TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un Sistema de Clasificación de Comportamientos Engañosos basado en Técnicas de Aprendizaje Automático. Aplicación a detección de opiniones fraudulentas y reclutadores radicalistas.
Título (inglés):	Development of a Classification System of Deceptive Behaviours based on Machine Learning Techniques. Application to fake reviews and radicalist recruiters detection.
Autor:	Rodrigo Barbado Esteban
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN
Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

DEVELOPMENT OF A CLASSIFICATION SYSTEM OF DECEPTIVE BEHAVIOURS BASED ON MACHINE LEARNING TECHNIQUES

**APPLICATION TO FAKE REVIEWS
AND RADICALIST RECRUITERS DETECTION**

RODRIGO BARBADO ESTEBAN

Junio de 2016

Resumen

En los últimos años Internet se ha ido introduciendo progresivamente en la vida de personas de todo el mundo. El poder de Internet es inmenso y sus beneficios múltiples para mejorar la comunicación entre personas y democratizar el acceso a la información.

Sin embargo, Internet también conlleva ciertos riesgos asociados no sólo a la seguridad sino también a la veracidad del contenido publicado. Uno de los riesgos relacionados con el contenido que se encuentra en la red tiene que ver con las intenciones del usuario que lo ha escrito, que no siempre son bien intencionadas. En este trabajo fin de grado, el objetivo principal es desarrollar un sistema de clasificación de para la detección de usuarios malintencionados.

En primer lugar se describirán las tecnologías que han hecho posible llevar a cabo este proyecto, en el cual existe la necesidad de obtener y almacenar datos de Internet para su posterior procesado. El problema de clasificación requiere guardar instancias de datos junto con la clase a la que pertenece, y se engloba en el marco del aprendizaje automático supervisado.

El sistema se aplicará en dos casos de estudio: detección de revisiones falsas en web sociales, y detección de reclutadores yihadistas. Para el primer caso, el proyecto trabajará con un corpus etiquetado de revisiones de restaurantes de la red social Yelp. Para el segundo, se trabajará con un corpus etiquetado de foros donde la comunidad Dark Web ha detectado actividad de reclutadores que promueven el radicalismo yihadista. En ambos casos se buscará lograr el mayor porcentaje de aciertos en la clasificación para ejemplos nuevos de entrada al sistema.

Por último, se expondrán las conclusiones del proyecto y las líneas de trabajo futuras.

Keywords: Aprendizaje automático, Python, Yelp, Dark Web, Scikit-learn, Scrapy, MongoDB, engaño, reclutamiento.

Abstract

In the last years the Internet has been progressively introducing into the lives of people from all over the world. The power of Internet is huge and it has multiple benefits to improve communication between people and democratize the access to information.

However, the Internet also carries some risks associated not only to security but also veracity of the published content. One of the risks related with content that can be found on the net has to do with the author intentions, which are not always well-intentioned. In this final grade project, the principal objective is developing a classification system for users with malicious intentions.

Firstly, there will be a description of the enabling technologies which made possible the realization of this project in which there is a need of obtaining and storing data extracted from Internet for its posterior process. The problem of classification requires saving data instances along with the class they belong to, being a case of supervised machine learning.

The system will be applied in two study cases: detection of fake reviews in social webs and yihadist recruiters detection. For the first case, the project will work with a labeled corpus os restaurants from the social network Yelp. For the other case, the work will be done with a labeled corpus from forums where the Dark Web community has detected activity from recruiters who promote yihadist radicalism. In both cases the objective will be achieving the greatest accuracy the classification for new input examples of the system.

Finally, conclusions of the project and lines of future work will be exposed.

Keywords: Machine learning, Python, Yelp, Dark Web, Scikit-learn, Scrapy, MongoDB, deception, recruitment.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

From the last years the Internet has been introducing progressively in the lives of everyone all over the world. The power of Internet is huge as it interconnects every single person that has access to it. The net of nets was developed with beneficial purposes of bringing lots of new services, but this has not ensured that all users behave ethically while using those services. As a result of that, every user has to assume various risks everytime he connects to the Internet. This project focuses on one of those risks, called deception [12], which is the act of misleading by a false appearance or statement. Internet is full of information which can be really useful for daily issues, but can we trust everything that appears on the web?

It is impossible to know the authorship of every text posted on the web, and users should not trust everything they read. However, when they are searching for something specifically or reading opinions that sound convincing for them, the general impulse is trusting these texts ignoring whether if the author wrote them with harmful intentions or not. In the last years this problem has grown remarkably due to the appearance of social networks.

Social networks are used by millions of people who post their photos, thoughts, sentiments or opinions. It is obvious that this amount of data stored on social networks can give powerful information if it is well processed. This project will gather data from the following sites:

- Yelp (http://yelp.com): in this site businesses of different categories are reviewed and rated by users. Users are not only limited to write reviews, they can also make friends as Yelp offers lots of facilities for it as many other social networks do. The reviews they filter appear at the end of each business page, and thanks to this it is possible to analyse which features they are using while filtering.
- DarkWebForums [4]: from this site there is information from typically private social websites where extremists interact. Some of the forums that can be found there focus on extremist religious and general Islamic discussions, and some radical Islamic organizations have presence there to recruit new members.

1.2 Project Goals

In general terms, this project aims at showing the possibilities offered by data analysis when trying to solve specific problems such as deception. The amount of information stored on the Internet is huge, and in many cases it is possible to obtain by different means. Extracting the information contained in data is a challenging task, but the reward is often really juicy.

More particularly, this project will show how important is organising and treating big amounts of data when trying to get rid of it. A bad architecture can turn a project complexity into unmanageable, and so designing every step carefully is crucial. Without these considerations, it would be impossible to fulfil the main goal of this project: exploring the utility of machine learning techniques for data analysis. This project will specifically focus on classification techniques, which try to identify which class an object belongs to. It will not only be important obtaining the best accuracy results, but also analysing in each case which features determine better if an instance belongs to a class or another in order to get conclusions from them.

1.3 Document structure

After this Introduction chapter, there is a Enabling technologies chapter (Chapter. 2) in which the main technologies used for this project are described, making an emphasis on Scikit-learn, MongoDB and Scrapy.

The next two chapters correspond to the two use cases in which the project is mainly divided. The reason of that division is that even if both cases correspond to deceptive behaviours on Internet, the fields of each of them are totally different and thus the features studied. Those chapters are called Yelp and Recruitment (Chapter. 3 and Chapter. 4 respectively) and both of them are compound of four main sections: Introduction, Corpus, Feature extracion and Classification model.

Finally there is a Conclusion and Future work chapter (Chapter. 5).

CHAPTER 2

Enabling technologies

2.1 Introduction

Machine learning [6] is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence . Depending on the nature of the learning “signal”, it can be classified into supervised learning, unsupervised learning and reinforcement learning. This project applies supervised learning techniques focusing on classification tasks.

- Supervised learning algorithms need labeled training data from which they produce an inferred function used for mapping new examples. Labeled training data is composed of examples with an input object, typically a vector, and its output value.
- Classification inputs are divided into classes, so the learning model must assign a class to each of the inputs. Classification and supervised learning are usually related as classification problems contain input objects whose output values are their classes.

To sum up, there is a need of training a classifier with labeled examples so it can learn patterns from them that will help for classifying new examples in the future. The following sections in this chapter describe technologies used to achieve this.

2.2 Scikit-learn

Scikit-learn is an open source machine learning library for Python, built on NumPy, SciPy and matplotlib [14]. It contains a great variety of classification algorithms from different categories such as Support Vector Machines, Nearest Neighbors or Decission Trees. Additionally, it is also possible to combine multiple classifiers with Ensemble methods. An example of Ensemble methods is a Voting Classifier, whose gets inputs are the outputs of the desired classifiers and learns from it. Classification in Scikit-learn follows the following architecture:

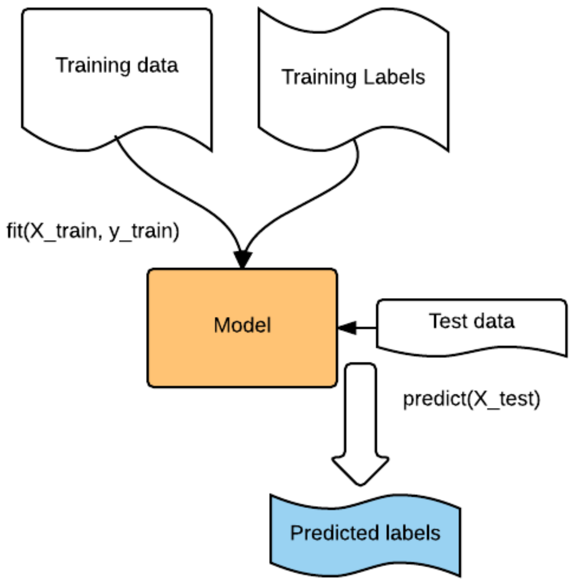

Figure 2.1: Model training and output prediction

Training data and test data are not raw data but features extracted from initial data and transformed into formats that can be understood by estimators. Scikit-learn provides feature extraction tools that convert objects to Numpy or SciPy representation which is used by Scikit estimators. The following vectorizers were used in this project:

- **Loading features from dictionaries**: DictVectorizer can be used to convert features feature arrays represented as lists of standard Python dictionary objects to NumPy or SciPy representation.
- **Text feature extraction**: Scikit-learn provides utilities for the most common ways to extract numerical features from text.

Going deeper into text feature extraction it is important to mention Term frequency-inverse document frequency (TF-IDF), which is a method to evaluate the importance of a word in a document by converting the textual representation of information into a Vector Space Model. A Vector Space Model is an algebraic model representing textual information as vectors, being their components for example the importance of a term. The aim is modelling documents into a vector space ignoring ordering of words but focusing on information about occurrences of words.

The first step is creating a model of the documents into a vector space creating a dictionary of terms that appear on them. However, there is a need of preprocessing texts as not all terms give information to discriminate between different types and they introduce noise that can difficult classification afterwards. Those words such as a, the, at, on... are called stop words and will be filtered to create the model in order of reducing noise. Additionally, there is another preprocessing task that consists on extracting the root or lemma of each word. This is done because there is more information on using a specific concept than its different forms, which can also introduce some noise.

The next step is calculating the TF or term frequency to represent each term into the vector space. The term frequency is the measure of how many times the term appears in the vocabulary. So, for each term of the vocabulary appearing on all the documents, each document will have a matrix of their appearances on it. With this, each document of the set is represented by a vector with zeros on the terms that did not appear and the number of appearances on the terms that did appear on it.

After obtaining those vectors, it is important normalizing them because a word appearing a number of times in a document has not the same relevance as if it appears on a document ten times smaller. Having all vectors normalized, the next step is calculating the inverse document frequency.

The inverse document frequency can be defined as $\log \frac{|D|}{1+|\{d:t\in d\}|}$, being $1+|\{d:t\in d\}|$ the number of documents where the term t appears and $|D|$ the number of documents in the corpus. This function could variate, as it tries to get the impact of a word in the corpus and in this case it smooths it by computing it into a logarithmic scale.

Once both term frequency and inverse document frequency are calculated, the TF-IDF for each term on a document is calculated by multiplying its term frequency and inverse document frequency. A high weight of the TF-IDF is reached with a local parameter which is a high term frequency in the document, and a global parameter which is a low document frequency of the term in the whole set (which means a high IDF).

2.3 MongoDB

As feature extraction for large amounts of data is a expensive operation in terms of time, storing processed features in a database is needed to work fluidly (http://mongodb.org). The solution adopted for that problem has been the use of MongoDB, which is a cross-platform document-oriented database classified as NoSQL.

It eschews the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas, named BSON. It was written in C++ and provides high performance, high availability and easy scalability. Queries on MongoDB provide a set of operators to define how the requesting methods select documents from collections.

2.4 Scrapy

Scrapy is a free and open source web crawling framework written in Python originally designed for web scraping. Its project architecture is built around 'spiders' or self-contained crawlers which are given a set of instructions (http://doc.scrapy.org/en/latest/intro/overview.html). A web crawler is an Internet bot which browses the Internet, and in this project it is needed to get extra information from the Yelp site.

With Scrapy the process of obtaining data from the web can be done in a quick way. The spider first needs the URLs, and then it iteratively accesses them. The next thing to be done is analysing the code of the web page. Once the specific information is found on the structure of the page, Scrapy offers functions to parse it and collect the information needed which can be easily saved as a file.

CHAPTER 3

Yelp

3.1 Introduction

The use of Internet for searching products and services has grown significantly in the last years. Internet has become a perfect shop window for businesses and their reputation relies on the opinions given by anonymous users, which is not always beneficial for businesses: bad opinions on the products of a business can transform into a bad reputation for the company. Additionally, as everyone can be on the Internet there is a fierce rivalry between businesses of the same fields so having a bad reputation may be critical. This fact is well known by companies but it is not usually easy to achieve a better valoration than competitors , and so some of them use unethical techniques in order to gain reputation. The majority of these techniques used are called fake reviews and consist of users who promote products or services of the interested organization, or discredit products from competitors. However, according to Jindal and Liu, not all fake reviews are equally harmful [7]. Fake negative reviews on good quality products are really harmful for enterprises, and along with fake positive reviews on poor quality products are harmful for consumers. Fake positive reviews on poor quality products are also harmful for competitors who offer average or good quality products but do not have so much reviews on them.

Another classification can be made concerning reviewers, whom we can define as professionals or non-professionals. Professional fake reviewers get paid for writing a large amount of fake reviews. They can be more easily detected, but discovering initial patterns to detect them is a challenging task. If they are discovered lately, the damage is really done. On the other hand we have non-professional fake reviewers or people who are not paid for writing fake reviews and do not write them regularly. They write fake reviews to promote their own products or products from their familiars or close friends and to discredit their competitors. In this group there are also fake reviewers who are just trying to have a good time writing. In addition, we can also find group spamming instead of individual.

Detecting fake reviews has become an important task for web sites such as Amazon, TripAdvisor or Yelp. These sites are focused on consumers and so, they rely on the veracity of the opinions exposed on them. If someone goes to a restaurant recommended by Yelp and they serve a poor quality food, not only the restaurant loses a client but also Yelp does. Consequently, these web pages are developing systems to detect fraudulent reviews.

In this work the objective is to discover more about the filter used by Yelp. Yelp was founded on 2004 and at the end of 2014 they had more than 2700 employees. From this fact we can deduce that Yelp filter might be working well, probably over-filtering in some cases as even Yelp has recognised. Nevertheless it is reasonable that Yelp prefers having more false positives than fake reviews because a fake review can be more harmful than a trustful review that has been filtered. As Yelp displays its filtered reviews at the end of each business page, obtaining labelled data for training is feasible using a web crawler.

Previous research on Yelp filter has been focused on linguistic features from reviews and behavioural features from users. Mukherjee et al. concluded that behavioural features performed much better than linguistic features in the case of Yelp [11].

This work proposes a new behavioural features which focus on Yelp as a social network, and that combined with the rest of behavioural features conduct to great accuracy results.

3.2 Related work

The problem of fake reviews has been studied since 2007, when Jindal and Liu [7] focused their attention on analysing and detecting review spamming. They first analysed the case of Amazon and concluded that manually labelling fake reviews might be challenging because fake reviewers could carefully craft their reviews in order to make them more reliable for the rest of users. Consequently, Jindal and Liu proposed considering duplicates or nearly-duplicates as spam in order to create a first model to detect fake reviews and a framework for fake review detection [7] in which three feature types are identified: review centric features, reviewer centric features and product centric features.

Another area of research has been the development of annotated corpus, where a number of approaches have been followed. Li et al. crawled Epinions [8], a site where users could evaluate the posted reviews by giving them scoring. They also suggest new features and a new model using semi-supervised learning, considering that a user who writes one fake review is suspicious of writing more.

Ott et al. [13] manufactured a corpus by using mechanical Turkers to create fake reviews for twenty Chicago hotels. In this experiment text features performed really well on classifying the reviews written by Turkers from TripAdvisor reviews which were considered as trustful examples. However, Mukherjee et al. [11] tried to use text features in order to classify fake reviews in Yelp obtaining remarkably worse results compared to the success obtained in [13]. Mukherjee concluded that mechanical Turkers did not a good job writing those fake reviews because word distributions between fake and trustful reviews were too much different. Consequently, he used behavioural features in order to improve the classification model on Yelp resulting in encouraging results.

Fei et al. propose another technique which was applied on Amazon corpus and consists on detecting review bursts [2] or the increase in the review number of a product in a short period of time.

3.3 Corpus

The confection of Yelp corpus was carried out following three steps. The first of them was getting the public academic dataset offered by Yelp for educational purposes [1]. This dataset, which has been the base for this project, contains the following elements in JSON format:

- Business objects: 18459 items with information about the name of the business, its category, review count, average stars given to it, its Yelp URL and many others with less relevance in this project.

- Review objects: 330071 items with information about its business identifier, user identifier, text, date, votes (useful, funny or cool) and the star rating given in the review to the business.

- User objects: 261746 items with information about the name of the user, his review count, average stars given, the votes received across his reviews and his the URL of his profile site inside Yelp.

Using supervised learning techniques requires having an annotated or labelled corpus, but Yelp does not offer information about which reviews are filtered. The solution of this problem was the development of a web scraper capable of searching every business appearing at the dataset offered by Yelp. The scraper developed in this project has the aim of annotating automatically the Yelp dataset, by identifying which reviews of the dataset have been filtered by Yelp, which are labelled as fake.This task was possible because each review of the dataset had its business identifier, and each business had its URL from Yelp. A field called ”fake” was added to each of the review objects, and its value could be True if the review matched a review located in the filtered section or False if not.

From the 330071 reviews contained on it only 8199 were fake, which means that less than a 3% of reviews were fake. However, this percentage is notably higher [11] (14%). It is not a good practice to start with such an unbalanced dataset because it would led to corrupted results , so a segmentation of the corpus was needed. The decision was to use all fake reviews from the restaurant domain, which were 4804, and choosing another 4804 restaurant domain trustful reviews randomly. This could also help for analysing textual features later on, as the vocabulary context was the same for all of them. With these reviews selected for the project, the division for training and test was designed as follows:

Set	Trustful Reviews	Fake Reviews
Train	3535	3535
Test	1269	1269

At this point, the development of a supervised learning model in order to classify reviews was possible. However, Yelp offers more information on the web than the one contained in the dataset specially regarding users. It was impossible to know beforehand if this new information could be relevant for the model, but the hypothesis of Yelp filtering according to user activity was strong enough to extract those new features. Thus, the third step on corpus creation was the development of another scraper capable of extracting additional information that appears on the profile site of each user. Once again this was possible because Yelp offered on its dataset the URL site of users. From this second scraper it could be possible to add the following information about each user:

- Number of friends
- Number of photos uploaded
- Number of tips given.
- Number of updates made on reviews
- Number of times the user was the first to review a business
- Boolean indicating if a user uploaded a profile photo or no
- Profile description

All the process that was carried out to obtain the final corpus is described in the following image, in which the final corpus refers to the selected restaurant domain annotated corpus with extra features.

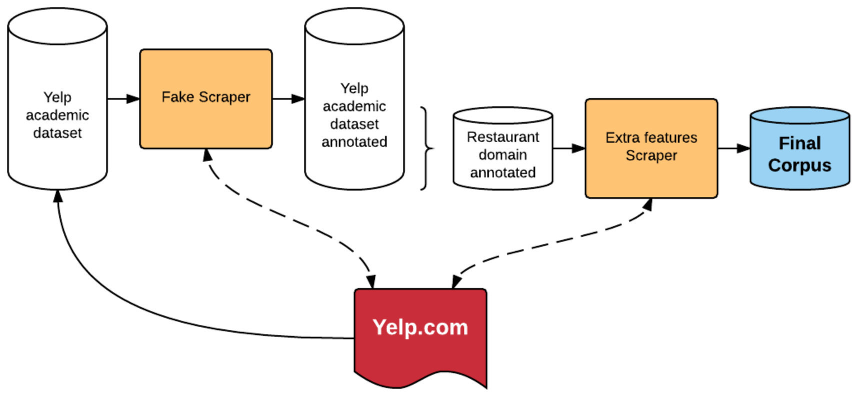

Figure 3.1: Corpus creation architecture

3.4 Feature extraction

As it was explained before, the corpus contains information about reviews and users. This makes possible the analysis of the problem from various points of view: focusing on content and metadata of the review or in data from the user who posted the review. This division results in two different types of features to be extracted: review centric features and behavioural features.

3.4.1 Review centric features

Linguistic or textual features are the ones extracted exclusively from the content of the review or from its metadata. The only metadata present in the corpus are the useful, cool or funny votes obtained by the review, but as the possibility of voting a review is only given while it has not been filtered, it can give unreal information about the impact of votes on reviews. Another thing that was taken into account was that analysing star rating as an isolated feature does not really give information, and will be used in the following section as they have to be analysed in the context of how an user behaves giving ratings. Some metrical features regarding the text of a review are number of words, words per sentence, capital letters, first and second pronouns, exclamations and questions, paragraphs, long words and punctuation marks. The TF-IDF was discarded for this use case as it introduced a lot of noise to the system, but the utility of it will be shown in the Recruitment chapter along with another techniques.

3.4.2 Behavioural features

The corpus provides a great amount of data from users from which valuable information can be extracted. In Yelp it is possible to find users who have both fake and trustful reviews. However, professional fake reviewers exist [7] and so focusing on how the reviewer behaves can be a good idea to detect them. Moreover, non-professional fake reviewers could also be detected with these features as they can be for example users that register in Yelp only for writing a positive review on a familiar business.

Focusing on behavioural features means doing it on users so there will always be errors in those users who have fake and non fake reviews, but in a normal situation a user who has a great percentage of fake reviews could be considered a fake reviewer and so his non filtered reviews should probably have been filtered. An opposite situation could take place regarding a user who has a majority of non fake reviews and a filtered one, case in which the user could be perfectly considered as trustful reviewer and his filtered review would be a fake positive.

Inside behavioural features there is another possible division between features obtained by making operations over the Corpus and features extracted directly from user information available in Yelp. The first of them will be obtained by processing information stored in the three main tables of the Corpus (users, businesses and reviews) while the others are the ones obtained with a web scraper from user profile pages in Yelp.

3.4.2.1 Features obtained from analysing relations between users, businesses and reviews

The Corpus contains three tables with information about businesses, users and reviews respectively. Each of its instances has an unique identifier: business_id, user_id or review_id. Yelp consists on users who post reviews on businesses, which means that a review needs a user identifier (its author) and a business identifier (its target). As a consequence, reviews in the Corpus also contain the fields business_id and user_id and so this relation is represented.

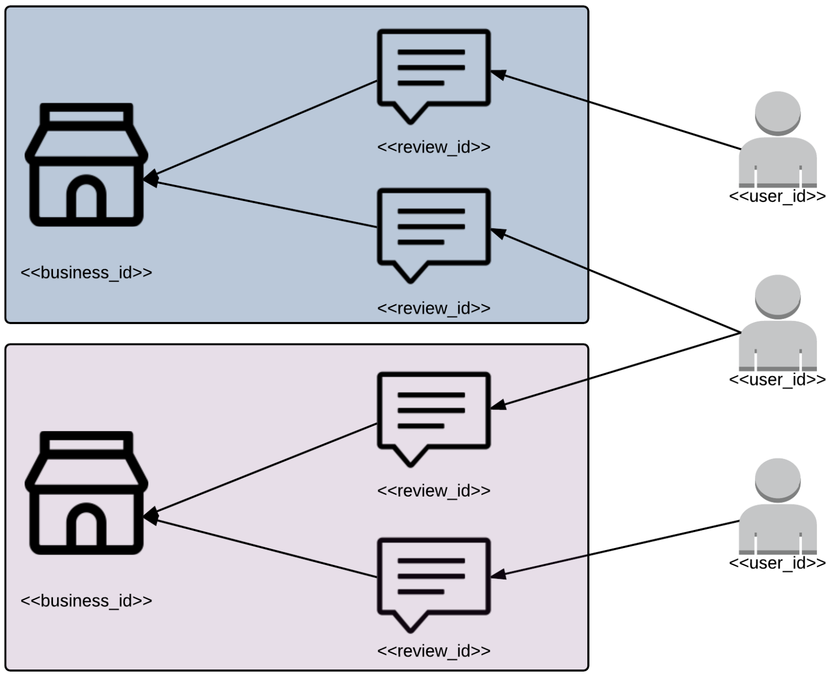

Figure 3.2: Relation between businesses, reviews and users

In the Corpus chapter it was described that there were approximately 7000 reviews used for training (from more than 6000 different reviewers). As behavioural features are related to reviewers, it will be necessary to process information from all their reviews. In addition, some of the behavioural features will need information from businesses and so the businesses table will be also used. All the information will be extracted from the Yelp academic dataset annotated defined in Figure 1. The following paragraphs describe each of the extracted features and some reasons for their extraction.

Average review length: the average word length from all the reviews from the user, so the steps are finding and then splitting them into words. The hypothesis was that trustful reviewers probably care more about giving a complete description of the business while fake reviewers tend to write quick reviews to finish their work early.

User average rating: in Yelp, each review contains an integer rating with values from 1 to 5. It is possible to think that fake reviewers will have extreme rating values (either too low or too high). However, fake reviewers can also give high rating to the ”friendly” business and low rating to its main competitors, so this will be tried to refine with more features as it could result in a more moderated mean which could confuse them with trustful reviewers .

User rating standard deviation: the procedure was getting all the reviews from the user and calculating their standard deviation. For fake reviewers there are two feasible scenarios here: they can give only extreme ratings from one polarity (low deviation) or extreme values with opposite polarities (high deviation) as it was explained before. The idea on trustful reviewers is that they will have values scattered over all ratings. This could mean higher standard deviations compared to the ones from fake reviewers with only high or low ratings, and also lower standard deviations compared to fake reviewers with both high and low extreme values.

Other rating distribution features from users: it is possible to think that probably a better way of getting information about rating distributions is obtaining more concrete information about them instead of means and deviations. The following four features were extracted with the intention of refining a bit more on rating distributions, always having in mind that extracting such specific features could also provoke over-fitting lately (but also considering that this can not be known beforehand). In addition they could be also useful for learning more about rating distributions in Yelp. Ratings between 1 and 2 stars were considered negative, between 4 and 5 positive and the rest as medium. The four of them are related ones with others, so good idea could be using them separately for classification and then deciding which of them gives more information.

- **Percentage of positive reviews**: this could clearly identify fake reviewers paid to improve the reputation of certain businesses.
- **Percentage of negative reviews**: similar case than the previous feature, but with fake reviewers paid to worsen reputation from competitors.
- **Rating values**: this feature determines the combinations of negative, positive and medium ratings given by the user.
- **Only a rating value**: a more concrete version of the previous feature that indicates

the unique rating value used from the user in case he only gives ratings from one category.

Rating deviation from businesses: this feature involves working with the three tables of the Corpus. As the majority of reviews are considered non fake, reviews with a great deviation from the average rating of the business are candidates of being fake. This can be analysed individually from each review or from a user perspective, which is the way elected in this section. There are some considerations that have to be taken for this feature: not every user has the same perception of quality when going to a restaurant. There are multiple factors such as time waiting, price or kindness from employees which have more weight on ratings given by some users than others and which can even be different depending on the day the user uses the service (in restaurant domain, the day he goes to it). For that reason this feature was studied from the point of view of the user: if a reviewer has in a majority of reviews a strange deviation from the mean rating of businesses, the hypothesis is that he is behaving suspiciously. In addition, and as this can be contrasted from various reviews, this hypothesis reduces the randomness of the factors described above which were more probable when analysing each review individually than in groups of reviews (it is possible to wait 10 minutes in a restaurant whose mean wait time is 5 minutes, but less possible than waiting 10 minutes in each restaurant from a group of 20 with a mean wait of 5 minutes).

User review count: The assumption here is that non fake reviewers are users who frequently use the services of Yelp whereas fake reviewers only write reviews when they are ordered. Yelp trustful users even like gaining reputation on the site and writing good quality reviews helps them. There are more features regarding reputation and analysing Yelp also as a social network, but they will be described in the section related to features obtained from the scraper.

Maximum content similarity: this feature gives the maximum cosine similarity between a pair of reviews of the user. The cosine similarity between two vectors or two documents on a vector space is a measure that calculates the cosine of the angle between them, which can be seen as a comparison between documents. The hypothesis for this feature is that a fake reviewer may probably have a template for making reviews faster or even write the same review in more than one business.

Posting more than one review in a day: the supposition made is that a fake reviewer in charge of giving a positive review to a specific business and bad reviews its competitors could do all his work in one day whereas for example trustful reviewers do not usually go to various restaurants on the same day.

Percentage of reviews on weekends: in a study made on Chinese Yelp [9] it was suggested that trustful reviewers write reviews based on their personal experiences and are more likely to post reviews on Sundays after returning from dinner parties or hangouts that happen over weekends. On contraposition, fake reviewers tend to write more in other weekends as they might be paid for it.

3.4.2.2 Features obtained directly from Yelp using a web scraper

Yelp as a social network provides a site for each user where he can post additional information. Additionally, this page also contains user statistics which can also provide valuable information. The general hypothesis for gathering this data is that active Yelp reviewers use this site not only for writing or reading reviews but also for making friends and having a good time. In social networks people try to interact with others and share opinions. However, fake reviewers probably do not focus on the interaction offered by Yelp as they post their reviews as a task from their jobs or even only create their account for a one-day use.

The gathering of new information was done by the scraper described in the Corpus section. These were the extracted fields:

- Number of friends: having friends implies interaction with other users. It implies user activity in Yelp, even if he does not write a great amount of reviews. If the hypothesis of thinking that trustful users are in Yelp to use it as any other social network, this feature may be one of the most relevant ones.
- Number of photos: Yelp reviews can contain photos of the reviewed businesses. Posting a review with a photo makes it more reliable, but if the photo is fake the reviewer could be easily discovered by other users of the restaurant or even the restaurant could report it. This fact makes data about photos uploaded an interesting feature to take into account as graphic evidence of what is being described has an important impact. In this issue there is one thing to be cleared first: reviews filtered by Yelp do not contain photos, which can mean that they did not contain a photo in any moment or that they contained it but Yelp removed it. This is the reason why photos can not be analysed as a characteristic of a review.
- Profile photo: a deep analysis of a user profile photo could give rich information. At first sight it is possible to see some possible inconsistencies like a user with a man profile photo and a woman name, or a profile photo obtained from another photo that can be found easily on the Internet. This kind of situations can either mean

that the reviewer does not want to give personal information or that the reviewer did not made a great effort trying to hide he is a fake reviewer. Another possible thing is identifying if the profile photo is different from the default one. This idea is also related to the one of using Yelp as a social network: users who want to make friends are more open in the sense of giving personal information like showing their faces or at least uploading a photo they like. Users with a profile photo give a sensation of activity on the social network, and this is the point in this feature. The task here was to differentiate the default photo Yelp offers from any photo uploaded by the user. By inspecting the HTML font code of Yelp it is easy to find the link of that default photo, and so this feature results in a boolean indicating if the link associated with the user profile photo is the one of the default URL or not.

- Number of review updates: updating a review can mean various things. A trustful reviewer usually cares of making a great review to gain votes which are translated into reputation. On the other hand, a fake reviewer can also think that a review is not going to be reliable and so he changes its content. Anyway, this can only be known by extracting and proving that feature later.

- Profile description: in Yelp users can write a kind of state below their name in their profile site. One more time, this can be an indicator of activity: users who care about those details are usually seeking that other users will see their profiles.

- First reviews: number of times the user made the first review on a business. In this case the supposition is that the first review has a greater impact than the following reviews. For example, if a business has a negative first review it will have at the beginning a bad reputation that may be traduced into less people using its services. This situation can be reverted, but the initial damage has to be considered. It is different having only a review that is negative than having five positives and one negative. That concept of impact can also be done in the way of giving a positive review on a business site from a friend that has been just created.

- Bookmarks: on Yelp users can save information about the places they most like or are interested in thanks to bookmarks. Bookmarks do not show social activity on the site, but users who have places bookmarked are supposed to be interested on them and so they use Yelp for its principal goal. It is hard to think that a fake reviewer bookmarks places as he will probably only enter the site with the aim of writing fake reviews.

- Tips: apart from writing reviews there is another way of giving opinions called which is known as Tips, which are short texts that appear on top of the page.

As it was said in the Review Centric Features section, Yelp offers the possibility of voting reviews in the sense of being useful, funny or cool. The count of review votes is visible on each user site and gives clues about reviewing quality. However, this information can be an unfair advantage for classification as filtered reviews can not be voted once they are filtered, and for that reason votes information was not included as a behavioural feature neither.

3.4.3 Conclusions

Having a good Corpus is necessary but not sufficient for making a good classifier. A Corpus contains raw data which has to be processed in order to obtain valuable information from it. This task has some problems that have to be solved in order to obtain a successful feature extraction. Organising correctly this information is always important, but in the case of having a large Corpus there is another issue that has to be taken into account: time taken for feature extraction. The step after having those features extracted will be experimenting how they perform in the creation of the classification model. However, if this extraction was made everytime the program runs, this would suppose a great waste of time. It took hours to obtaining behavioural features of section 4.2.1, so doing these operations more than one time makes the project unapproachable. The solution was creating a new table in the database to store all user features. As in the Corpus there was information about users completed with information obtained with the scraper, the new table contains a copy of the information stored for each user identified by a user_id and was completed with the features extracted with features from section 4.2.1.

The result of this section is, as it is shown in the figure below, a set of reviews (that has been partitioned into a training a testing set), connected to a database containing behavioural features from each user of those reviews obtained as it has been explained in this section.

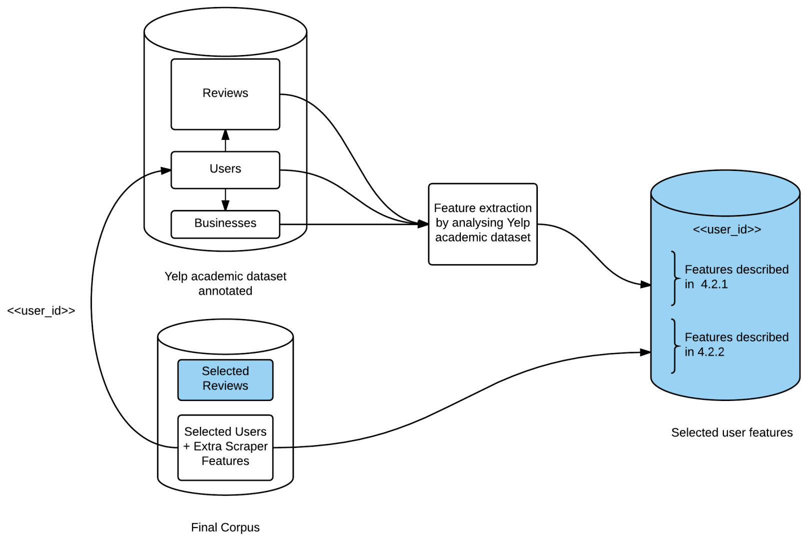
Reviews
Users
Businesses
Yelp academic dataset annotated
Feature extraction by analysing Yelp academic dataset
<<user_id>>
Features described in 4.2.1
Features described in 4.2.2
Selected user features
<<user_id>>
Selected Reviews
Selected Users + Extra Scraper Features
Final Corpus

Figure 3.3: Creation of new database containing user behavioural features

3.5 Classification model

Feature extraction results in a dataset which contains all the information that can be used for classification. However, it is impossible to know if this information will be useful before extracting it. The next task is obtaining a good classification model, and for this there are two key tasks: selecting the best classification algorithms (and optimizing its parameters) and the features that give more information for classifying. A common problem is called over-fitting and consists on training the classifier with too many features so that the classifier is too much adapted to the training set. This causes that the classifier learns by heart the training examples, but is not capable of classifying new ones.

These are some examples of classification algorithms that were used in this project:

- Ada Boost: meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. [3]
- Support Vector Machines: they are based on the concept of decision planes that define decision boundaries and have a cost parameter (C) which has to be modified in order of assuming more or less errors while deciding the boundaries.
- Random Forests and Extra Trees: they are based on the growth of many classification

trees and to classify a new object from an input vector, each tree gives a classification which is translated into a vote [5]. Each tree is made from a random subset obtained from the former training set, so the first objective of the algorithm is dividing the training set into M subsets that will be used to create M decision trees. When those trees are created, each of them will vote which class the instance belongs, and then the forest chooses the class with more votes over all the trees. This algorithm uses bootstrap aggregation or bagging, which is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms, also reducing variance.

- K-Nearest Neighbours: this algorithm is considered lazy learning because the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority vote of its neighbours with the object being assigned to the class most common among its k nearest neighbours.

- Gaussian Naive Bayes: algorithm that applies Bayes theorem with strong independence assumptions between the features.

All the results of the following experiments were obtained by using a ten-fold cross validation. This evaluation method consists on splitting the training set into ten subsets, and evaluating them in ten iterations. On each of those iterations, the test set will be one of the ten subsets that were made, and the other nine subsets will make up the training set. At the end, each of those subsets will have been tested with a training set of the other nine. The results will show the accuracy of the model, which means the percentage of correctly classified instances from the total number of instances.

3.5.1 Classification algorithm selection

As it is impossible to know which algorithm will perform best beforehand, the first thing to do is running a experiment proving the most common classification algorithms in order of getting a first idea of which are the best ones. As this experiment is carried out using all the features that were extracted, the results have to be later contrasted using the selected features. This first approach is useful to guide the following steps of the experiment and as in this case there are two types of features as it has been explained in previous chapters (review centric features and behavioural features), it will also show if both of them work equally or if one type works better than the other. All these experiments were done with default parameters offered by Scikit-learn.

The results obtained were the following, expressing the mean of the accuracy for each of the ten folds and their deviation :

Classification algorithm	**Accuracy Review Centric**	**Accuracy Behavioural**
LinearSVC	0.51 (+/- 0.02)	0.7359 (+/- 0.08)
Gaussian Naive Bayes	0.53 (+/- 0.03)	0.7261 (+/- 0.02)
K-Neighbors	0.52 (+/- 0.02)	0.8522 (+/- 0.02)
Decission Tree	0.53 (+/- 0.02)	0.8457 (+/- 0.01)
Logistic Regression	0.60 (+/- 0.02)	0.8489 (+/- 0.1)
Random Forest	0.55 (+/- 0.02)	0.8895 (+/- 0.01)
Ada Boost	0.60 (+/- 0.02)	0.8847 (+/- 0.01)
Extra Trees Classifier	0.56 (+/- 0.03)	0.8627 (+/- 0.03)

At this point, the algorithms that performed best were Random Forest and Ada Boost with behavioural features. However, as all the algorithms were proved with their default parameters it is possible that the best ones change after optimization.

The difference between the results obtained from using behavioural and review centric features is important, but it is possible that training with both of them together can improve the results. A first conclusion from this section is that detecting fake reviewers is mainly based on analysing their behaviour.

To conclude with this subsection, there was a trial of standardization and normalization of features, but the results obtained were clearly poorer. This had an explanation: outlier values on features may play an important role when trying to classify suspicious behaviours, so making transformations on the data can hide those outliers and thus a lot of information is lost.

3.5.2 Feature selection

Selecting the best features to train the classifier is not a simple task if the number of features grow. In a set of N features, there are subsets of N, N-1, N-2...1 features to prove. With a high number of N, proving all these subsets is a hard task. Scikit learn has its own feature selection methods, but this project proposes a new algorithm for this task which achieved slightly better results than Scikit methods. This algorithm is based on looking for feature importances on classification in order to remove the less important features, and its complete explanation is on the Appendix A.1.

3.5.2.1 Review Centric Features

There were only seven review centric features used, so making feature selection in this situation did not improve results. However, it is possible to make an analysis of feature importances to see what are the most weighted features for the classifier. This experiment was carried out using Ada Boost Classifier, as it was the one which best worked.

Feature	**Importance**	**Feature**	**Importance**
Number of Words	0.30	First vs Second Pronouns	0.26
Number of capital letters	0.20	Words per sentence	0.14
Number of paragraphs	0.10	Exclamations vs Questions	0.02

3.5.2.2 Behavioural Features

As behavioural features were the ones that best performed, an important task was looking if there was a margin of improvement regarding them. The experiments were made using the proposed feature selection algorithm, and using the classifier that performed best which was the Random Forest Classifier with default parameters, the accuracy results obtained over all the iterations were the following, being the X-axis the number of features that were deleted from the original set:

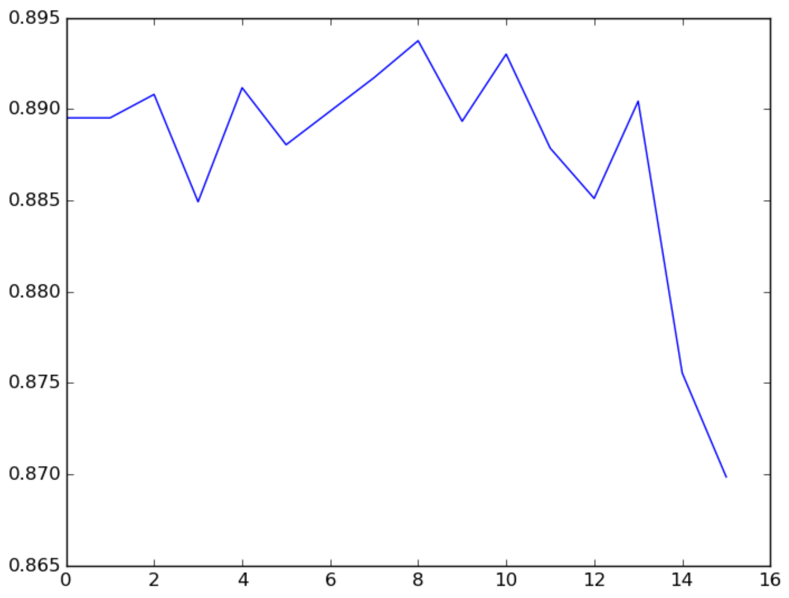

Figure 3.4: Random forest accuracies over the first part of the feature selection algorithm.

The best accuracy obtained was 0.8936 after removing eight features. After adding the discarded features one by one following the second part of the algorithm, there were no improvements so this was the best accuracy obtained at this point. The features selected and their importances for the classifier were the following:

Feature	**Importance**	**Feature**	**Importance**
Review Count	0.2504	Friends	0.1985
Reviewer average length	0.0997	Has profile photo	0.0807
Number of photos	0.0766	Firsts	0.0753
Bookmarks	0.0708	Tips	0.0528
Percentage positive reviews	0.0274	Average rating	0.0271
Updates	0.023473	Weekend percentage	0.0188

The hypothesis made about treating Yelp as a social network seems to be certain by seeing those feature importances, as the number of friends is the second most relevant feature for the classifier and features like having profile photo or the number of uploaded photos are also important. Moreover, with only ”social” features, Random Forest and Ada Boost had an accuracy of 0.86, and LinearSVC obtained a great growth having an accuracy 0f 0.83. Without using any of the ”social features” the accuracy decreased more in general terms: the Random Forest achieved an accuracy of 0.81 and the best result was obtained by Ada Boost with 0.84.

The most relevant feature is review count, and the evolution of feature importances until the maximum accuracy was reached is shown on the table below. There are some curious facts such as the evolution of the number of friends importance (purple line) When the self standard deviation is removed (second feature removed), its importance increases from less than 0.2 to more than 0.3. However, when the removed feature is having or not a profile description, its importance goes back to 0.2. Another interesting fact is the importance of the number of first reviews (blue line): when there is only one feature deleted, it is the second most important feature even above review count (orange line), but then it oscillates and it is finally the sixth most important. Regarding the average review length feature, it remains almost constant over all iterations with a value of around 0.09, son it seems that it was not really affected by other features.

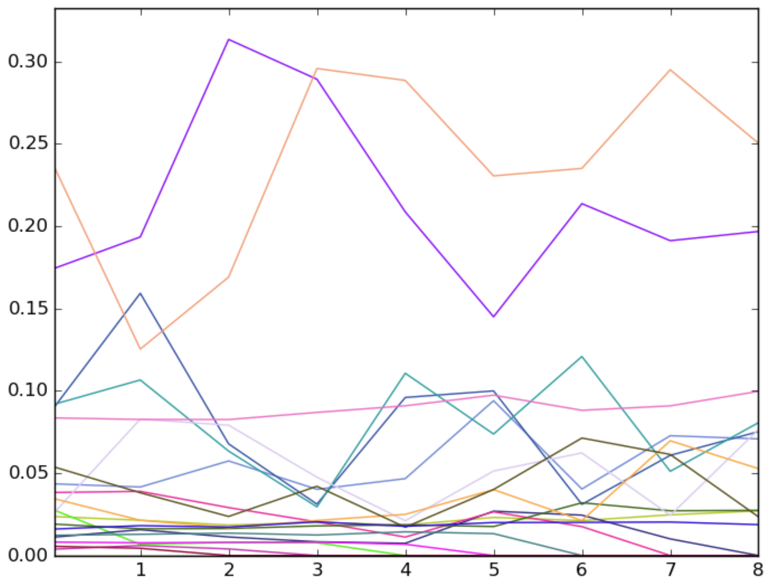

Figure 3.5: Evolution of feature importances until best accuracy was reached.

There were also improvements when proving the feature selection with other algorithms, but the best accuracy was still achieved by the Random Forest. Ada Boost obtained an accuracy of 0.8875, and the Extra Tress Classifier almost reached 0.87.

3.5.2.3 Combining behavioural and review centric features.

As it was shown on previous sections, there was a huge difference between the accuracy obtained when using behavioural or review centric features. That difference makes difficult combining both of them and obtaining better results as review centric ones do not really perform well. A little improvement was achieved, and as review centric features focus on the review and not only on the user, the will probably discriminate better between reviews of a reviewer (in the case that the reviewer has posted both fake and non fake reviews). Another trial was training first a classifier with only review centric features, and using the output of that classifier as an input of a classifier trained with all the behavioural features. The results were the same as the ones obtained with only behavioural features, so the first method was chosen.

The feature importances on this case were the ones shown in this table, achieving an accuracy of 0.8961:

Feature	**Relative importance**
Review Count	0.2881
Friends	0.1576
Updates	0.0931
Number of photos	0.0733
Reviewer average length	0.0484
Review Length	0.0459
Tips	0.0433
Bookmarks	0.0423
Only a rating value	0.0336
Number of capital letters	0.0327
Words per sentence	0.0326
Has profile photo	0.0276
1st vs 2nd pronouns	0.0270
Firsts	0.0250
Exclamations vs questions	0.0162
Percentage negative reviews	0.0134

The two most important features remain the same as when using only behavioural features, confirming that activity on the site discriminates properly if a user writes fakes reviews (case in which the only reviews written are the ones he is told to write) or if he uses Yelp in the right way reviewing the businesses he comes through. This theory is reinforced by the extra possibilities offered by Yelp to make friends on it: users who are looking for friends in Yelp tend to be trustful users. The most important review centric feature when combining all of them is the review length, with almost the same weight as the reviewer average length of all his reviews. The biggest curiosity is the new influence of the number of updates, which was before almost the last one of the chosen ones, but now is the third most important. In addition, as in the previous experiment using only behavioural features, the maximum cosine similarity between reviews of the user did not have a great impact. From this fact we can suppose that there it is not common seeing duplicates or near duplicates of reviews from the same user, so fake reviewers do not use templates to write their reviews more quickly.

The best new of the results combining both review centric and behavioural features was not the improvement of the accuracy results obtained with the ten fold cross validation (this improvement was really low, and Random forests have a little variance on its results depending on which trees are created) but the inclusion of features which only depend of the review. This will probably help discriminating between reviews of a user that has posted both fake and trustful reviews.

3.5.3 Classifier parameter tuning

All the results obtained in previous subsections were obtained using default parameters offered by Scikit-learn for each classification algorithm. These parameters can be modified in order of improving the accuracy of the classifier depending on the input data. The best results were obtained using a random forest, but the goal of this subsection is going to be improving all the classifiers used before to see if parameter tuning makes some of them achieve much better results and also because the final try of improving accuracy is going to be combining all the classifiers used, which will be done in next subsection. As each algorithm has a different basis, they will also have different parameters (except for Gaussian Naive Bayes, which has not parameters to be modified). All the documentation about parameter adjusting was found on scikit-learn.org, and the figures representing the following processes can be found on the Appendix A.2.

3.5.3.1 Random Forest and Extra Trees Classifier

The main parameters to be adjusted offered by Scikit-learn are the following (in italic will appear the name given by Scikit):

- Number of trees in the forest (*n_estimators*): the larger, the better but it will also take more time to be computed. Additionally, the results will stop improving after a critical number of trees.
- Size of random subsets of features to consider when splitting a node of the tree (*max_features*): the lower the greater reduction of variance, but also the greater the increase in bias.

One way of optimizing Random Forest results is measuring the out-of-bag (OOB) error. As it was previously explained, in Random Forest classifiers each tree has its own training subset or bootstrap sample. The out-of-bag error is the average error for each instance calculated using predictions from trees that do not contain the instance in their respective bootstrap sample. This fact allows the Random Forest being fit and validated while being trained [5].

The minimum OOB error was obtained with 80 trees and no maximum number of trees, reaching a classification accuracy of 0.9006.

In the case of Extra Trees Classifier, the procedure followed was the same. The results grew significantly compared to the ones obtained with default parameters, obtaining an accuracy of more than 0.89 using 209 estimators and without maximum number of features.

3.5.3.2 Ada Boost

Ada Boost on Scikit-learn is implemented with the algorithm AdaBoost-SAMME [18], which has two versions: the real boosting algorithm and the discrete. The results obtained modifying the number of estimators were almost the same with all of them, but the least error was achieved using the Real algorithm with 400 estimators.

3.5.3.3 Nearest Neighbors

In the case of nearest neighbors, the most important parameter to optimize is the number of neighbors that are going to be used to classify each distance. The optimal number of neighbors or k was 10, reaching an accuracy of 0.852941.

After selecting the best k parameter, the following task was searching the optimal weighting function of the k-nearest neighbors which classify each instance. There are two possible functions offered by Scikit-learn to carry out that task:

- Uniform, which weights every neighbor equally.
- Distance, which weights points by the inverse of their distance.

The best one in this case was the Uniform function, which was the one used by default, so the accuracy did not improve modifying this parameter.

3.5.3.4 Linear SVC

The goal of parameter selection in the case of Support Vector Machines is obtaining a risk minimization for the following equation:

$$C \sum_{i=1,n} \mathcal{L}(f(x_i), y_i) + \Omega(w)$$

where

- C is used to set the amount of regularization
- L is a loss function of our samples and model parameters
- Omega is a penalty function of our model parameters

The C parameter tells the SVM optimization how much it has to avoid misclassifying training examples. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. On the other hand, a small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

If we consider the loss function to be individual error per sample, then the data-fit term, or the sum of the error for each sample, will increase as we add more samples. The penalization term, however, will not increase.

The optimal value obtained for C parameter was 0.01, with an accuracy of 0.848162, which means an increase of almost 0.1 from the default cost parameter.

3.5.3.5 Logistic Regression

In logistic regression there is also a C parameter as in the case of support vector machines. The best accuracy obtained was 0.8506 with a C value of 0.07.

3.5.4 Ensemble of classifiers

The goal of ensemble methods is to combine the predictions of various estimators build with a given learning algorithm in order to improve generalizability over a single estimator. There are two families of ensemble methods which can be distinguished:

- In averaging methods, the principle is to build several estimators independently and then to average their predictions. The combined estimator is usually better than the base estimators individually as its variance is reduced. Examples of this are bagging methods used for example in Random Forests.

- In boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The goal is to combine several weak models to produce a powerful ensemble. An example of boosting has been used in this project with the Ada Boost classifier.

This section will try improve the results obtained by the tuned classifiers of the previous section by combining all of them using a Voting Classifier. The idea behind the voting classifier implementation is to combine conceptually different machine learning classifiers and use a voting system to predict the class labels. This model can help avoiding the weaknesses of the individual classifiers. The voting can be made in two ways:

- Majority/Hard Voting: the predicted class label for a particular instance is the class label that represents the majority of the class labels predicted by each individual classifier.

- Weighted Average Probabilities (Soft Voting): it returns the class label as argmax of the sum of predicted probabilities.Each classifier will have a weight, and after each classifier predicts the class label, their class probabilities are multiplied by the classifier weight and averaged. The final class label is then obtained from the class label with the highest average probability.

Training results using the ensemble were not satisfactory as the results obtained were below the ones obtained only using the random forest classifier.

3.5.5 Final results

This section will show the final results obtained from using the set which was saved for test. With this set, which was formed by 2538 instances, the accuracy obtained was 0.861308116627. This accuracy was lower than the one obtained in training results, so it is possible that some over-fitting was introduced. Another important metric to be analysed is the confusion matrix, which shows the mistakes made on classification.

	Classified as Fake	**Classified as Trustful**
Fake	1078	191
Trustful	161	1108

The confusion matrix shows that 46% of mistakes in classification were made classifying trustful reviews as fake, while 54% were made classifying trustful fake reviews as trustful. As the test set had the same number of trustful and fake reviews, the classifier has learned only slightly better how to classify trustful reviews than fake ones. From this fact, it is possible to conclude that trustful reviews will almost always be correctly classified but some fake reviews will be classified as trustful ones by mistake. This could be seen positive for common users as they know that the trustful reviews the post will not be filtered, but the cost they have to pay is that some of the reviews they will read may be fake. The cost assumed by misclassifying fake or trustful reviews should be studied by every enterprise that filters content on their sites, and they might design their filters taking this into consideration. The following metrics are indicators for those issues:

	Precision	**Recall**	**F1-score**	**Support**
Trustful	0.87	0.85	0.86	1269
Fake	0.85	0.87	0.86	1269
avg/total	0.86	0.86	0.86	2538

Where,

- Precision: ratio between true positives and true positives plus false positives. The precision is intuitively the ability of the classifier not to label as positive a label that is negative: $\frac{tp}{tp+fp}$.
- Recall: ration between true positives and true positives plus false negatives. The recall is intuitively the ability of the classifier to find all the positive samples: $\frac{tp}{tp+fn}$.
- F1-score: can be interpreted as a weighted average of precision and recall, whose relative contributions are equal: $\frac{2*(precision*recall)}{precision+recall}$.

To conclude with this chapter, it has been shown that data taken from social networks can be used for obtaining conclusions that a human eye could not deduce. For the specific goal of detecting fake reviews, despite the information seems to be on the text written the majority of clues given by data are not really on the text but on external features. So, a task that intuitively could be done by anyone who reads the review is better carried out with an automatic system that analyses all the data that surrounds that specific text.

CHAPTER 4

Recruitment

4.1 Introduction

Due to the power of communication offered by Internet and its unregulated nature, terrorist groups have started using it in order of recruiting new members since the last decade. According to recent research made by Torok, cyber tools are most influential at the onset of the activity of a future member of an extremist group, which is the recruitment and radicalization phase [16]. As the Internet offers the possibility of creating online communities without any entity censoring them, terrorist groups take rid of it in their aim of spreading literature and training materials for their members or future members. These organizations engage in directed communication and advertising, recruiting members on sites like Second Life, Facebook and radicalised religious web forums [16]. An investigation report on FBI counterterrorism intelligence failures leading up to the Fort Hood shooting on November 7, 2009 cited a data explosion and workload as contributing factors to analyst and agent oversights. As there were nearly 20000 electronic documents that had to be analysed, classification methods would be useful for pre-processing text documents and reducing the workload of human analysts [17].

In this project, as in an article made by Jacob R Scanlon and Matthew S Gerber

[15], a violent extremist group will refer to an organization that uses violent means, like terrorism, to disrupt legitimate authority, whereas terrorists are common types of violent extremist groups that act with the specific goal of influencing public opinion or inciting political change. Furthermore, violent extremist recruitment is any attempt by a group or individual involved in violent extremism to recruit, radicalise, or persuade another person to aid a violent movement and, consequently, cyber-recruitment is the recruitment activity that makes use of the Internet.

The goal of this chapter will be identifying recruitment activities of violent extremist organizations within online social media, specifically messages recruiting individuals for joining those organizations. As in the case of Yelp, machine learning techniques will be applied to classify instances between two classes which are in this case messages belonging to cyber-recruitment or not. For these classification purposes, a violent extremist cyber-recruitment message is any message that attempts to persuade the reader to join a violent extremist organization. This classification system can provide help for identifying the intention or incitement of violent activity within online communities.

4.2 Corpus

The corpus used in this project was a result of the experiments carried out in [15]. In this article, the data analysed was obtained from the so-called "dark web" content, which is defined as information from typically private social websites where extremists interact. The requirements for the corpus were the following ones:

- The collected data should come from popular sources where violent extremist groups and their sympathisers were in contact.
- The collected data should cover a contemporary time-frame in order to be considered relevant to contemporary anti-extremist efforts.
- The collected data must use the English language or be translatable to English using an automatic process.

The data was collected from a repository of social media messages compiled from 28 online discussion forums called The Dark Web Portal. Those forums focus on extremist religious and general Islamic discussions, many of which sympathise with radical Islamic organizations. The Dark Web Project provides translation services and has information from at least seven English-language forums being the most relevant the ones from the

Ansar AlJihad Network, a set of invitation-only jihadist forums in Arabic and English known to be popular with western jihadists. There are a total of 299,040 messages from Ansar AlJihad obtained between 2008 and 2012 in The Dark Web Project. From all those messages, 29,492 belonging to 382 members are compiled from English forums known as Ansar1. This subset contained contemporary English discussions between jihadists and jihadist sympathizers, but they were not annotated. For annotation, independent judges gave their point of view about 192 posts and validated the labelled data with an specific algorithm, and afterwards there was an additional annotation of 100 posts. With those annotated posts obtained in [15], they trained a classification model that achieved great accuracies for those posts with a cross validation evaluation and from which the rest of the almost 30,000 posts were classified lately.

This corpus was published on a csv format and contains the following fields for each post:

- messageID: Identifier of the post.
- threadName: Name of the thread in which the post was written.
- threadName: Identifier of the thread in which the post was written.
- authorName: Name of the user who wrote the post.
- authorID: Identifier of the user who wrote the post.
- dateYMD: Date when the post was written in YMD format.
- dateTime: In theory, it should contain the hour when the post was written but all of the instances have the same hour, so no information can be extracted from this field.
- messageBody: Text of the post.
- recruitment: Class of the post. If it belongs to recruitment class, this field is 1 and 0 if not.

From all of the posts contained on the dataset 2337 were labelled as recruitment messages, which means that less than a 10%. As the corpus was initially so unbalanced, the strategy was choosing all recruitment messages for the experiments and adding the same amount of non recruitment messages. In this chapter there will be features extracted from creating vocabulary models as it will be described in the following section. These models are created from the training corpus, and evaluating them with cross validation would be a difficult task as for each fold a model should be created from the rest of folds. For this reason, training evaluation will be carried out having a reserved subset for it. With this subset, all training experiments can be done without learning from the final test set, which will be only used to evaluate the final classification model. The dataset was partitioned into training and testing sets with a proportion of 75%-25% and inside the training dataset a 15% was reserved for validating training experiments.

Set	Non recruitment posts	Recruitment posts
Train	1490	1490
Train validation	263	263
Test	584	584

4.3 Feature extraction

As the corpus contains information about a forum site, the greatest amount of information will be extracted from the text from each post. In the case of Yelp it was possible to obtain a great number of valuable features related to the behaviour of each user, but in this case the information about users is more limited as there is no extra information such as profile description, photos or followers. In addition, it is reasonable to think that recruitment messages are more evident than fake reviews as a recruitment message has the aim of attracting the user while fake reviews are written trying to hide its real intention. For this reason, this section will explain techniques for making a deeper language analysis and additionally there will be a behavioural feature extraction in order to see if those features can also discriminate if a user recruits or not. As in the chapter of Yelp, features will be classified into review centric and behavioural.

4.3.1 Post centric features

This subsection will describe the techniques that were followed to extract features regarding the content of each post. The goal of online recruiters is spreading their message to achieve the greatest number of sympathisers, so the language they use may be different from the language used by others. Additionally to the differences on language used by recruiters and non recruiters, textual metrics can also be crucial to classify both types of users.

4.3.1.1 TF-IDF

The features obtained from this technique are, for each document, a vector with the TF-IDF of each of the terms of the vocabulary obtained from all the training documents that appear on it. As the model has to be obtained from all the training set, it is not going to be evaluated using cross validation. Obtaining TF-IDF features has the aim of detecting words that are more relevant or used in a class than in other, that is to say, modelling the vocabulary used by recruiters and non recruiters.

4.3.1.2 LDA

LDA stands for Latent Dirichlet allocation and is a topic model that generates topics based on word frequency from a set of documents, useful for finding accurate mixtures of topics within a given document set.

For doing this, as in the case of the TF-IDF, the first task is preprocessing data by removing stop words and getting the lemmas of terms.

The following step is creating a document term matrix with the count of appearances of words (once again, similar to the case of TF-IDF). Once this is done, the algorithm obtains a number of topics according to its most used words having specified the number of topics and words per topic beforehand. So, for each topic, there is a list of words with their proportional factor related to their weight on the topic.

Once the model of topics with their most important words is created, it is possible to obtain which are the topics related to new text entries and their weight or probability. This can be an interesting feature as probably recruiters write more often about certain topics than non recruiters.

4.3.1.3 Mutual information

In probability and information theory, the mutual information of two random variables is a measure of mutual dependence between two variables. The concept of mutual information is linked to the entropy of a random variable, a notion in information theory that defines the amount of information within a random variable. The definition of information in information theory can be expressed as [10]:

$$I(x) = log[\frac{1}{p(x)}]$$

The smaller the probability of event x occurs, the greater amount of information it provides. This fact, applied to text classification, means that lower frequency words will have a greater weight. For the case of mutual information, the formula is:

$$I(x, y) = log[\frac{p(xy)}{p(x)p(y)}]$$

In this case, y will be the class the text belongs to (recruitment or not), x will be each term appearing in the the posts set and the probability of a term appearing in a document of class y is denoted as p(xy).

The idea for this technique was crafting a dictionary with the mutual information between each word and each class, that is to say, the entropy o a word of belonging to a recruitment posts and to non recruitment posts. With that, we could obtain features such as mutual information between a word and recruitment class, a word and non recruitment class and the difference between both entropies.

4.3.1.4 Parts of speech

Another way of analysing linguistic features is focusing on lexical categories of words. The aim here is collecting the statistics about those word categories from the training dataset and see if recruitment messages have specific patters that differ from non recruitment messages. The parts of speech to be analysed are adjectives, adpositions, adverbs, conjunctions, determiners and articles, nouns, numerals, particles, pronouns, verbs and punctuation marks.

4.3.1.5 Other review features

Apart from the techniques described previously, there are other features that can be extracted from texts which may make help in the task of classification.

- Number of words of the post: large posts tend to be seen more than short posts and contain more information, so a hypothesis is that recruiters may write larger reviews than non recruiters in order of giving more powerful arguments with higher visual impact.
- Number of capital letters: writing on capital letters on forums is usually seen as shouting. An outlier number of capital letters may mean that the user is trying to get the attention of readers for his posts among others, and this could be done for spreading recruitment messages easily.

- Quotes: it is possible to quote other users who have written previously on the thread to interact with them. This means direct communication between both users, so quotes could be used for recruiters to personally persuade other users for their cause. Quotes have a specific format on the forum which is "Quote: Originally Posted by (nick of the user)", so it is possible to detect them.

4.3.2 Behavioural features

In a forum, users write posts on threads created by them or by other users. The only activity made by users is creating or replying to posts, so the information related with their behaviour is more limited than in other types of social networks in which there are another concepts such as following a user or liking some content. The features that were extracted in this subsection were in majority an adaptation of review centric features for users, looking for some patterns that can discriminate between a recruiter and another user. It is important to take into account that not all the messages from a user who recruits are labelled as recruitment messages.

- Number of posts: the activity of users on the forum can be a good indicator for detecting people who are trying to recruit. A user with a great presence on the forum may be trying to get rid of the messages he is posting, and his account may be even used by more than one person.
- Number of threads created: another feature that may indicate a high presence on the forum. The corpus does not have a field to indicate the creator of the thread, so for this project it will be supposed that the user who opened the thread was the first of the ones who posted on it. As the date field is available, this is possible to do (with an error of a day, as the date field has days as its maximum granularity).
- User similarity between posts: with this feature it is possible to detect whether a user spams some specific texts or sentences that can be repeated trying to convince people to join their organization.
- Average number of capital letters: as it was said before, a user who writes in capital letters may be trying to attract other users to read his posts. In this case, this feature will be associated with the tendency of the user of writing in capital letters instead of analysing posts individually.
- Average number of words: the average number of words written by the user on each post may be significant as a recruiter may probably tend to write larger texts to

persuade better other users or, as with capital letters, to have a greater visual impact on the site.

- Average number of quotes: it was also described on the subsection regarding other review centric features, but in this case the feature will compute all the quotes made by the user.

4.4 Classification model

This section will show the accuracy of each of classification using the feature extraction techniques which were mentioned previously. As in the case of Yelp, there will be several classification algorithms used for this task and after proving all of them, the aim will be getting the best ones for each experiment and trying to combine different types of features in order of improving the accuracy obtained. However, as the dataset has a low number of recruitment posts and users who recruit have a lot of recruitment posts, it was not possible to prove behavioural features as both training and test datasets would have the same instances as they share users.

For **post centric features** several experiments were carried out. The idea was evaluating separately all the techniques mentioned previously related to review centric features and then trying to combine them in order of achieving a greater accuracy. As it was said before, all experiments were done using the training set, creating the model with the 85% of it and evaluating it with the other 15%. Additionally, the models created from the set vocabulary will be analysed to see if there are some clear patterns that indicate if a text has recruitment intentions or not. All these experiments were evaluated selecting default parameters for each classifier, and the ones with greatest results will be modified afterwards.

- Only using the TF-IDF the results almost reached an 80% of accuracy with the default classifiers in Scikit.
- For the case of mutual information, the results were similar to the ones obtained with the TF-IDF, but the classifiers which best performed were not the same.
- With LDA topics the results were worse, but still near the accuracies obtained with the other techniques.
- Using Part-Of-Speech tags did not perform as well as the other techniques, obtaining less than a 55 % of accuracy. For that reason, from now on this technique will not be used in any experiment.

- The following step was trying to combine the three techniques selected, that is to say, using TF-IDF features, topic features obtained from the LDA Model and weights obtained from the dictionary based on mutual information between words and classes. This experiment was also done with the models that were created before and validated with the validation set. As it can be seen in the table below, results slightly improved reaching even more than 80% of accuracy with the Linear SVC. However, the contribution in number of features from each of the techniques was really imbalanced (the LDA model contributes with 20 features related to the topic importances and the mutual information gives three features, whereas TF-IDF provides a feature for each of the words that appeared on the corpus that have not been filtered). In the next section there will be a feature selection to delete noise especially from those features extracted from the TF-IDF model.

Classification algorithm	**Acc. TF-IDF**	**Acc. M.I.**	**Acc. LDA**	**Acc. all**
LinearSVC	0.7985	0.7832	0.7547	0.8023
Gaussian Naive Bayes	0.6787	0.6749	0.7281	0.7490
K-Neighbors	0.7414	0.7909	0.7680	0.7928
Decission Tree	0.7490	0.7757	0.7319	0.7738
Logistic Regression	0.7909	0.7909	0.7585	0.7985
Random Forest	0.7833	0.7699	0.7357	0.7890
Ada Boost	0.7795	0.7738	0.7623	0.7852
Extra Trees Classifier	0.7699	0.7662	0.7319	0.7947

4.4.1 Feature selection

As it was explained before feature extraction results in instances with 8203 features, being 8180 of them related to the TF-IDF, 20 to the LDA model and 3 to mutual information. This means that there are two possibilities of selecting features: selecting them for each technique and then rejoining or selecting the best of all of them joined. The experiments were carried with the feature selector from Scikit called SelectKBest. In this case, the feature selection method proposed in the previous section was not used as the results obtained were worse than with the method os Scikit.

With the first method, the results slightly improved reaching the 80% of accuracy with various classifiers (the maximum was obtained with Random Forest, resulting in a 0.8137). The 8180 TF-IDF features were reduced to 350 and the LDA features were reduced from 20 to 10. This, adding the mutual information features, made a total of 363 features.

The second method, which was selecting features from the three techniques all together, also improved results generally but in a smoother way. However, the best one was almost the same as the one obtained with the other method (0.8136 with Logistic Regression). These results were achieved by reducing the total number of features to 300.

The results of feature selection were positive for both methods because even if the improvements were not really significant, the dimensionality of the problem was reduced and so the probabilities of having over-fitting afterwards are smaller.

4.4.2 Classifier parameter tuning

The majority of classifiers obtained a similar accuracy so modifying their parameters may be crucial for obtaining one that clearly performs better than the others. In this section there will be a trial of improving all those classifiers as in the following one they will be combined with an ensemble. The way of tuning the classifiers will be done as in the case of Yelp. The best results at this point were obtained with the first method of selecting features, so all the experiments shown here were the ones done this way. The figures of the process are shown on the Appendix A.3.

- With Random Forest and Extra Trees Classifiers the parameters were selected by calculating the Out-of-bag error rates, obtaining the optimal number of estimators ln 279 for Random Forest using the 'log2' parameter for the maximum number of features per tree and 336 estimators por Extra Trees without maximum number of features per tree.

- In the case of Linear SVC and logistic regression, the strategy was modifying the cost to obtain the best accuracy. The results were a cost of 0.39 for Linear SVC and 0.7 for Logistic regression.

- With the Kneighbors model the goal was again getting the best number of neighbors from which the algorithm decides its output. The best K obtained was 9.

- Finally, with Ada Boost the results indicated that the best one was the Real algorithm against the discrete using 350 estimators.

4.4.3 Ensemble of classifiers

The first experiment was carried out using the parameters obtained from the previous section for each classifier. The Ensemble Classifier having the rest of classifiers as inputs obtained an accuracy of 0.8042, which was under the results obtained from the Random Forest. Another experiment was done with an Ensemble classifier which had the other classifiers with default parameters as inputs, and it obtained the same result, that is to say, the Ensemble performed best with classifiers which performed worse separately. However, as these results were similar, the final classifier will be the result of using the Ensemble classifier with the tuned classifiers. It is also important to mention that in a previous experiment, Random Forest achieved more than a 81% of accuracy.

4.4.4 Final results

The selected classifier used to evaluate the test set was the Ensemble Classifier with the tuned classifiers, which were Linear SVC, Logistic Regression, KNeighbors, Gaussian Naive Bayes, Decission Tree, Random Forest, Extra Trees and Ada Boost. This model achieved an accuracy of 0.8287, which is higher than all the accuracies obtained previously with the validation set. This is a great fact as there has not been over-fitting from the training data.

The confusion matrix shows that the classifier performed almost the same way classifying both classes, it had only eight more errors classifying non recruitment messages as recruitment than viceversa. The rest of metrics can also be seen on the second table, which are the same than the ones used in the case of Yelp.

	Classified as N.R.	**Classified as R.**
Non Recruitment	480	104
Recruitment	96	488

	Precision	**Recall**	**F1-score**	**Support**
Non Recruitment	0.83	0.82	0.83	584
Recruitment	0.82	0.84	0.83	584
avg/total	0.83	0.83	0.83	1168

As a conclusion for this chapter, textual features have worked really well for this task. Thanks to these techniques, a computer can make a classification which is partly subjective. It is true that a human could also make this classification, but a computer can process thousands of texts in only a few seconds. Consequently, with this method it is possible to find recruiters on social networks with an automatic way of receiving data such as a scraper instead of using people looking for suspicious messages, which is a hard task in time and effort.

Another important point is that textual information can be easily interpreted by computers by transforming it into vectors. With this, the computer behaves almost like a brain, getting conclusions from data inputs by making relationships between them. Thanks to that, computers can really understand natural language for the purpose they are trained.

CHAPTER 5

Conclusions and future work

The results obtained on this project have been really encouraging, surpassing an 80% of accuracy on the two cases which were studied. The first objective, which was obtaining a reasonable high accuracy, was accomplished. However, this project had another objective going a step further: showing the relevance of data nowadays. Everyday millions of people send data over the Internet, and as it has been shown there are uncountable conclusions to be taken from it. The special attractive of this project was that the classes to be classified were not strictly objective but a interpretation of texts. Detecting fake reviews could not be done most of the times by reading the review, and in the case of recruitment the computer should really understand what was written. Classifying whether a pixel is black or white is done on a similar way but a human eye can detect it automatically.

With fake reviews the techniques followed have been analysing all the data which surrounded each review, leaving the text apart as it was not possible of detecting if it was fake reading it in an isolated way. The need of analysing all the information which surrounded each review made the task of classification almost impossible for humans and so developing systems that process all this information is crucial. The best accuracy obtained in previous works was evaluated using cross validation, obtaining an accuracy of 86.1% [11]. In this work, the accuracy obtained with cross validation reached a 90% and with the test set the result was 86.13%, which means that the results were really successful. The idea of collecting new information from Yelp using the web crawler gave the possibility of extracting features that treated Yelp as a social web. The hypotheses that were proposed on Yelp chapter were verified, and this was clearly reflected on feature importances obtained with the final classifier. Behavioural features performed clearly better than review centric features, which means that analysing the activity of users at web pages can give valuable information.

In the case of recruitment the instances could be more easily classified by a human reader of the posts but it is important to highlight that, as it has been shown, a computer trained to understand natural language is capable of it. Transforming textual information to vectors make computers understand what is written almost as if it was a human. Due to the nature of the Corpus, the emphasis on this use case was made on post centric features and it could be seen that using techniques such as the TF-IDF, LDA models or using information theory techniques such as Mutual information work really well with text classification. As the Corpus was extracted from a Corpus which was created as the result of another classifier there were no references of previous work with it. However, the first experiments done in [15] with only 192 instances obtained a 89% of accuracy compared to classification made my judges, so there are possible lines of improvement in this area.

Regarding future work, for the case of Yelp, future work could be creating a Senpy plugin which, given a Yelp URL, crawls data from its reviews and users and returns the prediction made by the model created. It would also be interesting combining features from Yelp competitors such as TripAdvisor to see if users behave similarly or to get more possible features. Other similar problem to deal with could be trusting on second-hand trading pages such as Wallapop. This problem is related to deception but not exactly the same, but could also be treated with machine learning techniques. Another features to work with might be related to author profiling, seeking powerful features given by gender, age or personality characteristics. Additionally, there is also more work to be done with profile photos as sometimes people use photos obtained from the Internet. In some cases those photos are chosen to give more reliability to the author, for example selecting a familiar photo.

Regarding recruitment, an encouraging challenge is gathering data from the most important social networks at the moment (Twitter, Facebook) to create a similar model but including specific features from each social network. Thanks to this, it could be possible to detect online recruiters crawling those networks and learning the relationships between recruiters or additional information which could help finding them in real life. More improvements in accuracy could be done using Deep Learning with models such as Word2Vec.

Bibliography

[1] Yelp academic dataset. `http://yelp.com/academic_dataset`.

[2] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Exploiting burstiness in reviews for review spammer detection. *ICWSM*, 13:175–184, 2013.

[3] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[4] Tianjun Fu, Ahmed Abbasi, and Hsinchun Chen. A focused crawler for dark web forums. *Journal of the American Society for Information Science and Technology*, 61(6):1213–1231, 2010.

[5] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[6] William L. Hosch. Artificial intelligence. `http://global.britannica.com/technology/machine-learning`.

[7] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM, 2007.

[8] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 2488, 2011.

[9] Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[10] Ji-Rui Li, Yan-Fang Mao, and Kai Yang. Improvement and application of tf* idf algorithm. In *Information Computing and Applications*, pages 121–127. Springer, 2011.

[11] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S Glance. What yelp fake review filter might be doing? In *ICWSM*, 2013.

[12] Myle Ott, Claire Cardie, and Jeff Hancock. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st international conference on World Wide Web*, pages 201–210. ACM, 2012.

[13] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Jacob R Scanlon and Matthew S Gerber. Automatic detection of cyber-recruitment by violent extremists. *Security Informatics*, 3(1):1–10, 2014.

[16] Robyn Torok. Make a bomb in your mum's kitchen': Cyber recruiting and socialisation of 'white moors' and home grown jihadists. available from: http://ro. ecu. edu. au/cgi/viewcontent. cgi? article= 1005&context= act [accessed 04.20. 11.

[17] WILLIAM H WEBSTER, J Davidson, A Bifulco, P Gottschalk, V Caretti, T Pham, et al. Final report of the. *European Online Grooming Project. Retrieved*, 12:12, 2012.

[18] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

APPENDIX A

Appendix

A.1 Feature selection algorithm

1. The first part of the algorithm consists of iteratively finding the feature with less importance on the previous iteration and removing it. So, if the initial feature set contains 10 features, the first iteration trains the classifier with all of them and removes the feature with less weight for the classifier from the 10 features that are used. Then, the accuracy of this iteration is evaluated by cross validation and added to a list of accuracies. Information about features used on each iteration are also stored to use them on following steps. The following figure shows the evolution of feature importance every time a feature is removed. This graph can be useful to detect if some features were related or if they performed great together.

2. Once all features have been extracted, it is possible to see the evolution of accuracy of each iteration that is translated into the evolution of accuracy obtained by removing the less significant feature on the previous iteration. There will be a maximum value within those accuracies, and this will be the starting point for the following step of the algorithm.

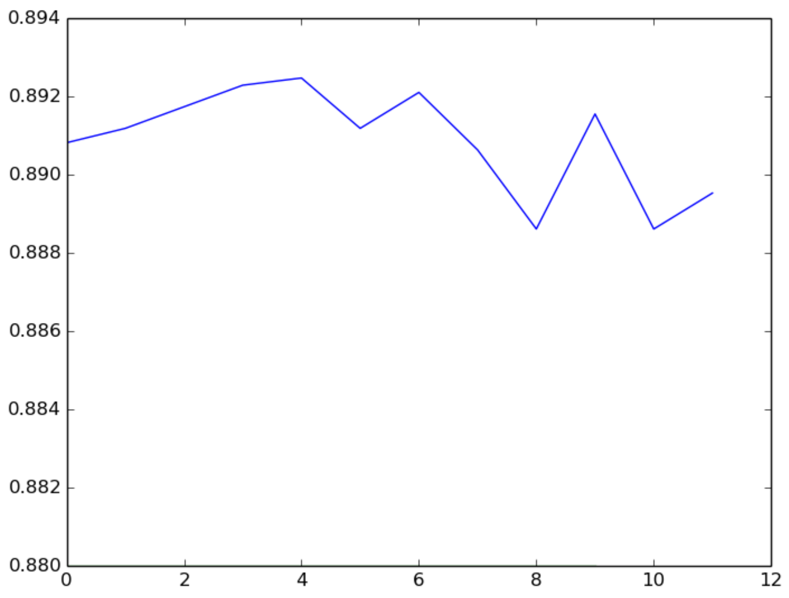

Figure A.1: Evolution of accuracy over the first iterations

3. Every time the previous graph drops, it means that there has been a loss of accuracy when removing a feature. The next thing is getting the features that were removed before obtaining the highest accuracy and trying to add them one by one in order to see if even than they were removed, probably in another situation they perform best than they did with more features.

4. After training the classifier with the features that performed best on the first part of the algorithm plus each of the features that were previously removed, if it is obtained a higher accuracy with one of those features, the selected feature is added to the set of best features and all of them are in this moment the ones that achieve the best results. As it is a new maximum, a new iteration starts adding the rest of removed features to the new set in order to obtain a better accuracy than the former best accuracy.

5. The algorithm stops looking for better features when there is no achievement of better results when adding iteratively each of the removed features.

A.2 Parameter classifier tuning on Yelp

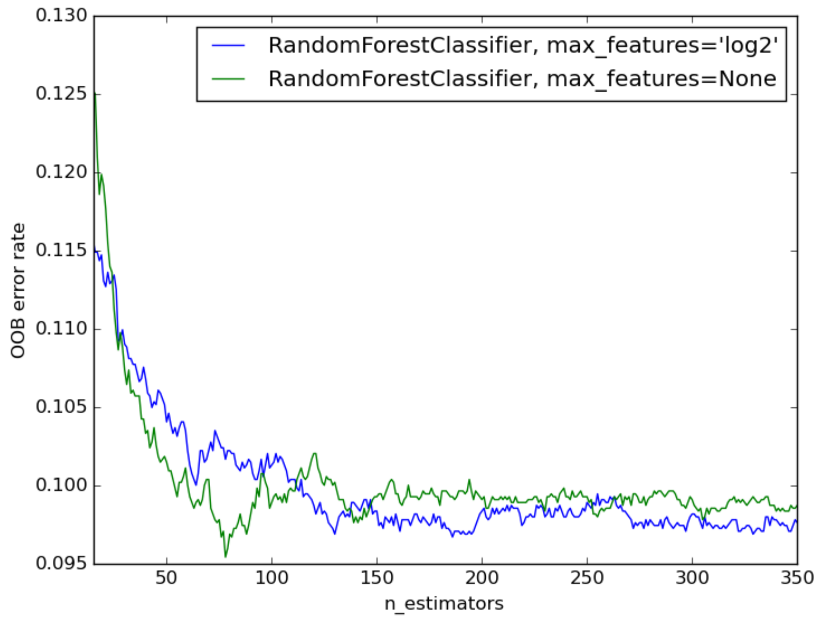

Figure A.2: Out-of-bag error Random Forest (Yelp).

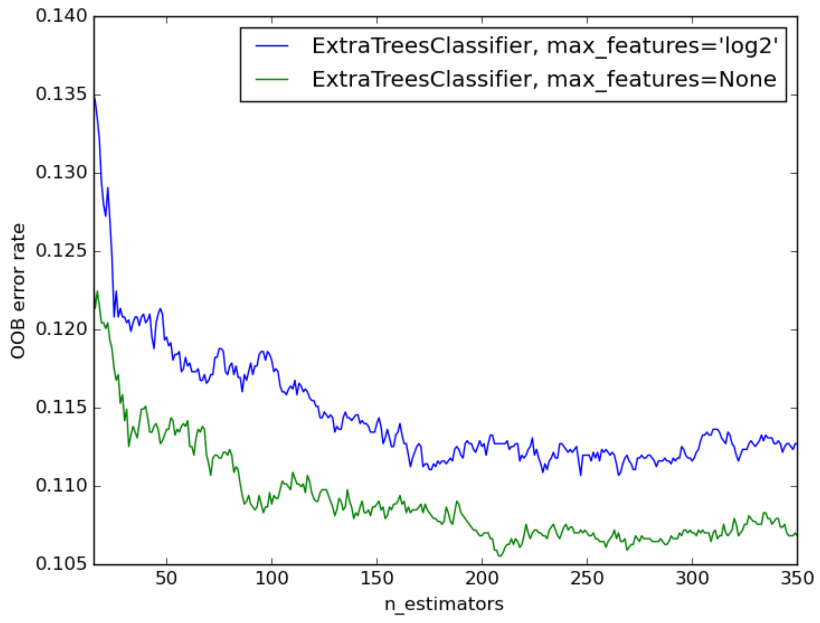

Figure A.3: Out-of-bag error Extra Trees (Yelp).

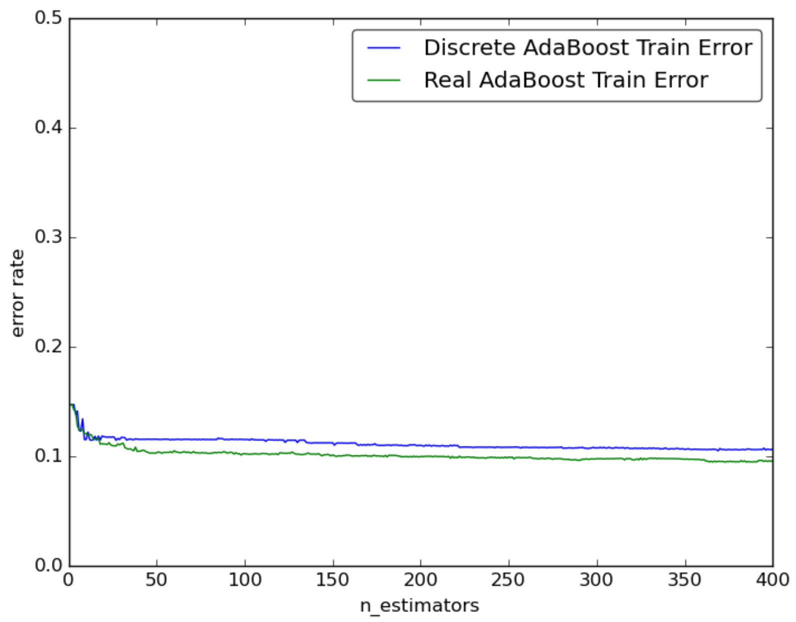

Figure A.4: Ada Boost tuning (Yelp).

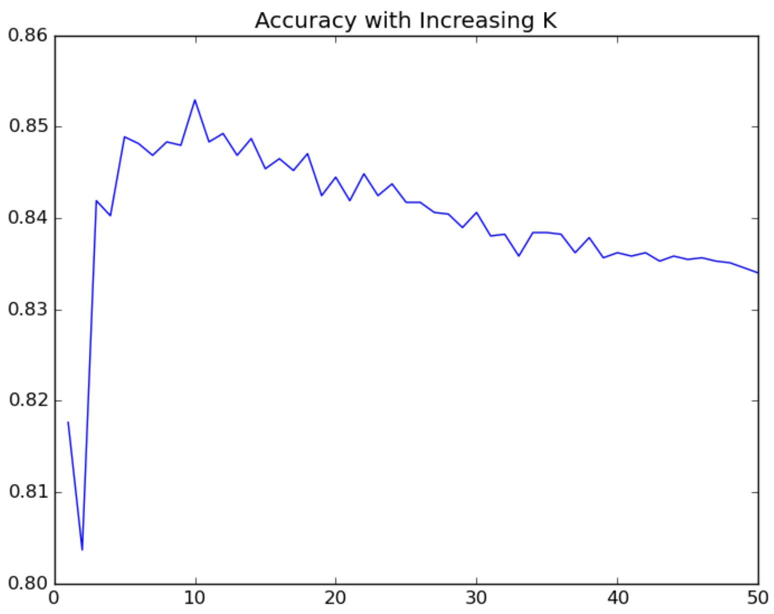

Figure A.5: Kneighbors tuning (Yelp).

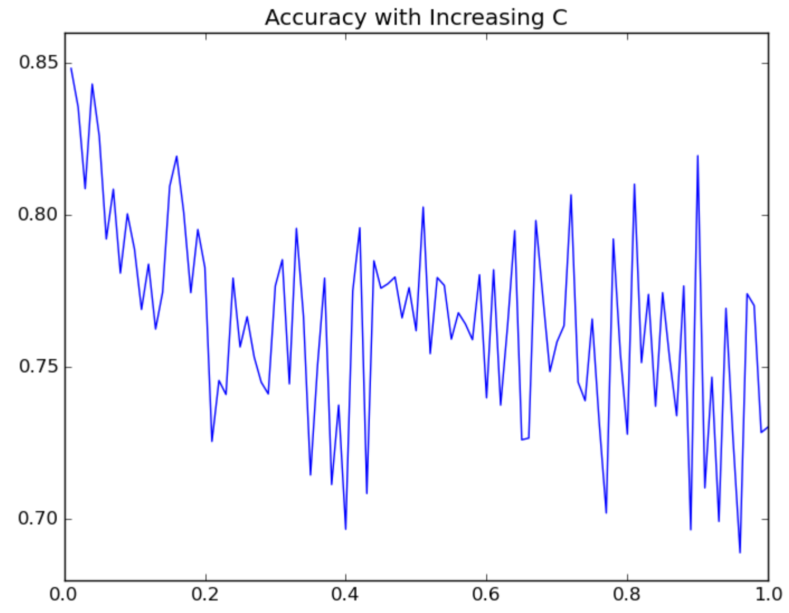

Figure A.6: Linear SVC tuning C parameter (Yelp).

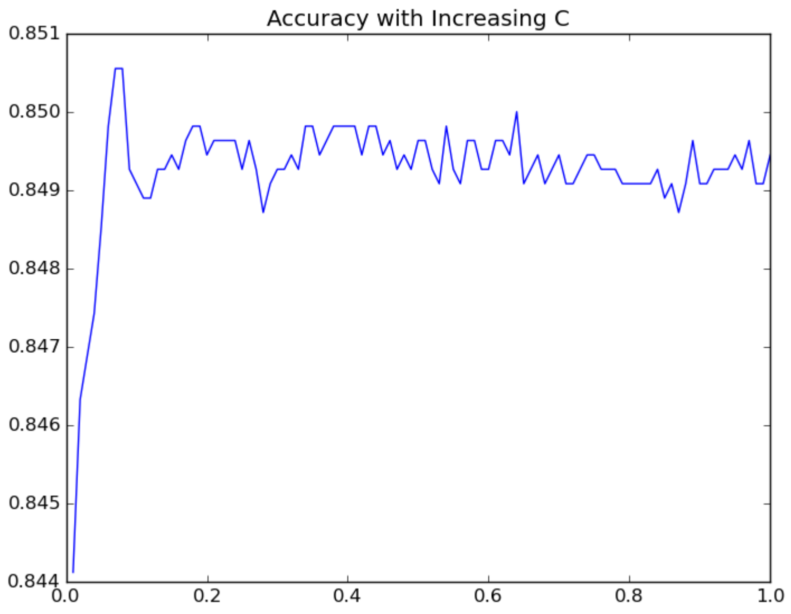

Figure A.7: Logistic regression tuning C parameter (Yelp).

A.3 Parameter classifier tuning on Recruitment

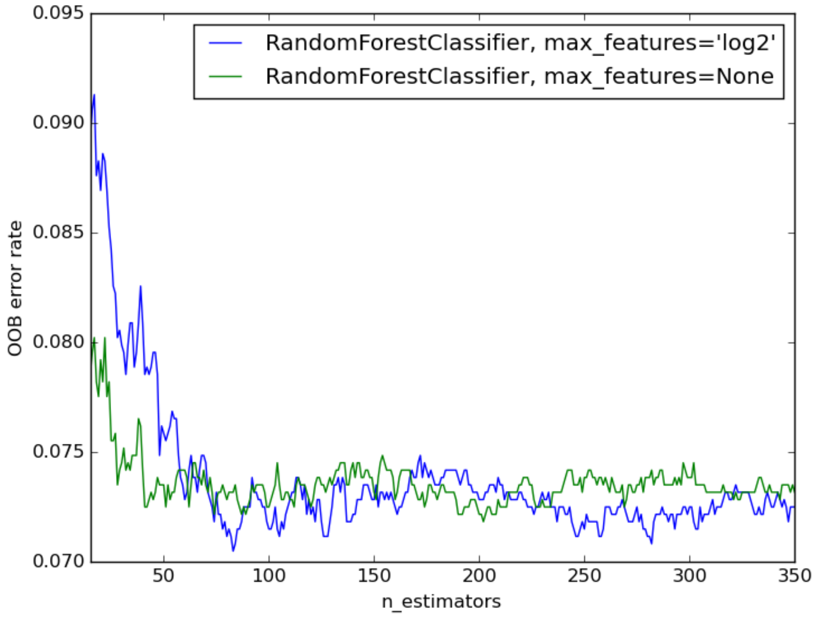

Figure A.8: Out-of-bag Random Forest (Recruitment).

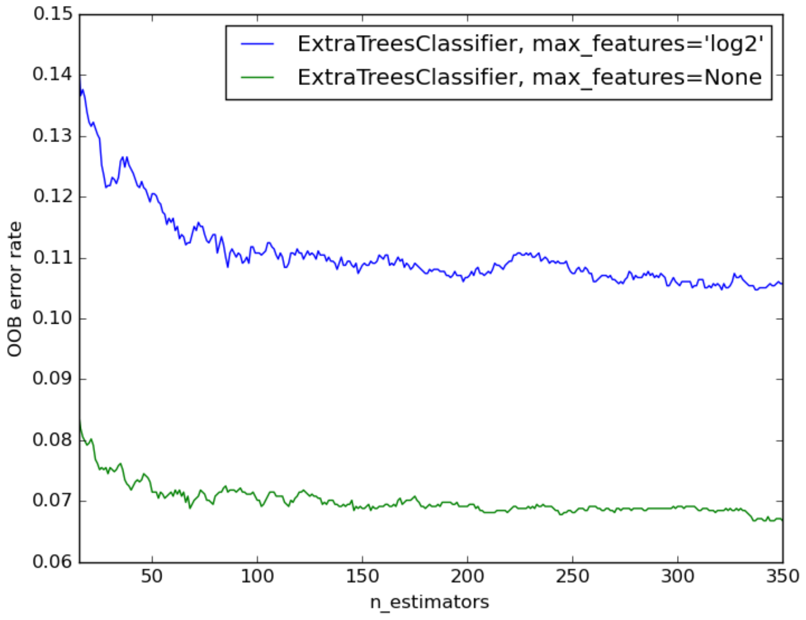

Figure A.9: Out-of-bag Extra Trees (Recruitment).

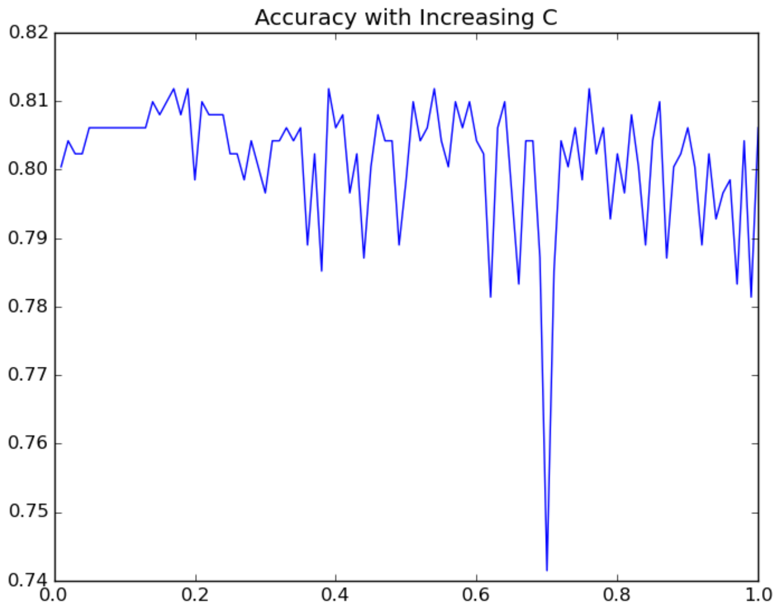

Figure A.10: Linear SVC tuning C parameter (Recruitment).

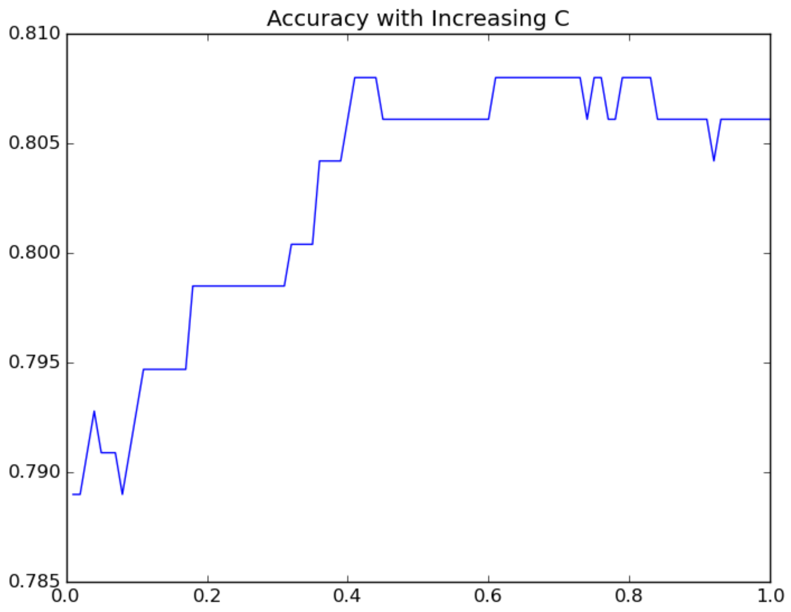

Figure A.11: Logistic regression tuning C parameter (Recruitment).

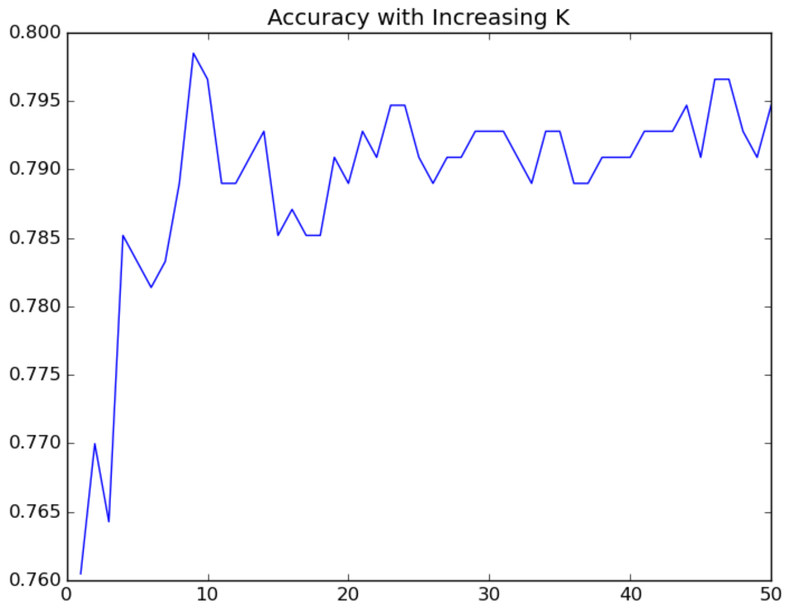

Figure A.12: Kneighbors tuning (Recruitment).

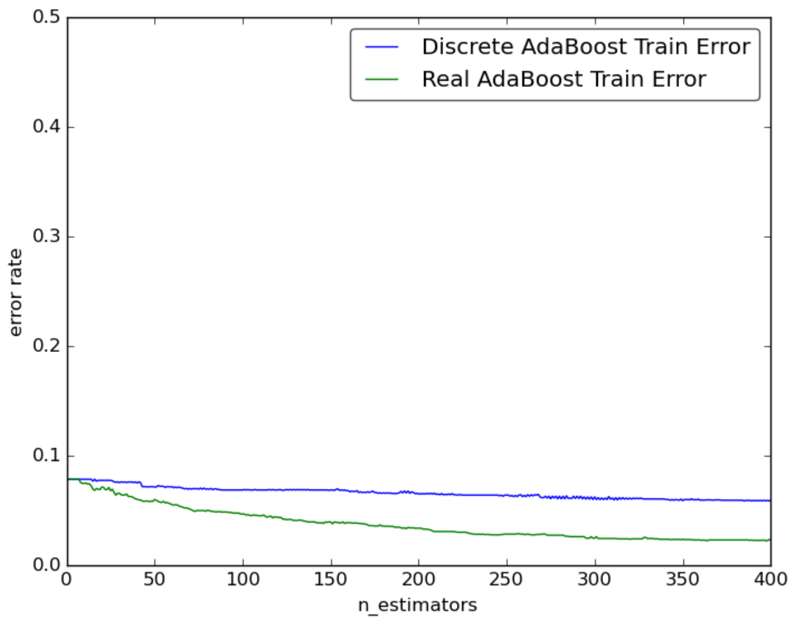

Figure A.13: Ada Boost tuning (Recruitment).