Dissertation Type: ML & NLP

DEPARTMENT OF COMPUTER SCIENCE

Detecting Contradictions in Online Dating Profiles

Josh Hamwee

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Thursday 28th May, 2020

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of BSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Josh Hamwee, Thursday 28th May, 2020

Contents

Abstract

With online dating becoming increasingly more popular, 'romance fraudsters' are more prevalent than ever, seeing a lack of security in the websites that are the basis of these operations. In May 2019 research was undertaken by Bristol University and others to detect these scammers, using machine learning and natural language processing. Owing to the success of this study I will try to expand on their work and will be focusing on the inconsistencies and contradictions within the profiles to detect potential scammers.

I have been granted access to a large dataset of roughly 60,000 profiles, of which a known portion are verified scammers, and will break up this data into a training and test set to evaluate the quality of my product. The data contains both images and text. The first challenge will be to extract as much information as possible from these sources using API's such as the Google Vision API and the COCO API. Once the dataset has been converted into a more useful form, I will use machine learning to classify each individual profile into either a scammer or not.

The overall goal of the system is to take in both images and descriptive text of an online dating profile. We will then return the probability of it being a scammers profile by combing multiple features of contradiction between the images and text.

The harder challenges involved will be to extract enough meaningful information from the images and text that will show discrepancies, and test whether a profile without images can hold enough information to classify it. The project falls into the categories of Machine Learning, Computer Vision and NLP.

The language I have chosen is Python for the majority of the programming due to its ease of use and compatibility with API's.

The project supervisor is Matthew Edwards, who worked on the original study, and hence has a strong understanding of what is required of me to complete this project on time and to a high standard.

Supporting Technologies

The following list provides the technologies used in the research at hand:

- Python 3 was used for all the programming in this project - Python 3
- A range of libraries to be used in conjunction with python, all install using pip3. Links to individual libraries can be found in the project execution section.
- Data harvesting methodology was implemented with inspiration from the previous work on the same topic. The links to this code can be found on https://github.com/gsuareztangil/automatic-romancescam-digger
- Both real and scam dating profiles were sourced from datingmore.com and scamdigger.com. Permission was granted by the owner of these sites.
- To be able to make use of the online dating profiles taken from the two previously mentioned websites, I had to be granted permission by the board of ethics. Conor Houghton signed off on the application with ID 95202 on the 25th October 2019.

Acknowledgements

A big thanks goes to my supervisor Matthew Edwards, who was there to help and guide me through the whole project. He always made himself available for my questions, no matter how elementary they were, and always provided the highest quality of feedback.

I would also like to thank the NHS staff and all essential workers who are putting their life on the line to ensure that this country is still operating to the highest standards through the struggles brought to light by Covid-19.

Chapter 1

Introduction

1.1 The Problem

For many years people have been using the internet to find their star-crossed lover using online dating sites. These sites have been growing at a significant rate with an estimated 240 million users and a growth of nearly 10% year on year [1]. With such a large user base, fraudsters have delved into this world of online dating to take advantage of vulnerable men and women for their financial gain. In 2018, roughly 1400 cases [2] of online dating fraud were reported in the UK alone, with a loss of £12.5 million, with less than a million returned to the victims. The size of this scam is even more prevalent in the US, with a estimated loss of $143 million in 2018 and a median of $2,600 per case, which is approximately seven times higher than any other online fraud [3].

Although large amounts of money are being lost to these scams, there is a lack of research into methods of automatically detecting these fraudsters. The reason why these dating sites and honest users need an automatic way of detecting these profiles is because humans are notoriously bad at recognising these profiles [4]. However, recently there has been new developments in the detection of these fraudsters using machine learning which proved very successful [5]. They used a plethora of features from a large dataset of dating profiles to detect fraudulent activity, and during their research they noticed a large amount of contradictions in the profiles of these fraudsters. This leads me onto the problem in which I will attempt to solve, detecting these fraudsters by using the contradictions in their profiles.

Although the solution will make use of machine learning to produce the classifier that will detect these profiles, one of the major hurdles in accomplishing this will be the measurements of inconsistencies and contradictions. There has been a small amount of research into how we can represent and quantify contradictions, and one of the major challenges for this project will be to synthesise a reliable and detailed method that can express the level of inconsistency within a profile.

1.2 Contributions

The high-level objective of this project is to implement a classifier that uses the contradictions within a dating profile to detect fraudsters. More specifically, the concrete aims of this project are as follows:

1. The first contribution will be the background research and survey of literature surrounding both measuring contradictions and the deception in online dating sites. This should give the reader a solid understanding of the background knowledge needed for the remainder of the paper.
2. Collect a large dataset of online dating profiles of both the fraudulent and genuine users.
3. Develop a framework to represent and quantify the inconsistency of a dating profile, with the potential for the framework to be expanded into other applications.
4. Implement a system to extract the necessary data of the contradictions in the profiles that will be required to evaluate the classifier.

5. Use the data gathered to train a classifier to accurately detect the fraudsters in online dating profiles using the identified inconsistencies.
6. Implement the a selection of contradiction features alongside the features used in Suarez's et al. [5].

1.3 Hypotheses

Introducing a hypothesis is important as it keeps the project on track in the right direction and can be used to evaluate the outcome of the project to what we believed it would be. The hypotheses for this project are as follows:

1. Online dating profiles can be used in conjunction with a range of APIs to extract data that shows the inconsistencies within the profiles. These inconsistencies can be used to train a classifier to detect fraudulent profiles. This can be tested with a range of accuracy measurements such as F1-score, precision and recall.
2. Online scammers tend to have more inconsistencies within their profiles. This can be tested in the Critical Evaluation section when looking at how well the classifiers have worked.
3. Analysis of the classifiers should provide some insight into whether using contradictions is a feasible method in scam detection.

1.4 Thesis Structure

The thesis is divided into multiple chapters and is detailed as follows:

- **Chapter 2 - Technical Background -** in which I will outline the relevant research required to carry out the rest of the project.
- **Chapter 3 - Project Execution -** in which I will describe the processes that I undertook to build the model that classifies the fraudulent profiles.
- **Chapter 4 - Critical Evaluation -** in which I will discuss the accuracy of the models that I have built and the possible future methods that may be undertaken to improve on my results.
- **Chapter 5 - Conclusion -** in which I will describe the achieved goals of the project, challenges that I faced throughout the project and an analysis of the original hypothesis to the end results.

Chapter 2

Technical Background

This chapter provides an overview of the background literature that will support the research throughout the thesis. It should provide the reader with the knowledge required to follow the Project Execution. More concretely, I will be going into some details of deception in online dating sites and related work that takes advantage of this deception. I will also discuss some basic mathematics of contradiction that may help in the analysis of our data set.

2.1 Deception in Online Dating

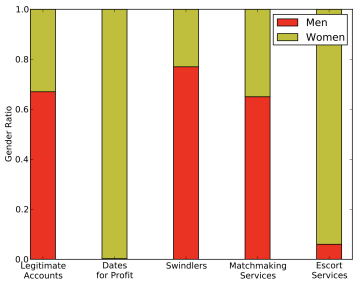
Deception in online dating profiles exists in two forms. A user may lie in their profile for a romantic advantage, or for a financial one. Both these profiles contain overlap as the imposter is creating a false profile. There has been a wider range of research into the user trying to gain a romantic advantage. Although the aim of this thesis is to try and identify the scammers, the underlying method of deceit from both categories is very similar, and hence research in this field has a slight grey area between the two.

2.1.1 Trust in Others

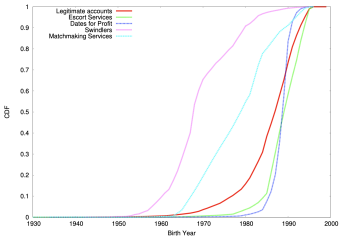
To gain an insight into the eyes of one of these fraudsters we have to understand how they gain the trust of the victim. A study by Hoffman in 2011, showed that if a user was trusting of others then they were 0.65 times less likely to use online dating [6]. Hoffman hypothesises that this "may be because individuals feel they can trust people they meet in person, but they feel different about trusting people they meet online". This is interesting as it adds an extra hurdle for the scammer; how do they overcome this lack of trust held by the victim?

The victim, upon a match on a dating site, will have an initial pre-conception with the trust in the scammer solely based off the profile. There are a multitude of factors that contribute to this trust such as the quality of the profile picture. Research by McGloin & Denes [7] provided me with some insights into how we perceive trust in online profiles and what factors contribute to this. Two of their positive hypotheses are relevant. Firstly, that the perceived attractiveness of the profile picture increased the trust the user had in that profile, and secondly, higher resolution and quality profile pictures provide a greater trust in the profile than ones that are not . This initial impression of the profile can have long-lasting implications for the victim because if their initial trust of the scammer is high, then they are more likely to trust them in the future. This can be described as the halo effect [8], which states that the initial impression or quality of a person will affect the way you perceive them in the future.

This initial pre-conception of trust is relevant to this study, as it is important to understand how these fraudsters are crafting their profiles to gain the trust of their victims. If a profile is ridden with contradictions, then how does this effect the initial trust the victim has on the profile? A fraudster would ideally try to reduce contradictions in their profile to increase their perceived trustworthiness, so these contradictions which we sought after may not be detectable to the human eye.



(a) Gender distribution of different scam accounts



(b) Age distribution of different scam accounts

Figure 2.1: Graphs taken from research by Huang et al. [9] on the Taxonomy of Dating Scammers

2.1.2 Characteristics of a Fake Profile

Understanding the different types of scammers present in these dating sites is a necessity, as it will help us to understand why these accounts have been created, and hence how they have been shifted to suit different scenarios. A recent study provided us with a detailed insight into the taxonomy of online dating scammers [9] in China. They identified four different categories in which these fake profiles fall under. Firstly, there are *Escort Service Advertisements* which contribute to the bulk of the spam accounts. These do not pose as much of a threat as they are only advertising legitimate escort services, and hence fall more under the category of spam. The second malicious category is *Dates for Profit*, which leverages the fact that the victims will want to meet in person. Owners of establishments would hire someone to create fake profiles and encourage the victim to go to said establishment to drive business up. This has a relatively high success rate as the scammer uses leverage on the victim to meet with an attractive woman. This also has a very low detection rate as very often the victim does not know they have been scammed as they did go on a date with an actual person. *Swindlers* usually take advantage of vulnerable people looking for romance, by building a strong relationship via online messaging until a time comes where they ask for money. This is the main way in which fraudsters operate for a financial gain and therefore the main target for our classifier. The final category is *Matchmaking Services* which are the least common. Typically they match with other users and use their personal information to try and persuade them to change to their matchmaking service. They do not operate for a financial gain, and this is more of an issue for the dating site itself.

Huang et al. [9] also provides some detailed analysis of the demographics of the different categories such as gender and age distribution which gives us an insight into how these scammers create their accounts. Figures 2.1 are taken from this paper and show the distributions of each demographic very well. Some interesting takeaway points from this paper:

- The gender distribution of the swindler category has a majority of being male. This means that the target for this type of fraud will mainly be heterosexual women, but potentially homosexual men. This does not mean that heterosexual men are not a target, however they tend to be approached less often.
- The age distribution of swindlers tends to be much higher than that of the legitimate accounts, hence the target for this scam will most likely be older than that of the average user of the site.
- Although not referenced above, their final analysis of these categories was on marital status. The results were that for most scams they chose their status to be single, however for the swindlers over 80% were widowed or divorced. We can deduce that this is most likely to try and gain sympathy from their victim.

2.1.3 Linguistic Analysis

One key factor in the analysis of the dating profiles is the description of the user. Each profile usually contains a short paragraph describing who they are and what they enjoy, and I believe that this description will be key to the classification of the profile. Research by Nagarajan and Hearst [10] gives a detailed analysis of the similarities and differences on language used between men and women on these sites. The takeaway points of this research are that there are some key differences in the language used, and hence

a possibility in reversing this evidence to predict the gender from the description. The relevance to this research is that if we were able to find a way of predicting the gender of the author of the description, then there is a possibility of a contradiction to the gender specified in the profile if a man created a fake profile of a woman or vice versa.

With the possibility of drawing this data from the description of the profile I researched into more detail about gender prediction using text. The field of Natural Language Processing is expanding more than ever, and hence plenty of research has been around this exact scenario. Loo et al. in 2016 [11] built a text-based classifier for age and gender. The results gave an accuracy of roughly 70% for the gender detection and the age detection accuracy is between 75% & 90% relative to the age in question. One of the issues with this method of age detection is that it requires a longer passage of text to work at a high accuracy, whereas the gender detection is still as accurate with a smaller sample.

I analysed more papers that gave some more concrete information in the methods used to detect the gender of the user. Traditional methods of NLP gender detection used bag of words and n-grams to classify the text, however a more interesting approach was to use deep learning [12]. They trained a Windowed Recurrent Convolution Neural Network which out performed all previous methods of gender detection.

2.2 Contradiction

The need for a formal technique to measure contradictions is becoming more of a necessity for machine learning and computer science in general [13]. One might think that these inconsistencies are not very desirable, however they hold a wide range of knowledge if measured correctly. Bertossi's research in 2005, cited previously, gives a very detailed review of why we need formal methods to measure inconsistency and a few methods to do this. In short, the need for a measure of inconsistency stems from the fact that when using formal logic you cannot have data that is both of the type p & $\neg p$. However in the real world data is often contradictory and still needs to be reasoned with. Throughout this section one should gain an understanding in the the definitions of contradictions and some methods to measure these.

2.2.1 Defining Inconsistencies

Research by Hunter et al. [14] gives a detailed description of different definitions of inconsistent information. They state that there are a plethora of ways to define an inconsistency, however the following two definitions relate more closely to contradiction rather than inconsistencies as a whole and hence more relevant for this research.

- **Inconsistency as conflicting inferences** "The knowledge-base $\triangle$ is inconsistent when there is the inference of both $\triangle \vdash \alpha$ and $\triangle \vdash \neg\alpha$ for some $\alpha \in L_i$."
- **Inconsistency as an inconsistent truth value** "Let β be an inconsistent truth value. Let $\alpha \in \triangle$ If for all models of $\triangle$, α is assigned β, then α is inconsistent, and hence $\triangle$ is inconsistent. "

The first definition of inconsistency states that if α is both seen as true and false, then there exists an inconsistency. This is similar to the second definition, which states that if α is assigned an inconsistent truth value, then for all models of $\triangle$ it is inconsistent. The first of these definitions is what I will use for the formal definition of contradiction. The research then follows on to define some dimensions that will be useful to in analysing the contradictions.

- **Atomic Inconsistency** An atomic inconsistency is the indivisible and discrete representation of contradictory information. There are two choices that we have when choosing this, firstly a minimal inconsistent subset of formulae, or secondly a propositional letter assignment to the inconsistent truth.
- **Number of Inconsistencies** Now that there is an atomic instantiation of a contradiction, we can now count them.
- **Size of the Inconsistency** With both the atomic definition of an inconsistency and the possibility to count them, we can deduce the size of an inconsistency. For example if we use a minimal inconsistent subset for the definition of an atomic inconsistency we can say that if $\mid \triangle_1 \mid > \mid \triangle_2 \mid$ then $\triangle_1$ is a larger inconsistency than $\triangle_2$. This will be a useful dimension in the future when we are comparing different contradictions and their relevance.

- **Ratio of Information to Noise** This is an important factor to take into consideration when measuring contradictions. If there is wide range of information that is consistent and a relatively small amount that is inconsistent, then that inconsistent information holds less of a weighting on whether the whole set of information is contradictory.

These dimensions and definitions declared above have laid out a firm grounding that will prove useful when evaluating different methods to measure contradiction. The research by Hunter et al. continues on to look at five different methods to do this, which will be looked into more detail in the following section. Even if the methods used to measure contradiction are not relevant to the research at hand, the definitions provided have formulated a good starting point in how to concretely define a contradiction, and how a collection of them can be collated together to provide us with more information.

2.2.2 Methods of Measuring Contradictions

With a firm understanding of what a contradiction is, we can look into different methods of measuring the inconsistency of information. Upon reading further into the research by Hunter et al. [14] a considerable issue presented itself in relation to the research at hand. Due to the way in which the data we will be using is presented to us, there is no concrete way of defining if a minimal inconsistent sub set is either true or false, as we will be measuring the contradiction with a probability measure. This therefore means that four out of the five methods presented to us in Hunter's research will not apply to this scenario. However, one of the methods described was relevant.

Probabilistic Measure of Consistency

There exists a framework first introduced by Knight in 2002 [15] that introduces a measure of consistency between [0,1] for each set of inconsistent formulae. If the set is contradictory, then the measure of consistency is 0 and for a set of formulae that is consistent, then the measure of consistency is 1. The size of this consistency value is directly proportional to size of the minimal inconsistent subsets. The formal definition is as follows.

Definition 2.1. A probability function on L is a function $P : L \to [0, 1]$ s.t.

$$\begin{cases} if \models \alpha & P(\alpha) = 1 \\ if \models \neg P(\alpha \wedge \beta) & P(\alpha \vee \beta) = P(\alpha) + P(\beta) \end{cases}$$

This first definition gives a probability distribution on interpretations and states that the probability of a formula is the sum of its models.

Definition 2.2. Let $\triangle$ be a knowledge base:

- $\triangle$ is η-consistent if there exists a probability function P such that $P(\alpha) \geq \eta$ for all $\alpha \in \triangle$
- $\triangle$ is maximally η-consistent if η is maximal (i.e. if $\gamma > \alpha$ then $\triangle$ is not γ-consistent)

This second definition now states that η-consistency can be used as measure of contradiction. What it means is that the more formulae that are needed to produce the inconsistency, the less the inconsistency is important. Hunter then goes on to describe two measure of information derived from these definitions, however the definitions of η-consistency is detailed enough for the research at hand and may be used as a measure for inconsistent subsets in the future.

The takeaway points from this section are that we now have a measure of inconsistency, however we have not developed our models to calculate this measure of inconsistency. Further discussion around this topic can be found in the Project Execution section where I will lay out the functions to calculate the contradiction. I will make use of the definitions provided above to lay out a solid set of rules that one can follow to extract inconsistencies from a range of different features.

2.3 Machine Learning and Fraud Detection

One of the the most fundamental areas that requires background research is that of previous methods of fraud detection. As this paper is in some ways an extension of previous work on scam detection using machine learning, a detailed review of the previous work will be very beneficial for this investigation. In this section I will discuss the methods used in past research to classify these scammers and more importantly the possible influence that these past research papers will have on my investigation.

Type	Real Profiles	Scam Profiles	All
Male	57.75%	63.76%	60.48%
Groups	0.50%	2.22%	1.28%
Children	5.21%	3.38%	4.38%
Food	1.86%	3.62%	2.66%
Animals	0.77%	1.08 %	0.91%
Discriminant	13.76%	17.77%	15.58%

(a) Topics outputted from the image description model

ID	Source	Name	Type	$\lvert\theta_i\rvert$
θ_M	Demographics	Age	NF	237
		Gender	CF	
		Latitude	NF	
		Longitude	NF	
		Country	CF	
		Ethnicity	CF	
		Occupation	CF	
		Marital Status	CF	
θ_C	Captions	set(*entities*)	CF	363
		set(*actions*)	CF	
		set(*modifiers*)	CF	
θ_S	Descriptions	set(*ngrams*)	SBF	105,893

(b) Proposed set of features to be used in classification

Figure 2.2: Tables taken from research by Suarez et al. [5]

2.3.1 Automatically Dismantling Online Dating Fraud

The roots of this project stem from the research by Suarez-Tangil et al. [5] that looks into the detection of scam dating profiles using machine learning. Their paper showed some noteworthy results that has allowed us to delve deeper into the detection of these fraudsters using the contradictions in their profiles. Firstly we will analyse how they characterised these profiles, and then how they used these characterisations to classify the users.

Characterising Dating Profiles

The data collected for this research was a collection of real user profiles from datingmore.com and a collection of scam users from scamdigger.com. Once a large enough dataset was collected, this information was used to to characterise the dating profiles, and extract valuable features that can be used in the classification model.

The research splits the data collected into three succinct categories which, if we use the same websites to source our data, will be beneficial to us to gain an insight into what we have to work with:

- **Demographic** Basic user information such as age, gender and ethnicity,
- **Images** Most profiles tend to have at least one image, with some users tending not to upload images of their face,
- **Description** A small paragraph of text written by the user describing themselves.

Demographics The next step in the research was to gather some meaningful data from the sources given to them. Due to not being constrained by the topic of contradiction they looked at a wide range of features within the profiles. Age, gender and ethnicity were referred to first, giving some key information about how the average age of female scam profiles is 30, and male scam profiles 50. Further data was extracted from the demographics such as location which used a basic method of contradiction by comparing the value in the demographics data to the geocoded latitude and longitude in the users image.

Images The image data was used in two separate methods. Firstly it was used to compare the geotagged location of the image and the users defined location, but they were also passed through a convolutional Neural Network (NN) combined with a language-generating recurrent NN. This model generates a detailed description of the image as seen in Figure 2.2, which can then be used in conjunction with most common topics occurring in both scam and real users to make a prediction. This method to process the images based on their descriptions seem to work effectively, however, to extend the scope of the output data to be measured based on contradiction, a comparison has to be made with the demographics data. Therefore we shall not require as much detail from the images, most likely just a prediction of age, gender and ethnicity.

Descriptions One of the more alluring sections within this characterisation operation was the analysis on the profile descriptions. As mentioned earlier in this research section, I looked at some potential methods to analyse the descriptions of the users using a Natural Language Processing (NLP) API. The relevant analysis from the Suarez et al. research informs us that a much higher percentage of scam users provide a description - 5,027 out of 5,402 - whereas the real users provided a description at a rate of 5,274

out of 14,720. The lack of real users who provide a description may cause some issues down the line as a large proportion of these profiles will not contain any contradictory data stemming from the description.

Although some of the finer details in this section of the research were interesting, the primary outcome from this analysis is the foundations of a well defined process of how to build a successful dataset to be used in the final model:

1. **Scraping** The initial task to perform is to scrape user profiles from the two websites referenced above. This can be performed with the Python library BeautifulSoup [16] which parses HTML data.
2. **Characterisation** Once the profiles have been collected some analysis is required on what information we have available to us, such as age, gender and ethnicity. With these values we can start to experiment in extracting contradictory values.
3. **Testing** Some testing may be required to evaluate the performance from a range of APIs.
4. **Feature Extraction** The final step in this data gathering process will be to make the API requests for each profile, and with the results process the value of inconsistencies for each feature that is required.

Classifying False Profiles

Now that we have gauged a firm understanding of how this models data is gathered and pre-processed, our next aim is to investigate the methods of classifying the profiles that this model used. The collection of features gathered from the previous step have been listed out in Figure 2.2, and as we can see they have been split into their relative sources and types. A general overview of the section is to gain an understanding of the following definitions, which have been taken directly from their research [5].

1. A set of prediction models $\mathcal{P} = \{P_1, \ldots, P_i\}$ that output the probability
$$\theta_i\left(\phi_1, \ldots, \phi_n\right) = P_i\left[X = \text{scam} \,|\, (\phi_1, \ldots, \phi_n)\right]$$
of each profile X being scam given a feature vector $(\phi_1, \ldots, \phi_n)$ obtained from different profile sections i.
2. A weighted model $f(\mathcal{P}) = \sum w_i \cdot P_i$ that combines all individual predictions in $\mathcal{P}$. Here, each individual classifier P_i is weighted by w_i according to the accuracy given on a validation set that is different from the training one. This will also serve as a way to calibrate individual probabilities. The final classifier will then output a decision based on a vote such that
$$f = \begin{cases} \text{scam} & \text{if } f(\mathcal{P}) < \tau \\ \text{real} & \text{otherwise} \end{cases}$$
where τ is a threshold typically set to $\left\lfloor \frac{\sum w_i}{2} \right\rfloor + 1$

Base Estimators The idea behind this model is that each category of feature is modelled with a suitable classifier, for example the demographics data used Naïve Bayes, as it worked effectively when there was data missing. However a large proportion of data was not missing and hence it wasn't the most suitable choice. For the data that did not have any holes in it, a Random Forest Model was selected, and the final outcome was a joint Random Forests and Naïve Bayes Model. With each category of feature type classified, satisfying part one of the system above, it leads us onto the Ensemble Classifier.

Ensemble Classifier An ensemble classifier is one that combines the predictions of a previous set of classifiers to provide a more accurate classification. In the evaluation of this classifier, it out performed the use of a single SVM classifier to combine the set of classifications discussed earlier. Going forward, I believe that this method may suit the classification problem at hand because the accuracy of our classifiers may not be sufficient.
To conclude the research into this paper, I feel that it has provided us with some valuable insight into the direction I will pursue with this project. We have gathered a strong method into how we are going to gather our data, whilst also gaining an understanding of how we can approach the classification step of our model.

2.3.2 Machine Learning Classifiers

Following on from the evaluation of the research by Suarez-Tangil et al. [5], the review did not provide enough detail into the choice of machine learning classifiers used. In this section I will give a description on the potential classifiers that could be used on our dataset, and why they are suitable for the task at hand.

Naïve Bayes

The first classifier to be considered to implement is the Naïve Bayes Classifier. It is one of the simplest supervised learning algorithms that operates at high speeds and accuracy. It assumes that each feature is independent of each other, even though they potentially may not be. This leads to a disadvantage, as in practice it is very unlikely to receive a set of features that are completely independent of each other. Research by Arar et al [17] looks into the effect that dependant features have on the model, and it does significantly reduce the accuracy of the classification. It then proposes a new method of classification called Feature Dependant Naïve Bayes that is more effective at classification with dependant features. The inconsistency features that will be extracted will most likely have some dependency on each other, hence it may be beneficial to look at this new proposed method.

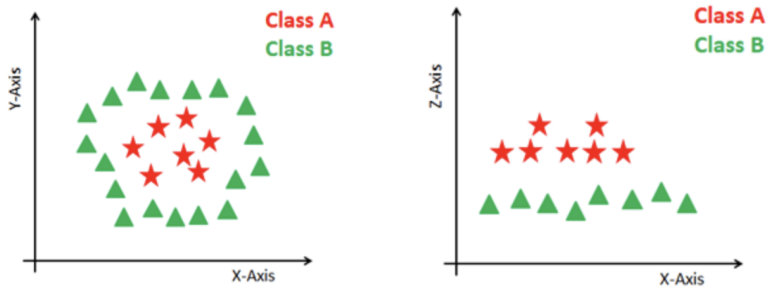
One of the more prominent reasons in which this classifier may suit the data is because it will most likely have imbalanced classes. As previously mentioned, the user profile data which was harvested in Suarez-Tangil et al [5] was not a 50/50 split between scammers and real users. It was more along the lines of 25/75, and hence we have unbalanced classes. Naïve Bayes classifier is not adversely affected by this, and consequently should be suitable for the requirements of this project.

Logistic Regression

One of the most frequently used classifiers is that of logistic regression. It is used to predict binary classes and hence suits the requirements for this classification problem, as a profile is dichotomous, scammer or not scammer. The logistic regression tutorial provided by DataCamp [18] shows some insight into the advantages and disadvantages of this classifier. Similar to the Naïve Bayes classifier, it handles unbalanced data sets very well, whilst also not having to assume any independence between features.

However, this model is also has some disadvantages that hinder its performance, such as the **assumption of linearity** between the dependant variable and independent variables. This therefore means that if the data does not follow a linear relationship, which it most likely won't, then the classifier will not work without feature transformations. If that is the case then it will not be possible to use this classifier for our required needs. A more suitable choice of classifier would be a Support Vector Machine with a radial kernel, described in the following section.

Support Vector Machine



(a) Non-linear inseparable plane

(b) Same data after the SVM kernel has been applied

Figure 2.3: Graphs taken from DataCamp [19]

Support Vector Machines can be used to classify multiple continuous and categorical variables. The method behind its implementation is to construct a hyper-plane in multidimensional space. This plane can then be used to categorise the data into its respective classes. Mentioned in the previous section, the

data that is to be classified may be non-linear. This is where the SVM kernels come into play, as they can be used to transform the input space into a higher dimensional space [19]. Figure 2.3 shows how one can use these kernels to solve a classification scenario that is non-linear.

However, as with all classifiers, it has its disadvantages. SVM's are not suited to big data-sets due to the large amount of time required for training. Secondly the classifiers do not work well when there is an overlap between classes, which may prove itself a problem if the APIs used extract similar information. One method to overcome this may be to calculate the inconsistency between predicted values from the API, which could be used as an extra feature in the training process.

Chapter 3

Project Execution

In this chapter I will be explaining the methods used to execute this project, and the challenges faced. Before getting started on the main body of the project, it is necessary for us to discuss a few planning prerequisites such as ethics and the choice of programming language. The following list outlines the main contents of the chapter, and more significantly the process used to build our model:

- **Stage 1:** Data Collection
- **Stage 2:** Methods to Measure Inconsistency
- **Stage 3:** Feature Extraction
- **Stage 4:** Classifier Training

3.1 Planning

Ethical Approval The first step, in order to be granted permission to handle the sensitive data provided by the two websites, was to apply for approval from the University of Bristol Research Ethics Committee. The conditions that I have to abide by are as follows. Firstly I may not redistribute any of the data, and secondly, it must be stored on an encrypted server in the cloud such as the One Drive account provided by the university. There were further conditions which were met, and so I was granted approval by the Board of Ethics on the 25th October, so that I could move on to the next stage, planning the experimental design process.

Programming Language The choice of programming language did not require too much thought. I had had a reasonable amount of experience using Python in the past, and due to its ease of use and its wide range of tools for data analysis and machine learning, such as Matplotlib and Sklearn, it was the most suitable choice. One of the more prominent reasons for choosing Python is that it has a library called 'requests' that makes HTTP requests simple and stable [20]. This proved very useful when calling the APIs.

3.2 Experimental Design

3.2.1 Data Collection

Dating Profile Harvesting

To start experimenting with different methods to measure inconsistency on dating profiles, the first step to carry out was that of downloading those profiles. With access to the code from the Suarez et al. paper, I was able to look at the method in which they downloaded all of the profiles. The Python library, BeautifulSoup, referred to in the Technical Background, was used to parse the HTML data. This was then converted into JSON, using the `json.dump()` command, and then saved to an individual JSON file. The images are all saved under a different directory, and each profile contains the file name for their respective images.

Due to the data being extracted from two different sites, two separate programs were required, both with a similar process, however the methods of parsing the data with BeautifulSoup are slightly different, and the two websites require different methods to iterate through the page and harvest the URL for each individual user. Below are two separate JSON files, one returned from a scam profile, and the other an authentic one. The data has been modified to show all potential options that could be received, separated with slashes, and with user inputted data or data which contains a non concrete value an underscore is used.

```
{
    "gender": "male/female",
    "age": "_ y.o.",
    "location": "_",
    "status": "married/divorced/single/widowed",
    "username": "_",
    "ethnicity": "white/black/mixed/hispanic/asian/native american/
   middle eastern",
    "occupation": "-",
    "description": "-",
    "match_age": "from _ to _",
    "children": "no children/children",
    "orientation": "Straight/Gay",
    "religion": "_",
    "smoking": "light smoker/heavy smoker/non-smoker",
    "drinking": "social drinker/occasional drinker/never",
    "intent": "Serious Relationship/Fun/Friendship/Romance",
    "images": ["imgreal/_.jpg"]
}
```

Code Listing 3.1: JSON data of a real profile taken from datingmore.com

```
{
    "year_reported": "_",
    "month_reported": "_",
    "images": ["imgscam/_.jpg", "imgscam/_.jpg"],
    "username": "_",
    "name": "_",
    "age": "_",
    "location": "_, _",
    "ethnicity": "white/black/middle-eastern/mixed/asian/native american
   /pacific islander/other",
    "occupation": "_",
    "status": "single/married/divorced/widowed",
    "inet": "IP ADDRESS",
    "email": "_@_",
    "description": "_",
    "messages": ["_","_"],
    "justifications": ["_", "_"],
    "gender": "male/female"
}
```

Code Listing 3.2: JSON data of a scam profile taken from scamdigger.com

Once a single profile was extracted from both the scam and real websites, I left both programs to run to download as many profiles as possible. A total of 3,070 scam profiles were collected, and roughly 15,000 real user profiles. However, due to the lack of descriptions and images in some of the real profiles, the final tally was reduced down to 10,910.

Comparison With the full set of user data now in my possession, it is important that we compare what values are common in both the sets of profiles, as seen in listings 3.1 and 3.2 so that we can extract

the same features for both sets. The following list shows the features that appear in both profiles, with a short description of the type of data provided.

- **Gender** This is a binary value in both profiles of either male or female.
- **Age** This feature contains a continuous integer value, however, within the real profile it is given with the text 'y.o.' at the end. One would have to remove the UTF-8 formatted text at the end and use a type conversion function from string to int.
- **Ethnicity** Both profiles contain a set of possible answers as follows:

$$E_{\text{scam}} = \{white, black, middle\text{-}eastern, asian, native\text{-}american, pacific\text{-}islander, mixed, other\}$$
$$E_{\text{real}} = \{white, black, mixed, hispanic, asian, nativeamerican, middleeastern\}$$

- **Occupation** This is provided as a user defined text value which will makes it more complicated to use as a comparison feature.
- **Images** This is provided as an array of text values that represent the local position from the working directory of the specified images.
- **Description** This value is provided as a string in UTF-8 format.

3.2.2 Methods to Measure Inconsistency

The next stage for this project is to develop a set of rules to be used when we need to extract the values of inconsistency from the different features within the profiles. The primary requirement for this step is to decide on which features I will use to measure the contradictions, and as seen from the previous stage, we are limited to the common features. Furthermore, we are also limited to what data we can extract from the images and description.

Initial Ideas My initial thoughts on how I was going to measure the contradictions stem from the ideas previously referenced in the Technical Background, the probabilistic measure of consistency. However, I believe it is more important to represent the value as a probabilistic measure of inconsistency. To gain a probabilistic value I will require a probability that describes how confident one is in specific feature's predictions accuracy. To be able to extract this information a brief overview on the three feature types is needed.

The Three Feature Types For simplicity I have split the users data into three separate types, and for each of these types certain values can be extracted for potential inconsistency measurements:

- **User Defined Values** The values defined by the users are set in stone. It is not possible to extract any probabilistic measure of confidence from them and hence cannot be used by themselves to measure contradiction. However, they will be able to provide a value that can be used in conjunction with another probabilistic value to calculate a probabilistic value of inconsistency defined earlier.
- **Images** The images attached to the profile do not contain any defined values, but it will be possible to extract information regarding Age, Gender and Ethnicity using a set of APIs. The data extracted from the images can be used to compare values with the User Defined Data.
- **Description** Similar to the images, one can use an API to predict Gender and Age of the author of the text. Using this in conjunction with the User Defined Data a range of values of inconsistencies can be calculated to add to our feature set.

Although I have not discussed the specifics of the APIs that I will use on the images and descriptions, it is necessary for one to know what values will be returned before I define our inconsistency measure. At this point in the project I was researching and testing a range of APIs, however it is more relevant in the feature extraction and hence can be found in the next stage. For now, we will assume that we have the following values that can be used. The image APIs have returned an **age, gender and ethnicity** value all with a corresponding confidence values. The description API has returned **age and gender** values both with a corresponding confidence value.

Defining Inconsistency Now that I have assumed what data I have extracted from the APIs I started to look at some formal definitions of inconsistency. The first being that of defining what n-inconsistency is. It is not necessary to over complicate this definition, as I referenced the value of n-consistency in the Technical Background. The only difference being that instead of measuring how consistent the feature is, one measures how inconsistent the feature is.

Definition 3.1. Let f be a function which returns the value of n-inconsistency for the relevant feature set

$$f(a,b,c) = \begin{cases} n = 1 - c & \text{if } a = b \\ n = c & \text{if } a \neq b \end{cases}$$

Where I have defined

$$a \leftarrow \text{user defined value}$$
$$b \leftarrow \text{predicted value}$$
$$c \leftarrow \text{confidence of predicted value}$$

Definition 3.1 above was the preliminary model to measure the inconsistency of a pair of features. The logic behind it is not too complex, but more importantly it will not work for all features. For example, it would work perfectly well when measuring the inconsistency value of gender, as there is a discrete set of values that can be true. However, it would not work if we were trying to measure the inconsistency value when comparing age, due to there being another dimension to the comparison that I had to take into account, the distance of the predicted age from the user defined value. I felt that the optimal method to measure the inconsistency for the age feature set is by slightly appending the function from the previous definition. With a few adjustments I was able to come up with the following:

Definition 3.2. Let f be a function which returns the value of n-inconsistency for the age feature set

$$f(a,b,c) = \begin{cases} n = 1 - c & \text{if } |a - b| < 5 \\ n = c/2 & \text{if } |a - b| < 10 \\ n = c & \text{otherwise} \end{cases}$$

Where we have defined

$$a \leftarrow \text{user defined age}$$
$$b \leftarrow \text{predicted age}$$
$$c \leftarrow \text{confidence of predicted age}$$

As you can see I have slightly changed the definition from the previous one by checking the distance of the predicted age from the user defined value. This function will only be applicable to the age comparisons, which leaves us with one comparison to consider, the demographics. My first thought was that I would have to create a new function that considers all demographics categorising similar ones. However, it is actually possible to use the function from Definition 3.1 where the value of inconsistency is $1 - c$ if the demographics are the same, and c if the demographics are different, where c is the confidence in the predicted demographic.

3.2.3 Feature Extraction

Now that the functions to calculate the value of inconsistency between features have been defined I could move onto the next stage in the project execution, feature extraction. The overarching goal of this section is to end up with a collection of features that measure the inconsistency of the profiles, which can be used to train and test our classifier.

API Selection and Evaluation

As mentioned in the previous section, before we collate all of our data, we have to choose the correct APIs for our model. We are looking for two APIs that meet the following criteria. Firstly, an Image API that predicts gender, age and ethnicity with a confidence value for each prediction. Secondly, a Text API that can predict age and gender both with a confidence value for its predictions.

Image to Gender, Age & Ethnicity When first researching APIs that meet the requirements I concluded that the best place to start was the big tech companies Image Recognition APIs. For example both Amazon and Google provide access to an API that takes in images and returns a set of features about said image. However, upon using them I recognised that firstly the cost of use would be too high when considering 20,000 users, and secondly neither API returns the full requirements.

```
"Emotions": [
                {
                    "Type": "DISGUSTED",
                    "Confidence": 0.018225016072392464
                },
                {
                    "Type": "ANGRY",
                    "Confidence": 0.03304917365312576
                },
                {
                    "Type": "CONFUSED",
                    "Confidence": 0.11641046404838562
                },
                {
                    "Type": "CALM",
                    "Confidence": 0.1430344581604004
                }
            ]
```

Code Listing 3.3: Section of the returned API from Google Vision

Listing 3.3 was a snippet from the Google Vision API that made me consider changing the methodology of predicting contradiction, or at least considering a new feature. With a set of emotions provided from the image, I thought about extracting emotions from the description provided by the user, and comparing them. However, due to limited time and resources I felt that this feature set could be implemented at a later date.

In the end, two Image APIs were chosen. Firstly I used the FaceX API that predicts an age range, gender and gender confidence. The reason for choosing FaceX is that it is considerably cheaper than the majority of Image APIs, however the information received was limited, and meant that I required a second API, ClarifAI. This choice provided me with more detailed information such as a specific age, gender and ethnicity. All of these predictions were returned with confidence value, which meant that I could use each of the features in the contradiction model.

Description to Gender & Age The final API that I chose to use was called AI-Applied. The request took in a paragraph of text, and uses NLP to predict the users Age and Gender. The API was compatible with multiple languages, which proved to be necessary, as a large proportion of the users were Hispanic, and hence their descriptions were in Spanish. A simple `if` statement was used to check the users' ethnicity, and if the user was Hispanic I changed the `language_iso` variable from `"eng"` to `"spa"` in the http request. A more automated method of doing this is by using a simple language identification classifier such as the one incorporated into the langdetect Python library.

API Used	Features Returned
FaceX	- Age Range - Gender w/ Confidence
ClarifAI	- Age w/ Confidence - Gender w/ Confidence - Ethnicity w/ Confidence
AI-Applied	- Age w/ Confidence - Gender w/ Confidence

Table 3.1: Features returned from the APIs that will be used

Table 3.2 shows the list of features from all of the APIs that are used. This selection of features seems to meet all the desired requirements to be able to the inconsistency functions from the previous

section. One point that I would like to consider is that all the FaceX features returned are also part of the features from ClarifAI. This can only be advantageous as it minimises the error in the predictions as we can calculate multiple inconsistency values for gender in images.

With the APIs tested and evaluated, the next stage was running them on each individual profile. I did come across an issue when running the Image APIs on the real users. For some reason, roughly a third of the profiles collected did not contain an image or a description. This immediately reduced the dataset of real users, as without that data there was no potential avenue to measure contradiction, as there was no way of comparing non-existent values. Aside from that slight hindrance, there seemed to be no other issues, and I had collated roughly 15,000 user profiles with their API responses. All the users were stored as an anonymous JSON file, and then cleaned up to extricate any irrelevant data.

Extracting Inconsistency Values

The final stage in the feature extraction process was to apply the inconsistency functions created previously. The data to be processed is all stored in individual JSON files. Using the Python JSON module I iterated through each individual profile applying our set of rules, and then exported it to a new JSON file containing only the features. During the evaluation of the output of a single user, it came to my attention that I had not utilised the Age Range feature provided by the FaceX API. Although it does not contain a confidence value, I was able to utilise it by comparing the range to the user defined age value, and assigning an ordinal value dependant on the distance from the age range. The definition below shows the initial implementation.

Definition 3.3. Let f be a function which returns the value of n-inconsistency for the feature set containing an age range and no confidence value

$$f(a, b_{min}, b_{max},) = \begin{cases} n = 0 & \text{if } (a > b_{min}) \wedge (a < b_{max}) \\ n = 0.5 & \text{if } (0 < b_{min} - a < 5) \text{ or } (0 < (a - b_{max}) < 5) \\ n = 1 & \text{otherwise} \end{cases}$$

Where we have defined

$$\begin{aligned} a &\leftarrow \text{user defined age} \\ b_{min} &\leftarrow \text{predicted age lower boundary} \\ b_{max} &\leftarrow \text{predicted age upper boundary} \end{aligned}$$

Upon re-evaluation of this measure of inconsistency using data containing an age range, I felt that it was not as strong as the other features, as it is the only one without a continuous spread of data. When I first implemented the model I noticed that it was underperforming and thus I decided to change it to incorporate a more valuable measure of inconsistency. The definition below is a revised function that makes use of the distance of the predicted age. It would then be possible to use pre-processing tools to scale it between 0 & 1 at a later date.

Definition 3.4. Let f be a function which returns the value of n-inconsistency for the feature set containing an age range and no confidence value

$$f(a, b_{min}, b_{max},) = \begin{cases} n = |a - b_{max}| & \text{if } (|a - b_{max}| > |a - b_{min}| \\ n = |a - b_{min}| & \text{if } (|a - b_{max}| < |a - b_{min}| \\ n = 0 & \text{otherwise} \end{cases}$$

Where we have defined

$$\begin{aligned} a &\leftarrow \text{user defined age} \\ b_{min} &\leftarrow \text{predicted age lower boundary} \\ b_{max} &\leftarrow \text{predicted age upper boundary} \end{aligned}$$

Upon implementation of the function given in Definition 3.4, I now implemented a function for calculating each of the features I required. The next step in the extraction process was to apply the calculations to each profile, which was a simple process to implement due to pythons easy iteration of a full directory.

With all the profiles imported, passed through the functions, and exported as individual JSON files, it was now time to collate all the data into one file. This step is not completely necessary, as the individual profiles could be imported as JSON files during the implementation of the classifier, however it would most likely save time in the future if I combined all the real and scam user profiles into a single CSV file, with there respective class labels.

Table 3.2: A table containing each feature used to classify the profiles, with a short description of the source and data type.

Features	Data Type	Description
text_age	Continuous, numerical data between 0 & 1	The value of inconsistency evaluated with the predicted age range from the description, and the use defined age value.
text_gender	Continuous, numerical data between 0 & 1	Similar to *text_age*, this feature is the measure of inconsistency between the predicted gender from the description, and the user defined value for gender.
image1_age	Continuous numerical data where $x = \{x \in \mathbb{R} \| x > 0\}$	The inconsistency measure evaluated using the age range provided by the FaceX API.
image1_gender	Continuous, numerical data between 0 & 1	This is the second feature extracted from the same API as above. It is the measure of inconsistency between the user provided gender and the predicted genders from the images.
image2_age	Continuous, numerical data between 0 & 1	The ClarifAI API is the only one used that returns a specific age value with a confidence measure. Using these extracted values, the third and final feature for age inconsistency was calculated.
image2_gender	Continuous, numerical data between 0 & 1	Using the same API referenced in the previous feature, the inconsistency measure for gender is calculated with the confidence measure provided by the API.
image2_ethnicity	Continuous, numerical data between 0 & 1	Using the predicted demographic of the user from the images and the user defined ethnicity, the feature for ethnicity inconsistency was calculated.

3.2.4 Classifier Training

The final stage in the project execution was to train and test a range of classifiers using the features prepared earlier. To carry out this step I broke it down into three sections. The first being the pre-processing of the data.

Pre-Processing

The pre-processing of the data is one of the most important stages in this whole project. If the data becomes corrupt during this stage, then it may cause drastic changes to the result of the classifiers. The initial step to carry out is importing the data from a CSV file and splitting it up into an array of features and labels. I used the pandas Data-Frame to store these arrays, as it works seamlessly with ScikitLearn's pre-processing and training tools. Due to some of the data not being between the range of 0 and 1, I made use of the MinMaxScaling tool to scale the features that were not consistent with the rest.

A small requirement for using the classifiers with ScikitLearn is to make sure that the data does not contain any `Null` values. To accommodate this I used a function built into Pandas called `data.dropna()` which removes a whole user if one of their values is missing or `Null`. The final step required to remove the

unnecessary data is to re index the individual users. Using another function provided by Pandas called `data.sample(frac=1).reset_index(drop=True)` will carry out this process.

With the data now in its desired variables, it was time to split the data into training and test sets. Initially I used the TrainTestSplit function provided by ScikitLearn, which splits the data into an array of training and test data in a ratio that one specifies. I ensured that the shuffle features was set to `True` because the input data was in the format of real users first, then scam users.

Initial Training & Testing

With the data now imported and all required pre-processing completed, it was time to train a range of classifiers. Initially each classifier was trained on a 75%/25% train test data split. I used of a range of functions built into the ScikitLearn library that made the training stage more straightforward, and meant that I could experiment with a range of classifiers that I had not previously considered. I initially started with the classifiers mentioned in the Technical Background section, and collated their performance by plotting confusion matrices and accuracy metrics such as the f1-score. During this stage I noticed that the performance of the trained classifiers were not satisfactory. More details into why this may be can be found in the following section, Critical Evaluation.

Due to these classifiers under-performing, I felt that I should experiment with a selection of new models such as Random Forests, Decision Trees and k-Nearest Neighbours. These models returned slightly more satisfactory results, however the accuracy metrics were still not up to a high enough standard. This leads me on to the next stage, adjusting features and changing how we split our test and train data.

Changes in Pre-Processing and Feature Tuning

As mentioned in the first sub-section regarding pre-processing, we have to remove the data with missing values, which disappointingly brings the total number of users down considerably. One method to accommodate for this lack of data is by training the model on multiple splits of training and test data. I used 10-fold cross validation to split the data into 10 separate sets, and then trained the model with each individual set. ScikitLearn contains a function that could carry this out. A small snippet of how this operates on a Decision Tree classifier is demonstrated below.

```
#Splitting the in data
feature_cols = ['text_age','text_gender','image_age','image_gender','demographic_age','
    demographic_gender','demographic_ethnicity']
X = in_data[feature_cols]
y = in_data[['labels']]

#Pre-processing functions
min_max_scaler = preprocessing.MinMaxScaler()
X = min_max_scaler.fit_transform(X)

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()


cv = ShuffleSplit(n_splits=10,test_size=0.3)

# Train Decision Tree Classifer on each Cross Validation Split
scores = cross_val_score(clf,X,y,cv=cv)
```

Code Listing 3.4: Example code making use of 10 Fold Cross Validation on a Decision Tree Classifier Validation

One of the reasons that I felt that the models were not performing to their optimal accuracy was due to a lack of features. The current features extracted solely made use of comparisons between a user defined value and a predicted value, but there was a second dimension that was not utilised to its fullest extent, a comparison between two predicted values. Below, I have included a new definition for comparing two predicted values returned from the APIs, using the confidence values by averaging them and then returning a value of inconsistency.

Definition 3.5. Let f be a function which returns the value of n-inconsistency for the feature set

containing two predicted values for gender:

$$f(a, a_{conf}, b, b_{conf}) = \begin{cases} n = (a_{conf} + b_{conf})/2 & \text{if } (a \neq b) \\ n = ((1 - a_{conf}) + (1 - b_{conf}))/2 & \text{if } (a = b) \end{cases}$$

Where we have defined

$$\begin{aligned} a &\leftarrow \text{predicted gender API 1} \\ a_{conf} &\leftarrow \text{confidence gender API 1} \\ b &\leftarrow \text{predicted gender API 2} \\ b_{conf} &\leftarrow \text{confidence gender API 2} \end{aligned}$$

With this new method of being able to extract a value of inconsistency from two predicted values, I added two new features to our feature set. Firstly, a comparison of the two Image API Gender results. Secondly, a comparison of the Text API Age and Image API Age, that made use of the second age function that I had developed in the inconsistency section. With these two new features and cross validation implemented, I felt that it was time to progress with more experimentation on the classifiers.

Further Training & Testing

Further training of these classifiers was carried out in a similar vain as before, except that I included the new features and cross validation to potentially increase the accuracy of the model. Referring back to Suarez et al. [5], where they used an ensemble classifier to combine different classifiers, I felt that this may potentially work for my scenario. I experimented with different combinations of classifiers that performed well using a simple Voting Ensemble Classifier that is provided in the ScikitLearn library.

At this stage of the project I had developed a range of models, and felt that I had reached an acceptable standard of accuracy with the features provided. Now was an appropriate time to evaluate the classifiers, and more specifically evaluating how effective inconsistencies are as a feature to be used in scam detection.

Chapter 4

Critical Evaluation

4.1 Introduction

In this penultimate section, I will discuss the results by dividing them into three particular topics. Firstly an analysis of the features I had collected, and how they differ between real and fraudulent users. Secondly, the classifiers I trained and tested are discussed by comparing their respective metrics. Alongside the analysis of the classifiers, I looked into which features were more effective in the classification process. To bring the evaluation to a close, a final section to discuss how effective the use of contradictions are in the detection of scam users.

4.2 Feature Analysis

The first stage of analysis that I undertook was a basic comparison of how individual features represent the different users. Table 4.1 below, containing a set of figures, shows a list of the features on each row, with the real users represented on the left and the scam users on the right. I have included a histogram for each category of feature; age, gender and ethnicity, whilst also including two features that show the inconsistency between two sets of API data. Starting from the top down I will describe what the graphs represent, and their relevance. The addition of the mean inconsistency on each figure will help to compare between the two classes within a feature. Below the list of histograms, I have provided Table 4.2 which shows the proportion of users with greater than zero inconsistency.

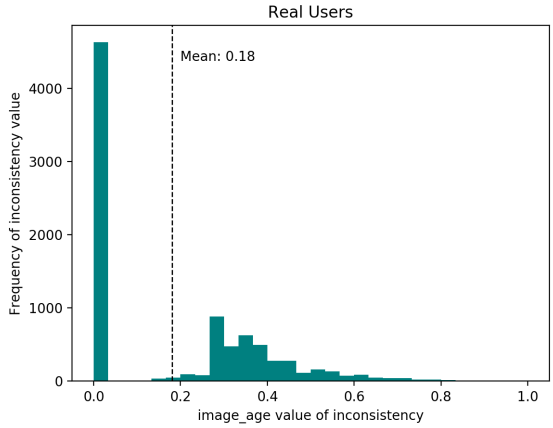
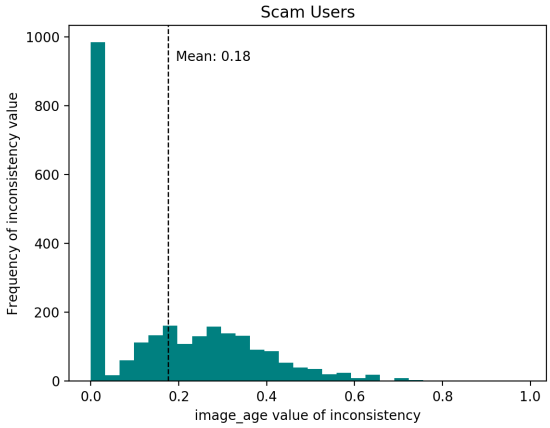
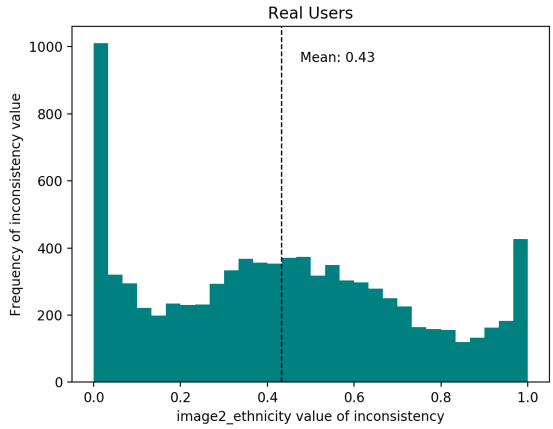
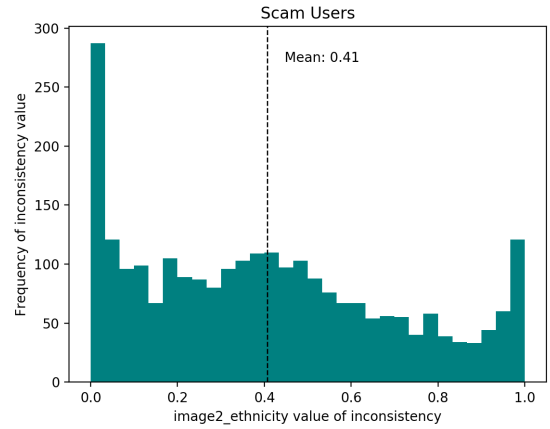
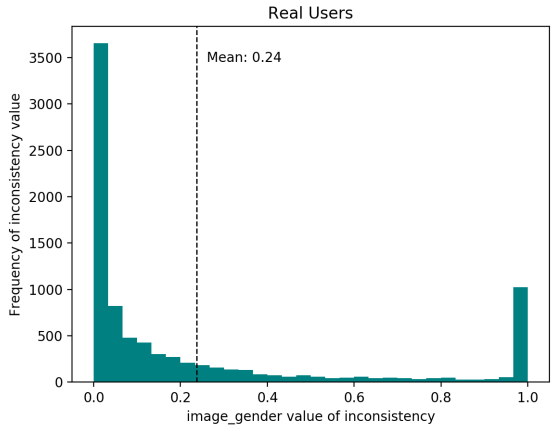
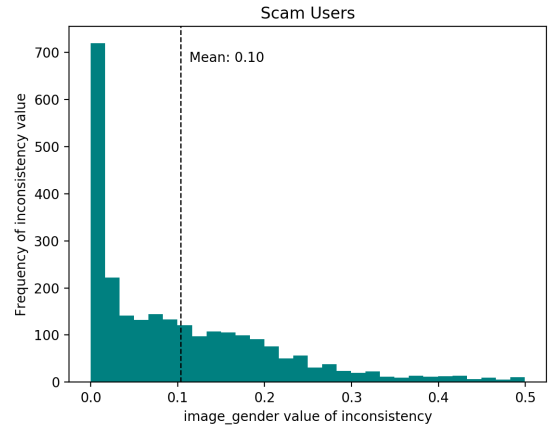
The first feature, *image_age*, does not provide a strong distinction between scam and real users as the majority of the both types have a value of inconsistency of 0. The mean value of inconsistency is the same for both users, however a larger proportion of scam users have an inconsistency of above zero, which is insightful, as it was the first time I have been able to confirm that one may be able to use inconsistencies to determine if a user is real. The second feature, *image2_ethnicity*, was underwhelming to say the least. I had expected that the ethnicity feature would provide the most contrast between scam and real users. But, in reality, the graphs show that the inconsistency values related to ethnicity tend to be the same for both user types. This will most likely mean that this feature will not provide a great deal of valuable information when classifying the users. The third feature in the table, *image_gender*, shows that the scam users may not necessarily have higher inconsistency values than the real users. For this feature, the mean value of inconsistency for real users is roughly 0.24 whereas for the scam users it is much lower with a value of 0.10. Although this feature contradicts our hypothesis that scam users will tend to have higher inconsistency values than the real users, it will still provide valuable information when classifying, as there is potential to distinguish between the two user types. However, a larger proportion of scam users do not have an inconsistency value of zero, and therefore we may be able to extract some more meaningful information from the feature.

The final two features listed in Table 4.1 are both features that represent values of inconsistency between predicted data. Both features, *text_image_age* and *image_image2_gender*, provide similar insight into whether this category of feature set will prove valuable when classifying profiles. The first being that both features show a slight shift towards higher inconsistency values for the scam users. This therefore means that there is a possibility that both of these features will add to the accuracy of a classifier.

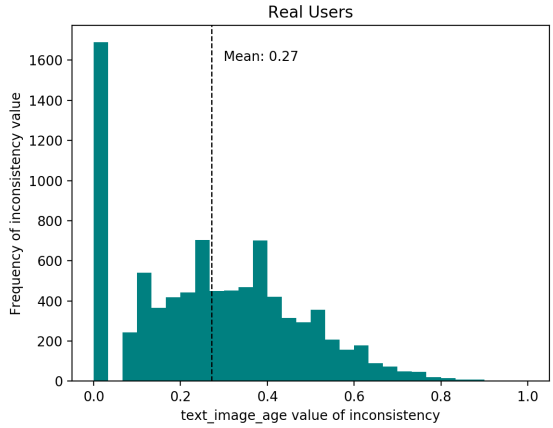
However, there is one reoccurring fault that all of these features show, that being a large portion of profiles contain an inconsistency value of 0. This can be seen in Table 4.2, where for some features the

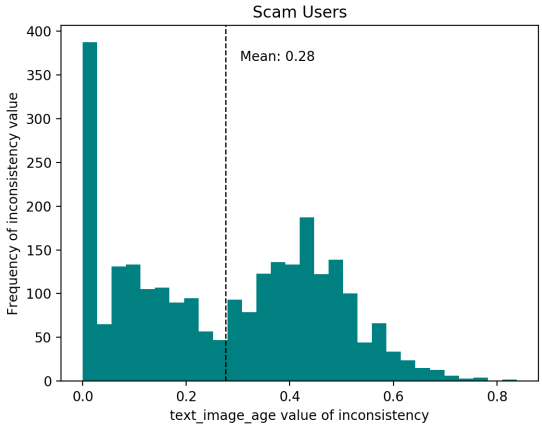
proportion of users with non zero inconsistency are as low as 50%. The issue with this is that if a profile's inconsistency values are all 0 then it would be impossible to determine what category the profile belongs to. If a large majority of profiles do fall into this scenario then the classifier will have a low prediction accuracy. Hopefully, with the full set of features, the amount of profiles that contain no valuable data will be minimised, and hence prove less of an issue.

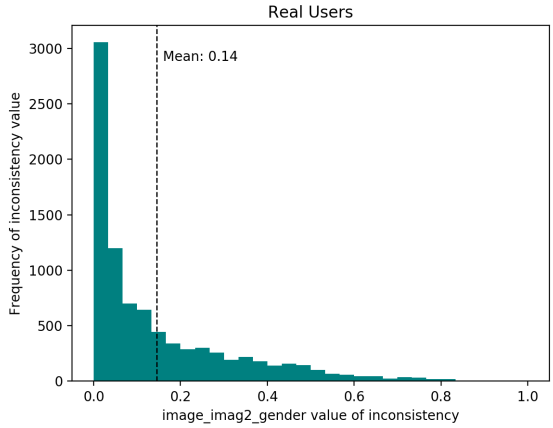
Table 4.1: A table showing histograms of a selection of features, split by real and scam users

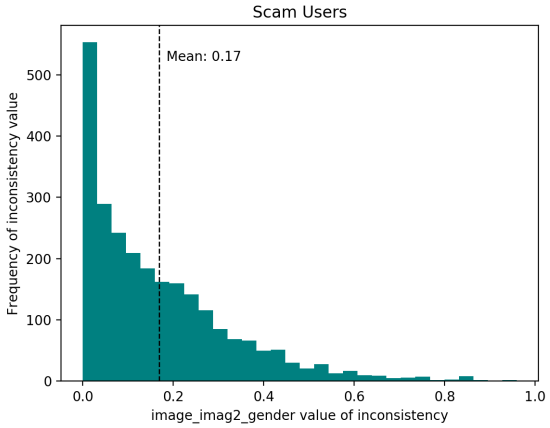
Real Users	Scam Users
	
	
	

Real Users | Scam Users









Feature	Real	Scam
image_age	0.46	0.61
image2_ethnicity	0.86	0.86
image_gender	0.52	0.57
text_image_age	0.80	0.82
image_image2_gender	0.65	0.72

Table 4.2: A table containing the proportions of users with inconsistency values greater than zero. All features referred to in the histograms are present.

4.3 Classifier Analysis

With a firm grasp of what our features represent and how valuable they may be, in this section I will evaluate how effective the classifiers I tested are. After considering the results from the previous section, I was not anticipating very accurate models due to how narrow the spread of data in the features were. To measure the performance of each classifier, I made use of four key metrics. Firstly a confusion matrix which shows the split of data contained into the following four categories: True Positive, False Positive, True Negative and False Negative. The final three measures of performance are Precision, Recall and F1-Score. These three metrics are defined in Definition 4.1.

Definition 4.1. Definition of Precision, Recall and F1-Score [21]:

$$precision = \frac{TP}{TP + FN}; \quad recall = \frac{TP}{TP + FN}; \quad f1 = 2. \frac{precision \,.\, recall}{precision + recall};$$

As mentioned in the Project Execution section, initially I did not utilise any cross validation methods when training the classifiers, however due to poor performance I went back and re-implemented cross validation for all classifiers. The results in this section will all be taken from models that have used 10 fold cross validation for training and testing, with the final results being an average of the scores from each split.

4.3.1 Naïve Bayes

		predicted class	
		real	**scam**
actual class	**real′**	986	760
	scam′	121	383

	Precision	Recall	F1-score
Real	0.89	0.56	0.69
Scam	0.34	0.76	0.47
avg/total	0.61	0.66	0.58

Table 4.3: Evaluation Metrics of Naïve Bayes Classifier

The first classifier to be trained with the data was Naïve Bayes. As mentioned in the Technical Background section, I was hoping for some strong results with this classifier due to it not being affected strongly by unbalanced classes. The results of this classifier can be seen in 4.3, and it appears to classify the scam users at a reasonably high rate, as can be seen with a recall of 0.76. However, this classifier is very poor at predicting a real user with an recall of almost 50%. I believe that the reason for this classifier not predicting as well as anticipated is most likely due to the features being too dependant on each other. For example, the feature set contains multiple features for age and gender.

I felt that the overall results for this classifier were very underwhelming, and that the data I had collected could be used in a much more effective manner. The choice of classifier was not suited to this dependant data, and hence needed to be changed. The most important takeaway point is that it classified more than half the scam users correctly. This was the first piece of evidence that using contradictions to detect fraudsters may be a plausible method. With this classifier performing well on the scam users, however with low precision, meaning that the predicted scam profile is most likely a real user, the motivation to carry forward will be to improve the precision of the classifier, while retaining the high recall.

4.3.2 Logistic Regression

The results outputted from the Logistic Regression model were very poor. As seen in Table 4.4, the classifier was not able to predict a single scam user correctly, leading to precision, recall and f1-scores of 0. As mentioned in the section regarding the Naïve Bayes classifier, this is most likely due to the features being too dependant on each other. After some brief research, I discovered that this classifier tends to operate much more poorly with unbalanced classes in comparison with the previous classifier. This may be another contributing factor to the poor results.

After some consideration it was clear that my initial preconception that the features were going to be independent, or that the assumption that they were, would not have a drastic effect on the classification. However, this assumption did not prove to be correct as seen from these results, and hence I needed to change the direction of this classification section if I was going to improve the accuracy of our model.

actual class \ predicted class	real	scam
real′	1732	0
scam′	519	0

	Precision	Recall	F1-score
Real	0.77	1.0	0.87
Scam	0	0	0
avg/total	0.38	0.5	0.43

Table 4.4: Evaluation Metrics of Logistic Regression Classifier

actual class \ predicted class	real	scam
real′	1486	244
scam′	253	267

	Precision	Recall	F1-score
Real	0.85	0.86	0.86
Scam	0.52	0.51	0.52
avg/total	0.69	0.69	0.68

Table 4.5: Evaluation Metrics of Decision Tree Classifier

4.3.3 Decision Tree

After the previous two classifiers' poor results, it was clear that I had to choose a more effective classifier. As there was no strong linear split of the data I felt that a Decision Tree may be an effective choice. As you can see in Table 4.5 the results were much more impressive than the previous two classifiers. The points worth mentioning are that firstly the precision for the scam class has improved from 0.3 in Naïve Bayes, up to 0.5 with the decision tree. This is a major improvement as more than half of the predicted scam users were part of that class. The f1-score for the scam users has also improved to over o.5, which is insightful as it shows that improvements can be made in the classification of the scam users, without changing the feature set.

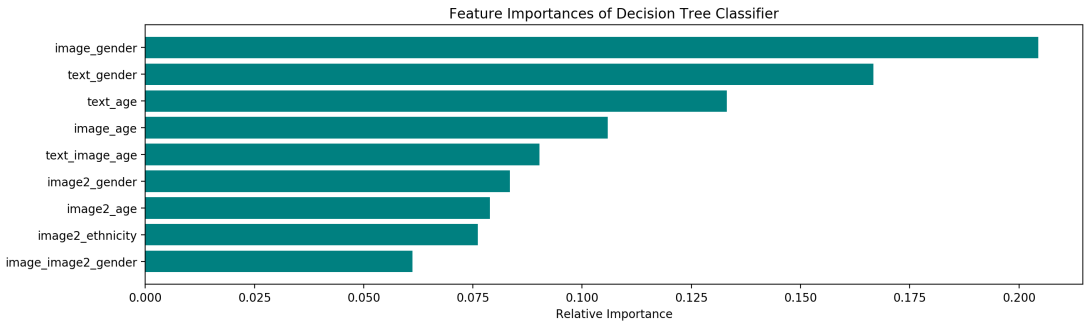


Figure 4.1: Feature Importance Graph for the Decision Tree Classifier

Feature Importance

As this was the first classifier that proved worthy of detailed analysis, I felt that a review of the feature importance was valuable as it would give an insight into whether the predictions from the Feature Analysis section were accurate. As seen in Figure 4.1, the features that tend to provide more of a weighting are those involving gender and age. The second image API features, *image2_* do not prove themselves to be as valuable as the features extracted from the other APIs, especially the ethnicity feature, which is suprising, as I felt that ethnicity had the largest potential to show contradictions. The reason for this poor performance is most likely due to the accuracy of the API, rather than the inconsistency measures that I had developed, as all the features from this API perform badly.

One of the more interesting points that can be extracted from this graph is that the API vs API contradiction value for age, *text_image_age*, tends to perform quite well. This was a surprise as I initially felt that the comparisons between APIs would not provide valuable information. This proved to be incorrect. The second feature that was extracted between two APIs, *image_image2_gender*, performed the worst of all of the features. This is most likely due to the reasoning given in the previous paragraph, that the second image API was not as accurate as the others in their predictions.

4.3.4 Random Forests

		predicted class: real	predicted class: scam
actual class	real′	1671	71
	scam′	274	234

	Precision	Recall	F1-score
Real	0.86	0.96	0.91
Scam	0.77	0.46	0.57
avg/total	0.81	0.71	0.74

Table 4.6: Evaluation Metrics of Random Forests Classifier

With the previous classifier, a Decision Tree, outperforming all other tested classifiers, the logical step in the next choice was a Random Forests classifier. The reasoning behind this is that a Random Forests classifier minimises the problems that stem from a Decision Tree classifier, such as error due to bias and variance. It overcomes these flaws by gathering a collection of Decision Trees built from subsets of the features, and then aggregating all the individual trees to limit over-fitting the data [22]. There are some issues with using this classifier, such as the training time required for each forest is considerably larger than that of the Decision Tree, and when computing this with 10-fold cross validation the time is considerably longer.

The results from this classifier are shown in Table 4.6. As expected, the results are similar, if not better, than that of the Decision Tree classifier. The scam value for Precision, 0.77, is very high, meaning that there is a very low probability of a real user being predicted as a scam user. This can be seen in the confusion matrix where only 71 real users were classified as a scam user. This classifier on the whole provides a reasonably accurate prediction, as it contains the highest average metrics, without regard to the Recall for scam users. This value of 0.46 shows us that the probability of a scam user being correctly classified is below 50%. This is worse than that of the classification in the Decision Tree, surprisingly as this model is made up of a collection of Decision Trees, one would expect the accuracy for all classes to increase.

Feature Importance

As seen in Figure 4.2, the average importance of all the features gathered from the second image API are considerably higher of that seen in the Decision Tree Classifier. This therefore may support the argument that these features do not provide valuable data, as when their weighting is increased, the scam recall value decreases. I experimented with the removal of these features, and it lowered the overall accuracy

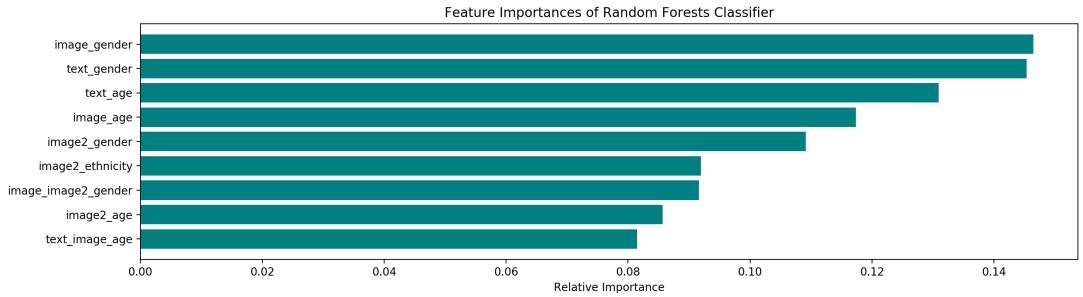


Figure 4.2: Feature Importance Graph for the Random Forests Classifier

of the classifier. This does not disprove the argument that the features from this API are not accurate, as upon removal of the features, the model can only be trained on 5 distinct features which was not sufficient for a Random Forests classifier.

A second interesting result from this graph is the performance of the *text_image_age* feature. In the previous classifier it performed much more effectively in comparison to the other features, however, in the Random Forests model it performed the worst, with the *image_image2_gender* feature not far in front. Upon experimentation of the removal of these two features, the classifier performed worse than before, which was to be expected, as the range of feature importance is very small, and therefore removing any of these features will hinder the performance of the classifier.

4.3.5 K-Nearest Neighbours

		predicted class	
		real	**scam**
actual class	**real′**	1595	156
	scam′	317	182

	Precision	Recall	F1-score
Real	0.83	0.91	0.87
Scam	0.54	0.37	0.44
avg/total	0.69	0.64	0.65

Table 4.7: Evaluation Metrics of k-NN Classifier

When analysing the results of the two first classifiers, Naïve Bayes and Logistic Regression, I came to the conclusion that the data was not easily separable by a linear model. This lead me to two choices of classifiers, either a Support Vector Machine with a non-linear kernel or K-Nearest Neighbours. Upon experimenting with both models, the SVM did not provide any classification, however the K-Nearest Neighbour classifier showed some promising results.

As seen in Table 4.7, the classifier performs well when predicting real users, with an f1-score of 0.87 which is as high as the Decision Tree classifier. However the performance of predicting scam users was underwhelming. Firstly, the recall was 0.37, which is the lowest of all classifiers, and secondly the precision of the scam class was 0.54, indicating that roughly half of the predicted scam users were real users. Upon further experimentation with the classifier, I could not improve on the results seen above, even by changing some of the hyperparameters such as the number of neighbours.

Although I had only made use of a single type of ensemble classifier, it would have been more advantageous if I had experimented with other methods such as a weighted classifier based on the individual performance of the sole classifiers. This may have provided a more promising result, however, due to classifiers working more effectively when the combined errors of the classifiers are poorly correlated, it may not have had too much of an affect on the model at hand. This is because all of the individual classifiers make use of the same feature sets, and therefore will all provide a similar error.

4.3.6 Ensemble Classifier

actual class \ predicted class	real	scam
real′	1556	183
scam′	227	283

	Precision	Recall	F1-score
Real	0.87	0.89	0.88
Scam	0.60	0.55	0.57
avg/total	0.74	0.72	0.73

Table 4.8: Evaluation Metrics of the Ensemble Classifier

With a range of classifiers now trained and tested, I felt that there was still more I could do to try and improve the accuracy of the models. With inspiration taken from Suarez et al. [5], as referenced in the Technical Background, making use of an ensemble classifier to combine previously trained classifiers may improve the accuracy of the model. I used a simple Voting Classifier with the voting function set to hard, as using the soft method is only recommended when the individual classifiers perform well by themselves, which, was not the case for this scenario. The ensemble was made up of the following previously mentioned classifiers:

- **Random Forests:** This was chosen due to the overall performance for both classes of real and scam users.
- **Naïve Bayes:** This classifier was selected because of its ability to classify the scam users.
- **Decision Tree:** This classifier was selected because it outperformed the accuracy of the Random Forests classifier when predicting scam users.

Table 4.8 above shows the results from this ensemble classifier. The results were on the whole were on par with that of our previously best classifier, Random Forests, which both have roughly the same f1-score. However, the aim for this stage in the classification was to improve the recall for scam users, and that is exactly what was achieved. Now more than half of the scam users were being predicted correctly, but it was still not able to classify the scam with the accuracy of Naïve Bayes.

4.4 Summary

Comparison with Previous Work

The results above give an insight into the effectiveness of contradictions when used in a classification problem, but it does not provide a comparison to previous methods. In this final section I will evaluate the most effective classifier in comparison to the classifiers used in Suarez et al. [5].

The recall of the ensemble classifier which gave the best results of all classifiers trained, for both classes, was roughly 0.72, and the recall for predicting scam users is roughly 0.55. In comparison, the results from Suarez's et al. classifier returned a recall of 0.97 for all classes, and n recall of 0.93 for predicting scam users. The comparison of the minority class recall results show that the original classifier outperformed the one developed in this research in achieving its original problem, classify false profiles. However there is a more valuable question to ask, are contradictions a suitable measure to detect scam or fraudulent data?

Contradictions and Online Dating

It is hard to deduce whether using only contradictions is a suitable choice for classifying other data besides online dating profiles. There are some key flaws with using this methodology for this classification problem. As mentioned in the Feature Analysis section, a plethora of real users have an average inconsistency

value higher than that of some scam users. This lead me to the conclusion that a large amount of real users tend to misrepresent their profiles to appear more romantically approachable, whereas scam users will try to minimise inconsistencies in their profile to give the impression of honesty. This conclusion makes more sense for the age features, as one may want to appear younger to match with younger profiles, however providing false information regarding ethnicity and gender does not seem to make sense. The categories in which I would assume real users tend to provide false information is their age, photos and description, which were all made use of in this classifier and may have led to poorer results.

This prompted me on to the conclusion that if one is going to solely use contradictions to detect fraud, then the real data must not be drawn from a system that encourages a user to lie about themselves, which online dating sites tend to do. This, however, may not be the only reason for the for underwhelming results from this classifier. It may potentially be down to the performance of the API's. Without knowing the accuracy of the API's used, I cannot be certain that this is the only reason for poor performance, but it may have contributed to a range of factors that hindered this project.

However, this does not make the research at hand irrelevant, as it shows that some of the inconsistency features obtained, especially relating to age and gender, provided a reasonably valuable feature in the classification problem. Although I was not able to implement the contradictory features alongside the features used in Suarez's et al. [5] classifier, I do believe that it would have increased the accuracy of the their model. Despite my beliefs, without the implementation of this extended classifier, one can not be certain in its potential.

Chapter 5

Conclusion

5.1 Summary

To conclude, I have summarised the project achievements in comparison to the original objectives and hypotheses. I have also discussed the current state of the project and any advancements that could be made to the project in the future.

5.1.1 Objectives

Below are the original objectives outlined in the introduction section:

1. The first contribution will be the background research and survey of literature surrounding both measuring contradictions and the deception in online dating sites. This should give the reader a solid understanding of the background knowledge needed for the remainder of the paper.
2. Collect a large dataset of online dating profiles of both the fraudulent and genuine users.
3. Develop a framework to represent and quantify the inconsistency of a dating profile, with the potential for the framework to be expanded into other applications.
4. Implement a system to extract the necessary data of the contradictions in the profiles that will be required to evaluate the classifier.
5. Use the data gathered to train a classifier to accurately detect the fraudsters in online dating profiles using the identified inconsistencies.
6. Implement the a selection of contradiction features alongside the features used in Suarez's et al. [5].

Before comparing the results of the project to the initial hypotheses, it is valuable to know which objectives have and have not been completed. I was able to complete all objectives, bar the final one, implementing a selection of contradiction features alongside features used in previous works from Suarez et al [5]. The repercussions of not implementing this are discussed in further detail in the upcoming section regarding the value of contradiction.

5.1.2 Hypotheses

The original hypotheses outlined in the introduction section were as follows:

1. Online dating profiles can be used in conjunction with a range of APIs to extract data that shows the inconsistencies within the profiles. These inconsistencies can be used to train a classifier to detect fraudulent profiles. This can be tested with a range of accuracy measurements such as F1-score, precision and recall.
2. Online scammers tend to have more inconsistencies within their profiles. This can be tested in the Critical Evaluation section when looking at how well the classifiers have worked.

3. Analysis of the classifiers should provide some insight into whether using contradictions is a feasible method in scam detection.

The primary hypothesis was confirmed to be true when collating the results provided in the critical evaluation. Although the hypothesis was proved to be correct, it does not provide too much of an insight into how accurate using inconsistencies are for fraud detection. This is because the hypothesis does not state any particular accuracy goals needed to detect these profiles, and hence, if even one fraudulent profile was detected, then the statement was proved to be correct.

The final two hypotheses are more closely related to how useful contradictions are for classification problems. The second hypothesis was proved to be incorrect in the critical evaluation section when analysing the features, and the third hypothesis was met, however, we can not draw too much information from the analysis of the classifiers as mentioned in the following section.

5.1.3 Value of Contradictions

The main objective of this project was to evaluate the use of contradictions in fraud detection, more specifically in online dating fraud. The results from this project provide only a small amount of evidence into whether using contradictions is a feasible choice in classification problems. I concluded that there were two primary reasons as to why the initial hypothesis, that contradictions are valuable for all classification problems, was inconclusive.

Firstly, due to the landscape of online dating, encouraging users to not be truthful with their profiles so as to appear more romantically approachable, perhaps it is not the most ideal forum to make use of contradictions. As discovered in the feature analysis section, a wide range of real users had higher inconsistency values than the scam users. This is most likely because of the natural tendency that real users would over sell themselves using data that is not necessarily truthful, leading to a profile ridden with contradictions. This lead me to the conclusion that online dating is not an ideal testing ground to evaluate how effective contradictions are.

The second reason for this is that without the implementation of the final objective, using contradictions alongside other features, I cannot conclude that inconsistencies provide a positive change in the accuracy of detecting fraudulent dating profiles.

Although there are two negative outcomes from this project, it has still provided some interesting results regarding contradiction. The final classifier performance was promising, and hence there is potential to increase the accuracy with more inconsistent features, such as those mentioned in the project execution regarding the comparison of the description and images.

5.2 Future Work

Finally, the points related to potential future work that could be done to improve on the current state of the project. The following list will outline the areas that could be expanded upon:

- Implement a range of the features collected from this study alongside a set of features that do not utilise contradictions, and evaluate the contribution in adding inconsistency related features.
- Test a new set of features that make use of contradictions within online dating profiles, for example detect the inconsistencies from the image and description that do not relate to age, gender and ethnicity.
- Implement a classifier to detect fraud using contradictions under a different environment. For example, in the detection of scam emails or in websites that are potentially fraudulent.

Bibliography

[1] Statista. *Online Dating Market Size*. 2019. URL: https://www.statista.com/outlook/372/100/online-dating/worldwide.

[2] UK Finance. *Fraud the Facts*. UK Finance, 2019. URL: https://www.ukfinance.org.uk/system/files/Fraud%20The%20Facts%202019%20-%20FINAL%20ONLINE.pdf.

[3] Federal Trade Comission. *Romance Scam Reported Losses*. 2019. URL: https://www.ftc.gov/news-events/blogs/data-spotlight/2019/02/romance-scams-rank-number-one-total-reported-losses.

[4] Monica T. Whitty. *Who can spot an online romance scam?* 2019. URL: https://www.emerald.com/insight/content/doi/10.1108/JFC-06-2018-0053/full/html.

[5] Guillermo Suarez-Tangil et al. *Automatically Dismantling Online Dating Fraud*. 2019. arXiv: 1905.12593 [cs.CR].

[6] Tanya Kang and Lindsay Hoffman. "Why Would You Decide to Use an Online Dating Site? Factors That Lead to Online Dating". In: *Communication Research Reports* 28 (July 2011), pp. 205–213. DOI: 10.1080/08824096.2011.566109.

[7] Rory McGloin and Amanda Denes. "Too hot to trust: Examining the relationship between attractiveness, trustworthiness, and desire to date in online dating". In: *New Media Society* 20 (Nov. 2016). DOI: 10.1177/1461444816675440.

[8] P Bak. "Sex differences in the attractiveness halo effect in the online dating environment". In: *Journal of Business and Media Psychology* 1.1 (2010), pp. 1–7.

[9] Jingmin Huang, Gianluca Stringhini, and Peng Yong. "Quit Playing Games With My Heart: Understanding Online Dating Scams". In: vol. 9148. July 2015. DOI: 10.1007/978-3-319-20550-2_12.

[10] Meenakshi Nagarajan and Marti A Hearst. "An examination of language use in online dating profiles". In: *Third International AAAI Conference on Weblogs and Social Media*. Jan. 2009.

[11] Janneke Loo. "Text-Based Age and Gender Prediction for Online Safety Monitoring". In: *International Journal of Cyber-Security and Digital Forensics* 5 (Jan. 2016), pp. 46–60. DOI: 10.17781/P002012.

[12] Aric Bartle and Jim Zheng. "Gender classification with deep learning". In: *Technical report*. The Stanford NLP Group., 2015.

[13] Leopoldo Bertossi, Anthony Hunter, and Torsten Schaub. "Introduction to inconsistency tolerance". In: *Inconsistency Tolerance*. Springer, 2005, pp. 1–14.

[14] Anthony Hunter and Sébastien Konieczny. "Approaches to Measuring Inconsistent Information". In: vol. 3300. Jan. 2005, pp. 191–236. DOI: 10.1007/978-3-540-30597-2_7.

[15] Kevin Knight. "Measuring Inconsistency". In: *Journal of Philosophical Logic* 31.1 (2002), pp. 77–98. ISSN: 00223611, 15730433. URL: http://www.jstor.org/stable/30226747.

[16] Leonard Richardson. *Python HTML Scraping Library*. URL: https://pypi.org/project/beautifulsoup4/.

[17] Omer Arar and Kürşat Ayan. "A Feature Dependent Naive Bayes Approach and Its Application to the Software Defect Prediction Problem". In: *Applied Soft Computing* 59 (May 2017). DOI: 10.1016/j.asoc.2017.05.043.

[18] Avinash Navlani. *Understanding Logistic Regression in Python*. Dec. 2019. URL: https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python.

[19] Avinash Navlani. *Understanding SVM in Python*. Dec. 2019. URL: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python.

[20] Kenneth Reitz. *Python Requests Library*. URL: https://pypi.org/project/requests/.

[21] Jason Brownlee. *How to Calculate Precision, Recall, and F-Measure*. Jan. 2020. URL: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

[22] Neil Liberman. *Decision Trees and Random Forests*. Jan. 2017. URL: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991.