from stemming in that lemmatization does not simply cut words, but uses lexical knowledge to obtain proper and existing base forms. It involves processes that take more time to compute than stemming.
  The *WordNetLemmatizer* from the NLTK library was used for this step. In addition, for the correct functioning of this lemmatizer, it is necessary to have the PoS tags and to define a dictionary of tags.

The ones that have been presented, are some of the most common text data pre-processing techniques. These can be adapted and modified according to the context and the specific case. Of course there are also other techniques that can, or should be, used depending on the situation and on the data.

Finally, empty or duplicated reviews after applying these transformations were removed, thus producing the final clean and preprocessed dataset, so that subsequent machine learning algorithms would work well on it and in a correct way. This final dataset consists of 1933 instances.

3.3 L^3 and L^3 wrapper usage

To carry out the sentiment analysis on this dataset, the Associative Classifier L^3 was chosen. In particular, this technique was chosen for the purpose of having an interpretable and readable model, instead of having a non-interpretable model like SVM and neural networks.
More specifically, thanks to L^3 many association rules are produced, with a higher or lower confidence. In this way we can have a complete and visual representation of the model, of the rules and therefore of the words that compose them. So, it is easier to identify at a glance, or with statistical and graphical methods, the concepts related to mobility that have been the vehicle of a positive or negative sentiment, together with their confidence.
In more details, L^3 was used through the L^3 wrapper [44] , that is a wrapper in which L^3 is run for ease of use and that has auxiliary functions.
L^3, through this wrapper, is intended for categorical/discrete attributes, and therefore requires an *object* data type as input. Consequently, the data of this work, which will be converted into numbers to obtain some meaningful representations that will be explained later, will be considered an *object*.
So, as mentioned earlier, the L^3 wrapper, in addition to providing accuracy, also provides some extra functionality. One of the most important, is the possibility of having a *human readable* representation of the rules. In particular, for this function, we need to pass the names of the columns.
The printed rules will have a series of fields containing the following information: <rule_id> <antecedent> <class label> <support count> <confidence(%)> <rule length>

3.4 Baselines

At this point, some baselines for the models devised in this project will be presented, in order to have a comparison and highlight the strengths of these methods and, if present, some critical points.
It is important to emphasise that, in most of the following baseline methods, and also in most of the new methods designed in this work, the sentences of the dataset are transformed into a *Bag of Words (BoW)* [45] representation as a last step before being fed to the various classifiers.

Bag of Words representation:
One of the issue with text processing is that it is unorganized, and machine learning algorithms prefer well-defined fixed-length inputs and outputs. Furthermore, they

cannot work directly with the unprocessed text, but the text must be converted into numbers and in particular, vectors of numbers, which reflect characteristics of the text. This process is called feature extraction.
A well-known and straightforward method of extracting features from text to feed into machine learning algorithms is the so called Bag of Words (BoW). It is a simplifying representation of data that is frequently employed in the field of NLP, indicating the multiplicity of words within a document, disregarding grammar and words order. Thus, the frequency of each word within a document is employed as a feature to train a classifier.
In practice, the BoW transforms arbitrary text into vectors of fixed length, equal to the number of distinct words in the documents (or a subset of words under consideration) and counts how many times each word appears. So finally the dataset will be a matrix of dimensions: the number of documents, on the rows, and the number of distinct words, on the columns. This process is often also called vectorization.
The idea is that if the content of the document is similar, so the document is similar and that, from the content alone, it is possible to get some insights about the meaning of the document.
This process involves two things: (i) a vocabulary of known words; (ii) a metric of the occurrence of known words.
Therefore, this model is called *bag* of words because the only thing that matters is whether the known words are present or not, without giving any information about the order and structure.

From an implementation point of view, the BoW representation is obtained either manually or by using the *CountVectorizer* function provided by the *Scikit-learn* library. This function, will also be used to perform other more general actions, such as calculating the frequency of words and automatically selecting the most frequent ones.

3.4.1 Baseline 1: N most frequent words

The first baseline to be analysed is based on the use of L^3, but with a different way of deriving features from the data than the methods created in this work.
In particular, this baseline method computes the frequency of words in the text, that is calculated using the *CountVectorizer* function provided by the *Scikit-learn* library. So, the n most frequent words, and therefore the most significant ones, are identified by this function and by setting the corresponding parameter $max\_features$, which determines the number of top words that will be found. At this point, the various sentences are filtered, keeping only those words in each sentence that are present in the set of the most frequent and significant words. Consequently, some sentences

will remain completely empty. It was chosen to eliminate these instances as it was assumed that they did not provide any information content. As a result, as the number of the most frequent words selected changes, the total number of instances in the dataset will also change.
The number n of most frequent words is chosen mainly on the basis of the corresponding number of features used in the methods designed in this work. But, in any case, a tuning of this parameter is carried out in order to have more results to make comparisons and evaluations. A number of words which is too large or too small is not typically chosen. Indeed, if the number of words is too large, the model will be less compact and too specific; moreover, with a high number of features, the training of L^3 classifier will be more difficult and much slower and therefore unmanageable, as will be emphasised several times in the next chapter on practical results. Vice versa, if the number of top words is too small, the model will lose its meaning because too many instances would be eliminated, as they will remain empty after filtering, and so too much data will be wasted. Therefore, intermediate values are chosen in order to find a balance, by making a selection of the most significant data, but without losing too much information.
After making this selection of data and features, the sentences are transformed into a Bag of Words representation to be fed to the classifier. So, the features that will be fed to the classifier will be the top n words and the value of these features, for each sentence, will indicate the presence or absence of those words in those sentences and will therefore have a value greater than or equal to 0.

3.4.2 SVM baselines

The second group of baselines analysed use the SVM algorithm for the classification instead of L^3 classifier, and is composed from four baselines.
The first of these baselines, is the so called *SVM baseline*. Here the data are modified in the same way as in the *Baseline 1*, in order to be compatible with the SVM classifier. For this reason, the word frequencies are calculated with the $CountVecotorizer$ method and then the n most frequent words are selected. On the basis of these, a Bag of Words is created. The resulting matrix will be fed to the classifier, which in this case is SVM.
Furthermore, in order to have complete and valid baselines, other three baselines have been created with the SVM classifier.
Specifically, SVM was used as a classifier with the data obtained after applying the feature extraction of *Methods 1, 2B* and *3*.
These methods will be explained in detail in the following of this chapter. We anticipate that they are data transformation and feature extraction methods, devised in this work, that are based on the use of Topic Modeling techniques.

3.4.3 Vader

The last tested baseline is based on Vader. This method, as explained in the previous chapter, is rule-based, and has been used as a competitive method to the one in this thesis, as L^3 also classify by making use of rules.
Unlike L^3 and SVM, in this method the data is not transformed and pre-processed to be passed to the classifier, but a string containing the original text is directly fed to Vader for each sentence.
Specifically, to calculate Vader's accuracy with our data, has been used the value of the *compound* attribute, produced by Vader using the *polarity_scores()* method. This attribute, as explained previously, is calculated by summing the valence scores of each word in the lexicon and then normalised to be between -1 and +1. This is the metric used when one wants a single overall measure of sentiment in a sentence. Specifically, a threshold is set and then the value of the compound metric is compared to the threshold value and a *positive, negative* or *neutral* sentiment is assigned to the sentence, according to the result of this comparison.
In this work, in which sentences are labeled with only *positive* or *negative* feelings and there is no *neutral* category, sentences are only assigned a *positive* or *negative* sentiment and the value 0 was used as the threshold value.
More specifically:

- if the value of *compound* is greater than 0, a *positive* label is attributed to the sentence;
- if the *compound* value is less than or equal to 0, a *negative* label is attributed to the sentence.

Once this was done, the accuracy, i.e. the number of sentences of the test set correctly predicted by Vader's algorithm with this threshold divided by the number of total sentences of the test set, was calculated.

3.5 The application of Topic Modelling

Topic Modeling, besides being used in the proper tasks, whose goal is to extract topics and classify documents as belonging to one topic or another, can also be used for other purposes.
In particular, in this work, it is not used as an end in itself, but it is used as an intermediate procedure, with the aim of extracting features from the data that give additional and complementary information, which helps to produce better final results, once fed to the L^3 classifier.
In this work, specifically, the *Gensim* [46] library was used for topic modeling. Gensim is a free open-source Python library for representing documents as semantic

vectors, in the most efficient way possible. It is intended to process digital text using unsupervised machine learning algorithms.
The Gensim algorithm that was used in this work is Latent Dirichlet Allocation (LDA). This, like other algorithms in this library, is an unsupervised algorithm that automatically discovers the semantic structure of documents by examining statistical patterns of co-occurrence in training data.
Specifically, the topics are found with Gensim's *LDAmulticore* function [47]. This function allows the Latent Dirichlet Allocation technique to be used online in Python. It allows all CPU cores to be used, to parallelize and accelerate model training. The parallelization uses multiprocessing.
The training algorithm:

- is streaming: training data can arrive sequentially, no random access is needed.
- runs in constant memory with respect to the amount of documents: the dimension of the training corpus does not influence the memory footprint, it can handle corpora bigger than RAM.

Importantly, this tool enables both the evaluation of the LDA model from a training dataset and the inference of topic distribution on fresh non-viewed documents. The model can also be refreshed with new data for online training.
This function can receive many parameters as input. The main ones, and the ones that have been used in this work, are:

- corpus: it is a set of document vectors or sparse matrix with shapes: *num_documents, num_terms*. In this project it is a bag of words obtained with the *doc2bow()* function called on the *id2word* dictionary.
- id2word: This is a dictionary that maps word IDs to words. It is used to measure the dimension of the vocabulary, for debugging and to print topics.
- num_topics: it is the number of hidden topics to be extracted from the training data. In our case the parameter k is passed as num_topic, which changes from time to time.
- passes: this parameter represents the number of passes through the corpus during training. In all models in this work it will be set to 10.
- iterations: it represents the maximum number of iterations through the dataset when deducing the distribution of the topic of a corpus. It will be set to 100 in all models.
- random_state: it is a *randomState* object. It is used for reproducibility, so that random operations always have the same output for the same input

parameters. Notice that the outputs may still differ due to undeterminism in the programming of the work process operating system. In all models it will be set to 10.

3.5.1 Parameter search

The most difficult part of the topic modeling task is to work out how many (and which) topics are present in the corpus of documents.
One of the parameters to pass to the *LDAmulticore* function, indeed, is precisely this number of topics.
One way to find the optimal value of this parameter, and also of other secondary parameters of the *LDAmulticore* function, is to search among the parameter space as exhaustively as possible, and finally choose the parameter (or combination of parameters) that maximises some metric or criterion.
To do this, in this work have been tried basically three approaches:

1. Extrinsic a posteriori evaluation after classification with L^3: the performances are evaluated directly on the result after applying L^3, and parameter values are chosen with the aim of maximising performance (accuracy).

2. Maximisation of a metric: in this work the *Coherence CV* [48] metric was used, which is a very common metric for topic modelling in the literature.
   The topic Coherence is defined as a measure that assesses a single topic by evaluating the degree of semantic similarity between words with a high score in the topic. This measure helps to distinguish between topics that are semantically interpretable and topics that are statistical inference artefacts.
   Therefore, it was tried to find a number of topics that would maximise this metric.

3. A visual analysis of the topics through visualisation tools: specifically *pyLDAvis* [49] was used in this work, which is an interactive topic visualisation tool.
   Through this visualisation and by trying out different *num_topic* values, it should be possible to work out which *num_topics* parameter value splits the documents better, by simply looking at the graph produced by this tool.

However, from a practical point of view, it was noted that the Coherence metric was not indicative for the specific task of this work, as it had low values even with a number of topics that turned out to be favourable for high accuracy after the application of the classifier. Moreover, it gave higher results for a number of topics that was definitely high and not compatible with the computational capabilities of L^3, as it will be underlined below.
Also the visual evaluation using the *pyLDAvis* tool did not help in choosing the best number of topics, as the topics were not easily distinguishable and assessable

by the human eye.
So, finally, the number of topics was chosen by means of an extrinsic a posteriori evaluation after classification with L^3.

3.5.2 Generation of methods

Through the use of topic modeling, the five methods used in this work were created. In particular, topic modeling was used to transform the data into features which were then passed to the model. In fact, it was thought that giving the extra information from the topic could be useful for classification purposes, adding information content at a higher level compared to features derived from the pure sentences alone.
In particular, 5 different methods have been designed through the help of topic modeling, that will be deepened in the following section.

3.6 The 5 methods with features obtained through topic modelling

In the following, the 5 methods created in this project will be presented and explained. In particular, these methods aim to transform the data finding features and so creating data compatible with the use of the L^3 classifier.
In general, for all methods, the parameters were tested and chosen by extrinsic evaluation, i.e. based on the accuracy after applying L^3 to the data obtained with the different methods.
The topics are found with Gensim's *LDAmulticore* function, as discussed above.

3.6.1 Method 1

In this method the n most important words in the k chosen topics are identified. So the two parameters to be evaluated are n and k: the parameter n will indicate the number of words chosen within a topic and the parameter k indicates the number of topics chosen.
The topics are found through the function *LDAmulticore* to which a certain parameter k, indicating the number of topics, is passed, and the other parameters are the ones cited above. The tuning of this parameter was done through the evaluation of the resulting accuracy on the test set, once the L^3 classifier was applied.
The most important words are found through the function *LDAmodel.show_topic()*, which finds the n most important words for the i-th topic. Also for the number n of top words, a tuning was performed according to the resulting accuracy once L^3

was applied, varying n in order to maximise it.
The most important words for each topic were then added to the list of top terms and, lastly, the duplicates were removed.
At this point, scrolling through the dataset again, each sentence is transformed into the corresponding sentence that contains only the words that are in the set of top words selected in the previous step.
Moreover, since the sentences are filtered by removing the words not in the set of top terms, it may happen that some sentences remain completely empty. These instances are identified and removed because they are deemed not to carry informative content for Sentiment Analysis.
Finally, the *CountVectorizer* function is used to obtain a matrix with the same number of columns, creating a Bag Of Words.
So, L^3 will be fed by a matrix that has as columns, and therefore as features, all the words in the set of selected words and on the rows the different documents. So in each cell there will be the number of times a given word appears in a certain document.

In the following methods, instances are mapped directly with the topics corresponding to that instance or to words of that instance, leaving out the actual text content. This is done using three different techniques.

3.6.2 Method 2A

The topic model is found with the function *LDAmulticore*, with the parameters cited before.
A number k of topics is used and each document is mapped to the corresponding probability distribution that the document belongs to each topic, obtained using the Gensim's function *get_document_topics*, that is applied to the LDA model. This function obtains distribution of the topics for the given document and returns a list of (int, float) that contains this distribution for the whole document. Each item in the list is a couple of ID of a topic and the probability that the document belongs to it.
Subsequently, a matrix will be created from this information, where on the columns there will be the topics found and on the rows there will be the various documents and each cell will contain the probability that each document belongs to a certain topic. This matrix is passed directly to L^3.
The parameter to evaluate is k, which is chosen according to the result of L^3 after applying this method.

3.6.3 Method 2B

Also in this method, the model is found with the function *LDAmulticore*, with the same parameters as previous models.
Here is used the Gensim *get_term_topics* function and is applied to the model obtained in this way. This function get the most relevant topics to the given word. More specifically, for each word, it returns the list of topics assigned to the word represented as pairs of their ID and their assigned probability, sorted by relevance. In particular in this work is chosen only one topic for each term (the one with the highest probability) and the data are mapped to the corresponding topics. So each row is transformed into a vector with a topic for each word in the sentence.
Then, to obtain a matrix with the same number of columns, the data are transformed manually in a Bag of Words representation, producing a matrix with the documents on the rows and the IDs of the topics on the columns. So, each cell represents the presence of that topic in that document.
The parameters to be evaluated are: k, which indicates the number of topics, as in the previous cases, and *unique*, which indicates whether the topics within a row are to be repeated or not.
Also these parameters and their values, are evaluated after the application of L^3 classifier, by looking the accuracy obtained on the test set and by trying to maximize it.

3.6.4 Method 2C

This method is a slight variation of the previous method.
Here again, the model is found with the function *LDAmulticore*, with the same parameters as previous models and the Gensim *get_term_topics* function is used, which returns the probability of belonging to one of the various topics for each term.
This time, several topics are chosen for each term, i.e. those with the highest probabilities until a total probability of 0.5 is reached (among the topics chosen for each term) and than each sentence is mapped to the corresponding topics, i.e. each row is transformed into a vector with one or more topics for each word in the sentence.
Then, to obtain a matrix with the same number of columns, the data are transformed manually in a Bag of Words representation, producing a matrix with the documents on the rows and the IDs of the topics on the columns. So, each cell represents the presence of that topic in that document.
The parameters to be evaluated are: k, which indicates the number of topics as in the previous cases and *unique*, which indicates whether the topics within a row are to be repeated or not.

3.6.5 Method 3

In this method, a concatenation of *Methods 1* and *2B* together is carried out.
That is, the n most important words for each of the k topics are selected and each row is mapped accordingly by selecting only the terms included in the chosen set of words, that form the features, and furthermore, a certain number of features k is added to each row, which represent the information of the presence of that topic or not in that sentence.
Consequently each row will be the concatenation of two pieces of information at two different levels, i.e. both the most relevant words and the topics present in a sentence. Indeed, it was thought that combining two pieces of information of different types could be beneficial because the additional content in each line is particularly significant and not redundant.
As far as the rows are concerned, are selected only those that are not discarded due to the application of *Method 1*, since according to the number of words and topics, some rows, once filtered, are empty and are discarded because they are not considered to have a useful information content.
Intuitively, to get a meaningful result, one would have to get a lot of features, as this method is the the concatenation of two methods, and it might not be feasible or make sense as the information would not be compact enough. But, it will be shown later, that this intuition is not necessarily true.
Here again the parameters are evaluated on the basis of accuracy after applying L^3 classifier and have been chosen with a focus on maximising it. They are the following: k, which indicates the number of topics for both *Method 1* and *Method 2*, and *unique*, which indicates whether the topics within a row are to be repeated or not.
Also here is used the *Gensim* library and its functionalities to find topics and top terms. In particular: the topics are found through the function *LDAmulticore* to which the parameter k is passed; the most important words are found through the function *LDAmodel.show_topic()*, which finds the n most important words for the i-th topic; finally, the *get_term_topics* function is used, which returns the probability of belonging to one of the various topics for each term.
At the end, the features obtained are concatenated in a matrix that was done by making a Bag of Words, in which the rows contain the documents and the columns contain the most important words for the different topics that have been selected and the IDs of these k topics. The cells contain the frequency of those words and topics for that document.

Chapter 4

Practical results

In this section will be discussed the practical results of the methods and theoretical implementations presented in the previous chapter. The results will be commented and explained, and sometimes accompanied by graphs and plots to further illustrate their information content.
The various methods were implemented in Python language and executed on a machine provided by the Politecnico of Turin, as they often required too much computational power to run locally or on free online Jupyter notebook services, e.g. Google Colaboratory [50]. The Google Colaboratory notebook was used instead to print the graphs and to compute the preliminary statistics.

4.1 Preliminary statistics on frequency

Before applying the various methods, preliminary analyses on frequency were carried out in order to explore the dataset and understand certain properties.

Words distribution in documents
The first analysis made concerns the length of the sentences and therefore the counting of the number of words per document. The results in this case are represented with a graph containing a histogram, as it is an intuitive representation.

As we can see from Figure 4.1, the sentences are in general small: most documents contain between 10 and 25 words.
In particular, the most frequently occurring number of words per document is 10 with 102 occurrences, followed by 9, 13 and 16 with 94 occurrences, and by 14 with 93 occurrences.
This could be a problem for some classification and topic modeling algorithms [51], aggravated by the fact that, with the additional pre-processing steps, the sentences

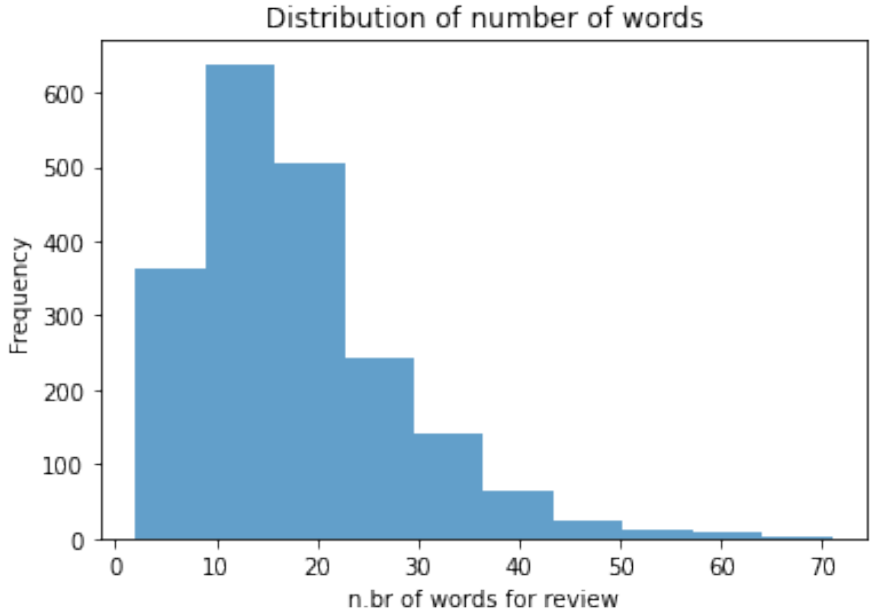

Figure 4.1: Frequencies of the number of words for review in the dataset

could be further reduced, until they contain a number of words close to 0 and therefore the algorithms may not have enough information for classification.

Frequency of distinct words

The second frequency analysis was carried out after applying pre-processing. The aim of this analysis was to see the frequency of individual words within the corpus, once preprocessed

In particular, the first n most frequent and therefore most important and significant words are printed (after having removed the stopwords).

To do this, the *CountVectorizer* function is used, in order to crate a Bag of Words. Subsequently, the graph is printed thanks to the *FreqDistVisualizer* function from *yellowbrick* [52], to which the features obtained from the *CountVectorizer* are passed, i.e. the words and their frequencies; the number n of the most frequent words to be printed is also passed as parameter to the *FreqDistVisualizer* function.

The *FreqDistVisualizer* is a method of visualising the frequency of tokens in datasets by means of the frequency distribution, which tells us how the total number of word tokens in the text is distributed among the vocabulary elements. It does not perform any normalisation or vectorialisation, and assumes that it receives text that has already been vectorized.

More in detail, after instantiating a *FreqDistVisualizer* object, a *fit* method is called

on it, which calculates the frequency distribution. The visualiser then draws a bar graph of the first n most frequent terms in the dataset, which is shown in Figure 4.2, where the 30 most frequent words are represented.

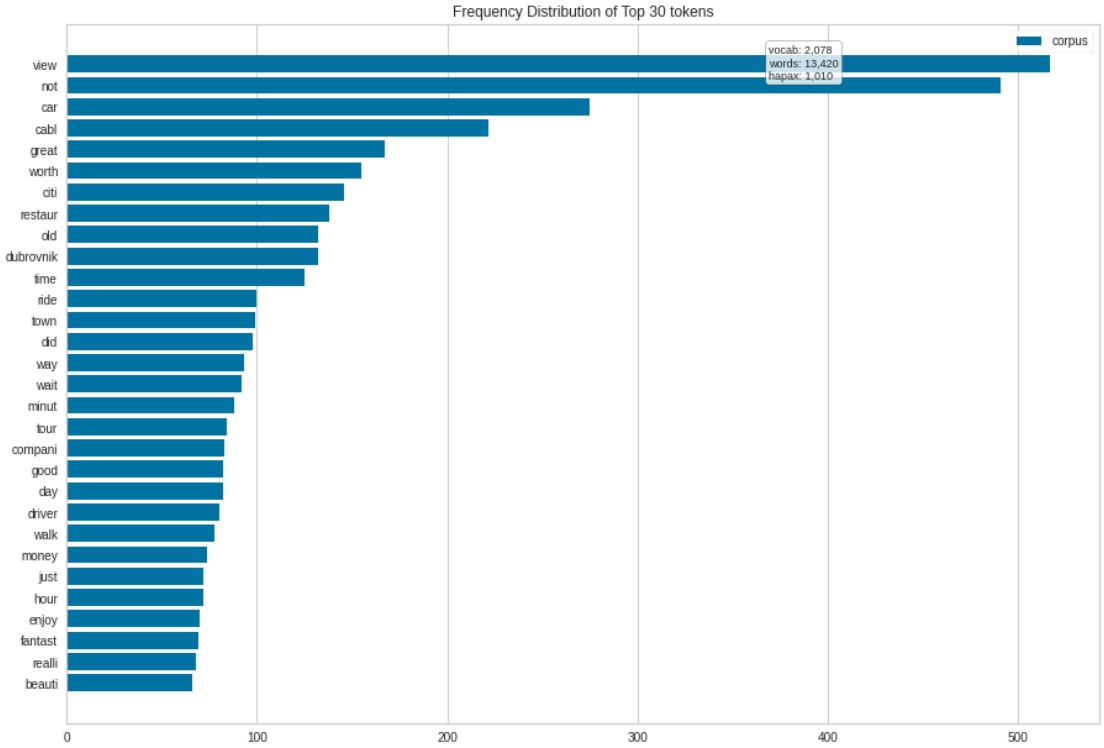

Figure 4.2: Frequencies distribution of the top 30 tokens in the dataset after pre-processing

The same kind of plot was also obtained using a vectorization with the *Term Frequency-Inverse Document Frequency (TF-IDF)* [53], [54]. This measure is a numerical statistic that aims to capture how relevant a word is to a document in a collection. The TF-IDF value grows in relation to the number of times a word occurs in the document and is weighted by the number of documents in the collection that include the word, which aids in regulating the fact that certain words occur more frequently in general. TF-IDF is one of the most common term weighting patterns nowadays.
It is applied with the function *TfidfVectorizer*, that convert a collection of raw documents to a matrix, which has as its values the result of the TF-IDF operation instead of the frequency of words in documents, as a simple Bag of Words.
The result of this operation is reported in Figure 4.3.

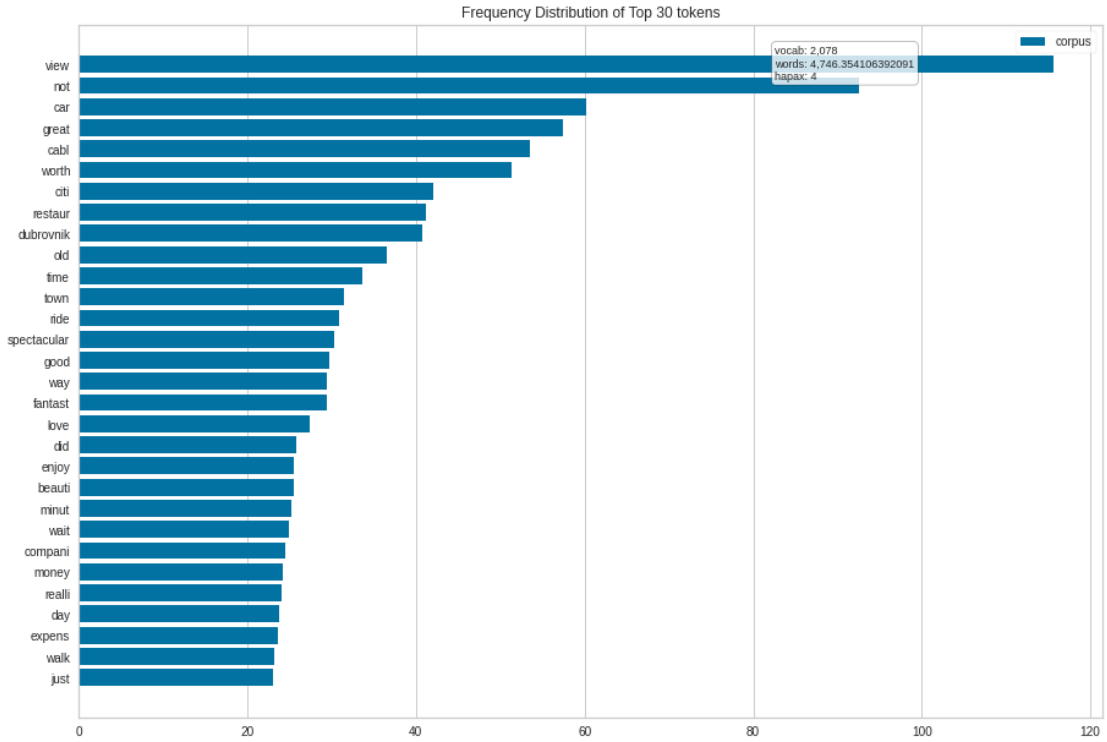

Figure 4.3: Frequencies distribution of the top 30 tokens in the dataset after pre-processing with a TF-IDF vectorization

As we can see from the plots, the distribution of word frequencies, when *CountVecorizer* or *TfidfVectorizer* are used, is very similar; the value of the frequencies changes a little, in particular the use of *TfidfVectorizer* allows a clearer distinction to be made.
In both plots, it can be seen that the most frequent word is *view*, which could indicate that it is an important factor for travellers. Among the most common words there are also *car*, which may indicate that many travellers use cars, and *city*, as can be expected since we are talking about traveling in a city.

4.2 Baseline results

Before analysing the results of the baselines and methods, it is necessary to make a few general considerations that apply to many of the methods that will be analysed.

Computational Time
It has been noted that the L^3 classifier has a training and classification speed which decreases exponentially as the number of features of the data fed to the classifier increases.
More specifically, it is quite fast (order of seconds/minutes) up to 20 features. But from 20 features onwards, it slows down substantially, until it takes hours when 24/25 features are used.
For this reason, when using L^3, the analysis has been limited to data with a maximum of 24 features for most cases, since with 25 features the time taken is not sustainable and the process often does not end.
On the other hand, the classification of SVM baselines is very fast, even with a large number of features, and there are not the problems that occur with L^3.
So, all the algorithms discussed in the following sections, both baseline and non-baseline (except Vader) have been tested on a machine provided by the Politecnico of Turin, as mentioned before. Indeed, since L^3 is very heavy, often, on online services as Google Colaboratory, the training fails to finish as the number of features increases, due to RAM or disk space problems, as well as being slower than on the machine of Politecnico.

Another important consideration, which applies to both baselines and methods, is that an attempt was made to explore the parameters search space as much as possible. However, doing an exhaustive search in this space is almost impossible and would have required too much time and resources, because it is a very huge space and is often even larger as it has a high number of parameter combinations. Therefore, in the following, the results for a sustainable number of attempts and parameter combinations will be reported, in order to explore the search space as exhaustively as possible and to give an idea of the trend of results, choosing the combinations and attempts that were deemed most significant.
Furthermore, not all the results obtained and the numerous attempts made will be reported in the descriptions and tables below, in order not to make the discussion too heavy and confusing.
The accuracies that will be reported in the results are obtained after applying the classifiers to the test set.

Finally, it should be recalled that the number of starting data for all methods is 1933 instances, obtained after applying the pre-processing techniques.

4.2.1 Baseline 1: N most frequent words

The first baseline that is analysed is the one called *N most frequent words*, here only the most frequent words in the corpus are taken into consideration and the sentences are filtered accordingly. Once the data have been transformed into the Bag of Words representation, they are fed to the L^3 classifier.
The parameter for which the tuning is to be carried out in this case is n, that concerns the number of words. The value of this parameter will represent the number of features that will be fed to the classifier.
The accuracy, precision and recall are collected for each value of this parameter.
The reference value is taken as $n = 17$ features, as it is the same as the reference value used in some methods created in this work and represents a significant number of features: i.e. they are not too few, causing the loss of too much information (and also data when the documents are filtered); nor are they too many, creating problems of classification speed, compactness and interpretability of the result.

With $n = 17$, and thus 17 features, the accuracy of this method is about 0.816 and the remaining sentences are 1379 (i.e. 71.3%), as 554 lines are discarded due to the operation of this method.
With different numbers of features, sometimes the results are higher. For example, the highest results are obtained with 22 and 23 features and have an accuracy of about 0.864 and 0.861 respectively, and 1507 (i.e. 78%) and 1520 (i.e. 78.6%) remaining data, respectively.
When tested beyond 24 features, the method is too slow and does not produce results at all.
The absolute highest accuracy is about 0.870 and is achieved with only 2 features. But this value was not considered significant because too much information and especially too much data is lost and wasted. In fact, at the end of the process there are only 861 (i.e. 44.5%) instances.
This baseline has worse results than the corresponding method created in this work with the aid of topic modeling, i.e. *Method 1*, as will be shown and discussed subsequently.
More details on the results can be found in Figure 4.4, where the exact accuracy, the precision, for positive and negative instances, and the recall, for positive and negative instances, are presented.

4.2.2 N most frequent words + SVM

The second baseline that is analysed is *N most frequent words + SVM*. Again only the most frequent words in the corpus are considered and the sentences are filtered accordingly. Once the data has been transformed into the Bag of Words

N° features	% remainig data	Accuracy	Precision	Recall
12	67,05	0,8061	P: 0,83 N: 0,77	P: 0,86 N: 0,73
17	71,34	0,8158	P: 0,81 N: 0,82	P: 0,89 N: 0,72
20	75,12	0,8417	P: 0,86 N: 0,82	P: 0,87 N: 0,81
22	77,96	**0,8635**	P: 0,87 N: 0,86	P: 0,89 N: 0,84
23	78,63	0,8606	P: 0,87 N: 0,85	P: 0,88 N: 0,84
24	79,41	0,8600	P: 0,89 N: 0,83	P: 0,85 N: 0,88

Figure 4.4: Complete results of the Baseline 1

representation, this time, unlike the previous baseline, it is fed to SVM classifier. The parameter for which tuning must be carried out also in this case is n, that represents the number of the most important words selected. The value of this parameter will represent the number of features that will be fed to the classifier. Accuracy, precision and recall are collected for each value of this parameter.
The reference value again is $n = 17$ features for the same reason as before.
With $n = 17$, and thus 17 features, the accuracy of this method is about 0.829 and the remaining sentences are 1379 (71.3%), as 554 of rows are discarded due to the operation of this method.
With different numbers of features, the results are sometimes higher. For example, with 23 and with 10 features is achieved 0.857 and 0.859 of accuracy respectively, with 1520 (78.6%) and 1223 (63.3%) data remaining respectively. With 10 features, although the accuracy is quite high, it is felt that too much information is lost.
The absolute highest accuracies are obtained by increasing the number of features. For example with 27, 45 and 50 features are obtained about 0.878, 0.899 and 0.888 accuracy respectively. This is quite understandable, since, in this way, increase the information and data, but it is a less compact representation and, above all, they are numbers of features incompatible with the timing of L^3.
More details on the results can be found in Figure 4.5, where the exact accuracy as the number of features changes, the precision, for positive and negative instances, and the recall, for positive and negative instances, are presented.

N° features	% remainig data	Accuracy	Precision	Recall
10	63,27	0,8589	P: 0,89 N: 0,81	P: 0,88 N: 0,83
17	71,34	0,8289	P: 0,83 N: 0,82	P: 0,88 N: 0,76
20	75,12	0,8438	P: 0,86 N: 0,82	P: 0,86 N: 0,82
23	78,63	0,8566	P: 0,86 N: 0,85	P: 0,88 N: 0,83
27	81,07	**0,8784**	P: 0,92 N: 0,84	P: 0,85 N: 0,91
45	88,15	**0,8988**	P: 0,89 N: 0,91	P: 0,92 N: 0,87
50	89,60	**0,8881**	P: 0,88 N: 0,89	P: 0,91 N: 0,87

Figure 4.5: Complete results of the Baseline 1 + SVM

4.2.3 Method 1 + SVM

This baseline uses the *Method 1 + SVM* classifiers. So the n most important words in the chosen k topics are taken and the sentences are filtered accordingly. Once the data has been transformed into the Bag of Words representation, it is fed to the SVM classifier.
The parameters for which tuning must be carried out in this case are two: k and n, which are respectively the number of topics and the number of main words for each topic. Each combination of parameters will create a different number of features, which equals the number of unique words found for those topics. The Bag of Words with these resulting features will be fed to SVM.
Accuracy, precision and recall are collected for different combinations of these parameters.
In this method, the reference configuration is $n = 4, k = 4$, that produces 13 features and obtain an accuracy of about 0.882 and the remaining sentences are 1339 (69.3%), since a number of rows are discarded due to the operation of this method.
With different combinations of parameters, that produce higher numbers of features, other slightly better results are obtained. For example by varying both n and k, the configuration $n = 9, k = 5$ produces 33 features, and it is obtained an accuracy of 0.899 and 1592 (82.4%) data; by varying k and setting $n = 9, k = 6$ are produced 39 features and an accuracy of 0.901 and with the configuration $n = 10, k = 7$

are produced 46 features and is obtained an accuracy of 0.891. These higher results are quite understandable, as, increasing the number of features, increases the information, but they are less compact representations and, moreover, they are a number of features incompatible with the timing of L^3.
A good accuracy of about 0.879 is obtained even with 24 features with the configuration $n = 5, k = 6$, that produces 1471 (76.1%) instances.
Probably, a more comprehensive and dense search for parameters would yield better results, these are just a few examples of good values. But an exhaustive search is not possible, so we are looking for some examples that give an indication of the accuracy trend.
More details on the results can be found in Figure 4.6, where the exact accuracy, the precision, for positive and negative instances, and the recall, for positive and negative instances, are presented.

n	k	N° features	% remaining data	Accuracy	Precision	Recall
4	4	13	69,27	**0,8824**	P: 0,89 N: 0,87	P: 0,91 N: 0,84
4	5	17	72,74	0,8685	P: 0,89 N: 0,84	P: 0,89 N: 0,86
5	6	24	76,10	0,8786	P: 0,91 N: 0,84	P: 0,87 N: 0,89
6	6	30	80,70	0,8757	P: 0,91 N: 0,84	P: 0,85 N: 0,90
7	5	25	76,88	0,8737	P: 0,92 N: 0,83	P: 0.85 N: 0,91
9	5	33	82,36	**0,8992**	P: 0,91 N: 0,88	P: 0,90 N: 0,90
9	6	39	85,15	**0,9007**	P: 0,91 N: 0,89	P: 0,91 N: 0,90
10	7	46	85,77	**0,8905**	P: 0,92 N: 0,86	P: 0,88 N: 0,91

Figure 4.6: Complete results of Method 1 + SVM

4.2.4 Method 2B + SVM

This baseline uses *Method 2B + SVM*. This time all words are considered and are simply associated with a topic. The phrases are not filtered and the dataset will therefore always be fully exploited. The data is then transformed into the Bag of Words representation and fed to the SVM classifier.
The parameter for which the tuning must be carried out in this case are two: k, that represent the number of topics; *unique*, which takes binary values (*yes* or *not*), and indicates whether to consider repeated topics one or more times. The topics

will be the features and the value of the parameter k will represent the number of features that will be fed to the classifier.
Accuracy, precision and recall are collected as the value of this parameter changes. In general, the value of accuracy with this method is lower than with the previous methods.
The reference combination is $k = 17, unique = yes$.
With this combination (and therefore 17 features), the accuracy of this method is about 0.814. When the is set $unique = no$, the accuracy is about 0.817.
Analyzing the results with a different number of topics, and therefore of features, can be observed that, for example, with the configurations $k = 22, unique = yes$ and $k = 23, unique = yes$, you get approximately 0.817 and 0.825 accuracy respectively; when is set $unique = no$, the accuracy is about 0.843 and 0.837 respectively. With $k = 25, unique = yes$ and $k = 28, unique = yes$ are obtained lower results, respectively 0.782 and 0.809 of accuracy; when $unique = no$ the accuracy is about 0.801 and 0.809 respectively. Finally with $k = 33, unique = yes$ and $k = 33, unique = no$ the highest results are found, that are 0.853 and 0.848 of accuracy approximately.
It is important to point out that the SVM classifier is quick to train, unlike L^3 which is very slow with the *Method 2B*. Consequently, no other tests were carried out with a higher number of features because the baseline would not have been comparable to the actual method.
More details on the results can be found in Figure 4.7, where the exact accuracy as the number of features varies, the precision, for positive and negative instances, and the recall, for positive and negative instances, are reported.

4.2.5 Method 3 + SVM

This baseline uses *Method 3 + SVM*, which is the union of *Methods 1* and *2B*. So, only a subset of words (the most important ones) are considered and the phrases are filtered accordingly. The data is then transformed into the Bag of Words representation and sent to the SVM classifier. The features that will be provided to the classifier are the concatenation of selected words + the k topics.
The parameters for which the tuning must be carried out in this case are three: number of top words per topic, n; number of topics, k; *unique*, which takes binary values (*yes* or *not*), and indicates whether to consider repeated topics one or more times.
Accuracy, precision and recall are collected as the values of these parameters vary. The reference is the set of parameters formed by: $k = 4, n = 4, unique = yes$, which produces 17 features and 1339 (69.3%) data. With these parameters the accuracy is about 0.891. By varying the *unique* parameter and setting it to $unique = no$, we

K/Features	Unique	Accuracy	Precision	Recall
17	Y	0,8135	P: 0,84 N: 0,79	P: 0,78 N: 0,85
17	N	0,8166	P: 0,84 N: 0,80	P: 0,79 N: 0,85
22	Y	0,8166	P: 0,84 N: 0,80	P: 0,78 N: 0,86
22	N	**0,8433**	P: 0,88 N: 0,82	P: 0,80 N: 0,89
23	Y	0,8245	P: 0,88 N: 0,78	P: 0,75 N: 0,90
23	N	**0,8370**	P: 0,84 N: 0,84	P: 0,84 N: 0,84
25	Y	0,7821	P: 0,79 N: 0,78	P: 0,77 N: 0,80
25	N	0,8009	P: 0,84 N: 0,84	P: 0,75 N: 0,85
28	Y	0,8088	P: 0,82 N: 0,80	P: 0,79 N: 0,83
28	N	0,8088	N: 0,81 P: 0,80	P: 0,80 N: 0,81
33	Y	**0,8527**	P: 0,87 N: 0,84	P: 0,82 N: 0,88
33	N	**0,8480**	P: 0,87 N: 0,83	P: 0,82 N: 0,88

Figure 4.7: Complete results of Method 2B + SVM

get an accuracy of: 0.871 . Also varying n and k and analyzing the configuration formed by: $k = 5, n = 5, unique = yes$ (producing 24 features and 1433 data, i.e. 74.1%), the accuracy obtained is about 0.867; when $unique = no$ the accuracy is 0.886.
By further varying the parameters and trying the configurations: $k = 6, n = 3, unique = yes$ and $k = 6, n = 3, unique = no$ (producing 21 features and 1320 data, i.e. 68.3%), the accuracies obtained are approximately 0.856 and 0.865 respectively.
It can be noted that the highest results are obtained with a number of features of 17. Probably, by varying the parameters in order to obtain a higher number of features, even higher accuracies could have been obtained, but this was not done as L^3 does not works efficiently with a large number of features (greater than 24

for example) and it would not have been possible to make a comparison.

More details on the results can be found in Figure 4.8, where the following are reported: the exact accuracy as the combination of parameters (and so the number of features) varies, the precision, for positive and negative instances, and the recall, for positive and negative instances.

k	n	N° features	% remaining data	Unique	Accuracy	Precision	Recall
4	4	17	69,27	Y	**0,8914**	P: 0,91 N: 0,87	P: 0,91 N: 0,87
4	4	17	69,27	N	0,8710	P:0,90 N:0,84	P:0,88 N:0,86
4	5	21	72,74	Y	0,8642	P:0,90 N: 0,82	P:0,86 N: 0,87
4	5	21	72,74	N	**0,8836**	P:0,89 N:0,87	P:0,91 N:0,85
5	5	24	74,13	Y	0,8668	P:0,88 N: 0,85	P:0,89 N:0,84
5	5	24	74,13	N	**0,8858**	P: 0,90 N:0,87	P:0,90 N:0,87
6	3	21	68,29	Y	0,8555	P:0,89 N:0,82	P:0,86 N:0,85
6	3	21	68,29	N	0,8647	P:0,90 N:0,82	P:0,87 N:0,86

Figure 4.8: Complete results of Method 3 + SVM

4.2.6 Vader

As regards the results of the *Vader* method applied to our dataset, this algorithm works quite well.
It is important to remember that this algorithm is based on scores given to words and sentences and no training is carried out.
The results that are produced by setting the threshold to 0 for the *compound* attribute, as explained in the previous chapter, are as follows:

- Incorrectly predicted instances = 340, i.e. 17.0%.
- Instances predicted correctly = 1660, i.e. 83.0%.

So, this method obtain 83% of accuracy.
As will be seen in the following, the methods devised in this work will achieve accuracies that tend to be higher than that of *Vader*.

A couple of examples of instances predicted incorrectly are as follows:

- *Homelands War exhibition just behind the top station was interesting and moving .* - True: POSITIVE - predicted: NEGATIVE

- *I would never again book with Select Dubrovnik as they do not have the tourists ' best interest in mind .* - True: NEGATIVE - Predicted: POSITIVE

Where *True* represents the ground truth and *Predicted* represents the predicted label.

4.3 Results of the 5 methods used for Sentiment Analysis

4.3.1 Method 1

In *Method 1*, the n most important words in the chosen k topics are taken and the sentences are filtered accordingly, discarding some rows. The parameters for which tuning must be carried out in this case are two: k and n, which are respectively the number of topics and the number of the most important words in each topic. Each combination of parameters will create a different number of features, which correspond to the number of unique top words found for all those topics.
Once the data has been transformed into the Bag of Words representation, the resulting features will be fed to the L^3 classifier.
Accuracy, precision and recall are collected for different combinations of these parameters.
The reference configuration for this method is $k = 5$, $n = 5$ yielding 19 features and 1433 data (i.e. 74.1% of the total). With this configuration is obtained an accuracy of about 0.873.
If the parameter n varies, placing $n = 6$, is produced a higher number of features and data (24 and 1471 respectively, i.e. 76.1%), but the accuracy remains approximately the same, equal to 0.872.
By varying k instead, thus placing $k = 6$ and $n = 5$ are produced 25 features and 1510 (78.1%) remaining data, with an accuracy of about 0.880.
By further varying k and placing $k = 7$ and $n = 5$ are produced 25 features and 1486 (76.9%) remaining data, with a higher accuracy of about 0.884 .
Varying further k and n and setting $k = 4$, $n = 4$, a slightly higher accuracy is obtained, about 0.887, but the number of features is reduced to 13 and consequently also the data, which become 1339 (69.3%), thus losing some extra information content, but this can also be viewed as a further step of data selection.
So the maximum performance of this method, with the combinations of parameters tested in this work, is around 0.884 or 0.887 of accuracies, depending on whether the reduced number of data of this case is considered a problem or not.

The results of this method are considered very good as it outperforms most of the baselines.
For example, it is much better than Vader, which is rule-based, and it also outperforms *Baseline 1*, which is a related method, but without the aid of topic modelling.
As for *Baseline 1 + SVM*, it has in general lower results than *Method 1*, except in some cases where *Baseline 1 + SVM* performs better than *Method 1*, but with a higher number of features.
Finally, comparing *Method 1* with L^3 and *Method 1 + SVM*, the results appear in general comparable.
Specifically, with the same number of features (13 and 25 in particular), the results of *Method 1 + SVM*, appear a bit lower. By increasing the number of features, for example using 33 or 39 features, the performances with *Method 1 + SVM* are higher: 0.899 and 0.901 respectively (a similar thing happens when we increase the features of *Baseline 1 + SVM*).
This is not a bad thing, since not only with *Method 1* we obtain higher values with a lower number of features, but we also obtain a readable and interpretable method, that can be a vehicle for useful information on the mobility issues, as will be discussed in detail at the end of this chapter.

Obviously one could do a more exhaustive search in parameter space and try other combinations to produce different numbers of output features, even higher than 24/25, perhaps using more powerful machines for calculation (as, with more than 25 features, this method is too slow) and maybe with more features can be produced also higher accuracies.
More details of the results can be found in Figure 4.9, where the exact accuracy, precision, for positive and negative instances, and recall, for positive and negative instances, are presented.

4.3.2 Method 2A

In *Method 2A* each sentence is mapped with the probability of belonging to each of the k topics. The features obtained, therefore, are equivalent to the number k of topics chosen and are fed to the L^3 classifier. The parameter to tune is only k.
Accuracy, precision and recall are collected for different values of this parameter.
The reference configuration is to set $k = 15$. With this configuration a very low accuracy is obtained, about 0.614. As the value of k increases, the accuracy increases: for $k = 19$, an accuracy of 0.649 is obtained; for $k = 21$, we get an accuracy of 0.729.
It can be seen that as k increases, the performance tends to increase, even if it

k	n	N° features	% remaining data	Accuracy	Precision	Recall
4	4	13	69,27	**0,8869**	P: 0,91 N: 0,86	P: 0,90 N: 0,87
4	5	17	72,74	0,8578	P: 0,86 N: 0,86	P: 0,90 N: 0,79
5	5	19	74,13	0,8732	P: 0,90 N: 0,84	P: 0,88 N: 0,86
5	6	24	76,10	0,8724	P: 0,92 N: 0,82	P: 0,86 N: 0,90
6	4	21	73,82	**0,8769**	P: 0,88 N: 0,87	P: 0,90 N: 0,85
6	5	25	78,12	**0,8798**	P: 0,91 N: 0,85	P: 0,87 N: 0,89
7	5	25	76,88	**0,8839**	P: 0,91 N: 0,86	P: 0,88 N: 0,89

Figure 4.9: Complete results of Method 1

remains quite low and furthermore with $k = 23$ and $k = 24$ the accuracy has again a drop (0.715 and 0.708 respectively).
Therefore, in general, this method produces lower results and is not competitive with the other methods and with the baselines, and for this reason a baseline using this method for data transformation + SVM as classifier was not created.
In addition, higher values were not tested because the method is very slow and fails to conclude the training.
More details of the results can be found in Figure 4.10, where the exact accuracy, precision, for positive and negative instances, and recall, for positive and negative instances, are presented.
Probably the poor results of this method are due to the fact that the topic-only representation at sentence level is too summarizing and is not sufficient to describe the sentences in an exhaustive way and to give enough information for the L^3 classifier, which therefore cannot construct sufficiently precise and correct rules that correctly discriminate positive from negative reviews.

4.3.3 Method 2B

In *Method 2B*, each word in a sentence is mapped to the most likely topic for that word. Subsequently the data is then transformed into the Bag of Words representation and sent to the L^3 classifier. So, the number of features will correspond to the parameter k which indicates the number of topics.
The parameters on which the tuning is carried out are: k, which corresponds to the number of topics, and *unique* which has binary values (*yes* or *no*) and

k	Accuracy	Precision	Recall
15	0,6144	P: 0,85 N: 0,57	P: 0,28 N: 0,95
19	0,6489	P: 0,81 N: 0,60	P: 0,38 N: 0,91
21	0,7288	P: 0,83 N: 0,68	P: 0,57 N: 0,88
23	0,7147	P: 0,82 N: 0,66	P: 0,54 N: 0,89
24	0,7085	P: 0,87 N: 0,65	P: 0,48 N: 0,93

Figure 4.10: Complete results of Method 2A

indicates whether the topic is counted only once or more times in the Bag of Words representation, when the topic is repeated several times in the same sentence.
Accuracy, precision and recall are collected as the values of these parameters vary. The reference configuration is $k = 15, unique = yes$, which has an accuracy of about 0.817. By varying k, with $k = 19$ the accuracy is about 0.814, for $k = 20$ we obtain an accuracy of approximately 0.821 and for $k = 22$ we obtain an accuracy of approximately 0.782.
Similarly, in the configurations with the same values of k and value of $unique = no$, we get the following values of accuracies: for $k = 15$ is obtained an accuracy of about 0.829, for $k = 19$ the accuracy is about 0.801, for $k = 20$ it is about 0.809 and for $k = 22$ it is about 0.792.
It was not possible to try higher values of k, as L^3, with the features obtained with this method, is even slower than in other methods and cannot produce results above the 22 features.
It is important to note that this method, although based on the topics, gives better results than the *Method 2A*. This is due to the fact that the data is mapped to topics with a higher granularity. Indeed, instead of assigning a topic (by means of a probability distribution) to a sentence, as in *Method 2A*, here the topic is assigned to each word of the sentence.
However, the results are less accurate than *Method 1*. Evidently, although the representation of the topics is more specific than before, it is still more generic than taking the exact words and selecting them.
Finally, the results of this method using L^3 are comparable or slightly lower than those of the same method but using SVM classifier (*Method 2B + SVM*), with the

same number of features. As is to be expected, SVM, produces some higher results with a number of features higher than 22, as SVM computation is faster even if the number of features increases, unlike L^3.
More details of the results can be found in Figure 4.11, where the exact accuracy, precision, for positive and negative instances, and recall, for positive and negative instances, are presented.

k	Unique	Accuracy	Precision	Recall
11	Y	0,7868	P: 0,81 N: 0,77	P: 0,75 N: 0,82
11	N	0,7837	P: 0,77 N: 0,80	P: 0,80 N: 0,77
15	Y	0,8166	P: 0,83 N: 0,81	P: 0,80 N: 0,83
15	N	**0,8292**	P: 0,83 N: 0,83	P: 0,83 N: 0,83
19	Y	**0,8135**	P: 0,81 N: 0,82	P: 0,81 N: 0,81
19	N	0,8009	P: 0,80 N: 0,81	P: 0,81 N: 0,79
20	Y	**0,8213**	P: 0,84 N: 0,81	P: 0,80 N: 0,85
20	N	0,8088	P: 0,81 N: 0,81	P: 0,81 N: 0,81
22	Y	0,7821	P: 0,79 N: 0,78	P: 0,77 N: 0,80
22	N	0,7915	P: 0,79 N: 0,79	P: 0,79 N: 0,79

Figure 4.11: Complete results of Method 2B

4.3.4 Method 2C

In *Method 2C*, which is a slight variation of *Method 2B*, the number of features will correspond to the parameter k which indicates the number of topics.
The parameters on which the tuning is performed are: k, which corresponds to the number of topics, and *unique* which has binary values (*yes* or *no*) and indicates

whether the topic is counted once or several times in the Bag of Words representation, when the topic is repeated several times in the same sentence.
Accuracy, precision and recall are collected as the values of these parameters vary. Taking as a reference configuration $k = 19$, $unique = yes$, we can see that the accuracy is very low (about 0.654) and setting $unique = no$, it increases a little, but remains low, equal to about 0.707.
Increasing progressively the number of features, we observe that the accuracy increases and is always higher when the parameter $unique = no$, so the multiplicity of topics matters more in this method than in others.
However, it remains low around 24 features: it is about 0.693 for $unique = yes$ and 0.765 for $unique = no$.
In addition, higher values of k have not been tested for $unique = yes$ due to slow computation. For $unique = no$, the computation is a bit faster, so other higher values of k have been tried, but they did not give very high performances: the accuracy did not exceed about 0.792 (with 29 features), and by increasing the number of features beyond 35, the computation becomes very slow also in this setting.
Therefore, this method gives inferior results and is not comparable with the other methods and with the baselines. Evidently, taking more topics for a word is misleading for the L^3 classifier.
In the future we could try to change the value of the threshold to explore more this technique.
More details on the results can be found in Figure 4.12, where the exact accuracy, the precision, for positive and negative instances, and the recall, for positive and negative instances, are presented.

4.3.5 Method 3

Method 3 is the union of *Method 1* and *Method 2B*, so only a subset of words (the most important) are considered and the phrases are filtered accordingly. The data is then transformed into the Bag of Words representation and sent to the L^3 classifier. The features that will be provided to the classifier are the selected words + the k topics.
The parameters for which the tuning must be carried out in this case are 3: number of main words per topic, n; number of topics, k; *unique*, which takes binary values (*yes* or *no*) and indicates whether to consider repeated topics one or more times.
Accuracy, precision and recall are collected as the values of these parameters vary. The reference is the set of parameters formed by: $n = 4, k = 3, unique = yes$, which produces 13 characteristics and 1205 (62.3%) data. With these parameters

k	Unique	Accuracy	Precision	Recall
19	Y	0,6536	P: 0,60 N: 0,81	P: 0,90 N: 0,41
19	N	0,7069	P: 0,69 N: 0,73	P: 0,75 N: 0,66
22	Y	0,6677	P: 0,62 N: 0,79	P: 0,87 N: 0,47
22	N	0,7273	P: 0,71 N: 0,75	P: 0,78 N: 0,68
24	Y	0,6928	P: 0,78 N: 0,65	P: 0,54 N: 0,85
24	N	0,7649	P: 0,74 N: 0,79	P: 0,80 N: 0,73
29	N	**0,7915**	P: 0,80 N: 0,79	P: 0,78 N: 0,80
36	N	0,7508	P: 0,73 N: 0,78	P: 0,80 N: 0,70

Figure 4.12: Complete results of Method 2C

the accuracy is about 0.859. By varying the *unique* parameter and setting it to $unique = no$, we get the same accuracy.
Varying k and analyzing the configuration formed by: $n = 4, k = 4, unique = yes$ (producing 17 characteristics and 1339 (69.3%) data), the accuracy obtained is about 0.896. When $unique = no$, the accuracy is approximately 0.864. By further varying the parameter k and trying the configurations: $n = 4, k = 5, unique = yes$ and $n = 4, k = 5, unique = no$ (producing 21 characteristics and 1406 (72.7%) data), the accuracy obtained is approximately 0.841 in the first case and 0.864 in the second.
By varying also n and setting $n = 5, k = 5, unique = yes$, we obtain 24 features and 1433 (74.1%) data and an accuracy of about 0.867; when $unique = no$ the accuracy is about 0.871.
It can be seen that the highest result is obtained with a number of features equal to 17 and $unique = yes$.
This is the absolute highest result of all the methods created in this work and the baselines (for a number of features compatible with L^3). The corresponding baseline *Method 3 + SVM*, with 17 features and $unique = yes$, achieved a slightly lower accuracy, having also the disadvantage of being a non-interpretable method.

Method 3 integrates *Method 1*, which with the corresponding combination of k and n obtained 13 features and a slightly lower accuracy of 0.887. Indeed, by adding topics to the top words, there are two different types of information and at two different levels of granularity, favoring the classification with L^3 positively.

It is possible that by varying the parameters in an ad-hoc way to obtain a certain number of features, even greater accuracies could have been obtained, but this has not been done as it is not possible to explore the entire search space.
More details on the results of this method can be found in Figure 4.13, where the following are reported for different parameter combinations (and thus for different number of features): the exact accuracy, the precision, for positive and negative instances, and the recall, for positive and negative instances.

k	n	N° features	% remaining data	Unique	Accuracy	Precision	Recall
4	3	13	62,34	Y	0,8593	P: 0,90 N: 0,81	P: 0,87 N: 0,85
4	3	13	62,34	N	0,8593	P: 0,90 N: 0,81	P: 0,87 N: 0,85
4	4	17	69,27	Y	**0,8959**	P: 0,91 N: 0,87	P: 0,91 N: 0,88
4	4	17	69,27	N	0,8643	P: 0,86 N: 0,88	P: 0,92 N: 0,78
4	5	21	72,74	Y	0,8405	P: 0,85 N: 0,83	P: 0,89 N: 0,77
4	5	21	72,74	N	0,8642	P: 0,86 N: 0,87	P: 0,92 N: 0,79
5	4	21	70,77	Y	0,8407	P: 0,86 N: 0,82	P: 0,87 N: 0,80
5	4	21	70,77	N	0,8451	P: 0,85 N: 0,85	P: 0,90 N: 0,78
5	5	24	74,13	Y	0,8668	P: 0,88 N: 0,85	P: 0,89 N: 0,83
5	5	24	74,13	N	**0,8710**	P: 0,88 N: 0,87	P: 0,90 N: 0,83

Figure 4.13: Complete results of Method 3

4.4 Interpretability

4.4.1 Graphical representations and L^3 *Human readable*

This section deals with the problem of interpretability of the results and of the model. Indeed, one of the objectives of this thesis is to extract useful information, concepts and words associated with sustainable mobility, which can be used by

third parties. In particular the extraction of this information, could be useful for the authorities to have a better knowledge of the strengths and weaknesses of the services, according to the users opinion, and to be able to do something concrete with the aim to improve the service and the customer satisfaction.
Also for these reasons it was chosen to use an Associative Classifier that produces rules, which make the model explicit and can provide useful insights for the purposes expressed above.
In particular, L^3, with the use of the L^3 wrapper, has the so-called *Human readable* representation, thanks to which one has access to the rules (I and II level rules), in extended format.
More specifically, the printed rules will have a series of fields containing the following information:
<rule_id> <antecedent> <class label> <support count> <confidence(%)> <rule length>.
The field named antecedent in particular will have the following form:
$attr_1$*:value,* $attr_2$*:value,* $attr_3$*:value ... ,* $attr_n$*:value*.
In our case, the attributes will be the features used in that specific model, so they will be typically the top words and/or topics, and the values of the features will be typically 0 or more than 0, thus indicating the presence or absence of that attribute in that rule and its frequency.
Therefore, the intuition of this work, was to use this readable representation in order to analyse the results and capture the predominant concepts.
In particular, the first level rules will be read and analysed, as they are in the order of hundreds, unlike the second level rules which are in the order of thousands and hundreds of thousands and could not be read by the human eye.
In addition, a compact and graphical representation of the rules has been created. It aids overall visual understanding and allows trends and tendencies to be identified thanks to its summary information, which would not have been possible by reading the specific rules in detail one by one (this representation was created on Google Colaboratory).

In the following will be shown the *Human readable* representation and the graphical and compact representation of the I level rules of *Method 1* and *Method 3*, which are the methods that produced the best results.

Figure 4.14 shows an extract of the human readable representation produced by the L^3 classifier (*Method 1* with $k = 4, n = 4$).

Figures 4.15, 4.16, 4.17 and 4.18, instead, show the graphical and compact representation, divided by positive and negative instances, of *Method 1* and *Method 3* in the best configurations: $k = 4, n = 4$ and $k = 4, n = 4, unique = yes$, respectively.

0	compani:0,great:1	POSITIVE	97	100.0	2
1	compani:0,not:0,worth:1	POSITIVE	70	100.0	3
2	cabl:0,car:0,citi:0,compani:0,dubrovnik:0,not:0,restaur:0,time:0,view:0,wait:0	POSITIVE	55	100.0	10
3	cabl:0,citi:0,compani:1,dubrovnik:0,love:0,restaur:0,view:0,worth:0	NEGATIVE	43	100.0	8
4	cabl:0,car:0,compani:0,love:1,wait:0	POSITIVE	38	100.0	5
5	compani:0,great:0,love:1,not:0,wait:0	POSITIVE	38	100.0	5
6	cabl:0,citi:0,dubrovnik:0,great:0,love:0,not:0,restaur:0,time:0,view:0,wait:0,worth:0	NEGATIVE	36	100.0	11
7	compani:0,restaur:1,view:1	POSITIVE	26	100.0	3
8	car:0,citi:0,dubrovnik:0,great:0,love:0,not:0,restaur:0,time:0,view:0,wait:0,worth:0	NEGATIVE	22	100.0	11
9	cabl:0,car:1,citi:0,dubrovnik:0,great:0,love:0,restaur:0,time:0,wait:0,worth:0	NEGATIVE	20	100.0	10
10	cabl:0,car:1,citi:0,dubrovnik:0,great:0,love:0,not:0,restaur:0,wait:0,worth:0	NEGATIVE	19	100.0	10
11	cabl:0,car:0,compani:0,time:0,view:2,wait:0	POSITIVE	13	100.0	6
12	citi:0,dubrovnik:0,great:0,love:0,not:1,restaur:0,time:0,view:0,wait:1,worth:0	NEGATIVE	10	100.0	10
13	cabl:1,car:1,citi:0,compani:0,great:0,love:0,restaur:0,time:1	POSITIVE	9	100.0	8
14	car:2,citi:0,compani:0,great:0,love:0,restaur:0,time:0,worth:0	NEGATIVE	8	100.0	8
15	cabl:1,compani:0,dubrovnik:1,great:0,love:0,time:0,view:1,wait:0	POSITIVE	8	100.0	8
16	cabl:0,car:0,compani:0,great:0,love:0,not:0,time:1,view:1,wait:0,worth:0	POSITIVE	6	100.0	10
17	citi:1,compani:0,dubrovnik:1,love:0,not:0,restaur:0,time:0,view:1,wait:0,worth:0	POSITIVE	6	100.0	10
18	car:1,citi:1,compani:0,love:0,not:0,time:0,view:1,worth:0	POSITIVE	6	100.0	8
19	cabl:0,citi:0,compani:0,dubrovnik:0,great:0,love:0,not:2,restaur:0,time:0,view:0,wait:1,worth:0	NEGATIVE	5	100.0	12
20	citi:1,dubrovnik:0,great:0,love:0,not:1,view:0,wait:0	NEGATIVE	5	100.0	7

Figure 4.14: Example of Human readable representation

The reported plots have the appearance of a confusion matrix. In the rows are present the various rules created from the model, on the columns are present the features, that are the top words and/or the topics.
In every cell is present the number 0, 1, or more, that represents the presence of that attribute in that rule and its frequency.
The color indicates if they are rules with positive (blue) or negative (green) class label.
Finally, the intensity of the color indicates the *confidence* level of the rule: the more intense the color, the more confidence the rule will have, i.e. the rule is more reliable and true, vice versa when the color is less intense.
In order to make the concept clearer, the definition of *confidence* of a rule is the following: the confidence of an association rule is a percentage value that indicates how frequently the consequent (*head*) of the rule is verified among all groups that contain the antecedent (*body*) of the rule. The confidence indicates how trustworthy this rule is. Numerically, the confidence of a rule is the $\frac{m}{n}$ ratio as a percentage, where m is the amount of groups that contain the head and body of the rule together; n is the number of groups that contain only the body of the rule.
In the plots shown here, the confidence of all rules is quite high (> 0.7), because only the I level rules are shown, which tend to have higher confidence and reliability than the II level rules.

These plots, when analysed, can provide very interesting information on how the model works.

In particular, we can see how both the presence and the absence of certain words is an integral part of the rules that are produced, as well as the co-occurrence of terms together.
It can be noted, for example, that the absence of words like *love* and *great* is necessary to determine that a sentence is negative. This observation is quite intuitive, but one can also discover less immediate things.
For example, the absence of the word *company* is necessary for positivity of the sentence. This is, perhaps, because it tends to be that, when company is mentioned, it is to criticise something.
One can also observe for example that for many positive rules, the word *wait* is required to be absent.
It can also be noted that, for negative rules, the word *worth* often does not have to be there. Instead, it can be present when it is also accompanied by the word *not*.
In the negative rules we can also observe that the presence of the word *not* is often required, even more than once, and instead the word *restore* is required not to be present.
Finally, can also be noticed that in the negative rules it is often required that the word *view* is not present, unlike for the positive rules where, when it is present, its value is required to be 1 in general.

With regard specifically to *Method 3*, can be made similar observations with respect to words and can be also made additional considerations with respect to topics.
Indeed, with this method, can also be seen the presence or absence of topics, that tends to influence the class label of the rules.
For example, can be seen that topic 0 is particularly requested to be 1 in the positive rules, unlike topic 2, which is instead particularly requested to be 1 in the negative ones.

Therefore, it is also important to analyse the topics because, not only do they give improvements in accuracy, but these could also be explored in the future, by looking at the words from which they are composed and, in this way, it is possible to analyse in more detail the results of these methods.

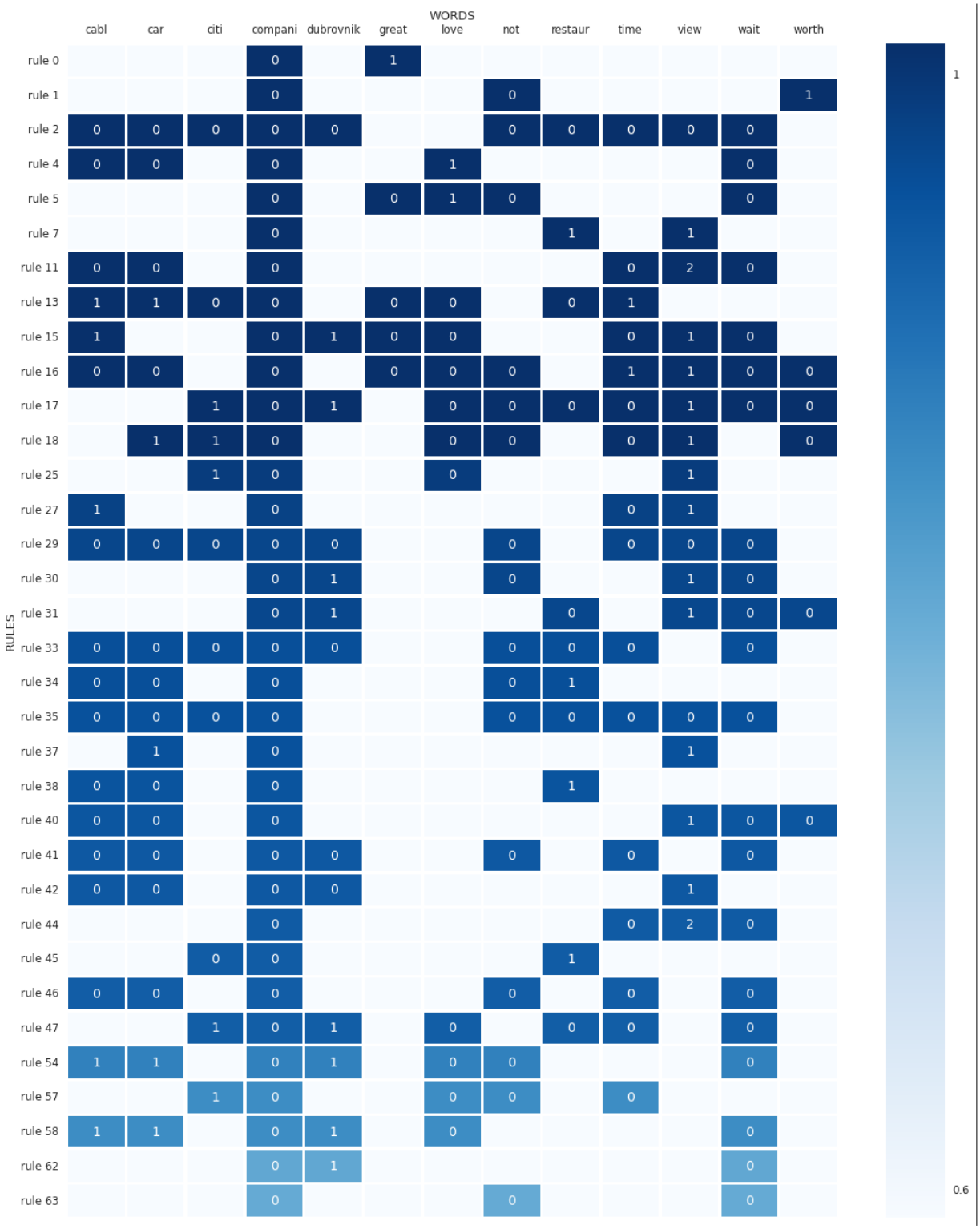

Figure 4.15: Rules obtained with Method 1 for positive instances

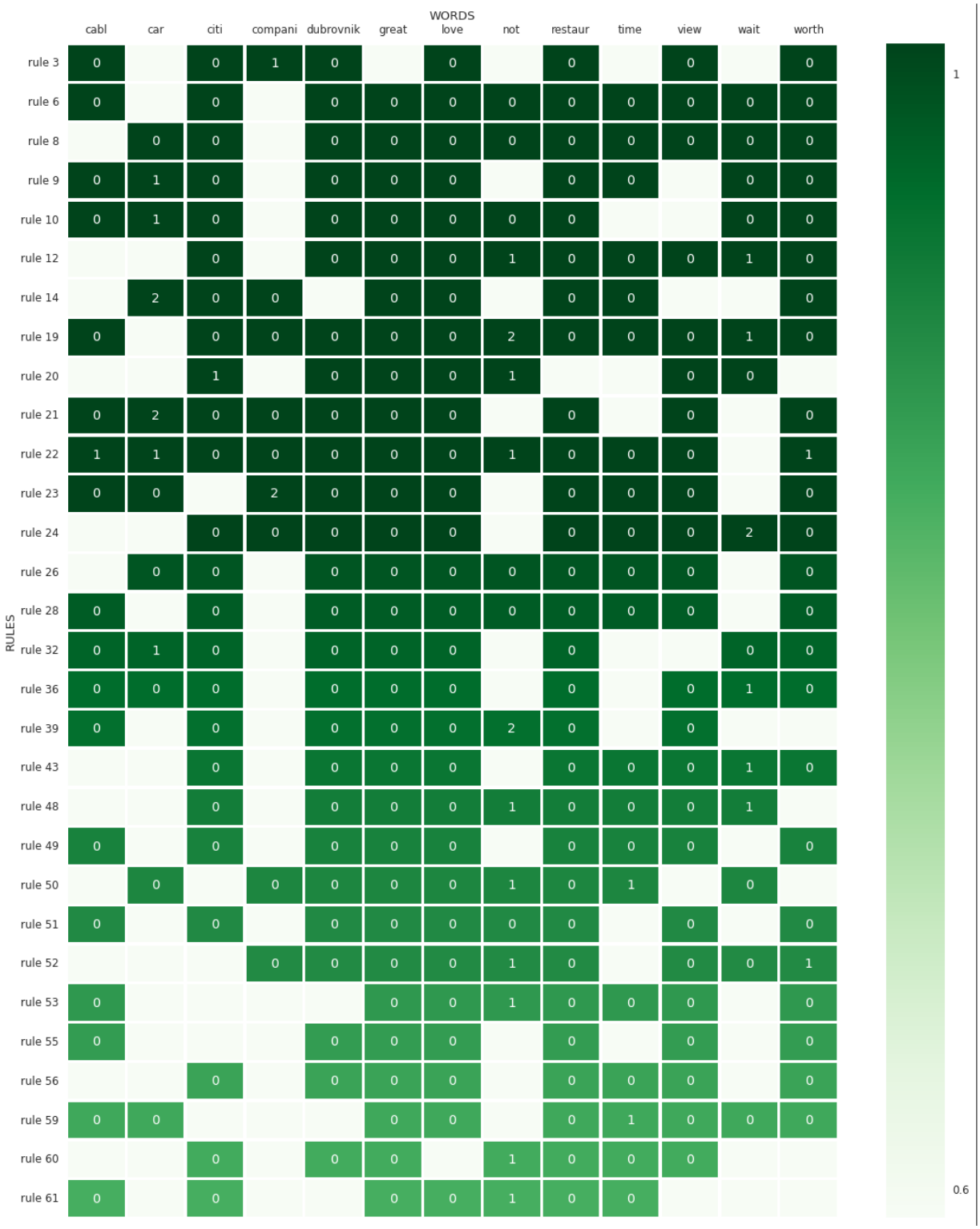

Figure 4.16: Rules obtained with Method 1 for negative instances

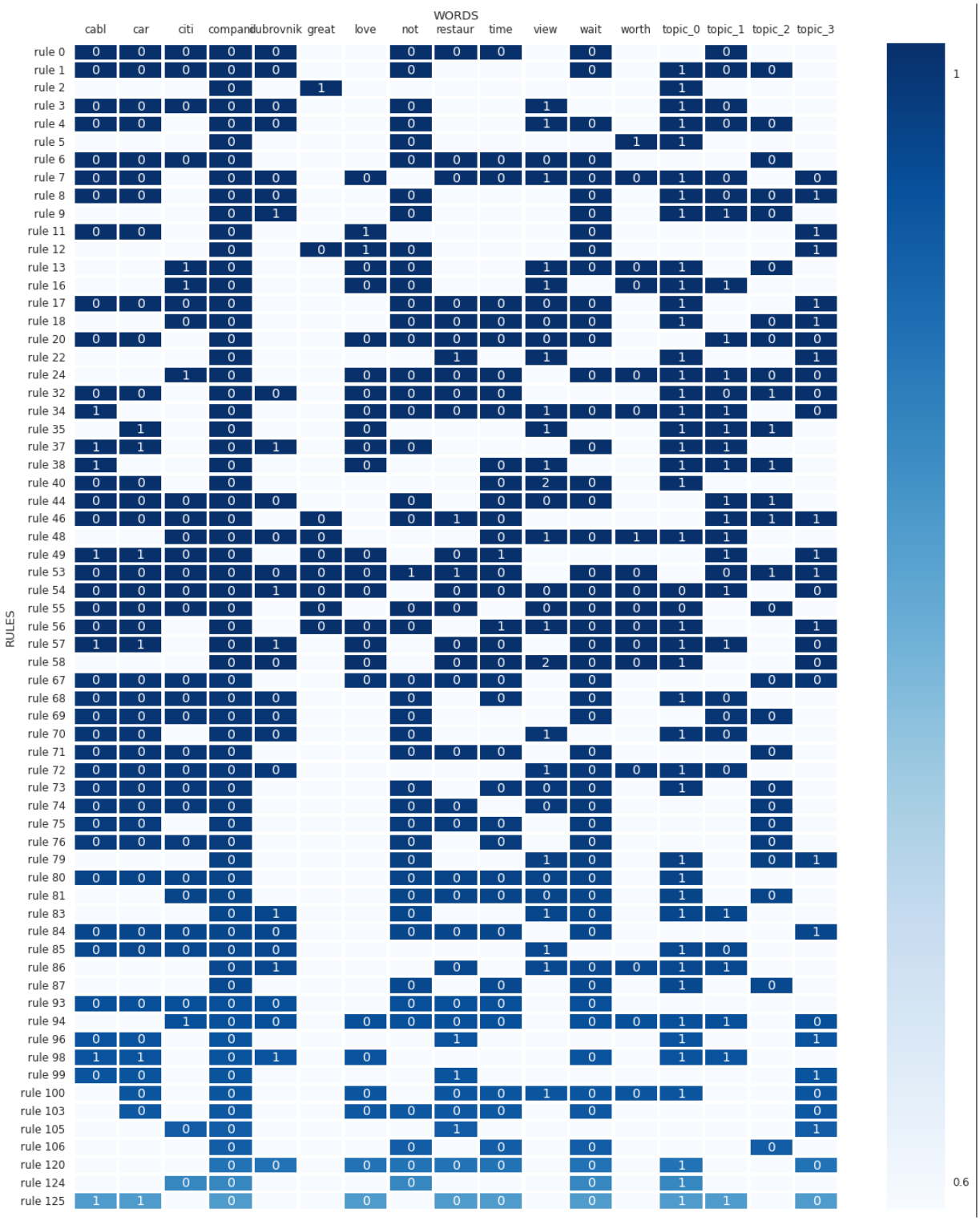

Figure 4.17: Rules obtained with Method 3 for positive instances

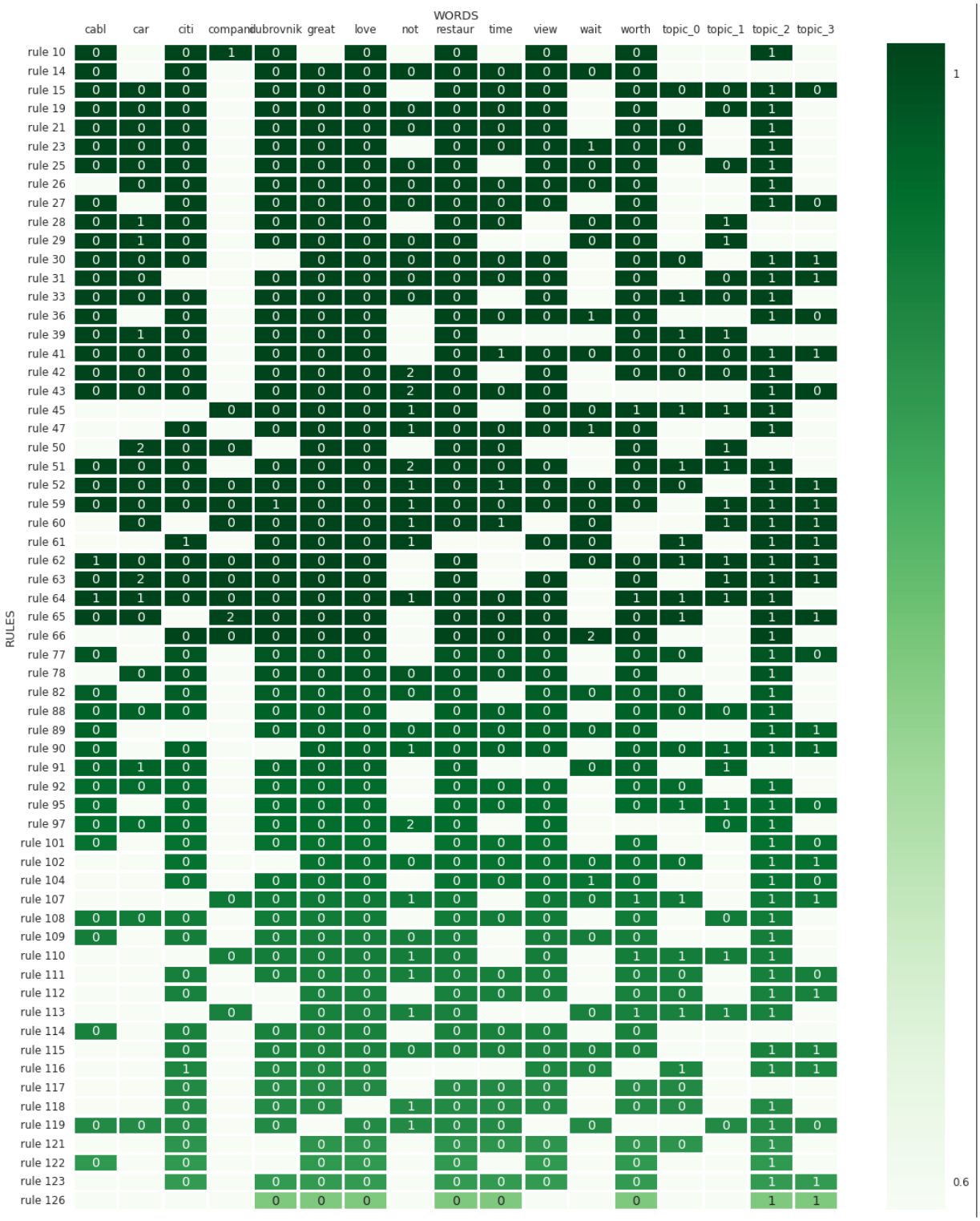

Figure 4.18: Rules obtained with Method 3 for negative instances

4.4.2 Further study: an alternative and complementary representation with SHAP

This section will present a further analysis carried out in this work, with the aim of making a preliminary analysis in the field that will be covered, that will give some ideas for future works or for the continuation of this work. Consequently, it will not be an exhaustive treatment, but only a mention of the most significant and relevant results obtained.
Specifically, the *SHapley Additive exPlanations (SHAP)* method has been used, with the aim of providing an alternative and complementary representation to the one argued in the paragraph above, so as to highlight the strengths of our method and at the same time complete and extend it, giving an increasingly comprehensive interpretation of the model obtained.

SHAP is a game theoretic method, based on Shapley values [55] and their extensions, of explaining the output of machine learning models; it links optimal credit allocation with local explanations [56], [57].
From a mathematical point of view, Shapley values from game theory have theoretical guarantees, which are applied to local explanations of predictions of machine learning models.
By using SHAP, Shapley values are calculated by feeding each feature, one by one, into a conditional expectation function of the model output, and assigning the shift generated at each step to the feature that was introduced, then averaging this process over all available ordering of features [58].

The SHAP method can be applied both to models such as L^3, that is already explainable by itself, but mainly it can be applied to other machine learning algorithms, such as SVM, which otherwise would not be interpretable.
In this thesis work it has only been applied to SVM, both because SVM is one of the mothods that has been treated, and also to show the potential of this additional layer applied to algorithms that cannot be interpreted directly. It has been used the *Method 1* to obtain the features.

Through the use of SHAP, a series of graphs is produced. The graphs allow the *global interpretation* of the model, i.e. the impact of the input characteristics on the model as a whole, and, more importantly, allow the *local interpretation*, which reveals the impact of the input features on the individual predictions (i.e. for a individual sample).

In this thesis, specifically, have been produced some graphs using the *Explainer* and the *kernelExpainer* from [56], [57], which can provide useful and complementary

information to our graphical representation. They will be analysed in the following.

1. *Summary plot*

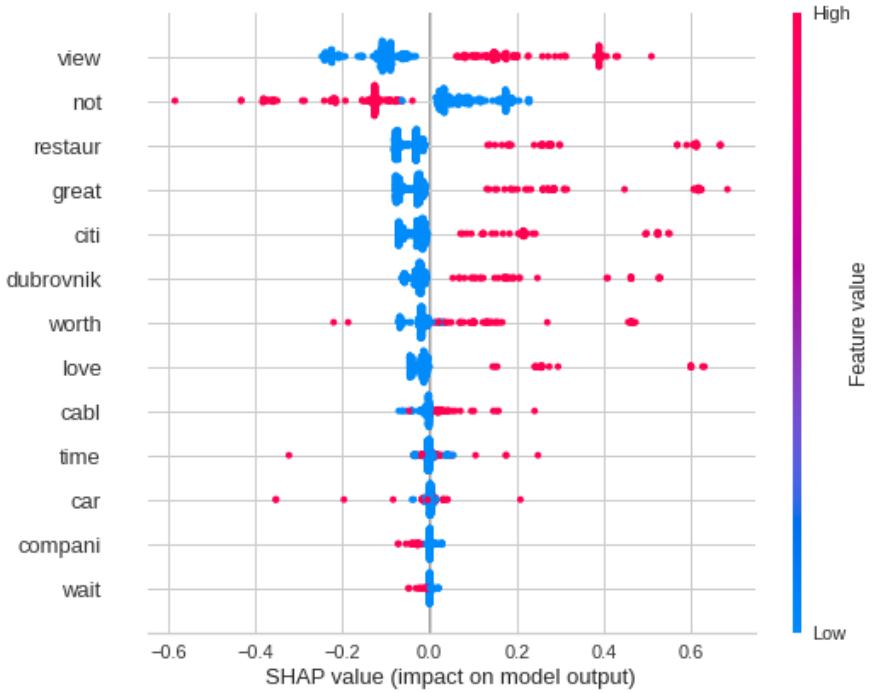

Figure 4.19: Summary plot

SHAP Summary plot shows the magnitude, prevalence and direction of the effect of a feature and it shows the positive and negative relationships of the predictors with the target variable. It avoids conflating the magnitude and prevalence of an effect into a single number, and thus reveals the rare large effects.
This plot is shown in Figure 4.19, where in the x-axis there is the SHAP value, that is the impact that feature has on the model prediction for that instance, and on the y-axis there is the indication of the features. In this plot, each point corresponds to a single instance, when several points stop at the same x position, they accumulate to show the density [58].
The graph reveals also the feature importance, since variables are ranked in descending order, and the direction of effects, e.g., how the presence of the *view* attribute tends to lead to positive sentiment and its absence leads to negative sentiment; vice versa, for example, for the *not* attribute. The color indicates whether that variable is present (in red) or absent (in blue) for that

observation.
The summary plot also shows the distribution of effect sizes, such as the long right tails of many attributes. These long tails mean that attributes with a low global relevance can be highly relevant for specific instances.
It can also be seen that most of the rare predictions are to the right and therefore there are more ways to classify abnormally positive than negative, except as regards the feature *not*.

2. *Dependence plot*
SHAP dependence plot shows how a feature's value on x-axis impacts the prediction on y-axis of every sample. It shows the marginal effect that one or two attributes have on the expected output of a model and shows if the link between the target and the variable is linear, monotonic or more complicated. In addition, the Python SHAP module automatically integrates another feature with which the feature chosen interacts the most, so we can see how the two variables interact [59].
In Figure 4.20, can be seen that the joint presence of *not=1* and *worth=1*, tends to reduce the SHAP value, driving to a negative prediction.
Another example of this plot can be found in Figure 4.21.

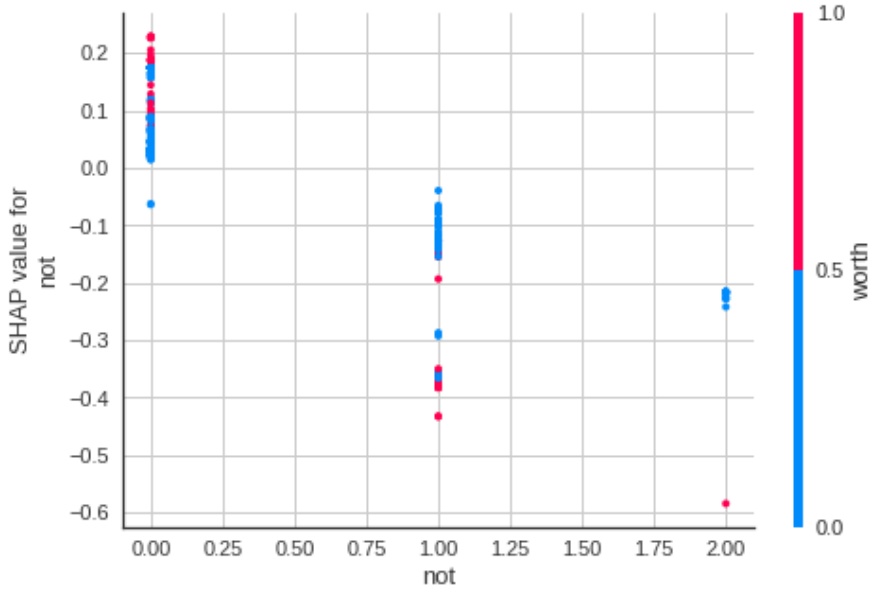

Figure 4.20: Dependence plot 1

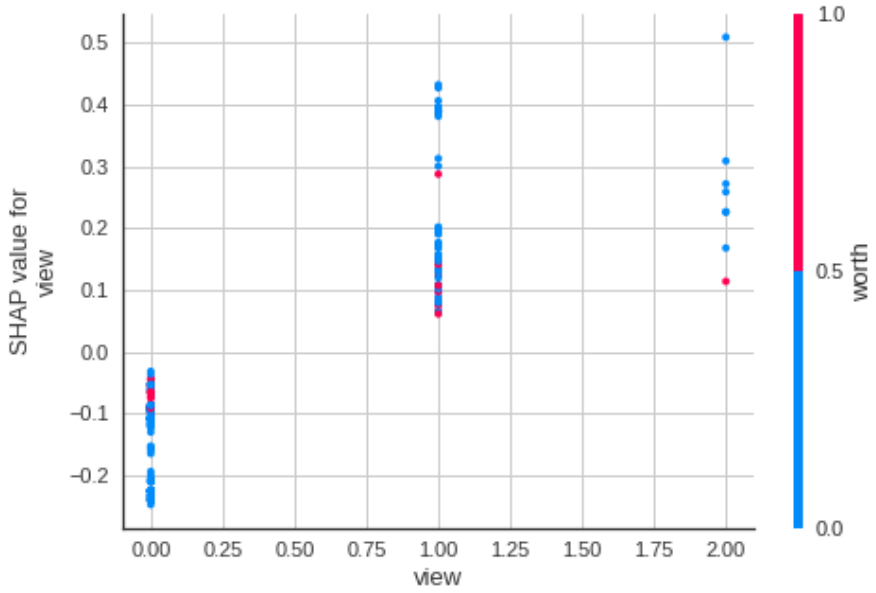

Figure 4.21: Dependence plot 2

3. *Force plot*
   The force plot depicts the contribution of each feature to the process of moving the value of the decision score from the base value (the average model output over the dataset) to the value predicted by the classifier. Features that push the prediction higher are shown in red, those that push the prediction lower are shown in blue and the lengths of the bars are the corresponding feature attributions [57].
   It is a *local explanation*, in fact a single instance is passed at a time.

   Let's describe in more details the components of this plot.

   - The output value is the prediction for that observation.
   - The base value is "the value that would be predicted if we did not know any features for the current output." [60]; in simple terms, it is the mean prediction.
   - Red/blue: features that drive the forecast higher (positive prediction) are displayed in red, and those that drive the forecast lower (negative prediction) are displayed in blue.

     In Figure 4.22, can be noticed that the prediction is 1 (positive) and that the features *not=0, view=1, city=1* push the prediction to an high value, vice-versa the features *restaur=0, great=0*, that drive the prediction

to a low value.
Similar observations can be done for the graph in Figure 4.23, in which the prediction value is 0, i.e. negative prediction [61].

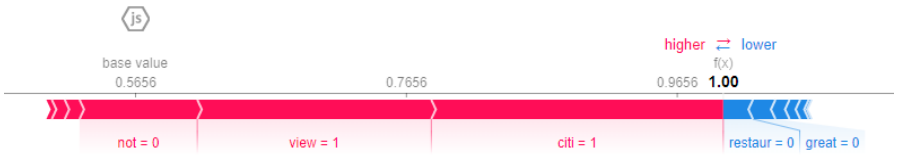

Figure 4.22: Force plot 1

Figure 4.23: Force plot 2

4. *Waterfall plot*
   The Waterfall plot is an alternative and extended representation of the force plot, but contains a similar information content. These plots are used to show explanations for individual predictions.
   The bottom of a waterfall diagram begins as the expected value of the model output, and then each row shows how the positive (red) or negative (blue) contribution of each feature shifts the value from the expected model output on the background dataset to the model output for this prediction.
   A couple of examples of a waterfall plots for two istances are reported in Figures 4.24 and 4.25, where the outputs are respectively a positive and a negative prediction and the features are sorted in order of importance.

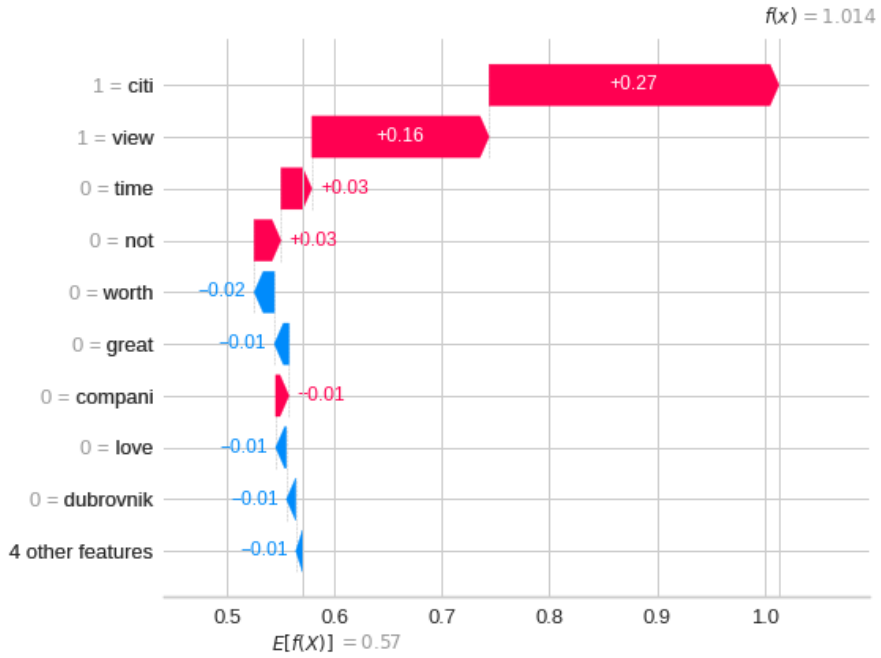

Figure 4.24: Waterfall plot 1

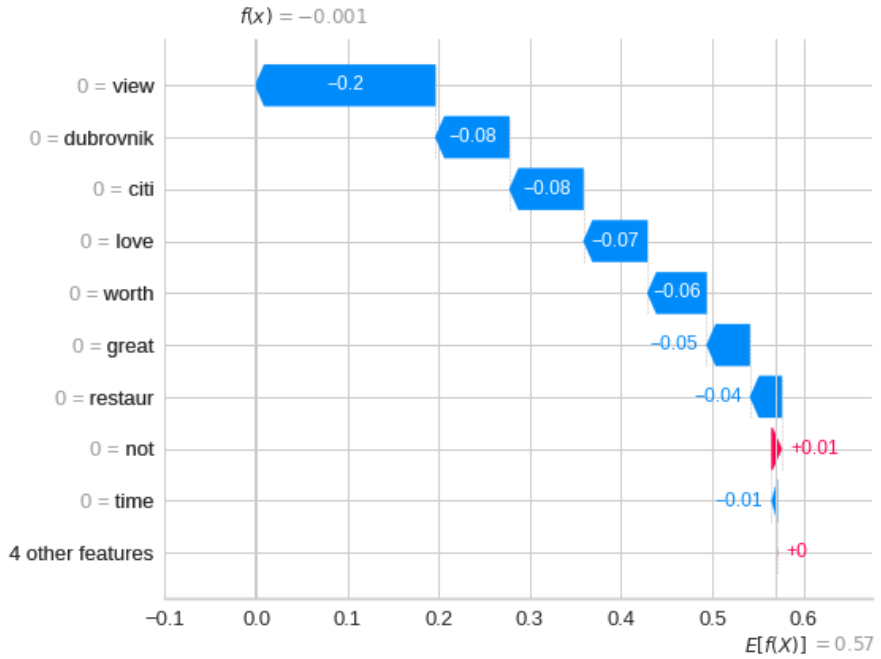

Figure 4.25: Waterfall plot 2

5. *Bar chart*
   The Bar chart plot in Figure 4.26 represents the average absolute value of the SHAP values for each attribute. The objective is to obtain a standard bar graph in which the importance of each attribute in determining a negative or positive sentiment is highlighted.
   As can be seen from Figure 4.26, the most important attribute is *view*, followed by *not* and *great*.

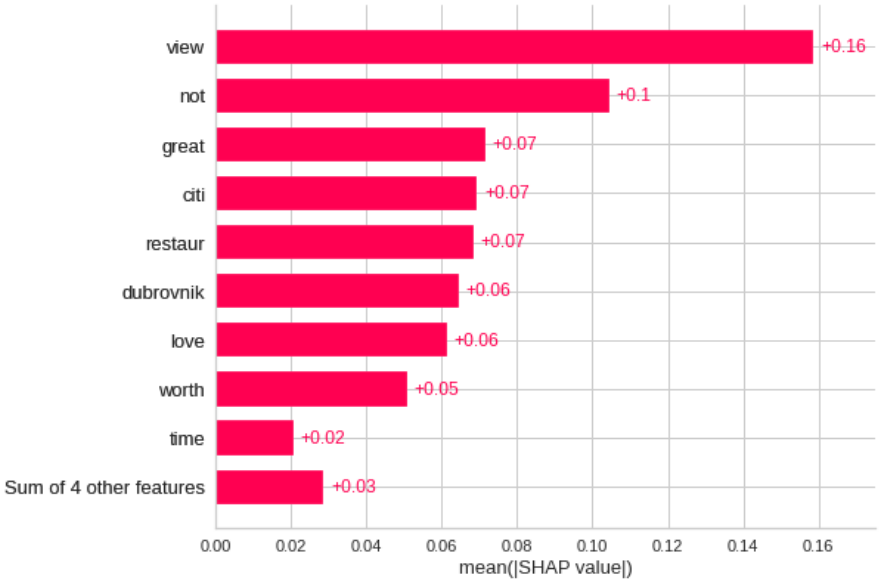

Figure 4.26: Bar chart

As can be seen from these example graphs, SHAP certainly provides a rich and accurate visualisation of the model. However, a point in favour of our rules representation, which remains implicit in SHAP, is the highlighting of co-occurrence of terms together, or more in general the correlation of terms with each other and the label in output, as SHAP focuses mainly on the impact of features taken individually.

These were just a few examples of the plots that can be made with SHAP. There are other plots that can be investigated, both generic for all machine learning algorithms and more specific for tree classifiers or natural language models, like those in the *Hugging Face* transformers library, for example.

An interesting further analys that could be done in the future is to apply SHAP to individual instances of the dataset and identify the L^3 rules that classify these instances. In this way, could be possible to see at a glance the differences between

the two methods of interpretation and visualisation and the added value of using them together.

Chapter 5

Conclusion

To draw conclusions from this thesis work, it is stressed that, in order to obtain a good Sentiment Analysis model, textual content on social networks, generated by users voluntarily, on transport and sustainability, was exploited.
Indeed, it is thought to be more useful to directly analyse comments freely provided on social networks, rather than polls, official statistics and traditional surveys carried out by public administrations. Indeed, free comments, are more reflective of the real opinion of users, who feel free to dwell on the aspects that struck them most. A potentially valid idea is also that of integrating these two kinds of data sources.
This analysis is particularly useful because transport pollution has increased a lot in recent years, and, more in general, transport practices have an impact on sustainability from a social, economic and environmental perspective [62], [63]. Consequently, politicians are looking for ways to address these problems by increasing the efficiency and sustainability of the transport system.
Sentiment Analysis can therefore help them to this aim. In particular, it can help to analyse the surveys and opinions expressed by users and so policy makers can make more informed decisions, as they have more information to work with.
The results obtained in this work demonstrate the accuracy and the validity of this approach, which can be extended to other datasets and contexts.

Now will be analyzed and summarized the major findings from this thesis work, that are the following:

1. A new methodology to perform Sentiment Analysis has been defined, i.e. through the use of rules and Associative Classifier and with the help of Topic Modeling.

2. The best results obtained are quite good. They are also better than the performance of the baselines with the SVM classifier (both those that use

topic modeling and those that do not), when models with the same number of features are compared.
More specifically, the best results of *Method 1* and *Method 3* emerge, which achieve the accuracies of 0.887 and 0.896, respectively. These are the highest results obtained in this work. They are also higher than the accuracies of the corresponding baselines with the SVM classifier applied to data transformed using *Method 1* and *Method 3* (0.882 and 0.891 of accuracies, respectively).
This is an important result even if the gap is not very high. Indeed, it must be considered that SVM is one of the best performing algorithms in machine learning and also, specifically, in Sentiment Analysis. So the fact that our method produces a slightly better result than SVM, is a very valuable point. Moreover, SVM produces a model that is not readable and not interpretable, unlike the rules produced by our method. These rules are perfectly readable, both with the human eye, with algorithms and with graphs.

3. Through the use of L^3 and the production of association rules, we were able to obtain a readable and summarizing model. It provides the main words and concept associations, regarding positive and negative opinions, about sustainable transport.
Through this model, a graphical representation has been obtained that allows an immediate analysis of these concepts. In particular, it allows to analyse the correlation and co-occurrence of concepts and terms together. The insights obtained from this representation can be used by the administrations to improve the service offered.
In addition, the issue of interpretability was further explored by implementing an alternative representation using SHAP. This provides additional information about the models and the contribution of the features. It can be used in conjunction with our representation.

The negative aspect of the method defined in this thesis work is that the process is very slow as the number of features fed to the model increases. In particular, the computation time starts to become unsustainable beyond 23 features.

5.1 Future work

Many of the things that have been done in this thesis work and the methodology developed, in the future could be extended to other works of the same type, analyzing larger corpora or other types of data. This methodology can also be applied in other different contexts.
On a practical side, the improvements that could be carried out are as follows. Probably, a better parameter tuning technique can be implemented, so as to cover

the search space more exhaustively and find better configurations.
Moreover, it could be found a way to parallelize the process or to use more powerful machines. In this way, the L^3 classifier could also be used with a greater number of features.
In addition, could be investigated the Topic Modeling aspects. Can be found a metric that works correctly for our purpose and that allows us to create better subdivided topics, instead of relying on extrinsic evaluation after applying L^3 classifier.
One could also try other Topic Modeling algorithms, which maybe work better for this dataset and task. For example, *Non-Negative Matrix Factorization (NMF)* algorithm has good performance even with a limited size dataset, like ours [64].
Finally, the interpretability aspect of the model could be further investigated. The preliminary analysis made in this thesis work on SHAP can be deepened and the complementarity of this method with the rules produced by L^3 can be studied. For example, one could apply SHAP to individual instances of the dataset and devise a way to identify the L^3 rules that classify these instances. In this way, could be possible to see at a glance the differences between the two methods of interpretation and visualisation, and the added value of using them together.

Bibliography

[1] B.B. Gupta, D. Perakovi?, A.A. Abd El-Latif, and D. Gupta. *Data Mining Approaches for Big Data and Sentiment Analysis in Social Media*. Advances in Data Mining and Database Management. IGI Global, 2021. ISBN: 9781799884156. URL: https://books.google.it/books?id=jP16zgEACAAJ (cit. on p. 2).

[2] Ainhoa Serna, Aitor Soroa, and Rodrigo Agerri. «Applying Deep Learning Techniques for Sentiment Analysis to Assess Sustainable Transport». In: *Sustainability* 13.4 (2021). ISSN: 2071-1050. DOI: 10.3390/su13042397. URL: https://www.mdpi.com/2071-1050/13/4/2397 (cit. on pp. 2, 29, 30).

[3] Elena Baralis, Silvia Chiusano, and Paolo Garza. «A Lazy Approach to Associative Classification». In: *IEEE Transactions on Knowledge and Data Engineering* 20.2 (2008), pp. 156–171. DOI: 10.1109/TKDE.2007.190677 (cit. on p. 2).

[4] Venkata Sai Rishita Middi, Middi Raju, and Tanvir Ahmed Harris. «Machine translation using natural language processing». In: *MATEC Web of Conferences* 277 (Jan. 2019), p. 02004. DOI: 10.1051/matecconf/201927702004 (cit. on p. 5).

[5] Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Z. Chen, and Yonghui Wu. «Direct speech-to-speech translation with a sequence-to-sequence model». In: *ArXiv* abs/1904.06037 (2019) (cit. on p. 5).

[6] Martin Popel, Markéta Tomková, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondrej Bojar, and Z. Žabokrtský. «Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals». In: *Nature Communications* 11 (2020) (cit. on p. 5).

[7] Vyacheslav Lyashenko, F. Laariedh, Lana Sotnik, and Ayaz Ahmad. «Recognition of Voice Commands Based on Neural Network». In: *TEM Journal* 10 (May 2021), pp. 583–591. DOI: 10.18421/TEM102-13 (cit. on pp. 5, 6).

[8] Ted Zhang, Dengxin Dai, Tinne Tuytelaars, Marie-Francine Moens, and Luc Van Gool. «Speech-Based Visual Question Answering». In: *ArXiv* abs/1705.00464 (2017) (cit. on p. 5).

[9] Catalin Ungurean and Dragos Burileanu. «An advanced NLP framework for high-quality Text-to-Speech synthesis». In: (May 2011). DOI: 10.1109/SPED.2011.5940733 (cit. on p. 5).

[10] Ishitva Awasthi, Kuntal Gupta, Prabjot Singh Bhogal, Sahejpreet Singh Anand, and Piyush Kumar Soni. «Natural Language Processing (NLP) based Text Summarization - A Survey». In: *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. 2021, pp. 1310–1317. DOI: 10.1109/ICICT50816.2021.9358703 (cit. on p. 5).

[11] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. «Text Summarization Techniques: A Brief Survey». In: *International Journal of Advanced Computer Science and Applications* 8.10 (2017). DOI: 10.14569/IJACSA.2017.081052. URL: http://dx.doi.org/10.14569/IJACSA.2017.081052 (cit. on p. 5).

[12] Rachit Garg, Arvind W Kiwelekar, Laxman D Netak, and Akshay Ghodake. «i-Pulse: A NLP based novel approach for employee engagement in logistics organization». In: *International Journal of Information Management Data Insights* 1.1 (Apr. 2021), p. 100011. ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2021.100011. URL: http://dx.doi.org/10.1016/j.jjimei.2021.100011 (cit. on p. 5).

[13] Kavitha Raju. «Speech Based Voice Recognition System for Natural Language Processing». In: *International Journal of Computer Science and Information Technogy* (Aug. 2014) (cit. on p. 6).

[14] Meena Rambocas. «Marketing research: The role of sentiment analysis». In: *FEP WORKING PAPER SERIES* (Apr. 2013) (cit. on p. 7).

[15] Cane Leung and Stephen Chan. «Sentiment Analysis of Product Reviews». In: (Jan. 2008) (cit. on p. 7).

[16] Nicola Capuano, Luca Greco, Pierluigi Ritrovato, and Mario Vento. «Sentiment analysis for customer relationship management: an incremental learning approach». eng. In: *Applied intelligence (Dordrecht, Netherlands)* 51.6 (2020), pp. 3339–3352. ISSN: 0924-669X (cit. on p. 7).

[17] Georgios Petasis, Dimitris Spiliotopoulos, Nikos Tsirakis, and P. Tsantilas. «Large-scale Sentiment Analysis for Reputation Management». In: Sept. 2013 (cit. on p. 7).

[18] Federico Neri, Carlo Aliprandi, Federico Capeci, Montse Cuadros, and Tomas By. «Sentiment Analysis on Social Media». In: Aug. 2012. DOI: 10.1109/ASONAM.2012.164 (cit. on p. 7).

[19] Thilageswari a/p Sinnasamy and Nilam Nur Amir Sjaif. «A Survey on Sentiment Analysis Approaches in e-Commerce». In: *International Journal of Advanced Computer Science and Applications* 12.10 (2021). DOI: 10.14569/IJACSA.2021.0121074. URL: http://dx.doi.org/10.14569/IJACSA.2021.0121074 (cit. on p. 7).

[20] Antony Samuels and John Mcgonical. *Sentiment Analysis on Customer Responses*. July 2020 (cit. on p. 7).

[21] Zhaoxia Wang, Chee Seng Chong, Landy Lan, Yinping Yang, Seng Ho, and Joo Tong. «Fine-grained sentiment analysis of social media with emotion sensing». In: Dec. 2016, pp. 1361–1364. DOI: 10.1109/FTC.2016.7821783 (cit. on p. 8).

[22] Bharat Gaind, Varun Syal, and Sneha Padgalwar. «Emotion Detection and Analysis on Social Media». In: *ArXiv* abs/1901.08458 (2019) (cit. on p. 8).

[23] Francisca Acheampong, Chen Wenyu, and Henry Nunoo-Mensah. *Text-Based Emotion Detection: Advances, Challenges and Opportunities*. Apr. 2020 (cit. on p. 8).

[24] Haoyue Liu, Ishani Chatterjee, Mengchu Zhou, Sean Lu, and Abdullah Abusorrah. «Aspect-Based Sentiment Analysis: A Survey of Deep Learning Methods». In: *IEEE Transactions on Computational Social Systems* PP (Nov. 2020), pp. 1–18. DOI: 10.1109/TCSS.2020.3033302 (cit. on p. 8).

[25] R. Sunitha Syam Mohan E. «Survey On Aspect Based Sentiment Analysis Using Machine Learning Techniques». In: *European Journal of Molecular amp; Clinical Medicine* 7.10 (2021), pp. 1664–1684. ISSN: 2515-8260. eprint: https://ejmcm.com/article__fc9dd6d90bfe72a036f8a9ea0ede617e6773.pdf. URL: https://ejmcm.com/article_6773.html (cit. on p. 8).

[26] Souvic Chakraborty, Pawan Goyal, and Animesh Mukherjee. «Aspect-based Sentiment Analysis of Scientific Reviews». In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020* (Aug. 2020). DOI: 10.1145/3383583.3398541. URL: http://dx.doi.org/10.1145/3383583.3398541 (cit. on p. 8).

[27] Say Hong Lye and Phoey Lee Teh. «Customer Intent Prediction using Sentiment Analysis Techniques». In: *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1. 2021, pp. 185–190. DOI: 10.1109/IDAACS53288.2021.9660391 (cit. on p. 8).

[28] Cohan Sujay Carlos and Madhulika Yalamanchi. «Intention Analysis for Sales, Marketing and Customer Service». In: *Proceedings of COLING 2012: Demonstration Papers*. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 33–40. URL: https://aclanthology.org/C12-3005 (cit. on p. 8).

[29] C.J. Hutto and Eric Gilbert. «VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text». In: Jan. 2015 (cit. on p. 9).

[30] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. arXiv: cs/0205028 [cs.CL] (cit. on pp. 10, 30).

[31] Vikramkumar, B Vijaykumar, and Trilochan. «Bayes and Naive Bayes Classifier». In: *ArXiv* abs/1404.0933 (2014) (cit. on p. 12).

[32] Moo Chung. *Introduction to logistic regression*. Aug. 2020 (cit. on p. 12).

[33] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. «A Training Algorithm for Optimal Margin Classifiers». In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. URL: https://doi.org/10.1145/130385.130401 (cit. on p. 12).

[34] Gwanghoon Yoo and Jeesun Nam. «A Hybrid Approach to Sentiment Analysis Enhanced by Sentiment Lexicons and Polarity Shifting Devices». In: *The 13th Workshop on Asian Language Resources*. Ed. by Kiyoaki Shirai. The 13th Workshop on Asian Language Resources. Kiyoaki Shirai. Miyazaki, Japan, May 2018, pp. 21–28. URL: https://hal.archives-ouvertes.fr/hal-01795217 (cit. on p. 23).

[35] Massimo La Rosa, Antonino Fiannaca, Riccardo Rizzo, and Alfonso Urso. «Probabilistic topic modeling for the analysis and classification of genomic sequences». In: *BMC Bioinformatics* 16 (Apr. 2015), S2. DOI: 10.1186/1471-2105-16-S6-S2 (cit. on p. 24).

[36] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. «An overview of topic modeling and its current applications in bioinformatics». In: *SpringerPlus* 5 (Sept. 2016). DOI: 10.1186/s40064-016-3252-8 (cit. on p. 24).

[37] Maarten Grootendorst. *Concept*. Nov. 2021. URL: https://towardsdatascience.com/topic-modeling-on-images-why-not-aad331d03246 (cit. on p. 24).

[38] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. «Latent dirichlet allocation». In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://portal.acm.org/citation.cfm?id=944937 (cit. on p. 24).

[39] Alexis Conneau et al. «Unsupervised Cross-lingual Representation Learning at Scale». In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747 (cit. on p. 29).

[40] Fabian Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (Jan. 2012) (cit. on p. 32).

[41] Peter Willett. «The Porter stemming algorithm: Then and now». In: *Program electronic library and information systems* 40 (July 2006). DOI: 10.1108/00330330610681295 (cit. on p. 33).

[42] Martin F. Porter. *Snowball: A language for stemming algorithms*. Published online. Accessed 11.03.2008, 15.00h. Oct. 2001. URL: http://snowball.tartarus.org/texts/introduction.html (cit. on p. 33).

[43] Chris D. Paice. «Another Stemmer.» In: *SIGIR Forum* 24.3 (1990), pp. 56–61. URL: http://dblp.uni-trier.de/db/journals/sigir/sigir24.html#Paice90 (cit. on p. 33).

[44] A. Cognolato G. Attanasio. *L3 Python Wrapper*. https://github.com/g8a9/l3wrapper. 2020 (cit. on p. 34).

[45] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. «Understanding bag-of-words model: A statistical framework». In: *International Journal of Machine Learning and Cybernetics* 1 (Dec. 2010), pp. 43–52. DOI: 10.1007/s13042-010-0001-0 (cit. on p. 34).

[46] Radim Rehurek and Petr Sojka. «Gensim–python framework for vector space modelling». In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011) (cit. on p. 37).

[47] Matthew Hoffman, Francis Bach, and David Blei. «Online Learning for Latent Dirichlet Allocation». In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper/2010/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf (cit. on p. 38).

[48] Michael Röder, Andreas Both, and Alexander Hinneburg. «Exploring the Space of Topic Coherence Measures». In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 399–408. ISBN: 9781450333177. DOI: 10.1145/2684822.2685324. URL: https://doi.org/10.1145/2684822.2685324 (cit. on p. 39).

[49] Carson Sievert and Kenneth Shirley. «LDAvis: A method for visualizing and interpreting topics». In: June 2014. DOI: 10.13140/2.1.1394.3043 (cit. on p. 39).

[50] Ekaba Bisong. «Google Colaboratory». In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, pp. 59–64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7 (cit. on p. 45).

[51] Liangjie Hong and Brian D. Davison. «Empirical Study of Topic Modeling in Twitter». In: *Proceedings of the First Workshop on Social Media Analytics*. SOMA '10. Washington D.C., District of Columbia: Association for Computing Machinery, 2010, pp. 80–88. ISBN: 9781450302173. DOI: 10.1145/1964858.1964870. URL: https://doi.org/10.1145/1964858.1964870 (cit. on p. 45).

[52] Benjamin Bengfort and Rebecca Bilbro. «Yellowbrick: Visualizing the Scikit-Learn Model Selection Process». In: *Journal of Open Source Software* 4.35 (2019), p. 1075. DOI: 10.21105/joss.01075. URL: https://doi.org/10.21105/joss.01075 (cit. on p. 46).

[53] Shahzad Qaiser and Ramsha Ali. «Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents». In: *International Journal of Computer Applications* 181 (July 2018). DOI: 10.5120/ijca2018917395 (cit. on p. 47).

[54] «TF–IDF». In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 986–987. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_832. URL: https://doi.org/10.1007/978-0-387-30164-8_832 (cit. on p. 47).

[55] Michael Maschler, Eilon Solan, and Shmuel Zamir. «The Shapley value». In: *Game Theory*. Cambridge University Press, 2013, pp. 748–781. DOI: 10.1017/CBO9780511794216.019 (cit. on p. 72).

[56] Scott M Lundberg and Su-In Lee. «A Unified Approach to Interpreting Model Predictions». In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf (cit. on p. 72).

[57] *Shap: A game theoretix approach to explain the output of any machine learning model*. 2018. URL: https://github.com/slundberg/shap (cit. on pp. 72, 75).

[58] Scott M. Lundberg et al. *Explainable AI for Trees: From Local Explanations to Global Understanding*. 2019. DOI: 10.48550/ARXIV.1905.04610. URL: https://arxiv.org/abs/1905.04610 (cit. on pp. 72, 73).

[59] Towards Data Science. *Explain Any Models with the SHAP Values*. URL: https://towardsdatascience.com/explain-any-models-with-the-shap-values-use-the-kernelexplainer-79de9464897a (cit. on p. 74).

[60] Scott M Lundberg and Su-In Lee. «A Unified Approach to Interpreting Model Predictions». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf (cit. on p. 75).

[61] Towards Data Science. *Explain your Models with the SHAP Values*. URL: https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d (cit. on p. 76).

[62] Henrik Gudmundsson, Ralph P. Hall, Greg Marsden, and Josias Zietsman. «Sustainable Transportation - Indicators, Frameworks, and Performance Management». English. In: Springer Texts in Business and Economics. Springer, 2015. ISBN: 978-3-662-46923-1. DOI: 10.1007/978-3-662-46924-8 (cit. on p. 81).

[63] Todd Litman and David G. Burwell. «Issues in sustainable transportation». In: *International Journal of Global Environmental Issues* 6 (2006), pp. 331–347 (cit. on p. 81).

[64] Zhikui Chen, Shan Jin, Runze Liu, and Jianing Zhang. «A Deep Non-negative Matrix Factorization Model for Big Data Representation Learning». In: *Frontiers in Neurorobotics* 15 (2021). ISSN: 1662-5218. DOI: 10.3389/fnbot.2021.701194. URL: https://www.frontiersin.org/article/10.3389/fnbot.2021.701194 (cit. on p. 83).