Explaining Sentiment Prediction by Generating Exemplars in the Latent Space

B.S. Thesis

Candidate
Orestis Panagiotis Lampridis

Supervisors
Riccardo Guidotti
Salvatore Ruggieri
Grigoris Tsoumakas

September 2019
Explaining Sentiment Prediction by Generating Exemplars in the Latent Space

Δημιουργία Υποδειγμάτων στον Λανθάνοντα Χώρο για την Επεξήγηση Προβλέψεων Συναισθήματος

Orestis Panagiotis Lampridis

10th September 2019
Abstract

In this thesis, we present an approach to explain the decisions of black box models for text classification. To this end, we build on the Local Interpretable Model-Agnostic Explanations (LIME) algorithm, which is a standard reference in the literature. LIME works by generating a neighborhood of samples around an instance for which we require an explanation. When it comes to text classification, LIME is only able to generate neighbors by randomly removing words or bi-grams from a given text. As a result, the synthetic data generated may not result into semantically meaningful textual sentences. In order to improve the generated neighborhood, we use a variational auto encoder that is able to generate neighbors of the initial instance which are close to it semantically, by randomly tweaking the latent space. Then we use these sentences along with their respective labels taken by the respective black box model in order to train a decision tree classifier from which we can then extract the exemplars as a final explanation. We conduct an experimental evaluation of the approach on two datasets and two black box models. Experimental results demonstrate that our proposed approach is able to produce accurate explanations.

Keywords: Black Box, Machine Learning, Interpretability, Explanations, Sentiment Prediction
Περίληψη

Στην παρούσα πτυχιακή εργασία παρουσιάζουμε μία τεχνική για την επεξήγηση των αποφάσεων αδιαφανών μοντέλων μηχανικής μάθησης (μαύρων κουτιών) σε προβλήματα κατηγοριοποίησης δεδομένων κειμένου. Για τον σκοπό αυτό, χρησιμοποιήθηκε ο αλγόριθμος Local Interpretable Model-Agnostic Explanations (LIME). Ο αλγόριθμος LIME λειτουργεί δημιουργώντας μια γειτονιά δειγμάτων γύρω από ένα στιγμιότυπο για το οποίο χρειαζόμαστε μια εξήγηση. Όταν πρόκειται για κατηγοριοποίηση κειμένου, ο αλγόριθμος LIME είναι σε θέση να παράγει γείτονες μόνο διαγράφοντας τυχαία λέξεις ή συνδυασμούς λέξεων από ένα κείμενο. Ως αποτέλεσμα, τα συνθετικά δεδομένα που παράγονται δεν αντιπροσωπεύουν πραγματικά δεδομένα. Προκειμένου να βελτιωθούν οι παραγόμενες γείτονες, εισάγουμε ένα variational auto encoder το οποίο είναι σε θέση να παράγει γείτονες του αρχικού στιγμιότυπου, οι οποίοι είναι σημασιολογικά κοντά σε αυτό, αλλάζοντας τυχαία τον λανθάνοντα χώρο. Στη συνέχεια, χρησιμοποιούμε αυτές τις προτάσεις μαζί με τις αντίστοιχες ετικέτες που έχουν ληφθεί από το αντίστοιχο μοντέλο μαύρου κουτιού, για να εκπαιδεύσουμε ένα δέντρο απόφασης από το οποίο μπορούμε στη συνέχεια να λάβουμε επεξηγήσεις. Παρουσιάζουμε τα αποτελέσματα μιας πειραματικής αξιολόγησης σε δύο σύνολα δεδομένων και με δύο μαύρα κουτιά. Τα πειραματικά αποτελέσματα στα σύνολα δεδομένων δείχνουν ότι με το προτεινόμενο μας μοντέλο μπορούν να παραχθούν ακριβείς επεξηγήσεις.

Λέξεις Κλειδιά: Μαύρο Κουτί, Μηχανική Μάθηση, Ερμηνευσιμότητα, Επεξηγήσεις, Πρόβλεψη Συναισθήματος
Acknowledgements

Before presenting this work and its results, I would like to thank some of the people I met and worked with while working on my thesis.

I would like to express my gratitude to prof. Grigorios Tsoumakas for the trust he showed in me to undertake this unconventional project under his supervision, for helping me set it all up and for his overall assistance. Next, I would like to praise prof. Salvatore Ruggieri for accepting to be my co-supervisor in Italy, for making me feel at home from the first day I met with him and for the valuable guidance he gave me. Of course, I would like to thank my other co-supervisor Riccardo Guidotti for the endless stream of brilliant ideas that he had about the project. He always took the time to help me understand his vision and also to discuss any obstacles I faced. I would also like to acknowledge Ioannis Mollas for consistently being willing to help me and for happily giving me advice whenever and wherever I needed it. It was an honor to have collaborated with each and every one of these people. This work wouldn’t have been possible without them.

In addition, I would like to thank two PhD students that prof. Grigorios Tsoumakas connected me with. Namely, Alexios Gidiotis for helping me work out some initial problems with seq2seq models and Despina Christou for allowing me to run experiments on her machine. Naturally, I would also like to thank all the friends I had by my side during this academic journey both at home and abroad.

Lastly, I would like to thank my family. My parents, Dimitris and Elina, as well as my sister Lydia, for always being there for me and supporting me during my academic course.

Thank you all so much,

Orestis Lampridis
# Contents

1 **Introduction** 14  
  1.1 Topic Description .......................... 14  
  1.2 Contribution of the Thesis ......................... 15  
  1.3 Structure .................................. 15  

2 **Machine Learning** 17  
  2.1 Types of Data ................................ 18  
  2.2 Types of Methods ................................ 19  
  2.3 Algorithmic Approaches on Supervised Learning .... 20  
    2.3.1 K-Nearest Neighbors ....................... 20  
    2.3.2 Linear Models ............................. 21  
    2.3.3 Naive Bayes ................................ 23  
    2.3.4 Decision Trees .............................. 24  
    2.3.5 Decision Rules .............................. 24  
    2.3.6 Random Forests .............................. 26  
    2.3.7 Support Vector Machines ...................... 27  
    2.3.8 Neural Networks and Deep Learning .......... 28  
  2.4 Needs for Interpretability ......................... 30  

3 **Explainable Machine Learning** 32  
  3.1 Interpretable Models ............................ 32  
    3.1.1 Interpretability ......................... 33  
    3.1.2 Dimensions of Interpretability .......... 33  
    3.1.3 Desired Features of Interpretable Models .. 34  
    3.1.4 Comprehensibility and Complexity ........ 35  
    3.1.5 Types of Data for Interpretable Models .. 36  
  3.2 Recognized Interpretable Models .................. 36  
    3.2.1 Supervised Models ......................... 36  
    3.2.2 Unsupervised Models ....................... 36  
    3.2.3 Fuzzy Systems ............................. 37  
  3.3 Making Black Box Models Interpretable ........... 38  

8
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1 Reverse Engineering Analysis</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2 Design of Explanations</td>
<td>39</td>
</tr>
<tr>
<td>3.4 Explaining the Predictions of Any Classifier</td>
<td>39</td>
</tr>
<tr>
<td>3.4.1 LIME</td>
<td>39</td>
</tr>
<tr>
<td>3.4.2 Limitations of LIME on textual data</td>
<td>40</td>
</tr>
<tr>
<td>4 Design</td>
<td>42</td>
</tr>
<tr>
<td>4.1 Sequence to Sequence Learning</td>
<td>42</td>
</tr>
<tr>
<td>4.2 Variational Auto Encoders</td>
<td>43</td>
</tr>
<tr>
<td>4.3 General Architecture of X-SPELLS</td>
<td>44</td>
</tr>
<tr>
<td>4.4 Final Explanations</td>
<td>45</td>
</tr>
<tr>
<td>5 Implementation</td>
<td>47</td>
</tr>
<tr>
<td>5.1 Pre-processing of Datasets</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Black Box Models</td>
<td>48</td>
</tr>
<tr>
<td>5.2.1 Random Forest</td>
<td>48</td>
</tr>
<tr>
<td>5.2.2 Deep Neural Network</td>
<td>48</td>
</tr>
<tr>
<td>5.3 Variational Auto Encoder</td>
<td>49</td>
</tr>
<tr>
<td>5.4 Neighborhood Generation</td>
<td>51</td>
</tr>
<tr>
<td>5.4.1 From Sentences to Latent Space</td>
<td>51</td>
</tr>
<tr>
<td>5.4.2 Random Generation</td>
<td>51</td>
</tr>
<tr>
<td>5.4.3 Decoding back to Original Feature Space</td>
<td>53</td>
</tr>
<tr>
<td>5.5 Labeling and Class Balancing</td>
<td>53</td>
</tr>
<tr>
<td>5.6 Decision Tree Learning</td>
<td>55</td>
</tr>
<tr>
<td>5.7 Extraction of Exemplars</td>
<td>55</td>
</tr>
<tr>
<td>6 Evaluation</td>
<td>57</td>
</tr>
<tr>
<td>6.1 Datasets for Experimentation</td>
<td>57</td>
</tr>
<tr>
<td>6.2 Predictive Performances of Black Box Models</td>
<td>58</td>
</tr>
<tr>
<td>6.2.1 Performances of Random Forest</td>
<td>59</td>
</tr>
<tr>
<td>6.2.2 Performances of DNN</td>
<td>60</td>
</tr>
<tr>
<td>6.3 Explanations using LIME</td>
<td>61</td>
</tr>
<tr>
<td>6.4 LIME vs X-SPELLS: Quantitative Evaluation</td>
<td>64</td>
</tr>
<tr>
<td>6.5 LIME vs X-SPELLS: Qualitative Evaluation</td>
<td>65</td>
</tr>
<tr>
<td>7 Conclusions</td>
<td>69</td>
</tr>
<tr>
<td>7.1 Future Work</td>
<td>69</td>
</tr>
</tbody>
</table>
## List of Figures

2.1 Example of k-NN classification ........................................... 21
2.2 Example of Linear Regression ............................................ 22
2.3 Another example of a decision tree that visualizes the problem on whether we should lend someone a car or not. .................... 25
2.4 Example of Random Forest classification. ............................. 26
2.5 Example of a linear SVM .................................................. 27
2.6 Example of non-linear SVM classification ............................. 28
2.7 Example of a simple Neural Network ................................. 29

3.1 Explaining individual predictions to a human who makes decisions. Source: Marco Tulio Ribeiro [1]. ................................. 32
3.2 Visualization of the difference between boolean logic and fuzzy logic. ................................................................. 37
3.3 Structure of a fuzzy rule-based system. Fuzzy rule base contains the fuzzy rules along with the membership functions. ............. 38
3.4 How LIME is able to locally approximate a given record, shown as the red bold cross in this figure, and then generate synthetic neighbors. .................................................. 40

4.1 The three parts of the seq2seq model. The encoder, the encoder vector, also referred to as the state vector and finally the decoder. 43
4.2 The parts that constitute a VAE model when tasked at reconstructing movie reviews. ..................................................... 44
4.3 Architecture of X-SPELLS. .................................................. 45

5.1 Example problem solved by using SMOTE [2]. ....................... 54

6.1 The explanation given by LIME on the 6th instance of the hate tweet dataset when explaining the predictions of the random forest on the top and the DNN on the bottom. ....................... 62
6.2  The explanation given by LIME on the 107th instance of the polarity dataset when explaining the predictions of the random forest on the top and the DNN on the bottom. . . . . . . . . . . . . . . . 63
List of Tables

4.1 Example of an explanation when given input sentences using X-SPELLS. ......................................................... 46
6.1 The classification report of the random forest classifier for the hate tweet dataset. ...................................................... 60
6.2 The classification report of the random forest classifier for the polarity dataset. ....................................................... 60
6.3 The classification report of the DNN classifier for the hate tweet dataset. ................................................................. 60
6.4 The classification report of the DNN classifier for the polarity dataset. ................................................................. 61
6.5 Fidelity for each dataset when using LIME and X-SPELLS to explain the random forest. .............................................. 64
6.6 Fidelity for each dataset when using LIME and X-SPELLS to explain the DNN. .............................................................. 64
6.7 The explanation when given input sentences from the hate tweets dataset using X-SPELLS. .......................................... 66
6.8 The explanation when given input sentences from the hate tweets dataset using LIME. .............................................. 66
6.9 The explanation when given input sentences from the polarity dataset using X-SPELLS. .................................................. 67
6.10 The explanation when given input sentences from the polarity dataset using LIME. .................................................. 67
Chapter 1

Introduction

1.1 Topic Description

The recent years have witnessed a rapid increase in the use of machine learning models in a wide range of application fields, including businesses, self-driving cars, medicine, public policy, and many others. A large part of those machine learning models are black boxes, i.e., their overall functioning and the logic behind their decisions for a given input instance are not clearly understandable to humans.

In sensitive fields such as business decision support, where a potential wrong decision could lead to the loss of business capital, or in those of medicine and self-driving cars, where a potential wrong choice could lead to the death of a human being, the ability to know why a machine learning system made a certain decision is of grave importance and reinforces the idea that the decisions of these systems need to be known and be easily understood. This fact, along with recent European law regulations [3] have created the rising field in computer science called explainable machine learning. The subject of this field is to make the decision logic of black boxes understandable to humans.

Significant academic progress has been made in the last few years in the development of various techniques to explain the predictions of black box models given in input an instance (a tuple, a sentence, an image, etc.). The class of approaches that take advantage of black box behavior in the neighborhood of the instance is known as local explainers, and it is becoming the most used and studied in the field. One the most popular approaches belonging to this class is the Local Interpretable Model-Agnostic Explanations (LIME) [1] system. LIME generates a neighborhood of random samples around an instance for which an explanation is required, then it queries the black box on the instances in the neighborhood, and finally it builds an interpretable classifier (a linear model) out of them. The linear model provides an explanation of which attributes of the instance weigh the most
in the decision logic of the black box.

1.2 Contribution of the Thesis

In this thesis, we propose an original local explainer that improves over LIME for the specific input of textual data called X-SPELLS: eXplaining Sentiment Prediction by generating exempLars in the Latent Space. The improvement over LIME consists of generating better neighborhoods. Another difference is that X-SPELLS relies on decision tree classifiers for generating explanations. Experimental validation is performed for the task of sentiment prediction.

The way we approach the improved generation of neighborhoods, is by creating neighbors which are semantically closer to the instance we want to explain by using a variational auto encoder. Firstly, we encode the instance into its latent space representation. Then, by randomly tweaking this vector several times and decoding each of these newly created vectors, we end up with new neighbors which are semantically close to our initial instance.

Afterwards, we train a decision tree on this neighborhood with the labels for each neighbor assigned from the respective black box model we want to explain. After training the decision tree, we find the leaf in which the initial instance is located and then we can find the exemplars, which are the neighbors which were classified as belonging to the same leaf. From this set of exemplars, we take the most common set of words. The final explanation given consists of a number of randomly chosen exemplars along with the most common words found in them.

1.3 Structure

This thesis is structured into the following chapters:

- **Chapter 2 - Machine Learning.** In this chapter and also in the next, we will describe the scientific and knowledge basis and background behind the field of machine learning which will be necessary to develop a foundation that will aid in the understanding of the solution that we propose. In particular, we will describe the basic concepts of the field of Machine Learning and analyze the most used algorithmic approaches used. In the end of this chapter, we will describe why there are needs for interpretability in our current age, which will allow us to set the stage for the next chapter.

- **Chapter 3 - Explainable Machine Learning.** Similarly to the previous chapter, here we will describe the basic concepts in the recent rising scientific field of explainable machine learning. In the end, we will describe
a technique called LIME, which is used to explain the predictions of any classifiers and we will detect a problem that exists with this approach when it is used to explain classifiers on textual data.

• **Chapter 4 - Design.** In this chapter, after introducing several key concepts, we will describe the general layout of our approach to explain the predictions of classifiers on textual data in order to combat the aforementioned problem that LIME faces.

• **Chapter 5 - Implementation.** Here, we will build on top of everything we introduced in the previous chapter and get into more technical details regarding the implementation of our approach, along with the implementation of the black box models that were used in our research.

• **Chapter 6 - Evaluation.** In this chapter, firstly, we will present our datasets and then we will evaluate our black box models. Afterwards, we will showcase how LIME works to produce explanations. Lastly, we will present our final implementation, and we will compare the accuracy against LIME.

• **Chapter 7 - Conclusions.** Finally, in this chapter, we will present the conclusions from our evaluation of the implementation and we will suggest possible future improvements and future research based on our approach.
Chapter 2

Machine Learning

Machine learning, sometimes referred to as predictive analytics or statistical learning, is a sub-field of artificial intelligence, at the crossroads of other fields such as computational statistics and computer science. It differs however from traditional computational approaches. In traditional programming, algorithms are sets of explicitly programmed instructions in order to allow the computer to solve specific problems. Machine learning algorithms instead, allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning allows computers to construct models from sample data in order to automate decision-making processes based on data inputs.

The term machine learning was originally coined by Arthur Samuel in 1959 who first described it as sub-field of computer science that gives computers the ability to learn without being explicitly programmed [4]. Following is the most used mathematical definition, which was given by Tom Mitchell. “A machine is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E” [5].

Applications of machine learning today are so indented into reality, it is difficult to grasp the full impact of them in everyday life activities. Almost any technology user today has benefited in some way from machine learning. A few use cases include automatic recommendations catered to each user’s personal preference. For example which products to buy (e.g. on Amazon), which movies to watch (e.g. on Netflix), which songs to listen to (e.g. on Spotify). Another example is the recognition of a user’s friends in the photos they upload on Facebook. All of the above websites contain various different machine learning models that enable them to do the aforementioned tasks.

Machine learning is also used to solve several problems that exist in the scientific community. For example, analyzing DNA sequences, finding new person-
alized ways to treat diseases such as cancer, discovering distant planets, stars and even new particles. It is also used to push technological advances forward. Self driving cars were thought to be an idea straight out of science fiction movies. However, using machine learning techniques, autonomous cars have become a reality and may soon be available in the consumer market.

In the following sub-sections, we will delve into what machine learning is about by first looking into some problems that machine learning algorithms can solve and then by analyzing the most common methods used to solve each one of them. Afterwards, we will see a few general algorithmic approaches to solve problems in the field of machine learning, first by looking into the most simple approaches and then build up into the more complex ones. A few of the most well-known methods that we will analyze are k-nearest neighbors, decision trees, support vector machines and neural networks. Finally, we will discuss why there are needs for interpretable models in machine learning.

### 2.1 Types of Data

Before exploring the different types of methods used in machine learning we need to have an understanding of the different data types available from a machine learning perspective. Most data types are usually categorized into the following basic types: numerical data, categorical data, boolean data, image data and text data. The most used types of data used as input for interpretable models are the following:

- **Tabular data.** These kinds of data consist of datasets in which each feature is either numerical, categorical or boolean.

- **Image data.** These kinds of data consist of datasets that basically contain images. Many machine learning models work with labeled images. Generally, they are used for tasks like image recognition.

- **Textual data.** With textual data, we indicate datasets that basically consist of phrases or words. Usually, to use text data we turn it into some function by using a representation (e.g. the bag-of-words model). In most cases, they are used for tasks like hate speech detection, spam detection or topic classification.

In machine learning classification problems, other types of data are also used like time series data, sequences, networks and a few others. However, types of data different from tabular, images and text are not generally used as input for potential interpretable models that are proposed in various papers in the state of the art [6].
In this work, we are going to focus on textual data and more specifically, textual data used for hate speech detection and polarity classification.

### 2.2 Types of Methods

Some problems on which machine learning algorithms can be applied on:

- Identifying which digit is written from handwritten digits on a piece of paper. In this problem the input is a scanned image of the handwritten digit. The output is the actual digit. To create the dataset needed for this machine learning model, one needs to collect many pieces of paper, read the digits, identify them and store the desired outcome.

- Identifying topics in a set of blog posts. In this problem the input would be large collections of text data. How many topics there are or what these topics are about will not be known ahead of time. As a result, there are no known outputs for this problem.

- A computer learning how to play a game against a human opponent.

By inspecting the above examples we can easily understand that there are differences between them. In the first example, there is a specific input and output. In the second example, there is a specific input but not a specific outcome. In the third example, there is neither an input nor an output. There are many different algorithms used in machine learning that differ in the way they deal with a given task. They fall into three main categories depending on the type of data that are used for input, the output they produce and the type of task that they are intended to solve. These categories are classified as:

- **Supervised Learning.** The first example falls into this category. There is a dataset for which the outcome of interest is already known. The aim then is to predict the outcome for new unseen data. Furthermore, in supervised learning, the task is called *classification* if the output is categorical such as “black” or “white” or “spam” and “no spam” and *regression* if the output is numerical such as “Euros” or “height”.

- **Unsupervised Learning.** The second example falls into this category. Also called *clustering* and *association*, in this method of machine learning there is no specific outcome of interest but only clusters of data that are needed to be found. In essence there are no labels in the dataset.
• **Reinforcement Learning.** The final example is an example of reinforcement learning and is about trying to maximize a reward by taking a set of actions in an environment. It differs from the above methods in the sense that there is no training data given as input that contains the answer key so the model must decide what to do in a given task.

There is also a category classified as **Semi-supervised Learning** algorithms. These are algorithms that fall somewhere between supervised and unsupervised learning. They work by using both labeled and unlabeled data for training. Usually a small amount of them consists of labeled data and a larger amount of unlabeled data. Systems that use this method can considerably improve learning accuracy.

### 2.3 Algorithmic Approaches on Supervised Learning

Before diving into the most common algorithmic approaches used in machine learning, it can be helpful to first define and understand what the terms regression and correlation are from the field of statistics. They are commonly used techniques for investigating the relationship between quantitative variables. Regression describes how an independent variable is numerically related to a dependent variable. Regression analysis can be used to anticipate the dependent variable when the independent variable is known and thus it has prediction capabilities. Different regression techniques may differ based on the number of independent variables and the type of relationship between them. Correlation is a measure of association used to represent the linear relationship between two variables that are not specified as dependent or independent.

New approaches in machine learning are continuously under development. We will go through a few of the most popular approaches that are being used at the time of writing.

#### 2.3.1 K-Nearest Neighbors

K-Nearest Neighbors algorithm is a pattern recognition algorithm that can be used for both classification as well as regression purposes. Often abbreviated as k-NN, the k in k-Nearest Neighbors is a positive and usually small integer number which shows how many neighbors or closest data points from the dataset will be used for the prediction. To make a prediction for a new data point, the algorithm finds the nearest neighbors of the new data point.

In its simplest version, the k-NN algorithm only considers one nearest neighbor, which is the closest training data point for the point which we want to make
a prediction about. The prediction is then simply the already known output of that training point.

Let’s look at an example of k-NN. In the diagram below (Figure 2.1), there are blue square objects and red triangle objects. These belong to two separate classes: the blue square class and the red triangle class. If $k = 3$, we need to look to the closest 3 neighbors, which is shown in the diagram as a solid line circle. We can see that our new object is assigned to the red triangle class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$, we need to look to the closest 5 neighbors, which is shown in the diagram as a dashed line circle. This time it is assigned to the blue square class, as there 3 squares and 2 triangles inside the dashed circle.

![Figure 2.1: Example of k-NN classification](image)

While k-NN algorithm is very simple, it is generally not used in practice, as the prediction is slow for large datasets and it is unable to perform well on datasets with many features. Also it is not easy to transfer a k-NN model (e.g. if we try to use it on a different machine). In order to make a prediction the k-NN model needs to have access to the whole dataset in contrast with other models, that we will view later on, which just output a function. However, it’s a good method to try before moving into more advanced techniques.

### 2.3.2 Linear Models

Linear models are used to make a prediction using a linear function of the input features. For regression, the general equation for a linear model is:

$$y = w_0 * x_0 + w_1 * x_1 + ... + w_p * x_p + b$$
Here, $x_0$ to $x_p$ denotes the number of features of a single data point, while $w$ and $b$ are parameters of the model that are learned. The output of the linear model algorithm is given by $y$. Another way to see the output $y$ is as being a weighted sum of the input features, with weights given by the entries of $w$.

There are many different linear models used for regression. The difference between them is how the model parameters $w$ and $b$ are calculated, based on the given data points. The most popular method is called linear regression and can be seen in the figure below.

![Figure 2.2: Example of Linear Regression](image)

Linear regression, also known as ordinary least squares, is a type of regression model where the number of features is only one. The purpose of linear regression is to find the red line in the above graph which can also be referred to as the best fit straight line. The line can be modelled based on the linear equation shown below.

$$y = w_0 x_0 + b$$

Here, $w_0$ is the slope and $b$ is the intercept (y-axis offset). By using least square minimization, we are able to find the values $w_0$ and $b$ that plot the line that best fits the given data points. The way these values are found is by minimizing the mean squared error between predictions and the true regression targets, $y$, on the training set. The mean squared error is the sum of the squared differences between the predictions and the true values [7].

$$y = w_0 x_0 + w_1 x_1 + \ldots + w_p x_p + b > 0$$

Linear models are also widely used for classification purposes. Instead of returning the weighted sum of attributes, as in linear models used for regression,
here we set the predicted value to zero. If the function returns a value less than zero, then we predict that it belongs to class -1. Likewise, if it is greater than zero, we predict it belongs to class +1. This general prediction rule is common to all linear models for classification. However, there are many different ways to find the parameters $w$ and $b$.

There are many different algorithms used for learning linear models which differ with regards to how they measure how well a particular combination of parameters $w$ and $b$ fit the training data and with what kind of regularization they use. Two of the most commonly used algorithms are logistic regression and linear support vector machines.

### 2.3.3 Naive Bayes

Naive bayes (NB) models are classifiers that have a lot in common with linear models but tend to be even faster in training. The reason why naive bayes models are so effective is that they learn the parameters from looking at each feature separately and collect simple statistics by category from each feature. Naive Bayes consist of three main models.

- **Gaussian Naive Bayes**, in short GaussianNB, applied to any consistent and mostly very high-dimensional data.
- **Bernoulli Naive Bayes**, in short BernoulliNB, applied to binary data.
- **Multinomial Naive Bayes**, in short MultinomialNB applied to measurement data.

BernoulliNB and MultinomialNB are mainly used for the classification of text data. The BernoulliNB classifier measures how often each attribute of each class is not zero. GaussianNB and MultinomialNB, on the other hand, differ in the type of statistics they calculate. GaussianNB stores the average value as well as the standard deviation of each attribute for each category. MultinomialNB takes account of the average of each attribute for each category. To make a prediction, a data point is compared to the statistics for each of the categories and the most appropriate class is provided.

Naive bayes models share many of the strengths and weaknesses of linear models. They are very quick to train and the training process is easy to understand. The models also work very well with sparse data and with data of high dimensions. Naive Bayes models are often used in very large data sets, where linear models would require a very long time to train.
2.3.4 Decision Trees

Decision trees are models that support decisions by using a tree-like models of each decision and their possible consequences. They can be used for both regression and classification purposes.

To better understand decision trees we will consider an example. The problem we are looking at is the task of distinguishing an instance between four different fruits: pineapples, bananas, kiwi and strawberries. Our goal is to get the right answer by asking the fewest possible if/else questions. In our scenario, we could first ask whether the fruit is tropical. If the answer is “yes” we are left with two fruits, pineapples and bananas. Then we could ask another question to distinguish between those two. For example, a question could be whether the fruit is yellow or not. If the answer to whether the fruit is tropical was “no”, we can ask the question whether the fruit is red to distinguish between kiwis and strawberries. These series of questions can be expressed as a decision tree.

In the tree constructed in the above example, each node either represents a question on an attribute (e.g. whether a coin flips heads or tails) or a terminal node that contains the answer to that question. The paths from the root to the leaf represent the classification rules.

In machine learning terms, we essentially build a model to distinguish between four classes of fruits (pineapples, bananas, kiwis and strawberries) using the three features “is tropical”, “is yellow” and “is red”. Instead of building these models by hand, we can learn them from data by using supervised learning. In Figure 2.3 we can see a visualized example of a decision tree.

2.3.5 Decision Rules

Decision rules, also known as rule based systems, are functions that map observations to appropriate actions. Decision rules consist of simple IF-THEN statements made up of a condition and a prediction. Let’s consider an example of a simple decision rule. IF it snows AND IF it is winter, THEN it will snow tomorrow. The first part of our decision rule is the condition and the second part is the prediction. In reality we can use a single decision rule or a combination of several rules to make predictions.

Decision rules generally follow the structure: IF (condition) is true THEN make a prediction. Decision rules are probably considered as the most interpretable machine learning models because their structures linguistically resemble natural language and the way in which people think. Provided of course, that the condition is built from well-understood features and that the length of the condition is relatively short, meaning that there are not too many rules. In the present time, Machine learning systems can also learn decision rules through algorithms.
As already mentioned, the most common decision rules are of the form: IF (condition) is true THEN make a prediction. Decision rules of this form can also be made by a number of negations, conjunctions and disjunctions. Other types of decision rules stated in the literature are:

- **M-of-n Rules.** These are rules that given a set of n conditions, if m of them are confirmed, then the consistency of the rule is considered to be true.

- **List of Rules.** Given an ordered set of rules which are considered to be true, the consequent of the first rule is verified.

- **Falling Rule Lists.** These are rules that consist of if-then rules ordered by the probability of a specific outcome. The order signifies the example to be classified by the rule.

- **Decision Sets.** In decision sets an unordered set of classification rules is defined so that the rules are not connected by else statements. This way, each rule serves as an independent classifier, without taking into account other rules. Decision sets are thought to be one the most human understandable explanations that an interpretable system can output [8].
2.3.6 Random Forests

Random forests are supervised learning models, that work by building multiple decision trees and then merging them to get more accurate and stable predictions. Their biggest advantages are their ease of use and the quality of the results that they can produce, even without any significant parameter tuning. They can also be used for both classification and regression problems. Another advantage is that they inject randomness. For these reasons, they are one of the most used machine learning algorithms.

To have a better intuition of how random forests work we will consider a real life scenario. Let’s say we want to decide on a specific location to spend our vacation in the summer. In order to find the best place, we ask various people for advice. Initially, we will ask a friend who asks us where we have traveled to in the past and if we liked it or not. Based on the answers we have provided, our friend will give us advice. In essence, this is the way the decision tree algorithms work. The friend created rules to help guide his decision about what he should recommend, by taking into consideration the answers that we have given him.

Afterwards, we ask more and more friends to advise us and each time the friends ask us different questions, from which they can derive some recommendations for us. In the end, we will choose the places that where recommend to us the most. This whole process can be seen at Figure 2.4 and is how the random forest algorithm typically works.

Another great quality of the random forest algorithm is feature importance. It is very easy to measure the relative importance of each feature on the prediction of
the classifier. This is important, because a general rule in machine learning is that the more features we have, the more likely it is that our model will start to over-fit. By using feature importance, we can decide which features we may want to drop, in the case that they don’t contribute enough to the final prediction.

2.3.7 Support Vector Machines

Support vector machines, commonly referred to with their abbreviation as SVMs and also sometimes support vector networks, are supervised learning models that are used mostly for classification problems but sometimes also for regression analysis.

With SVMs, we plot each data item as a point in n-dimensional space (where n is the number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the optimal hyper-plane that divides the two classes with a clear gap, that is as wide as possible. New data is then mapped into the same space and is predicted to belong to one category depending on which side of the hyper-plane they fall on. An example of this is shown in the diagram below.

![Figure 2.5: Example of a linear SVM](image)

SVMs can also perform non-linear classification. An example of the way this can be accomplished can be observed in Figure 2.6. In the first diagram, we cannot draw a linear boundary. For this reason, we will add a third dimension and call it Z. This will lead us to the second diagram, in a three dimensional space. In this perspective, the data is now in two linearly separated groups. The hyper-plane, shown in the third diagram, will be the plane parallel to the X axis at a certain Z point. We pick the hyper-plane that will maximize the margin between the two classes of
data. Finally, as shown in the fourth diagram, by mapping back to two dimensions, we can see that the hyper-plane has become a circle that clearly separates our data.

Figure 2.6: Example of non-linear SVM classification

2.3.8 Neural Networks and Deep Learning

Artificial neural networks, more commonly known as simply neural networks and in short NNs, are computing systems that consist of a collection of connected units or nodes called artificial neurons. They are vaguely inspired by the biological neural networks that are found in human brains.

The signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some linear or non-linear function of the sum of its inputs. The connections between artificial neurons are called “edges”. Artificial neurons and edges typically have a weight that adjusts as the
learning algorithm proceeds. The weight increases or decreases in a way so that
the desired outcome can be achieved. This modification of weights constitutes the
traditional method of implementation of NNs and it has been applied successfully
in various fields. However, a NN also has the ability to modify its topology much
like in humans, where brain neuron can die or new synapses between them can
form.

Typically, NNs are aggregated into layers. Different layers may perform dif-
ferent kinds of transformations on their inputs. The output of one layer serves as
the input of the successive layer. Signals travel from the first layer, also called the
input layer, to the last layer, also called the output layer, possibly after traversing
the layers multiple times.

![Figure 2.7: Example of a simple Neural Network](image)

In the 1980s, most NNs were a single layer, as shown in Figure 2.7, due to the
high costs of computation. Nowadays we can afford to have more hidden layers.
These neural networks are called Deep Neural Networks (DNNs). Another way to
refer to DNNs is with the term Deep Learning.

In deep learning, algorithms can be either supervised and perform data clas-
sification, or unsupervised and perform pattern analysis. Computer vision and
speech recognition have both realized significant advances from Deep Learning
approaches. Using deep learning to classify and label images isn’t only better than
any other machine learning algorithms but it’s also starting to be better than actual
humans.

An example already mentioned in the introduction that uses deep learning is
the recognition of faces in photographs on Facebook, which has been successfully
CHAPTER 2. MACHINE LEARNING

worked out by using deep learning. “Asked whether two unfamiliar photos of faces show the same person, a human being will get it right 97.53 percent of the time. New software developed by researchers at Facebook can score 97.25 percent on the same challenge, regardless of variations in lighting or whether the person in the picture is directly facing the camera” [9].

2.4 Needs for Interpretability

So far, we have analyzed various different machine learning methods in order to solve different kinds of problems. Many of these models (e.g. random forests, SVMs, DNNs) are referred to as Black Boxes, in the sense that we can’t really inspect how the algorithm is accomplishing what it is accomplishing or why it made a certain decision. However, one could ask, if a machine learning model performs well, why not just trust the model and ignore why it made a certain decision?

In many cases, it is important to know why a machine learning model supports a particular decision. This is especially true in medicine and in autonomous vehicles, where a potentially wrong decision could very well lead to the death of human beings. As an example, in March of 2018 a self-driving Uber car struck and killed a pedestrian in Arizona [10]. However, because Uber cars don’t use interpretable models, no reasoning could be given as to why this decision was taken.

Also, the European Union recently introduced a new regulation called Right to Explanation in the General Data Protection Right (GDPR) as an attempt to deal with the potential problems stemming from the rising importance of machine learning algorithms. As a result, as of 2018 all European Union citizens are entitled to know why a decision was taken on their behalf (e.g. why their loan was rejected). For the above reasons, it is understandable that interpretability is not only important, but it is necessary for the advancement of machine learning as a trustworthy, responsible and open data science.

Finally, it should be noted that interpretable models are not always required. Interpretability is needed only if a machine learning model has an important impact, whether it is an economical (e.g. the loss of business capital) or a social (e.g. the death of a human being) one. For example, if someone wanted to build a deep learning model to classify fruits based on image data then there would be no actual problem if his model was wrong, nor would he owe anyone an explanation as to why a fruit was wrongly classified (e.g. a tomato was classified as a cucumber). However, if there was a business involved in the classification of the aforementioned fruits, then the business could lose money on a wrong decision and they would owe an explanation to a potential customer as to why this mistake in the classification happened. The fact that interpretability is not always needed is also important because, usually, when we try to make models interpretable we tend to
lose accuracy. Therefore, the goal is to find the golden ratio between interpretability and accuracy.
Chapter 3

Explainable Machine Learning

Explainable Machine Learning or Interpretable Machine Learning is a recent rising sub-field of machine learning that arose due to the needs for interpretability that exists in various machine learning models. It contrasts with the concept of the black boxes that was introduced in the previous chapter, it refers to models whose actions can be trusted and easily be understood by humans. Its scope aims to make black boxes interpretable or create similar ones which are understandable.

3.1 Interpretable Models

Before diving into the classification of the problems dealing with Black Box models and their proposed solutions, it is essential to first have a solid understanding of the meaning of an interpretable model. In the following subsections, we will introduce various new concepts such as interpretability, explainability and comprehensibility. We will analyze the various dimensions of interpretability along with the desired features that interpretable models need to have. Furthermore, we will present the set of machine learning models that are already recognized by
the state of the art as interpretable models. Ultimately, we will talk about ways to evaluate a model’s complexity.

3.1.1 Interpretability

Two definitions for interpretability are given by Christoph Molnar. “Interpretability is the degree to which a human can understand the cause of a decision”. Alternatively, interpretability is the degree to which a human can consistently predict the model’s result. The higher the interpretability of a model, the easier it is for someone to comprehend why certain decisions or predictions have been made. A model is better interpretable than another model if its decisions are easier for a human to comprehend than decisions from the other model [11]. It should also be argued that, even though the terms interpretable and explainable have been used interchangeably, a distinction between them should be made.

A black box model is explainable when it is able to provide the user with complete insight into what causes it to make a decision and thus making the user able to trust the model completely. Interpretable models are a great first step towards explainability, but in [12] it is stated that these models need to include mechanisms that are complete, with the capacity to defend their actions, provide relevant responses to questions and be audited. Explainable models are interpretable by default, but the reverse is not always true.

3.1.2 Dimensions of Interpretability

In the analysis of various interpretable machine learning models, it will be useful to identify a set of dimensions, that can help us characterize the interpretability of a model [13].

• **Global vs Local Interpretability.** Global interpretability denotes that the model is completely interpretable (i.e. we are able to understand all of the decisions made by the model). On the other hand, local interpretability is observed in a model when it is only partially interpretable, (i.e. we know the reasoning behind a single decision).

• **Time Limitation.** Time limitation denotes how much time the user can afford to spend to understand an explanation. When a decision needs to be taken quickly, then it is better to have a simpler and thus more easily understandable explanation. In applications where time is not an issue (e.g. scientific applications), the user would be willing to spend more time trying to fully understand a more complete and complex explanation.
• **Nature of User Expertise.** Nature of user expertise indicates how experienced the user is in the task. Users of machine learning models may come from all sorts of backgrounds (e.g. scientists, doctors, data scientists, engineers, researchers). Some of them may prefer a more refined and complex model, while others may not have the appropriate knowledge to understand complex models and may prefer a smaller and less clear one.

• **Area and Severity of Incompleteness.** The type of explanation that will be asked by each user may vary depending on the Severity of Incompleteness that each problem has. For example, one could ask a number of different questions about autonomous cars. A simple user may just be curious as to how they make decisions. A safety engineer may want to find out the specific conditions (e.g. sensor inputs) under which the car would drive off the road. Area and severity of incompleteness indicates which part of the problem formulation is incomplete, and how incomplete it is. We will speculate that the type of explanation needed will differ depending on the source of concern.

### 3.1.3 Desired Features of Interpretable Models

Arguably, interpretable machine learning Models need to provide explanations. In this manner, to understand an interpretable model, it is crucial to think about a set of features that these models need to have. Following is a list of desired features, which have been discussed in several papers in the state of the art [13].

• **Interpretability:** the degree to which the interpretable model and its outputs are understandable by humans. One of the problems in this feature is related to how interpretability can be measured, since it is to a high degree subjective. One solution for measuring interpretability is the complexity of the model in regards to the model size.

• **Accuracy:** the degree to which the interpretable model accurately makes a prediction based on data it hasn’t seen before. The accuracy of a model can be calculated using a number of different evaluation metrics (e.g. accuracy score, balanced Accuracy score, f1-score etc.). When building an interpretable model it is essential to maintain high levels of accuracy.

• **Fidelity:** the degree to which the interpretable model in question is able to accurately imitate a black box model. Often, it is better to create a new Interpretable Model that simulates a black box model. Fidelity shows how good an interpretable model is at the imitation of the behavior of a black box model. Fidelity, like accuracy, is measured with the same evaluations.
methods, with the only difference being that it is measured with regards to the outcome of the black box model it is imitating.

- **Ethicality.** Some articles define features which are related to ethical aspects, such as *Fairness* and *Privacy*, that machine learning models need to have. Fairness shows that the model guarantees the protection of people or groups against any kinds of direct or indirect discrimination (e.g. the race of a person shouldn’t define whether he can take a loan or not). Privacy, on the other hand, shows that the model does not reveal sensitive personal information about any person or groups. These two features have a high impact on the amount of trust that a human user can give to a machine learning model.

- **Usability:** the degree to which the interpretable model in question is able to provide information to users that assist them to accomplish a task with as much awareness as possible. Users tend to prefer and put more trust on interactive and flexible explanation models than textual and fixed ones.

Finally, a few other important features are **Reliability**, **Robustness**, **Causality**, **Scalability** and **Generality**. Reliability and robustness refer to the ability to maintain certain levels of performance regardless of small variations of parameters or from the input data. Causality refers to controlled input changes due to a disturbance that alters the behavior of the model. Furthermore, since we are in the "Big Data" era, it is advisable to have models capable of adapting to large input data. Scalability refers to these models. Finally, since the same model with different data can often be used in different application scenarios. Generality refers to the ability to have portable models that do not require specific training regimes or constraints.

### 3.1.4 Comprehensibility and Complexity

Many different interpretable models can provide explanations. Thus, we need to have a way to evaluate the human interpretability of the explanations outputted by different interpretable models. We will call this evaluation, with regards to each model, comprehensibility of a model. To help us understand the comprehensibility of an Interpretable Model we can evaluate its complexity. This evaluation is generally approached with a rough estimation related to the size of the model. Complexity is often used as an opposite term to interpretability.

Complexity can have many different meanings, depending on which model we are analyzing. Complexity for linear models is defined as the number of non-zero weights, while for decision trees it is defined as the depth of the tree [1]. The complexity of a decision rule, is measured by its length, which in turn is defined as the number of characteristic-value pairs in the condition. When given two rules with
similar frequency and accuracy, the rule with the smaller length will be preferred as it is more easily understandable and readable. In the case of lists of rules the complexity can be measured by taking into account the total number of characteristic-value pairs. The problem with this approach is that in ordered rule lists there are different test records which need a distinct number of rules for their evaluation. A better approach would be to find the average number of conditions evaluated to classify a set of test records [14].

3.1.5 Types of Data for Interpretable Models

Different types of data offer different levels of interpretability for humans. The most understandable data for humans are images and texts because they portray the most common and natural way that people use to get through in their everyday lives. However, the use of these data for predictive models requires various levels of processing, such as their transformation into vectors in order to make them easier to be processed by algorithms.

For the greater part, however, Machine Learning techniques work on data organized in tables which algorithms are able to handle as matrices. The advantages of this type of data are that they are more easily managed by these algorithms, without requiring any particular processing. Also, this type of data is simple enough to be understandable by humans. The disadvantage of tables is that the interpretation of the represented information requires a deeper understanding of the metadata. Only then will human users be able to associate a meaning to the values in the tables.

3.2 Recognized Interpretable Models

3.2.1 Supervised Models

A set of existing supervised machine learning models, are already acknowledged as being interpretable models. These include linear models (more specifically linear regression and logistic regression), decision trees, decision rules, k-nearest neighbors and naive bayes. These models are widely considered by the state of the art as being easily understandable by humans [11].

3.2.2 Unsupervised Models

There are also a few use cases of unsupervised machine learning methods mentioned in the literature (namely clustering), which are used in order to interpret supervised algorithms. These use cases include:
• Prototype Based Clustering.

• Multi-Subspace Discovery, which uses Clustering [15].

• Partition Aware Local Model, in short PALM algorithm, a method that is able to learn and summarize the structure of the training dataset in order to aid in the Machine Learning debugging process [16].

3.2.3 Fuzzy Systems

Fuzzy logic systems, also known as fuzzy control systems and fuzzy systems in short, are systems based on fuzzy logic. Fuzzy logic is a mathematical theory that analyzes analog input values in terms of logical variables that can take continuous values between 0 and 1. This contrast to traditional boolean logic, which is able to operate only on discrete values of either 1 or 0 (true or false, respectively). The way they work is by attempting to mimic human thinking, much like neural networks. However, instead of trying to represent the brain’s architecture, the main focus, by using fuzzy logic, is on how humans think in an approximate rather than precise way [17].

![Figure 3.2: Visualization of the difference between boolean logic and fuzzy logic.](image)

Usually the inputs to a fuzzy system are crisp values which have to be converted to fuzzy sets. The mapping from a crisp value to a fuzzy set is called fuzzification. The most widely used fuzzifier is the singleton fuzzifier. The inverse process is referred to as defuzzification.

Another important aspect of the fuzzy systems is the membership function. Typical membership functions are bell-shaped membership functions, triangular membership functions and trapezoidal membership functions. As an example, let’s
suppose we want to define the concept “around noon”. We can choose from a number of different membership functions that meet our understanding and our feeling about this concept.

Besides having the same advantages as neural networks, fuzzy systems are also highly interpretable. The main drawback is that fuzzy systems are much harder to implement. There is much more support for the development of neural networks available (e.g. libraries in various programming languages) and thus it is much easier for programmers to simply implement a non interpretable neural network in order to solve the same kind of problems.

3.3 Making Black Box Models Interpretable

To make Black Box models interpretable, we can distinguish between two categories, namely Reverse Engineering and Design of Explanation. In the first case, we try to interpret the predictions made by the black box. Given the decision records produced by the black box decision maker the task is to recreate an explanation for it. Generally, the data on which the model is trained is not known and thus the explanation is usually based solely on the predictions made by the model. In the second case we can create a model similar to that of the black box, which should give the same results as those of the black box and should also be understandable by humans. Given a dataset of training decision records the task is to develop an interpretable predictor model along with its explanations [6].

3.3.1 Reverse Engineering Analysis

Reverse engineering, also known as black box explanation, has been broken down into three sub-problems. These consist of Model Explanation, Outcome
Explanation and Model Inspection and they will be analyzed in the following sub-sections.

- **Model Explanation.** The Model Explanation problem consists of providing a general explanation of how the black box model works, through a globally interpretable and transparent model that is able to capture and replicate the behavior of the black box. More specifically, the problem consists of finding an explanation with the help of an interpretable global predictor which has been produced by utilizing the Black Box and the data on which it has been trained.

- **Outcome Explanation.** The outcome explanation problem consists of providing an explanation for the outcome of the black box model only on a specific instance. It is not necessary to explain the whole logic found in the black box, but only the reason for the prediction in a particular entry case. More specifically, the problem consists of finding an explanation with the help of an interpretable local predictor which has been produced by utilizing the Black Box and the data of a single instance, on which it has been trained.

- **Model Inspection.** The model inspection problem consists of providing a tabular, visual or textual representation in order to understand a specific property of a black box model. In contrast to the other two categories of problems, Model Inspections aims to help the human user understand, by offering a representation of several potential attributes of the Black Box model.

### 3.3.2 Design of Explanations

Design of explanations, also known as transparent box design consists of providing a local or global interpretable model directly. An example of this kind of model is a decision tree classifier from which local or global explanations can be obtained as decision rules by following the paths of the decision tree.

### 3.4 Explaining the Predictions of Any Classifier

#### 3.4.1 LIME

Local Interpretable Model-Agnostic Explanations, in short LIME, is a technique that is able to explain the predictions of any machine learning classifier [1]. LIME is able to solve problems that belong to the outcome explanation problems
presented in the previous section. This is achieved by providing a locally interpretable model that is able to explain the black box predictions with understandable human terms for only a particular case. This category of approaches, that takes advantage of the local prediction view, is becoming the most used and studied in the field of explainable machine learning.

LIME has a significant advantage over other interpretable machine learning techniques and that is because it does not depend on the type of data used, or on the type of black box to be interpreted. Lime can solve problems with almost any kind of data albeit by using different methods for different kinds of data. The way it works is by approximating the black box model by an interpretable one, such as a linear model. The main idea behind it is that it is much easier to approximate a black box model by a simple model in the neighborhood of the prediction that we need explained than trying to approximate the model globally. The explanation comes locally from the synthetic data that are generated randomly in the neighborhood of the initial record that needs to be interpreted and then weighted appropriately, according to the synthetic data proximity to the record.

Figure 3.4: How LIME is able to locally approximate a given record, shown as the red bold cross in this figure, and then generate synthetic neighbors.

3.4.2 Limitations of LIME on textual data

In this section, we point out a shortcoming of LIME, when dealing with text classification. As already mentioned, LIME works by generating a neighborhood of samples $Z$ around an instance $x$ for which we require an explanation. To better understand the issue LIME faces when dealing with text we will consider two examples.

When LIME uses tabular data and $x$ is of the form:

\[
age = 25, \ sex = male, \ income = 1200
\]
in \( Z \) the generated instances could be of the form:

\[
\text{age} = 26, \text{sex} = \text{male}, \text{income} = 1100
\]

\[
\text{age} = 25, \text{sex} = \text{female}, \text{income} = 1300
\]

In this case, we can say that the content of \( Z \) consists of synthetic data that could be real.

On the other hand, if we deal with text data and \( x \) is of the form “Dear John, I'm so happy to hear from you! Next week we can meet at the usual cafe.”, it is only able to generate neighbors of the form: “Dear, I'm happy to from you. Next we can meet at the.” or “John, I'm so to hear you. week at the usual cafe.” In this case, it just randomly removes words, bi-grams and even n-grams, where \( n \) the number of the words, but it does not generate synthetic data that could be real as in the tabular data case. Our proposal, presented in the next chapters, is precisely intended to overcome such a limitation.
Chapter 4
Design

In this chapter, we design our original proposal to overcome the limitations of LIME when dealing with textual data. It consists of a local model agnostic explainer for text classifiers called X-SPELLS (eXplaining Sentiment Prediction by generating ExempLars in the Latent Space). Before presenting the overall architecture, we introduce its main building block, namely Variational Auto Encoder models, which are a refinement of seq2seq models.

4.1 Sequence to Sequence Learning

Sequence to sequence models, also known as seq2seq, were introduced in 2014 in a paper by Google [18]. The authors use a deep Long Short-Term Memory (LSTM) artificial Recurrent Neural Network (RNN) to map a fixed-length input sequence to a vector, known as the encoder or state vector, and then another deep LSTM to decode the target sequence from that vector. This type of model aims to map input of fixed dimensionality to a fixed-length. Sequence to sequence models have had success in various natural language tasks such as translation, text summarization and question answering. The big advantage this model offers is the fact that it can map sequences of different lengths to each other.

To better understand the model we need to take a deeper look into each of its components shown in Figure 4.1. The model consists of three separate components. These are the encoder, the state vector and the decoder. The encoder consists of several LSTM recurrent components where each one takes a single element of the input sequence, processes it and returns a state vector. This state vector captures all the information from the input elements in order to help the decoder make accurate predictions in the next step. In essence, it contains the information about that sequence in the latent space. This state vector contains the latent feature space representation.
CHAPTER 4. DESIGN

4.2 Variational Auto Encoders

Variational Auto Encoders, in short VAEs, were first proposed in [19]. They are similar to seq2seq models, except that they are derived by placing an additional limitation on the loss function in such a way that the latent space is scattered and does not contain dead zones (see Figure 4.2). By avoiding these dead zones we can be sure that we don’t end up with a reconstructed entry where the data is not understandable. In this way, we can randomly test a vector from the latent feature space and create a decoded output from it which has semantic meaning [20].

The VAE decoder consists of several RNNs where each receives a hidden state from the previous one and produces an output as well as its own hidden state. It is trained to predict the next characters of the target sequence, given the previous characters of the target sequence. In particular, it is trained to convert the target sequences into the same sequences, but being offset by a time-step in the future, a training process referred to as “teacher forcing”. In order for the decoder to obtain
information about what it is supposed to produce, it is important that the encoder uses the state vector from the encoder as the initial state.

VAEs are called variational because they work by approaching the posterior distribution with a variational distribution. Therefore, the encoder emits the parameters for this variational distribution, which is a multifactorial Gaussian distribution, and then the latent representation is taken by sampling this distribution. The decoder then takes as input the latent representation and tries to reconstruct the original input from it.

**4.3 General Architecture of X-SPELLS**

In this thesis, we use the VAE model to encode the initially given text into its latent feature space representation. Then, by randomly tweaking the latent space and then using the VAE decoder we can generate random sentences. These sentences can be used in order to get the labels from the initial black box. We then use the labels along with the latent space representations to train a decision tree classifier. Finally, we locate in the decision tree the leaf in which the initially given text belongs to. This way, we can find the exemplars.

Given a text x to explain and a black box, the explanation provided by X-SPELLS is a set of exemplars that illustrate instances classified with the same outcome with x. From these exemplars, we keep the most common set of words and show them as the final explanation. On Figure 4.3 we can see a schema of X-SPELLS containing all of its parts.

With the encoder trained by the VAE (Figure 4.2), the input text x is turned into
its feature latent space representation $z$. Then, by using a random generator we are able to create a neighborhood of sentences based on $z$ called $H$. These sentences are then decoded by the VAE decoder and are labeled by using the black box. The labels $Y_b$ returned by the black box, are either 0 or 1. Subsequently, $H$ and $Y_b$ are used to train a decision tree classifier. From this decision tree, we can then obtain the exemplars.

The input text’s $x$ latent space vector becomes the first vector of $z$. This way, when we decode the first vector of $z$, we are able to use the black box model in order to find the prediction for our initial input text.

### 4.4 Final Explanations

After training the decision tree classifier, we are able to find the leaf in which the initial sentence we want to explain is located. By construction, in that same leaf, there will be other sentences which will be classified by the decision tree as being significantly close in semantic meaning with our initial sentence. We call these sentences exemplars. From this set of exemplars, we decided to take some of them randomly.

We also decided to keep the most common set of words that are found in the exemplars. In the end, the most common set of words, along with the randomly picked exemplars, represent the final explanation given by the X-SPELLS model.

In Table 4.1 we can see a complete example of how our model works given an input sentence. We will consider two sentences which come from datasets that contain tweets. Both datasets will be properly defined in the following chapters. The first sentence is part of a dataset that contains hate speech or neutral tweets. The second sentence comes from a dataset that contains tweets about positive or...
negative movie reviews.

In the first sentence the task is to explain the classification of whether the tweet contains hate speech or not and in the second sentence the task is to explain the classification of whether the review contained in the tweet is positive or negative. The first sentence is classified as belonging to the hate speech class and the second sentence is classified as belonging to the negative review class. Following are the explanations of these classifications, given by X-SPELLS.

On the first sentence’s explanation, when we look at the top words, we can see that offensive words such as ”nigga” and ”shit” correctly convey the hate speech contained in this tweet. On the second sentence respectively, we can see that words like ”bad”, ”predictable” and ”problem” are all words that correctly convey the negativity in this review.

<table>
<thead>
<tr>
<th>Input</th>
<th>Exemplars</th>
<th>Top words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicks love these ugly hipster faggots. Prolly cause they look like the dad that was never in their life.</td>
<td>’rt the was an as fashioned but im fuck’</td>
<td>menarstle</td>
</tr>
<tr>
<td></td>
<td>’rt the last smile as they just me to’</td>
<td>nigga</td>
</tr>
<tr>
<td></td>
<td>’rt the last smile as they does still’</td>
<td>shit</td>
</tr>
<tr>
<td></td>
<td>’rt i am a boon 4 as a nigga like shit”</td>
<td>smile</td>
</tr>
<tr>
<td></td>
<td>’rt menarstle czickenshack i go on drudge’</td>
<td>rt</td>
</tr>
<tr>
<td>Chaotic, self-indulgent and remarkably ugly to look at, it’s...</td>
<td>‘the movie is too much but its also one’</td>
<td>bad</td>
</tr>
<tr>
<td>like a series of pretentiously awful student films strung together into one feature-length horror.</td>
<td>’the movie is too much its’</td>
<td>predictable</td>
</tr>
<tr>
<td></td>
<td>’the problem is as its predictable’</td>
<td>problem</td>
</tr>
<tr>
<td></td>
<td>’the problem is as as many’</td>
<td>movie</td>
</tr>
<tr>
<td></td>
<td>’the movie is too much to its too bad’</td>
<td>end</td>
</tr>
</tbody>
</table>

Table 4.1: Example of an explanation when given input sentences using X-SPELLS.
Chapter 5

Implementation

In this chapter, we will delve deep into technical details regarding the implementation of everything introduced in the previous chapter. Everything was implemented using Python 3 and the following libraries: Scikit-Learn, TensorFlow, Keras, NLTK, Pandas, Numpy, imblearn and LIME. Firstly, we will talk about the pre-processing that was applied to our datasets and get into detail about the implementation of our black box models. Then, we will describe the implementation of the variational auto encoder along with the method we used to generate random sentences. Then, we will talk about a problem that arises when there are too few instances of one class and how we solve it by balancing the classes. Afterwards, we will talk about the training of the decision tree and how to extract the exemplars from it.

5.1 Pre-processing of Datasets

In this section, we will describe the pre-processing steps that we took and can be followed on any textual dataset. Firstly, we replaced punctuation with spaces, then we removed double spaces and put every character in lowercase. We also removed words that were less than 2 characters, because they were considered insignificant. We experimented on removing stop-words and using stemming\(^1\) in the form of lemmatization, but the results were not adequate because the explanations produced were less intuitive and clear than keeping stop-words and not performing stemming. For the above reasons, we decided to keep stop-words and not use stemming.

\(^1\)Stemming is the process used to reduce a word to its stem or root form. This way words like "working", "works", "worked" would all be reduced to the same word "work".
5.2 Black Box Models

In evaluating our approach, we need to experiment with real black box models. We decided to use two black box models to compare the results of LIME vs X-SPELLS: a Random Forest model and a Deep Neural Network model. For each dataset that we used, we split the data into 75% for training the black box models and 25% for training our VAE model (see next subsection).

5.2.1 Random Forest

Before training the Random Forest model, it was necessary to transform the textual data into vectors. This was done by using the Term Frequency–Inverse Document Frequency (TF-IDF\textsuperscript{2}) vectorizer of the class sklearn.feature_extraction.text, a vectorizer commonly used for text.

Afterwards, using RandomizedSearchCV and searching across many different combinations of parameters of the Random Forest algorithm, we found the best ones in order to use them for learning our random forest model.

5.2.2 Deep Neural Network

For the DNN, we decided to use a sequential model from the Keras library, which was trained on 100 epochs. The first layer consists of the embedding layer. To better understand the embedding layer’s purpose we will consider an example. Firstly, let’s think about how we can represent a sentence with a vector. In the case of the Random Forest, we used the TF-IDF vectorizer. To accomplish the same with a neural network, we need to use a sequencer which converts a sentence into an array of integers and then by using a padder we get those sequences and fill them with zeros in the words that were absent from the sentence.

By converting a sentence into a vector, we now have each word represented by a number. The embedding layer’s purpose is to convert every word into its own vector. The reason for this is that if we had one word, for example "apple" and we had another word, for example "lemon" and we represented these words with a vector of five values instead of just one, we could find their similarity easier in the 5-dimensional space than in the one dimensional space.

The embedding layer thus tries to fine-tune the right weights into each of our words, so that when a word like "apple" is transformed into a 5-dimensional vector it is close to the 5-dimensional transformation of the word "lemon" but farther from the transformation of a word like "frog".

The parameters of the embedding layer are the following:

\textsuperscript{2}TF-IDF works by summarizing how often a given word appears within a given document.
CHAPTER 5. IMPLEMENTATION

- \( \text{input\_dim} = 20000 \), which defines how many words we want to have in our vocabulary. In our case, it was set equal to 20000.

- \( \text{output\_dim} = 64 \), which defines with how many other words we want to associate a word. In our case, it was set equal to 64 words for each word.

- \( \text{input\_length} = 140 \), which trims the big sentences. This predisposes us to cut valuable words from our instances, but because our datasets contain only tweets, we set the input\_length equal to 140, which is the longest amount of characters a tweet is allowed to have. This way, we don’t lose any information from our data.

- \( \text{trainable} = \text{True} \), which lets the embedding layer fine-tune the imported weights, to take the actual training examples into consideration.

The layer architecture we then used is as follows. Firstly, we used a dropout layer of 0.25. Afterwards, we have three dense layers of 64 neurons, 512 neurons and then 128 neurons respectively. After that, we decided to use an LSTM layer of 100 neurons. Then, we have another set of three dense layers of 512 neurons, 64 neurons and 32 neurons respectively. All the dense layers have ‘relu’ as their activation function. In the end, we have a dense layer of 1 neuron with ‘sigmoid’ as its activation function. For compiling the model we used the following parameters: for loss we used ‘binary_crossentropy’, for optimizer we used ‘adam’ and for metrics we used [‘accuracy’].

5.3 Variational Auto Encoder

A variational auto encoder (VAE) model is a type of autoencoder, with additional constraints on the encoded representations being learned. To build an LSTM-based autoencoder, we first use an LSTM encoder to convert our input to a single vector containing information about the entire sequence and then repeat this vector n times (where n is the number of time-steps in the output sequence) and then we use an LSTM decoder to convert this constant sequence in the target sequence. A few useful sources, mainly from the Keras blog, that helped in the implementation of various part of the model are cited [21–23].

First, our encoder network maps input X samples to our latent distribution parameters, namely \( z\_\text{mean} \) and \( z\_\text{log}\_\sigma \) which we can use in order to sample new similar points from the latent space. We then randomly test similar z points from the latent normal distribution that is believed to generate the data through

---

3The rectified linear activation function is a linear function that will output the input directly if it is positive, otherwise it will output zero.
$z = z_{\text{mean}} + \exp (z_{\text{log\_sigma}}) \times \epsilon$, where \( \epsilon \) is a random normal tensor. Finally, a decoder network can map these sampled latent points back to reconstructed inputs.

Before getting into more details, we should first explain how exactly the latent feature space of a sentence is represented and generated. Following is a summary of our process. Firstly, we turn the sentences into 2 Numpy arrays, encoder_input_data and decoder_input_data respectively, which contain a one-hot vectorization\(^4\) of the sentences.

- **encoder_input_data** is a 3D array of shape (input_texts_cleaned, max_encoder_seq_length, num_encoder_tokens).
- **decoder_input_data** is a 3D array of shape (input_texts_cleaned, max_encoder_seq_length, num_encoder_tokens).

These two arrays are the same but decoder_input_data is offset by one timestep in the future. This means that decoder_input_data[:, t, :] will be the same as encoder_input_data[:, t + 1, :]. Following is an explanation about the parameters that define the shape of the above arrays.

- **input_texts_cleaned.** Contains all the sentences but with only the most common words kept. If a word is contained in a sentence that is not part of the most common words, it gets deleted from the sentence.
- **max_encoder_seq_length.** Contains the maximum sequence length for inputs.
- **num_encoder_tokens.** Number of unique input tokens. Essentially the number of unique words that exist. As mentioned before, this number is the union of the 4500 most common words of the respective dataset and the 1000 most common English words.

What we’ve done so far allows us to instantiate three models:

- A model that is able to predict decoder_target_data when given encoder_input_data and decoder_input_data.
- A VAE encoder which can map inputs to the latent space.
- A VAE decoder that can take points on the latent space and will output the corresponding reconstructed samples.

---

\(^4\)One hot vectorization is the process by which each piece of data, in our case text, is converted into a form that can be provided to machine learning models.
Aftersomeexperimentation, we decided to train our VAE model for 200 epochs. The model parameters are trained through two loss functions. These are the loss of reconstruction, according to which, the decoded samples need to match the original inputs and the Kullback–Leibler (KL) divergence, which is a measure of how one probability distribution is different from a second probability distribution. In our case, KL divergence measures the difference between the learned latent distribution and the prior distribution and acts as a regularization term.

The way our VAE model works is by keeping a dictionary of every word it has seen. Keeping all the words would result in a very big dictionary and as the latent dimensionality of each layer of the VAE depends on this number of words, it would require a substantial amount of RAM in order to train the VAE model. For this reason, we decided to keep only the most common words. Consequently, we kept the 4500 most common words taken from the respective dataset along with the 1000 most common English words [24].

5.4 Neighborhood Generation

In the following subsections we will discuss the process of how the neighborhoods are generated. We will also add an example so the reader can better understand what happens. Let’s suppose we have the following sentence taken from one of our datasets: "It’s a matter of pure weight ratios... A five ounce bird cannot carry a one pound coconut!". This sentence is represented as a one-hot vectorization by the encoder_input_data.

5.4.1 From Sentences to Latent Space

As already established, by using the VAE encoder, we are able to predict for any of the encoder_input_data, the corresponding latent space vectors. These vectors contain float numbers that represent each sentence in the latent space.

For the sentence we introduced in the previous subsection, the latent space vector that we end up with is a vector of this form:

\[-0.09199559, -0.04583299, -0.05029951, -0.00728882, ..., 0.08778154\]

5.4.2 Random Generation

What we need to do to create random sentences is to tweak these numbers randomly. This way, we can produce new sentences. The way we approached this randomness was by calculating for every of the encoder_input_data, the latent space
vectors. Then, for every one of these vectors, we found the minimum and the maximum number in each of the positions of the vector. This way we have an upper and lower bound for each position.

Now, to create random sentences, we take every time one random sentence from the training set. In the same way as before, we pass the encoder_input_data of that respective sentence to the encoder and we calculate the latent space vector of that specific sentence. Then, we start a loop and for each one of the positions of this vector, we compute a random number between the upper and lower bounds calculated before. At the same time, we also randomly calculate a probability from 0 to 1. If this probability is greater than 0.4, we substitute the old number in the latent space vector with the new one. This was done in order to try to create as many random sentences as possible while also having a few sentences that are close to our initial sentence.

Every time one loop runs and essentially a new sentence is generated, we check if this sentence is already contained in a unique decoded sentence list. If it is not contained, we add it to the list and continue the loop. If it is contained, we don’t add it and continue the loop. We decided to run the above loop until we reached 500 unique generated sentences or until we complete 5000 loops.

With this behavior, we can generate random sentences. A problem that arises sometimes is that no matter how much the latent space vector is changed we don’t end up with new unique sentences. Keeping in mind that in order to train the decision tree correctly we need many sentences, when we see that the sentences generated are less than 100, we decide not to train the decision tree and consequently not to explain this initial sentence.

Following from our example, we now end up with more latent space vectors. Initially we had the vector:

\[-0.09199559, -0.04583299, -0.05029951, -0.00728882, ..., 0.08778154\]

And now we have after randomly altering the values of our initial vector we end up with vectors of the form:

\[-0.09199559, 0.06011891, -0.19760853, -0.13546361, ..., 0.08778154\]

\[-0.09199559, -0.04583299, -0.05029951, 0.05674997, ..., 0.08778154\]

\[-0.05674997, 0.06849408, -0.05029951, -0.00728882, ..., 0.08778154\]

\[-0.16821258, -0.04583299, -0.05029951, -0.00728882, ..., -0.2107801\]

\[...\]

\[-0.06289171, -0.19760853, -0.03756742, -0.00728882, ..., 0.08778154\]
5.4.3 Decoding back to Original Feature Space

The random generation of previous subsection ends up with sentences in the latent space, which we have to decode back to the original feature space, i.e., to text. The steps that are followed for this purpose are adapted from the example of Keras blog in [21] and are the following:

- Generate an empty target sequence of size 1 with only the start-of-sequence character "\t".
- Feed the state vectors (which essentially contain the sentences in the latent space) sampled with the mean and std provided by the encoder along with the 1-char target sequence to the decoder to produce predictions for the next character.
- Sample the next character using these predictions by using argmax.
- Append the sampled character to the target sequence.
- Repeat from step 3 until we find the stop character or we hit the max length.

Finally following from our example, by decoding all of the previous latent space vectors we end up with:

"it’s a matter of pure weight ratios"

"who loves wearing fuzzy socks to work”
"yankees i love loud stupid and crackers”
"we wacko birds are still tweeting makedcisten”
"i want oreos who can help me”
...
"I love wearing fuzzy socks to work”

5.5 Labeling and Class Balancing

For every initial sentence that we want to explain, we now have in our disposal up to 500 randomly generated sentences. By using the black boxes we trained in the previous sub-sections, we are now able to label those sentences. However, a common problem in classification problems is due to class imbalance, which
occurs when one class in the training set dominates the other(s). For this reason, we need to make sure that our generated sentences contain about the same amount of both labels.

In our case, the classes generated from our sentences are not necessarily balanced. In order to overcome this problem, we use a method called Synthetic Minority Over-sampling Technique (SMOTE) [25]. SMOTE is able to synthesise new instances of the dominated class between already existing actual instances of that class. The way it works is by creating lines between existing instances of the class and then by imagining new, synthetic instances of that same class somewhere along those lines. SMOTE generates new instances in the latent space and then by using our VAE decoder we can decode our sentences to the original space.

![Figure 5.1: Example problem solved by using SMOTE [2].](image)

In our case, we use SMOTE when we have a class imbalance problem, namely one of the two classes has less than 40% instances labelled by the black box. For example, if the amount of sentences we have classified as hate speech is less than 40% than the total amount of hate speech and neutral speech combined or similarly if the amount of sentences we have classified as neutral speech is less than 40% than the total amount of hate speech and neutral speech combined, then we have an oversampling problem.

A problem that sometimes occurs, is that when there are too few samples of one class, SMOTE is not able to generate new data for that class. In order to avoid this problem, we decide to not explain initial sentences from our datasets in which the number of samples in any of the two classes is less than 6.
5.6 Decision Tree Learning

By using SMOTE, we have managed to solve the class imbalance problem. Next, we can use these labels along with the respective latent feature representation vectors of each sentence, in order to train an interpretable decision tree. For the implementation of the decision tree, we used DecisionTreeClassifier from the sklearn library. To find the best parameters for our decision tree, we used GridSearchCV from sklearn.model_selection with the following set of parameters:

- **param_grid**, with the following parameters names as keys:
  - **min_samples_split**, with possible values: [0.002, 0.01, 0.05, 0.1, 0.2].
  - **min_samples_leaf**, with possible values: [0.001, 0.01, 0.05, 0.1, 0.2].
  - **max_depth**, with possible values: [None, 2, 4, 6, 8, 10, 12, 16].
- **scoring** = ’f1’,
- **cv** = 5,
- **n_jobs** = -1,
- **iid** = False.

In order to test the quality of the decision tree, we will computed its fidelity, which is the accuracy between the classification of the sentences made by the black box model and the classification made by the decision tree. We present experimental results in the next chapter.

5.7 Extraction of Exemplars

Starting from the trained our decision tree model, we proceed by extracting the exemplars for our initial sentence to be explained. As already established, the exemplars are all those sentences that belong to the same leaf in which the initial sentence we want to explain, i.e. those for which the decision tree path provides a reason for similar classification.

To find the exemplars, we find the indices of all the samples that have a leaf index equal to the leaf index of our initial sentence. We chose to extract randomly 5 exemplars in order to present them as our explanation. We already have stored in two different lists the latent representation along with the original representation. So we just need to find all those indices in the original representation list. This way the exemplars are returned in the original feature space.
A problem that sometimes arises, is that there may be less than 5 exemplars in the same leaf. In that case, we decided not to show the explanation because the totality of the words contained would be too few and would likely not be that correct as to which ones appear the most times. So the final explanation produced would not be of good quality.

We also decided to keep the most common set of words that are found in the exemplars. In our case, we choose to pick the 5 most common words. From these words, we also choose to remove stop-words in order to disregard words of no significant importance. In the end, these 5 words along with the 5 randomly picked exemplars represent the final explanation given by the X-SPELLS model.
Chapter 6

Evaluation

6.1 Datasets for Experimentation

We will to use two datasets in order to test our approach. These datasets were chosen because they both contain tweets. This means that they consist of short texts, which makes the training of the VAE model significantly faster. The two datasets are as follows:

- **Hate tweets dataset** [26], which contains tweets that belong to one of three classes. These are either hate speech, offensive or neutral tweets. For simplification purposes and to convert the problem to a simple binary classification problem, we decided to keep only the neutral and hate speech tweets because we wanted to turn the problem to a binary one. We chose these two particular classes because it is easier to differentiate between them. It is hate speech or not hate speech meaning there is a clearer line of distinction. With this selection we are left with 5593 tweets. Out of these 1430 belong to the hate class and 4163 belong to the neutral class.

- **Polarity dataset** [27], which contains tweets that represent movie reviews on Twitter. These are either positive or negative reviews. There are 10660 tweets in total, out of which 5337 are negative movie reviews and 5323 are positive.

We split both datasets in the following way. We used 75% of the data as our training set and 25% of the data as our testing set for both black boxes. For the explanation of the black boxes, we also used the same testing set.
6.2 Predictive Performances of Black Box Models

Let us introduce a few metrics regarding the evaluation of classification models in general, which allow us to understand how good the two black box models we experiment on are. All of these metrics are computed on the test set. First, we introduce these notions:

• **True Positives** (TP), which are the correctly predicted positive values. For example, when the value of the actual class is positive and the value of the predicted class is also positive.

• **True Negatives** (TN), which are the correctly predicted negative values. For example, when the value of the actual class is negative and value of predicted class is also negative.

• **False Positives** (FP), which are the values that occur when our actual class contradicts with our predicted class. An example is, when the actual class is negative and the predicted class is positive.

• **False Negatives** (FN), which are similar to the False Positives, but when the actual class is positive but the predicted class is negative.

The following standard quality metrics are implemented in sklearn.metrics and more specifically accuracy_score and classification_report:

• **Accuracy**, which shows the ratio of how often our classifier is able to predict the correct label to the total number of observations. One could say that if we have high accuracy then our model is very good. While accuracy is a great measure, it only really works when we use symmetric datasets, whose values of false positive and false negatives are almost the same. For this reason, we also have to look at other parameters to evaluate the performance of our model.

\[
Accuracy = \frac{TP + TN}{TP + FP + FN + TN}
\]

• **Precision**, which shows the ratio of the correctly predicted positive observations to the total predicted positive observations. As an example, this metric answers to the following question: Out of all the movie reviews that were predicted as positive, how many are actually positive? High precision relates to a low false positive rate.

\[
Precision = \frac{TP}{TP + FP}
\]
• **Recall**, which shows the ratio of the correctly predicted positive observations to all the observations in the actual class. As an example, this metric answers to the following question: Out of all the movie reviews that were truly positive, how many did we predict?

\[
Recall = \frac{TP}{TP + FN}
\]

• **F1-score**, which considers both the precision and recall by computing their weighted average. Consequently, it takes into account both false positives and false negatives. F1-score is usually more useful than accuracy, especially when we have an uneven class distribution.

\[
F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision}
\]

• **Support**, which shows the number of occurrences of each class. For example, if we had a dataset comprising of 1000 tweets and 400 of them are of class 1 and 600 are of class 2, then the support for class 1 is 400 and the support for class 2 is 600.

The following metrics are also included in classification_report:

• **micro avg**, which gives the average of the total true positives, false negatives and false positives. For example, in a two class problem where we have class 1 and class 2 they are calculated as follows:

\[
Precision = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2}
\]

\[
Recall = \frac{TP1 + TP2}{TP1 + TP2 + FN1 + FN2}
\]

• **macro avg**, which gives the average of precision, recall and f1-score for the different labels.

• **weighted avg**, which gives a weighted average of precision, recall and f1-score where the weights are the support values.

### 6.2.1 Performances of Random Forest

The random forest model achieves accuracy score of about 92.57% on the testing data of the hate tweets dataset and a lower accuracy score of about 67.02% on the polarity dataset. The classification report for the hate tweets dataset can be seen on Table 6.1 and for the polarity dataset on Table 6.2.
CHAPTER 6. EVALUATION

6.2.2 Performances of DNN

Our DNN model achieves accuracy score of about 84.85% on the testing data of the hate tweets dataset and an accuracy score of about 63.02% on the polarity dataset. The classification report for the hate tweets dataset can be seen on Table 6.3 and for the polarity dataset on Table 6.4.

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.69</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td>2</td>
<td>0.91</td>
<td>0.88</td>
<td>0.90</td>
</tr>
<tr>
<td>micro avg</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>macro avg</td>
<td>0.80</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Table 6.3: The classification report of the DNN classifier for the hate tweet dataset.
6.3 Explanations using LIME

In this sub-section, we showcase how LIME works. More specifically, we show how LIME gives weights to words in a graphical way in order to explain the classification made by the black boxes we trained. We chose to show the top 10 features that contributed the most in the explanation. For the following results we decided to choose two different instances from both datasets randomly.

In Figure 6.1 and Figure 6.2, we can see the explanations given by LIME on both our black box models for both datasets. On the plots, in the hate tweet dataset, features with negative weight represent hate speech and features with positive weight represent neutral speech. In the polarity dataset, features with negative weight represent negativity and features with positive weight represent positivity in terms of the movie review.

The first instance that we will explain is the 6th instance from the hate tweet dataset which contains the sentence: "@_SoulSurvivor_ @zakbauer I dont have any problems with zak, but you seem like a faggot". On the first graph we can see that LIME’s weights on each word roughly agree on both the random forest and the DNN. LIME rightly gives a large negative weight to the word “faggot” and small weights to other irrelevant words. Consequently, this sentence would be classified as hate-speech due to the large weight that the word “faggot” has.

The second instance that we will explain is the 107th instance from the polarity dataset which contains the sentence: "Clumsy, obvious, preposterous, the movie will likely set the cause of woman warriors back decades.". In this case, we can notice that LIME’s weights on words are not in agreement across our two models. Both of the explanations correctly classify the sentence as a negative review but by taking into account different words. When explaining the random forest we can see that words with negative meaning such as "preposterous", "clumsy" and "obvious" correctly have negative weights. However, the largest negative weight is given by the word "movie", which is questionable. When explaining the DNN, the word "obvious" has the biggest negative weight and the weight of all the other words have almost irrelevant effect.
Figure 6.1: The explanation given by LIME on the 6th instance of the hate tweet dataset when explaining the predictions of the random forest on the top and the DNN on the bottom.
Figure 6.2: The explanation given by LIME on the 107th instance of the polarity dataset when explaining the predictions of the random forest on the top and the DNN on the bottom.
6.4 LIME vs X-SPELLS: Quantitative Evaluation

We evaluate our approach by comparing the fidelity of the classifications made by LIME and the classifications made by X-SPELLS on each of our datasets. As already established in section 3.1.3, fidelity shows the degree to which an interpretable model is able to accurately imitate a black box model. Thus, it is an appropriate measure to compare the performances of LIME and X-SPELLS.

To have a fair comparison we calculated the fidelity for the same sentences on both models and then computed the average. The results on our black box models can be seen below. First, we show the results when explaining the random forest and then the results when explaining the DNN.

<table>
<thead>
<tr>
<th></th>
<th>LIME</th>
<th>X-SPELLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hate Speech</td>
<td>0.62</td>
<td>1</td>
</tr>
<tr>
<td>Polarity</td>
<td>0.89</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 6.5: Fidelity for each dataset when using LIME and X-SPELLS to explain the random forest.

<table>
<thead>
<tr>
<th></th>
<th>LIME</th>
<th>X-SPELLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hate Speech</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>Polarity</td>
<td>0.91</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 6.6: Fidelity for each dataset when using LIME and X-SPELLS to explain the DNN.

On the first table we can see that when explaining the random forest classifier, the X-SPELLS model does a much better job than LIME. Especially on the hate speech dataset we can see that X-SPELLS accomplishes fidelity of 1 whilst LIME accomplishes a fidelity of only 0.62. On the polarity dataset the numbers are closer but still X-SPELLS outperforms LIME 0.96 to 0.89.

When explaining the deep neural network on the second table, we can see that the fidelities are not as distant as they were on the random forest. On the hate speech dataset we can see that X-SPELLS accomplishes fidelity of 1 contrasted to the fidelity of 0.92 of LIME. On the polarity dataset, X-SPELLS surpasses LIME with a fidelity of 0.97 compared to a fidelity of 0.91 on LIME.

All in all, we can conclude that X-SPELLS outperforms LIME in terms of fidelity on both datasets and on both black boxes. It should however be noted that it is not surprising that decision trees, which X-SPELLS uses, perform better
than linear models, which respectively LIME relies on, since linear models are more complex classification models. However, we believe that the better fidelity is also attributed to the better neighborhood generation approach of the X-SPELLS model.

6.5 LIME vs X-SPELLS: Qualitative Evaluation

In qualitative evaluation, we contrast the explanations given by LIME and those given by X-SPELLS on 4 input instances. We will use 2 sentences from the hate tweet dataset, one belonging to the neutral class and one belonging the hate class and 2 sentences from the polarity dataset, one belonging to the negative review class and one belonging to the positive review class. For each dataset, we will first show the explanations using X-SPELLS and afterwards, the explanations using LIME for the same set of sentences. For both LIME and X-SPELLS we explain the DNN, because the fidelity results in the quantitative evaluation were comparably similar. In regards to LIME, we choose to showcase the top 5 words that contribute most to the prediction.

In the following tables, we provide the reader an easy way to understand the explanations for both X-SPELLS and LIME. For each instance, we show the input sentence given, the prediction made by either X-SPELLS or LIME and then the true class of the respective input. In the case of X-SPELLS, we showcase the five exemplars that were chosen along with the five top words. In the case of LIME, we show the five top words along with their respective weights.

The first instance that we explain belongs to the hate class and is the sentence: “Chicks love these ugly hipsterfaggots. Prolly cause they look like the dad that was never in their life”. On Table 6.7 we can see that X-SPELLS correctly classifies the sentence as belonging to the hate class. When looking at the top words, we can see words like “nigga” and “shit” which correctly show the sentiment of the input. On Table 6.8 we can see that LIME incorrectly classifies the sentence as belonging to the neutral class. By looking at the top words, we can see that LIME gave too much weight on the positive words within the input sentence, such as “life” and “love” and no significant weight on the negative words. Thus, it made the wrong prediction.

The second instance that we explain belongs to the neutral class and it consists of the sentence: “They have a clumsy bird too lol what is life”. On Table 6.7 we can see that X-SPELLS correctly classifies the sentence as belonging to the neutral class. When looking at the top words, we can see words like “favorite” and “love” which correctly convey the sentiment of this given input. On Table 6.8 we can see that LIME correctly classifies the sentence as belonging to the neutral speech. By looking at the top words, we can see that LIME gave positive weights on all the
words with the most significant ones being "life" and "bird".

<table>
<thead>
<tr>
<th>Input</th>
<th>Prediction</th>
<th>True Class</th>
<th>Exemplars</th>
<th>Top words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicks love these ugly hipster faggots. Prolly cause they look like</td>
<td>Hate</td>
<td>Hate</td>
<td>'rt the was an as fashioned but im fuck'</td>
<td>menarsle</td>
</tr>
<tr>
<td>the dad that was never in their life</td>
<td></td>
<td></td>
<td>'rt the last smile as they just me to'</td>
<td>nigga</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'rt the last smile as they does still'</td>
<td>shit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'rt i am a boon 4 as a nigga like shit'</td>
<td>smile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'rt menarsle czickenshack i go on drudge'</td>
<td>rt</td>
</tr>
<tr>
<td>They have a clumsy bird too lol what is life</td>
<td>Neutral</td>
<td>Neutral</td>
<td>'its a plane its a bird nah its move'</td>
<td>told</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'i dont know what you been told but'</td>
<td>wear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'i love the domes in my favorite wear'</td>
<td>favorites</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'i love dat domes in my favorite bring'</td>
<td>love</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'i love monkey with my throat when we'</td>
<td></td>
</tr>
<tr>
<td>Table 6.7: The explanation when given input sentences from the hate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tweets dataset using X-SPELLS.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Prediction</th>
<th>True Class</th>
<th>Top words</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicks love these ugly hipster faggots. Prolly cause they look like</td>
<td>Neutral</td>
<td>Hate</td>
<td>life</td>
<td>0.130038</td>
</tr>
<tr>
<td>the dad that was never in their life</td>
<td></td>
<td></td>
<td>love</td>
<td>0.097469</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hipster</td>
<td>-0.053505</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dad</td>
<td>0.051236</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>never</td>
<td>0.037720</td>
</tr>
<tr>
<td>They have a clumsy bird too lol what is life</td>
<td>Neutral</td>
<td>Neutral</td>
<td>life</td>
<td>0.064772</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bird</td>
<td>0.064019</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lol</td>
<td>0.032086</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clumsy</td>
<td>0.013192</td>
</tr>
<tr>
<td>Table 6.8: The explanation when given input sentences from the hate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tweets dataset using LIME.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The third instance that we explain belongs to the negative movie review class and it consists of the sentence: "Looks and feels like a low-budget hybrid of Scarface or Carlito’s Way.". It is of greater interest because this sentence, while stated as being negative has no clear content. Scarface and Carlito’s Way are both movies that are widely favored by the critics. Metacritic, a website which uses a weighted average, assigned Scarface a score of 65 out of 100 based on 9 critics and Carlito’s way a score of 65 out of 100 based on 25 critics both of which indicate according to the website “generally favorable reviews”. [28, 29].

Thus the sentiment of this tweet would not necessarily be negative. On the opposite, we believe that the content of this review is more positive rather than negative and unsurprisingly, it seems like both X-SPELLS and LIME agree. On Table 6.9 we can see that X-SPELLS incorrectly classifies the sentence as belonging to the positive class. When looking at the top words, we can see words like "clever" and "funny". Among the exemplars there are also sentences with seemingly positive content. On Table 6.10, we can see that LIME also incorrectly classifies the
sentence as belonging to the positive class. By looking at the top words, we can see
the most significant ones being “way”, with the biggest positive weight of 0.65
and “hybrid”, with the biggest negative weight of -0.31.

The fourth and final instance that we explain belongs to the negative class and
it consists of the sentence: “Chaotic, self-indulgent and remarkably ugly to look
at, it’s...like a series of pretentiously awful student films strung together into one
feature-length horror”. On Table 6.9 we can see that X-SPELLS correctly classi-
fies the sentence as belonging to the neutral class. When looking at the top words,
we can see the words ”bad”, ”predictable” and ”problem” all of which correctly
convey the sentiment. Among the exemplars there are also sentences with ostensibly negative content. On Table 6.10, we can see that LIME also correctly classifies
the sentence as belonging to the negative class. By looking at the top words, we can see that LIME gave negative weights on the words ”chaotic”, ”student”, ”self”
and ”length” and a smaller positive weight on the word ”look”.

### Table 6.9: The explanation when given input sentences from the polarity dataset

<table>
<thead>
<tr>
<th>Input</th>
<th>Prediction</th>
<th>True Class</th>
<th>Exemplars</th>
<th>Top words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looks and feels like a low-budget hybrid of Scarface or Carlito’s Way.</td>
<td>Positive</td>
<td>Negative</td>
<td>’the director and sentimental cast appear to see this’</td>
<td>val</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’in the name of an allegedly and f lick the emperors’</td>
<td>clever</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’the stories of attraction gets the level of val party’</td>
<td>story</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’the plot is very clever but it down with funny enough’</td>
<td>funny</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’the plot is very clever but it down with funny as’</td>
<td>plot</td>
</tr>
<tr>
<td>Chaotic, self-indulgent and remarkably ugly to look at, it’s...</td>
<td>Negative</td>
<td>Negative</td>
<td>’the movie is too much but its also one’</td>
<td>bad</td>
</tr>
<tr>
<td>like a series of pretentiously awful student films strung together into one feature-length horror</td>
<td></td>
<td></td>
<td>’the movie is too much its’</td>
<td>predictable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’the problem is as its predictable’</td>
<td>problem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>’the problem is as as many’</td>
<td>end</td>
</tr>
</tbody>
</table>

Table 6.10: The explanation when given input sentences from the polarity dataset

<table>
<thead>
<tr>
<th>Input</th>
<th>Prediction</th>
<th>True Class</th>
<th>Top words</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looks and feels like a low-budget hybrid of Scarface or Carlito’s Way.</td>
<td>Positive</td>
<td>Negative</td>
<td>way, hybrid</td>
<td>0.658037</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>low, scarf</td>
<td>-0.318910</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>face, feels</td>
<td>0.115448</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>way, hybrid</td>
<td>-0.018211</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>low, scarf</td>
<td>-0.013485</td>
</tr>
<tr>
<td>Chaotic, self-indulgent and remarkably ugly to look at, it’s...</td>
<td>Negative</td>
<td>Negative</td>
<td>chaotic, student</td>
<td>-0.060571</td>
</tr>
<tr>
<td>like a series of pretentiously awful student films strung together into one feature-length horror</td>
<td></td>
<td></td>
<td>self, length</td>
<td>-0.058855</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>look, length</td>
<td>-0.058103</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>look, length</td>
<td>-0.046681</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>look, length</td>
<td>0.019574</td>
</tr>
</tbody>
</table>

To conclude, while LIME provides more meaningful explanation in terms of
the words it shows, it only enhances the words by applying to them their respective
weights. Aside from the top words that convey the sentiment of the sentence it is tasked to explain, the advantage of X-SPELLS over LIME, excluding the better fidelity, is that it also provides similar sentences in the form of the exemplars, which lead to the same classification from the black box.
Chapter 7

Conclusions

In this thesis, we introduced X-SPELLS, a black box model agnostic explainer for text classifiers, that through the use of a variational auto encoder, learns and uses the latent space for generating local neighborhoods. The explanations that X-SPELLS returns are made of a set of exemplars, bearing the same label corresponding to the text we want to explain and also by a set of words, which are the most prominent ones found in the set of exemplars.

Based on the quantitative evaluations made on the previous chapter we can see that our proposed model out-performs the state of the art technique called LIME in terms of fidelity. Based on the qualitative evaluation, we can also deduce that the explanations given in the form of exemplars and top words in them are reasonable and intuitive given their respective input.

In conclusion, this thesis’ work was accomplished with no prior experience on any of the component parts involved. Despite this fact, the experimental results are promising and show that, even in its simple form, the X-SPELLS approach is able to generate valid explanations.

Moreover, the design of X-SPELLS is general enough to account for improvements in each of the architecture components. Example of further improvements are proposed next.

7.1 Future Work

From this point, there are many extensions and future work directions possible. This work could be refined by using the general layout already proposed in this thesis and then by improving on each of the individual components. Several future steps can be considered for the improvement of the present work.

Firstly, data preprocessing and parameters of X-SPELLS can be extended to account for textual data longer than the size of tweets. Better pre-processing could
also be made on the datasets, with regards to removing some prominent words both when training the black box models and the VAE mode. For example, in the hate tweet dataset the word ‘rt’ should have been removed because not much informative. Additionally, experimentation can be conducted with additional black box models and/or different learning parameters.

With respect to the VAE, several variants could be experimented with, perhaps with different and more complex encoder and decoder layers. Better methods could also be used for the generation of the neighborhoods. Also, state of the art techniques could be experimented/compared with to try paraphrasing from the initial instance (for example using deep reinforcement learning for paraphrase generation [30]).

In this thesis, we used SMOTE to solve the class imbalancing problem. However, using SMOTE leads to the synthetic data generated to suffer from the same problem that LIME’s generated neighbors, as mentioned in section 3.4. For this reason, experimentation could be done firstly without using SMOTE and then by comparing the results with regards to the quality of the explanations with and without using SMOTE.

Finally, a substantial improvement would be to also include counter-factual exemplars. Counter-factuals are created from instances of one class with a minimal change which push them to be classified as belonging to the opposite class [31]. Following this, we could generate counter-factual rules in the latent space and exploit them to derive counter-exemplars in the original space. Then, we could utilize these counter-exemplars to show in the explanation why our initial instance did not belong to the opposite class.
Bibliography


