Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

Automated classification of pet scam websites

Ronel Mehmedov

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

Monday 13th September, 2021

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc 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.

Ronel Mehmedov, Monday 13th September, 2021

Contents

Executive Summary

Pet scams are a type of cyber fraud where criminals create website platforms to advertise fictitious pets and extract money without providing any goods to the victims. Scammers also use fake delivery websites to extract additional funds from victims on the pretext of delivery costs and charges. This project seeks to develop a classifier that differentiates between legitimate and fraudulent pet sale and delivery websites. In order to do this, I collected data online by creating a web scraper which extracts HTML and images from pet sale and delivery websites. Subsequently, I processed the data to create features that can be used for automated decision making. By testing these features as data points for a classifier, the project also explores the usefulness of these features in predicting whether a website is a pet scam or not, which is useful information that can be used by other researchers who seek to develop anti-fraud software in this context.

My research hypothesis is that a machine learning classifier can be developed that can detect pet scam websites with high degree of accuracy. This project explored two features that have been proved to connect pet scam websites in a previous study - HTML and image similarity. The project assessed their value as data points for a machine learning classifier. In addition to this, the project explored four more features which have not been previously discussed in literature regarding pet scams - presence of address, image to HTML ratio, menu options and financial data. I found that the feature that evaluates menu options has very high predictive accuracy in classifying pet sale websites as legitimate or fraudulent. I also found that the minimum, maximum and average price found in a website could be also a useful indicator in that regard as well as the ratio between HTML pages in a website and the number of images found in it. Moreover, I tested four different algorithms and identified that Random Forests provide the best performance for classifying pet advertising websites and Logistic Regression has the highest results for pet delivery websites. The important outcomes of this project can be summarized as follows:

- I developed a machine learning classifier that can predict whether a website is a fraudulent pet sale website with 97% accuracy. The classifier for delivery websites can predict whether a website is a fake delivery website with 89% accuracy.
- In order to do this, I created scraping scripts which extracted HTML and image files from 5808 websites. The scraper extracts statically served pages as well as asynchronously loaded content.
- I created two features that explored the usefulness of data points which have been previously highlighted to have high predictive accuracy in the context of pet scams.
- I analysed the data obtained and created four features that can be used to automate the detection of fraudulent websites in the context of pet scams. These features have not been discussed in previous pet scam literature and can be applied in subsequent projects in the same field.

The code used for scraping images and HTML, feature engineering and training the machine learning models is available for viewing at https://github.com/Ronel-Mehmedov/dissertation2021.git.

Supporting Technologies

- I used the Selenium, BeautifulSoup and Requests libraries for extracting HTML and image content from websites and tokenizing HTML pages to build a web scraper
- I used the Imagehash library in order to obtain perceptual hashes for images and compare their similarity
- I used the HTML_similarity library for comparing the HTML similarity between websites
- I used the Sci-Kit Learn library to train and test the machine learning models in this project

Acknowledgements

I would like to express my special thanks to my supervisor, Dr Matthew Edwards, for his help and guidance in this project.

Chapter 1

Introduction

Online fraud has increased dramatically as the popularity of the internet has grown. The annual losses from cyber fraud have increased by $700mn between 2019 and 2020 in the US [9]. According to reports published by the FBI, total losses in the region of $4.2bn were sustained in the US for 2020 [9]. According to estimates of the National Fraud Authority in the UK, losses due to cyber fraud total £1.3bn for the period between 1 January and 31 July 2021 which constitutes a 300% increase from the figures published for 2020 [5]. It is also established that as much as half of the UK population have been a target of a scam, and 8% have actually been defrauded [27].

Pet scams are a form of online non-delivery fraud whereby scammers use advertisements and online platforms to extract payment from victims. The usual medium for reaching potential victims is through websites that serve as landing pages for searches by people looking to buy pets. Though fraudsters do not impersonate particular businesses, they put great efforts into creating websites that look authentic [28]. The websites are then advertised on the web through ads or social media to increase the likelihood of the sites being visited. Part of the strategy in this regard is that the pets are often advertised at a significant discount to attract purchases from viewers of their website [28].

The content of the website is moderated so that it appears as an authentic business. For example, many pet scam websites have logos and text that suggest association with credible organisations responsible for the shipping or sale of pets [18]. Pet scam websites often have pages for 'testimonials' or 'reviews' where lengthy descriptions are provided by fictitious customers, explaining how satisfied they are with the service they have received. Moreover, fraudsters use descriptions and images that seek to trigger emotional response in visitors, knowing that people who have reached that far into the websites are very likely to have attachment to animals. The purpose of this is to make the victim reach out for the contact details provided by the scammers.

Once the victim does this, the conversation between victims and scammers is moved outside the website platform and into another medium such as WhatsApp [28]. At this stage the website has served its purpose of connecting scammers with potential victims and the last remaining step is to elicit payment for the purchase of the fictitious pet. Once the victim accedes to the requests for payment they are directed to do so via non-refundable payment method such as Western Union which provides anonymity for the scammers and precludes a recovery action for the funds transferred [28].

However, just like with traditional non-delivery fraud, pet scams usually do not stop there. Scammers continue to request payments under different justifications, capitalising on the emotional investment achieved and the sunken cost fallacy that many people fall into in these scenarios [18]. These justifications are often connected to difficulties with transporting the pet to the victim's location and scammers even employ another set of websites which copy the features of an authentic pet delivery company. The victims are directed to these websites where they can 'track' the delivery of the pet, which seeks to reinforce the perceived legitimacy of the whole transaction. This is used to provide never ending sequences of requests, based on 'accidents' or unexpected problems with the delivery of the pet [18]. The justifications quoted are often such as that the pet needs immediate medical care or the pet being stopped at customs [28]. Scammers have a never-ending list of pretexts at their disposal and the scam stops only when the victims are financially unable to provide more funds or understand that they have been defrauded.

1.0.1 Why it is important to address pet scams

The reasoning about why pet scams are a major problem is the same as with advance fee and non-delivery fraud - the financial harm caused by these crimes is increasing. In 2017, the Better Business Bureau in the US reported that that four out of five sponsored advertising links for sale of pets online are actually websites that have been made by criminals resident in Cameroon [4]. Moreover, the large amounts reported to be lost to scammers are likely to be a lot less than the actual amounts elicited due to under-reporting. This is most likely because victims are often reluctant to report the fraud due to feelings of embarrassment about being tricked by the scammers [29]. Cases of pet scams have increased by 400% in the last year and it is expected that this figure will continue to rise because COVID restrictions give fraudsters a plausible explanation why buyers cannot get their pets in person or see them before making a payment [23].

Just like phishing emails are sent in large numbers, pet scam sites are usually produced in large numbers too. By doing this, scammers not only increase their online presence but are also able to cover a wider range of pets and animals for people with different preferences. However, scammers also are under pressure to reuse website content to keep the cost of their operations low [18]. If a new and completely different website was created from scratch for each breed or every time a website is taken down, that would require a lot of time and effort on scammers to design novel designs for each instance. Therefore, scammers usually employ a set of recurring features to design their websites and change other features that can be more easily replaced and produce greater visual difference - such as text, images, layout colours, fonts and other css features. A research paper published by Price and Edwards successfully isolated clusters of pet scam websites that share significant amount of similarities between each other [18]. This is of major importance because it means that the process of detecting fraudulent websites for sale of pets can be automated on the basis of their recurring features.

This paper seeks to take this a step further and create a machine learning classifier that can automatically categorise pet sale and delivery websites as fraudulent or legitimate on the basis of their features. As part of the project, data will be collected through a scraper which will extract the HTML and image data of pet sale websites for further analysis. The paper then investigates how this data can be transformed into features. Finally, the paper will cover how a machine learning classifier can be made and the predictive accuracy of the features used to build it will be assessed. Building a classifier and assessing its predictive accuracy would also provide useful information about the importance of the features used in the context of detecting pet scams and similar instances of cyber fraud. The insight about the predictive value of each of the features can be used to develop a robust anti-scam advice for the general public. Also, if successful, the classifier can be deployed to pre-emptively detect and report fraudulent websites which would have direct reduction of the number of people being defrauded.

The objectives of the paper can thus be summarised as follows:

- Create a web scraper that extracts data from pet scam and legitimate pet sale websites for further analysis
- Implement code that captures collected data into features that can be used for machine learning purposes.
- Create a classifier that distinguishes between legitimate and illegitimate pet sale (and delivery) websites
- Explore the predictive accuracy of a set of features in determining whether a pet sale or delivery website is fraudulent

Chapter 2

Technical Background

2.1 Comparison of pet scams with other types of cyber fraud

Pet scams are essentially a type of non-delivery fraud. This is a cyber fraud where the seller receives funds as part of an online purchase and does not provide the items bought [15]. It is often the case that dedicated websites are set up by criminals for the sole purpose of extracting such payments but the false advertisements could also be made on popular auction platforms or online sales websites.

Pet scams can also be compared to phishing. This is a different type of cyber fraud which involves theft of user data such as credential or credit card details whereby the attacker deceives users to share such information by pretending to be a trusted entity. The information can be obtained by prompting the user to enter their details on a page that is controlled by the attacker or it could be collected through a script that is activated upon opening a link or website. Phishing and pet scams are similar in that they target the general public rather than a particular person or organisation. However, there are aspects of pet scams that differentiate them from phishing websites. First, phishing often relies on impersonating a legitimate business to prompt a click or response from the victim. In comparison, pet scam websites rarely try to impersonate established businesses and instead rely on copying the features that present them as authentic small-scale businesses [18]. Moreover, pet scams are different in that the websites used are not the direct means of defrauding the victim, only a platform to connect the fraudster to the victim with the view of continuing the scam over a private messaging platform.

Another form of cyber fraud that is targeted at the general public is advance fee fraud. The earliest forms of this attack had a distinct methodology. They started off with unsolicited requests to potential victims to provide money on the promise of a return or financial gain [1]. Whilst the pretext used for the requests are numerous, they usually fall under one of three main categories: investment propositions, inheritance claims and charity payments [20]. If the potential victim responds to the message they are then asked to send funds and the justifications given depend on the initial context used for contacting the victim but the most common appears to be processing fees [1]. Pet scams are more similar to advance fee fraud in this aspect because the method for obtaining funds from a victim is based on deceiving them to make a voluntary transfer as opposed to collecting information from victims who believe they are providing their details to a legitimate business, as is the case with phishing. Just like with advance fee fraud, pet scammers request payment from the victim after initial contact is established and additional payments are requested subsequently for various reasons.

However, pet scams are different in that they are mainly perpetrated through advertising websites and scammers let victims initiate the contact. Another difference is that with advance fee fraud the scammers elicit payments through promises of a financial return for the victim whereas pet scammers rely on the emotional appeal of the pets advertised on their websites. Due to this emotional attachment and the fact that victims are usually the party that initiates contact, once the fraud is apparent victims are reluctant to report the crime due to the feeling of embarrassment. This has been noted in the context of romance fraud, whereby a similar tactic is employed but on dating platforms [29]. Scammers in that context impersonate a party with romantic interest in the victim and the justifications for requesting funds are usually related to travel costs necessary for them to meet in person [29]. The reluctance to

report a romance fraud could be higher than a pet scam but similar feelings of shame and embarrassment lead to under-reporting of the crime which, in turn, delays the development of mechanisms for preventing it. This is partly the reason why pet scams have proliferated rapidly in a short period of time - the Better Business Bureau for example reported an annual increase of 39% in the period from 2017 to 2019 [4].

The coverage of pet scams in media and in academic works has so far been centered on highlighting the proliferation of the crime and dissecting the methodology employed by criminals to make suggestions on the data needed to prevent the crime [28]. Attempts to devise mechanisms against pet scams currently remain underexplored and only two academic papers have focused so far on developing prevention mechanisms.

The first one is a research project by Norazman and Zamin that developed an application which scans contents of emails and reports a likelihood of the email being fraudulent [17]. The assessment of the email's legitimacy was performed by counting the frequency of keywords that have been found to be prevalent in pet scam emails. This particular paper focused on addressing pet scam emails and is reliant on a set of keywords that have been frequently used in 50 emails that were reported as pet scam emails [17]. The limitation of this application is that the the model used captured keywords from a very small dataset and, therefore, it might not generalise well to new data if deployed in practice.

The other paper that explored a way of detecting pet scams was published by Price and Edwards and it focuses on analysing website content which is closer to the objective of the present project. This research showed that features of fraudulent websites can be used as data points to isolate clusters that share significant similarities [18]. The features investigated were direct links between websites, textual similarity, image similarity and HTML similarity. The authors found that 90% of the websites analysed in the project can be connected by combining all four features. The connections derived from each method were validated from domain registration data such as date of registration, IP address and who registered the website [18]. So if this data was the same across the websites flagged as connected, the relevant connections were deemed to be externally validated. The fact that Price and Edwards found validated connections between a significant majority of the fraudulent pet scam websites means that an automatic classifier could be an effective way of dealing with this type of cyber fraud.

2.2 Obstacles to law enforcement and victim-focused approaches against cyber fraud

The main strategies to prevent victimisation nowadays are heavily influenced by the difficulties in identifying and persecuting online fraudsters. First, the geographic location of the offenders is not immediately available and in many cases it is impossible to obtain [27]. This is because scammers can obfuscate their location by using hosting providers that do not verify their identity and then use encrypted connection, mainly through virtual private networks, to connect to these servers [8]. Where websites need to be hosted for the crime, fraudsters use services provided by Namecheap for example to hide identities.

Secondly, even if the location of the offenders is established, most of them are resident in separate legal jurisdictions from the victims [27]. Therefore, to prosecute scammers, law enforcement agencies often have to rely on cooperation from the authorities in the relevant country where the scammers are located. However, according to research, most cybercriminals reside in poor and disadvantaged countries and collaboration has been noted to be more difficult with these jurisdictions [8]. The multi-jurisdictional nature of these crimes has the added complexity that criminals are able to redirect their connection across other jurisdictions before their signal reaches the victim's country. Cybercriminals can use a network of IP addresses to divert their connection through many jurisdictions before reaching their ultimate target which means that even if the connection is tracked in the reverse direction, cooperation from many jurisdictions could be necessary to establish the starting point of the connection [8].

For these reasons, the modern approach for policing non-delivery fraud has shifted to raising awareness amongst the target group for victimisation and proactively reducing the possibility for engagement between victims and offenders [14]. If the broader population is more aware of the methodology and the distinctive features of cyber frauds they are better equipped to spot them and avoid them. In addition, authorities have tried campaigns whereby they identify potential victims and contact them with the view

of raising awareness about the likelihood of a fraud taking place. According to Webster and Drew, authorities can use financial transaction data to spot ongoing fraudulent activity and inform the targets to prevent repeat victimisation [27]. The authors point as an example the initiative of the the Australian Queensland police department to analyse 'financial transaction data to identify recipients of money sent by Queenslanders to high-risk nations, such as Nigeria and Ghana. The victims are then telephoned by FCCG detectives and asked about the circumstances concerning their monetary transfers' [27]. The victims are then given information about the high-risk of the funds being received by fraudsters. This had limited success as many victims chose to ignore the warnings due to the emotional attachment to the narrative provided by criminals[27].

Another approach is to make the secure option the default case for users regardless of their internet activity. For example, compulsory antivirus software has been suggested as a measure in that regard [14]. Furthermore, users could be subconsciously influenced to make them more cautious when engaging in an activity that is likely to pose a threat to them. For example, in the context of anti-phishing measures, the green URL indicator in Internet Explorer is a sign that the user is searching safely and should be alert otherwise [14]. Additional example is the safety lock icon which shows that the connection with a particular domain is encrypted and therefore more secure than HTTP connection. Another method to make web users more resilient to cyber fraud is to increase awareness by investing in education of the general public at early age about the methodology of criminals [14]. It is clear however, that each of these measures has limited effect as they do not target particular types of fraud in order to develop specific measures that are highly effective against them.

2.3 Automated cyber fraud detection methods

The need to develop tailored response to different types of cyber fraud has led to studies of the methodology of the relevant crimes and the development of technology that is highly effective in detecting the means of perpetrating the crime [28]. This allows the mediums for cyber fraud, such as emails or websites, to be targeted accordingly and either filtered out or taken off the webspace. There are currently no technical countermeasures against pet scams but such measures have been explored in the other types of fraud that pet scams is similar to and, therefore, parallels can be drawn from these areas.

In the context of non-delivery fraud, software has been developed by Almendra that classifies listings on an auction site as legitimate or fraudulent [2]. The software takes the content of the listing as input and the features it assesses include 'price, date (when the listing was published), product category and seller. We also included information related to the seller: reputation score, account age, and number of recent transactions'[2]. The classifier had true positive rate of 83% and false positive rate obtained was 11% [2]. Whilst this is useful in the context of non-delivery of goods purchased at auction websites, the methodology cannot be replicated for pet scams where a separate website platform is used. This is because the features used in the classifier measure the frequency of the listings and their timings which is not available for comparison in a pet scam website. Therefore, currently there is a need for research that combines different techniques to target pet scams specifically.

In the context of phishing attacks, automatic detection mechanisms have been developed to detect websites on the basis of the text used or the website features. Phishing attacks work by impersonating legitimate businesses or popular websites and, therefore, attempts to detect these fraudulent websites are focused on capturing the visual similarity with the target. Anti-phishing software has been developed that checks for similarity with the logo of the legitimate website that is being protected [7]. The reasoning behind this approach is that if a website is using a business's trademark then it is trying to impersonate it with an unlawful purpose. In other cases, the DOM structure is used to check for structural similarity in order to identify phishing websites [24]. Scammers have adapted to this by creating visually similar but structurally different websites [6]. However, anti-phishing software has also advanced to analyse visual similarity as well. One of the methods to implement this is to extract frequencies or 'super-signals' from the HTML structure of the assessed website and compare them the relevant counterparts from the official website [6]. Therefore, in the context of phishing, anti-fraud initiatives capitalize on the visual similarity between the target and fraudulent websites to detect the latter. This means that these techniques cannot be applied with the same efficacy against pet scam websites.

On the other hand, the fact that pet scams use independent websites means that these can be directly

removed from the web if they have features that have strong connection to websites or methods known to be used by cybercriminals. Therefore, having a good profile of what a pet scam website looks like could be a major contribution towards combating this kind of crime. For this reason, Whittaker and Button suggest that the best way of dealing with pet scams is to encourage reporting of the incidents and supporting the development of knowledge databases such as petscams.com that can be a useful resource for people to verify the legitimacy of the parties that they are dealing with [28]. In their research paper, Whittaker and Button reviewed in detail the methodology of pet scammers and compared the similarities of their techniques to other types of online fraud. The authors highlight the importance of websites that list fraudulent URLs as they provide the first step towards deconstructing the strategies employed by fraudsters which can be helpful to prevent them from reusing these techniques again [28]. The data shared in petscams.com ranges between contact details and domain names of fraudulent websites, and comments that share additional data for each website. The organisation accepts reports from the public via online forms and these are then reviewed by volunteers. The latter keep track of the number of complaints received per website and assess the legitimacy of websites before uploading the details of the website as a warning that they are illegitimate [28].

The limitation of the blacklist approach used by petscams.com is that they only collect URL data and contact details about fraudulent websites with the view of highlighting these to the public as fraudulent. A more direct anti-fraud mechanism can be developed by collecting the data from these URLs and analysing it. The process of collecting online data is commonly refereed to as scraping and is considered next.

2.4 Web scraping

Web scraping is the method of extracting and saving data from the World Wide Web to a hard drive or database for further processing[21]. The available methods to perform this are based on utilizing the Hyper-text Transfer Protocol which underlies most web pages today. Zhao points out that extracting data from websites can be broadly separated into two functional parts - acquiring web resources and extracting specific information from the received data[31].

The first part revolves around issuing an HTTP request to obtain the content of a website. In line with the HTTP protocol, once a request is sent and processed by the server of the target website, a response is received by the scraping program which can then be further processed. As Zhao states, this part of a crawler deals with defining functions to deal with authentication, redirections,cookies and other aspects of establishing a connection with a web server through the usual request-response cycle[31]. The second major aspect of a web crawler is the additional parsing of the data with the aim of extracting the information that is sought after. After this is obtained, the data is saved in spreadsheets or comma separated value files for further analysis.

Uzun, Tarik and Kirat distinguish three different categories of web scraping - wrapper based methods, DOM-based methods and machine learning based methods[26]. Of these, the DOM-based methods are the most commonly used in the context of extracting small to medium sets of data. Dedicated libraries such as Selenium and Requests can be used to interpret the DOM elements to construct a hierarchical structure of elements. Once this structure is built the program can navigate through the tree structure and extract the required information. For example, elements have attributes and values which can be used by the program to navigate through the HTML structure.

Importantly, these libraries allow for dynamic content to be loaded which would be otherwise omitted from the HTML parsed. In such websites, the source code includes embedded Javascript code and a live user would usually click buttons to trigger the events that call the relevant script. These scripts interact with the DOM that represents the loaded page and, through requests to the web server, generate snippets of HTML that are loaded into the web page without the need for the remaining parts of the web page to be requested from the server. This has the advantage of improving user experience but in the context of scraping the content of a website, it means that parts of the page's HTML are not available up-front and can be obtained only after the relevant scripts are executed. Therefore, the ability of Selenium and Requests to render the Javascript code of a web page means that these libraries can be employed to extract the full HTML of a page on disc.

Another alternative to DOM based methods is the string based scraping method. In these cases the HTML is parsed as a string rather than a structure of HTML tags. This method can provide faster performance for extracting particular pieces of information as the program can skip the whole process of recreating the tree structure of HTML tags[25]. Uzun points out that using regular expressions can lead to significant time reduction, however, the overall accuracy of the data extraction is compromised with this method. Uzun reported that that the data retrieved with regular expression only had only 43.5% accuracy in comparison with the data that was expected to be received[26]. For this reason, web scraping in this project will be made with Selenium, Requests and BeautifulSoup.

2.5 Machine learning algorithms

Machine learning is the process of using data to create a model that uses performance measures of different features to automate a decision making process. Machine learning models are created through statistical algorithms that are mathematical functions designed for pattern recognition [11]. After such algorithms are applied to data, they output a model which captures the relationship between the data points provided to the algorithm and the resulting predictive response.

Machine learning algorithms can be broadly divided into supervised and unsupervised. With supervised learning, the algorithm is passed labeled data, meaning that a matrix of data points has a corresponding vector of qualitative or quantitative response that represents the correct classification or quantity for each row in the matrix [11]. Popular supervised learning algorithms are Linear regression, Random Forest, K-nearest neighbor, Logistic regression, Support Vector Machine. For example, with linear regression the model is derived by estimating coefficient for each feature presented to the algorithm. The formula for estimating the coefficients can be expressed as follows:

$$y = b_0 + b_1 x_1 + b_p x_p$$

In this formula y is the quantitative prediction and x1, xp are the features provided with the data. The coefficients are marked as b0, b1, bp - they have to be estimated for each feature and are calculated through the method of least squares whereby the coefficients are selected so that the square of the difference between the actual response and the value output by the model is minimal [11]. This model assumes a linear relationship between the features and the response and is, therefore, rarely the algorithm that produces the best fitting model. However, due to its high interpretability it is a useful starting point in the selection of an algorithm and it provides a baseline result that other models can be compared against. All algorithms have their advantages and disadvantages, so their performance can vary according to the size and the structure of the data provided. The alternative, unsupervised learning, no labeled data is provided and the algorithm is designed to observe clusters or distances between the data instances in the matrix. Examples of unsupervised learning algorithms include Singular Value Decomposition, K-means clustering and Principal Component Analysis.

Moreover, in machine learning there is also the distinction between classification and regression problems. In the case of the former, the task is to categorise data into a selected number of groups based on their features. On the other hand, with regression the objective is to derive a model that produces a quantitative response. For example, linear regression is a classic model for regression problems whereas logistic regression is used for classification tasks. Detecting fraudulent pet sale websites is a classification task as the required output from the algorithm has to be a binary response - fake or legitimate. For this reason, the this project will explore algorithms that tend to perform well with classification tasks:

- K-Nearest Neighbours
- Logistic Regression
- Create a classifier that distinguishes between legitimate and illegitimate pet sale
- Random Forests
- Support Vector Machine

2.5.1 K-Nearest Neighbours

For each row in the features matrix the program calculates its distance from the other occurrences in the dataset. The resulting collection is ordered and the first K occurrences are taken into account - their mode of classification is attributed to the particular instance that is being assessed by the algorithm [11]. The K parameter can be tuned and it has a strong impact on the algorithm's performance with respect to underfitting and overfitting the training dataset. The main advantage of this algorithm is that it is easy to implement and because of it's simple approach takes little computation time. However, this is only because the algorithm implements instance based learning and it does not derive a function from the dataset to be then applied to feature cases [11]. For this reason, the algorithm's performance tends to decline with large datasets. Also, this method suffers from an additional penalty with increase in dimensions in the data and is prone to overfitting. This can be particularly problematic with missing values and outliers.

A KNN classifier has been successfully used in the area of medical fraud [10]. The classifier separated practitioners with good and bad medical practices on the basis of features like number of services provided which measures the likelihood of charging for greater duration than actually provided or over-servicing of patients whereby the latter are given more services than they need for their medical condition [10]. The authors explain that 'the KNN was selected for this study because it is a widely used profile-matching technique and it bases its classifications of each case on those of its nearest neighbours using different decision rules' [10]. KNN classification has also been successfully deployed in the area of credit card fraud to isolate transactions that are out of the usual pattern for the account holder [13]. The KNN algorithm is used in this project because it performs well on smaller datasets and it can be useful in establishing a baseline against which other algorithms can be tested.

2.5.2 Logistic Regression

Logistic regression is used to determine the probabilities or likelihoods. This algorithm can be used in cases were the dependent variable is binary but it can also be used to obtain rank ordering. Logistic regression can be utilised to categorize data by attributing coefficients to variables on the basis of labelled data. In this sense, the algorithm captures linear relationships from the features matrix, however, unlike linear regression, logistic regression converts the number derived from established coefficients into a number between 0 and 1 [11]. By using the Sigmoid function logistic regression introduces non-linearity into its output. This is done according to the formula shown in Figure 2.1:

$$\frac{e^{(\beta_0+\beta_1 x)}}{1+e^{(\beta_0+\beta_1 x)}}$$

Figure 2.1: Formula for Logistic Regression

This algorithm has the advantage that it makes no assumptions about distributions of classes in feature space, however, it constructs linear boundaries and that can be problematic for highly non-linear data [11]. Logistic regression is usually not prone to overfitting, however, this type of error increases in datasets with many features and regularization techniques are needed to address this [11]. Logistic regression has been successfully used for detecting fraudulent accounts on e-commerce platforms such as Ebay and Amazon [16]. The authors of the research chose Logistic Regression for the task because the algorithm provides a rank ordering on the classified data [16]. Whilst Logistic Regression classically outputs a probability between 0 and 1, it does so by converting bigger numbers into fractions using the Sigmoid function. Therefore, if the last step is omitted the algorithm can be used to obtain rank ordering. In this project, this was not done as a binary classification is the ultimate goal.

2.5.3 Random Forests

Random Forests is an algorithm that has the same methodology of Decision Trees but introduces measures to address its limitations, therefore it is important to review the way the latter operates first. Decision Trees are composed of decision nodes which contain conditions to split the data and leaf nodes that are used to derive the final classification for each instance in the dataset [30]. Starting from the root node, the algorithm tests the purity of the nodes obtained by splitting the data on the basis of each of the features. If the dependent variables are perfectly split in the two separate resulting nodes that would be an example of pure nodes and at that stage the algorithm stops splitting the nodes as a satisfactory way of splitting the data is already found [30]. However, if the split results in impure nodes then the process continues. At each split for each level, the algorithm tests the results on the basis of each feature to give optimal weight to splits that would produce a clearer classification. At each step, if the node itself has the highest purity compared with the possible splits, that node becomes a leaf node [30]. Once the training stage is complete, the code follows the tree structure established until one of the leaf nodes is reached which results in a classification.

The algorithm tries every possible split and chooses the one that maximises information gain [30]. Therefore, the algorithm is a greedy search in that it selects the best split for each level but it does not later backtrack to evaluate the effect of this choice on subsequent splits. This is why the random forest is usually the better performing version of the decision tree classifier. The Random Forests approach uses multiple decision trees and each node in this structure tests a random subset of the features provided by the dataset. Other than that, the same process is followed as with decision trees and the highest performing version is chosen as the final classifier. This preserves the main advantages of the tree based process but alleviates the problem created by relying purely on a greedy search. Random Forests is tested in this project as it often provides high performance compared to other classifiers [30].

2.5.4 Support Vector Machine

Support vector machines can solve both regression and classification problems. Support vector classifiers work by finding the best line that separates classes in a given dataset. Whilst the majority of the instances of the total number of classes can be separated by many different lines, SVC is used to find the line that maximises the distance between the line and the separate classes [11]. Many different lines are fit to the data and the best performing one is selected. The basis for the selection are support vectors. These are vectors that are plotted through the points of each class that are closest to the main fitted line and are parallel to it [11]. The distance between the line and the support vectors is the margin that is used as a metric to chose the best line. The line with the largest margin is thus selected as the best classifier. This method works even for data that is not linearly separable. The adaptation that makes this possible is to introduce a further dimension which allows the process above to be implemented [11]. Subsequently, the decision boundary obtained is projected back to the original dimensions using mathematical kernels [11]. The algorithm has been used in detecting mobile fraud, amongst other areas. The research compared call patterns, average duration and frequency of the calls as features [22]. The model classified malicious behaviour in cases where certain calls were out of the ordinary pattern for a particular user. The classifier had accuracy above 90%, and false positive rate below 10% [22]. This algorithm is tested in the present project because it performs very well with highly dimensional data due to the way it works which covers an aspect for which KNN for example has poor performance. By testing these different algorithms the likelihood of finding the best approach is increased.

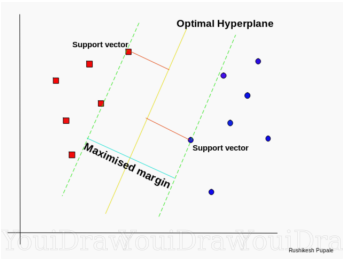


Figure 2.2: Margin maximisation - shared by Rushikesh Pupale

Chapter 3

Methodology

3.1 Collecting data from the web

3.1.1 Scraping target list of websites

In this project, web scraping was used to obtain data that can be used to differentiate between legitimate and scam pet websites. The raw data that is necessary to build a classifier in this context can be split into four categories - legitimate pet sale websites, legitimate delivery websites, scam delivery websites and scam pet sale websites. For the last two categories, petscams.com serves as an invaluable resource as it is a website that lists confirmed fake pet sale and delivery websites. For legitimate pet sale websites, https://marketplace.akc.org/ was used as a starting point - this website is a large collection of verified puppy breeders and their listings have been parsed to build a set of legitimate websites. Similarly, https://www.ipata.org/ lists accredited pet transport companies and this website was parsed to get a list of legitimate delivery websites.

The first part of the scraping process involved crawling these websites and extracting links to the target websites into csv files that can then be iterated over. The Selenium library was used for this purpose as the websites had asynchronously loaded content. As outlined above, Selenium executes the Javascript code that is embedded in the source code for the website and this allows all links to be found and crawled. Most importantly, the desired results for the https://www.ipata.org/ website were only available after a search button is clicked and Selenium has a convenient way of dealing with this - `search.send_keys(Keys.RETURN)` was enough to render the results that need to be parsed. It was apparent that the links were all included in web elements with the same class name so this was used to select the relevant elements. Finally, the anchor tags were selected within these elements and the href containing the website retrieved with `link.get_attribute('href')`. The same process was used to extract a list of websites from petscams.com and marketplace.akc.org.

After crawling the pages on marketplace.akc.org, 5,067 adverts were identified. Of these, some entries were invalid URLs that did not redirect to existing websites. In some cases the error was due to the authors of the posts in the marketplace website including manual text as explanation note next to their website link which was extracted together with the URL as it was part of the <a> tag content.Most of the invalid entries however, related to entries such as:

> http://Website currently being redone. Please call at 479-665177 with questions and for additional information

or

> http://None

After these entries were removed the final list of websites to be extracted from https://marketplace.akc.org/ was 4663. These contained valid URLs, however, some of the websites provided were facebook,

twitter and instagram pages. These links are obviously not useful for extraction as they are external websites that do not provide insight into the differences between legitimate and scam pet websites. Other URLs that were filtered out included those that did contain references to google and pinterest or contained the '@' sign and were therefore emails instead of websites. After removing these, 4,179 websites were left to crawl. During the extraction process, 38 websites from this group activated protections against crawling and did not return valid HTMLs. This conclusion is based on the response status 'Forbidden' that was received after issuing requests to these websites. Therefore, in the end 4,141 legitimate pet sale websites were successfully extracted.

The list of legitimate websites from IPATA yielded 39 entries at first. Four websites could not be extracted because they did not conform with the UTF-8 standard and so 35 websites were identified from this list in total. With regard to the scam websites, 8,344 scam pet sale websites were obtained from `www.petscams.com`. However, many of these were discovered to be offline and no longer available for extraction. Therefore, a short program was used to check the status code received from the response after issuing a request to these websites. Following this, the list of scam pet sale websites was reduced to 1,533 websites. It appears that in the period between obtaining this list and creating the code to crawl the websites, additional 233 websites were taken down. Therefore, the final number of scam pet sale websites identified was 1,300.

Similarly for scam delivery websites, 3,816 websites were originally obtained. After removing the websites that were offline at the time the list was created, the list was reduced to 459 websites. Same as with scam pet sale websites, additional websites were taken down before the extraction was finalised, in the case of scam delivery websites that number is 127. Therefore, the final number of fraudulent delivery websites obtained is 332.

3.1.2 HTML and images extraction method

The next part of the process was to scrape the target websites. Due to the large number of target websites, a number of options were considered, taking into account the runtime and the resultant HTML that was retrieved with each option. The first option considered was to crawl the websites with Selenium. The advantage of using this library is that it automatically loads the asynchronous content of a webpage once a session is initialised. The second option considered was sending requests with Python's Requests-HTML library. The library includes a `.render()` method which allows the program to load dynamic content like Selenium but it seems to have lower runtime duration. Furthermore, the sampled websites that were tested with both libraries produced equivalent results. For this reason, the requests library was used to extract HTML from the target websites. The third option that was considered was using the wget command and save results through a bash script. Whilst wget has the convenient capability of creating an exact mirror of the target website, it does not execute Javascript code and therefore it was considered that the retrieved HTML results could be missing the code that would have been generated from Javascript had it been executed with Selenium or Requests.

The code that deals with extracting HTML is `extractAllLinks.py`. The program accepts a URL and recursively parses links within the HTML structure obtained to find further sub-pages of the domain. A maximum limit is set at 80 pages per domain and the program does not parse additional pages after this. The program also saves the external links encountered in the HTML structure and also saves the header information about each domain. This information, along with the HTML obtained is saved in a separate folder for each domain by the driver function that calls `extractAllLinks.py`.

It was observed that a fixed time delay occurs between requests to different pages and this accounts for the initialisation of a new session. A threefold reduction in runtime was achieved by restructuring the code to reuse the same session every time the program sent a get request to a website. Furthermore, certain <a> tags contained links to a phone number or an email and thus the program was amended to include a clause that can catch the general exceptions thrown by the Request library.

The code for extracting images is contained in `extractImages.py` and follows the same process as the one for HTML extraction, except it collects image files found in the HTML files instead of saving the latter. For this reason, the way this code operates is not described separately.

Further processing was made after the websites were extracted. This did not affect the number of websites available for each group but some sub-pages were deleted for some of the domains. The aim of this was to remove HTML pages that had different names but were duplicates in the sense that they had the same content. It was observed that for some pages, links were detected by the crawling program as separate pages even though they were merely navigating to different parts of the same page. This happened because the code deemed two links as different even if the difference was following signs such as "?" and "#" which in the content of URLs indicate a specific position in the website or a query argument. Furthermore, the follow up processing verified that the documents extracted were valid HTML and the small number of documents that appeared to be in other format were removed. The numbers for the resulting dataset are provided in Table 3.1.

	Advertisement	Delivery
Legitimate	4141	35
Fraudulent	1300	332

Table 3.1: Table of websites extracted

3.1.3 Limitations of dataset

As described above, efforts were made to collect the largest portion of websites from each source. However, some websites could not be extracted because they were offline and no longer functioning. Moreover, some of the links obtained did not lead to functioning websites either. The main reason why some websites could not be collected was that they had anti-scraping mechanism in place. This conclusion is inferred from the fact that queries to them responded with 'Forbidden' error message. The `"User-Agent": "XY"` parameter was provided to all requests as some anti-scraping mechanisms review the headers to determine the handling of the requests. While this reduced the number of errors received, some websites did not return content despite this. Also, some websites had links stored in unconventional structure and the scraper could not identify any links to pages to be downloaded. Still, most websites from all sources were extracted in the scraping process.

The biggest limitation of the dataset is the number of sources that it is derived from. Due to the data being extracted from three online databases, it is possible that a machine learning model trained on such data is biased towards the websites listed on them. This shouldn't be a problem in the case of legitimate websites as one would expect a good amount of variance between the websites on `marketplace.akc.org`. However, the websites obtained from `petscams.com` could represent an incomplete profile of a pet scam website because many victims of cyber fraud are reluctant to report the offenders due to embarrassment or the low likelihood of recovering their funds.

3.2 Feature engineering

Feature engineering is the process of using domain knowledge to obtain properties, or attributes from raw data. These properties capture facts about a category of units and provide analytical information from which a machine learning model can be derived. The more narrow case of feature extraction is defined by Khalid and Nasreen as a method for transforming input data into low dimensional data whilst preserving the relevant information and dispensing with redundant and irrelevant information [12]. There are methods of automating the task of feature extraction but those were not deemed necessary for the present project. This is because this is the first project to create a classifier for fraudulent pet sale websites, and it is believed that significant results can be achieved by selecting features manually. Moreover, the an automatic feature extraction process would not be sufficient to develop features which require deeper understanding of the dataset.

The choice for the features used for the classifier was based on two factors. First, the two sets of legitimate and illegitimate websites were carefully examined and conclusions were drawn from their visual similarity. Secondly, the paper by Price and Edwards has highlighted features that have been used successfully to find clusters of websites among fraudulent websites [18]. Due to their proven value in

establishing a profile of fraudulent pet scam websites, HTML similarity and image similarity were the first ones to be used for the classifier. In the context of online pet scams, the raw data input are the websites represented by the relevant HTML, css and javascript for each domain. The features used in this project are:

- Image similarity - a binary feature that shows whether of a website shares similarity with one that is already flagged as fraudulent
- HTML similarity - a fractional value that measures HTML similarity of a website with one that is already flagged as fraudulent
- Address present - a binary feature which represents whether the website contains a valid UK or US postcode
- Image to HTML ratio - a continuous feature that measures the ratio of images obtained from scraping a website to the number of HTML pages scraped
- Menu options - a continuous feature that measures the ratio of keywords found in the menu options of a website and the number of options in the menu
- Minimum price - the minimum financial amount found in a website
- Maximum price - the maximum financial amount found in a website
- Average price - the average of all financial amounts found in a website

3.2.1 Shared images

As detailed previously, scammers face pressure to create large amount of websites and this often prompts them to reuse resources from pre-existing domains and images are often duplicated across fraudulent websites. This is supported by the paper published by Price and Edwards which highlights that 949 out of 1780 fraudulent pet sale websites shared at least one image [18]. The feature used in this project to reflect this was binary input for each website which represents whether it contains an image which is found in reference group of pet scam websites. The first step in creating this feature was to establish a reference group of websites which contained images with high number of occurrences in fraudulent websites. The codes for all images for these sites were stored in memory and all websites were tested whether they had any of these corresponding codes. Those who had a match were marked as websites sharing an image with the fraudulent reference group.

Perceptual hashing

In order to implement this, a hashing algorithm is necessary to translate images into comparable numerical values. In this project, I used a perceptual hashing algorithm to represent each image as a number. Theoretically, the advantage of perceptual hashing is that, unlike cryptographic hashing algorithms, the generated code is similar to the codes generated for perceptually similar images [18].

This method starts by reducing the size of the image to 32 x 32 pixels and converting it to grayscale which means that instead of the dimensions for red, green and blue, the resulting array represents variations between black and white [19]. After this, the Discrete Cosine Transform is applied to the image. This is a mathematical algorithm which is used to represent frequencies in an image. The top-left part of the resultant matrix always contains the lowest frequencies for the image and represents its general structure [19]. Due to this, the mean of the values of the low-frequency quadrant of the matrix are computed and these are subsequently used as a benchmark for creating the final hash [19]. Each bit is set to either 1 or 0, depending on whether the original value is above or below the mean. Finally the resulting output is represented as a hexadecimal integer which can be compared against other hashes derived from images.

I used a library, Imagehash, which implements the above algorithm. I wrote a python script, getHash.py, which iterates through the file structure containing all websites and creates a table. The first element of each row of this table is the name of the relevant website and all other entries for that row are the corresponding numerical representations of the images found for that website. Running the script

for each category of websites in this project I obtained csv files that are essentially 2D matrices of the above data - `codesForLegitDeliveryIMGS.csv`, `codesForScamDeliveryIMGS.csv`, `codesForIMGS.csv`, `codesForScamIMGS.csv`

Constructing 'shared image' feature

After this, in `getFrequencies.py`, I iterated each of the entries in the above csv files and built a dictionary where the key is the hash itself and the value is the number of times each code was encountered. In the end, I sorted the dictionary in ascending order created a list with the first hundred most encountered hashes. I repeated this process for delivery websites so this code yielded two new csv files - `targetHash.csv` and `targetDeliveryHash.csv`.

In a separate code, I removed certain hashes that accounted for a large number of images but were representing purely black and white images. Also, I built and inspected a histogram which represented the occurrence rates for codes in the fraudulent and legitimate hash codes lists. In `analyseResults.py`, the codes were checked to find the codes that had significant overlap across the two groups and then the corresponding hash codes were removed from the pool of target codes. I iterated over two of the four csv files created by `getHash.py`, namely, the csv containing a dictionary of website names and their corresponding codes. On each iteration, aka. for each hash code, I iterated the websites dictionary until I found a website that contained the hashcode and then I added that website to a final list which is the pool of websites that would be used as a reference group for the comparison. After this I iterated over a list of the selected websites and saved a separate list containing all hash codes from each of the selected websites. Effectively, this list contains all hash codes from the reference group which is used as a benchmark for all other websites to check if they have common images with the reference group. The pre-processing steps described above were made to obtain a reference group that includes the most occurring images in fraudulent websites which would, in turn, increase the likelihood of the resulting feature capturing a higher number of these websites.

The list mentioned above was built by running the same process for both delivery and advertising scam websites.This process resulted in a pool of banned images which could be used as a reference to construct a feature. This was done in `createTable.py`, the code here simply went through the list of websites and checked for each one whether it had a hashcode that matched one of the hash codes in the reference group. If a match was encountered, value of 1 was inserted into a table containing the websites as rows, and 0 if no match was encountered. Finally, in `removeBannedFromEval.py`, I removed the rows corresponding to websites which were present in the reference group. The purpose of this is to reflect the fact that these websites could not only yield '1' upon evaluation.

3.2.2 HTML similarity

The research paper published by Price and Edwards highlighted that HTML similarity between websites was most indicative of overall similarity between scam websites [18]. Therefore, in this project I followed the methodology used in the study above, namely, the HTML similarity was measured on the basis of the Jaccard index between the tag frequencies of two sites. Jaccard similarity index compares entries for two groups to measure similarity between the groups within a range from 0% to 100%. In this project, I used the HTML_similarity library to compute the Jaccard index between two pages. The crucial element that performs the relevant calculation is set out below:

```
def jaccSimilarity(doc1, doc2):
    doc1 = set(doc1)
    doc2 = set(doc2)
    intersect = len(doc1 & doc2)

    if len(doc1) == 0 and len(doc2) == 0:
            return 1.0

    denominator = len(doc1) + len(doc2) - intersect
    return intersect / max(denominator, 0.000001)
```

In this context, doc1 and doc2 are derived from the HTML files provided to the function and they contain the elements found in the corresponding files. The function used by the library to derive this is:

```
def get_classes(HTML):
    doc = Selector(text=HTML)
    classes = set(doc.xpath('//*[@class]/@class').extract())
    result = set()
    for cls in classes:
           for _cls in cls.split():
                  result.add(_cls)
    return result
```

Such comparison uses two HTML documents for comparison whereas domains usually compare many HTML pages as part of their structure. I decided to use only the main landing pages for comparison as they had the greatest level of similarity from my observations. The method I used to select the main page of each domain was two-fold. First, I checked if the domain contained an HTML file titled 'main' and returned this as the landing page if found. However, in the absence of this I selected the HTML file with the shortest name as many websites returned '/' for their root file.

Creating a reference group of websites and recording HTML similarity as a feature

As comparison between websites is necessary in this context, I created a reference group of websites for the purpose of constructing this feature and removed these websites from the evaluation set. My objective was to establish a reference group which is as small as possible, but also covers websites with HTML structure that is similar to a significant portion of the overall body of websites. This way the feature built on the basis of such reference group would detect an optimal number of fraudulent websites on the basis of HTML similarity without excluding large amounts of data from the evaluation set. To do this, I created dictionaries where the keys were website names and the values were python dictionaries. These dictionaries contained all websites of the same group (eg. fake delivery websites) and the corresponding similarity index of these with the key. For example, each site in the fake delivery websites is a key and the value of the key is a dictionary capturing the similarity indices between the key and all websites in the group. The dictionary was called 'frequencies' in the code as its purpose was to determine the websites with highest frequency in the dataset.

Once this dictionary was built, I used the following loop to obtain a reference group which contained the websites with the highest similarity frequency across the dataset:

```
while(countMax > 2):
  highestKey, countMax = getMax(frequencies)
  print("{} : {}".format(highestKey, countMax))
  mapping.append((highestKey, countMax))
  DeleteList(highestKey)
```

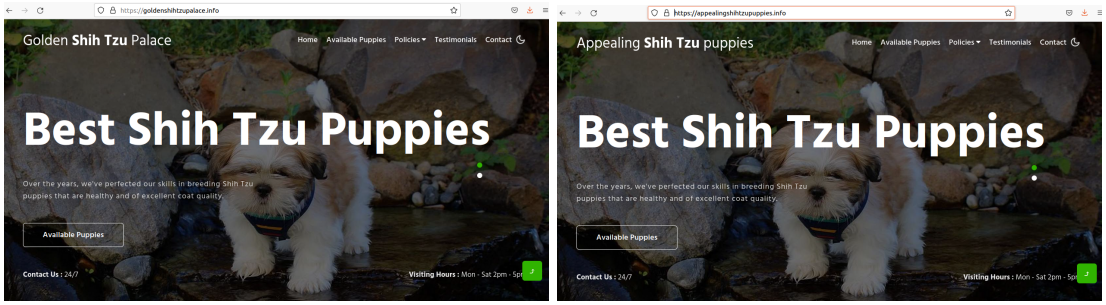
Here, `getMax` is the function that is responsible for finding the website with the highest number of similar websites (or websites with similarity index greater than 0.8 in the `frequencies` dictionary). Along with the website name, the function also returns the number of indices greater than 0.8 found for this website, called `countMax` in this instance. Upon each iteration I added to the final reference list the name of the website with highest HTML similarity across the dataset. To make this possible, I had to delete the added website to allow the `getMax` function to find the next website name in line.

However, I also removed from the dictionary all websites that had high similarity with the one flagged up by `getMax` upon each iteration. The intuition behind this is that I wanted each of the websites in the reference group to be representative of a different cluster. Therefore, by removing groups of similar websites, instead of simply the most frequent one on each iteration, I made sure that the next time `getMax` is called the function will select a website representative of a different cluster of websites in the dataset.

The exit condition for the loop is for the most frequently repeated website in the dictionary to be similar to 2 websites. At this stage any website added would be representative of only 3 websites and also the number of such websites in the dataset is very large so they cannot be removed from the evaluation set into the reference group.

I ran the process above twice - once with make delivery websites and another time with fake advertisement websites in order to obtain reference groups for both. After the reference groups were obtained, I executed a similar but abridged process to the one described above. This time, the code was run for all four groups of websites, including legitimate websites, and for each website a dictionary was built with indices measuring similarity with the websites in the reference group. For each website, the highest index was selected and this was the final value inserted into the features matrix. In the end, all websites that are part of the reference group were removed from the evaluation matrix.

A comparison between two different websites with Jaccard index of 1.0 is provided in Figure 3.1:



(a) goldenshihzupalace.info (b) appealingshihtzupuppies.info

Figure 3.1: Jaccard similarity of 1.0

3.2.3 Postcode present

The intuition behind checking if websites have an address in their content is that fraudsters should be more reserved to share such information. The cover that many fraudsters use to perpetrate their crime is the distance between them and the prospective buyer [28]. If no address is disclosed on the website, it would be easier for a scammer to inquire about the location of the victim and then claim their location to be at a prohibitive distance. This line of reasoning is more valid for advertisement websites but the feature was extracted for delivery websites too.

The approach in this case was to use regular expressions that capture UK and US postal addresses. Manual inspection of the websites highlighted that in some instances fraudulent websites had an address but no postcode but the extraction of that kind of data would not be possible with regular expressions alone and were not taken into account for this reason.

The baseline regular expressions used for US and UK postcodes were:

- \D\d{5}[-\s]?(?:[0-9]{4})?\D
- \D[A-Z]{1,2}\d[A-Z\d]? ?\d[A-Z]{2}\D

However, I made additional provisions in case the postcode was found somewhere in the beginning or the end of the string searched. In these edge cases the '\D' key would not match the postcode so the full list of the regular expressions I used is detailed below:

US Postcodes:

- ^\d{5}[-\s]?(?:[0-9]{4})?$
- \D\d{5}[-\s]?(?:[0-9]{4})?$
- \D\d5[-\s]?(?:[0-9]{4})?\D

- ^\d{5}[-\s]?(?:[0-9]{4})?\D

UK Postcodes:

- ^[A-Z]{1,2}\d[A-Z\d]? ?\d[A-Z]{2}$
- \D[A-Z]{1,2}\d[A-Z\d]? ?\d[A-Z]{2}$
- \D[A-Z]{1,2}\d[A-Z\d]? ?\d[A-Z]{2}\D
- ^[A-Z]{1,2}\d[A-Z\d]? ?\d[A-Z]{2}\D'

The code that created the feature started by iterating a list of all websites. The text for each HTML was retrieved with the functions below:

```
def visibletag(tag):
   if tag.parent.name in ['script', 'style', 'title', 'meta', '[document]', 'head']:
      return False
   if isinstance(tag, Comment):
      return False
   return True

def HTMLtext(body):
   soup = BeautifulSoup(body, 'HTML.parser')
   content = soup.findAll(text=True)
   visibleText = filter(visibleTag, content)
   return u" ".join(x.strip() for x in visibleText)
```

For each website, I applied the regular expression to check each of the pages for the domain - if at least one match was encountered the relevant website was assigned a value of 1. The only technical difficulty that had to be dealt with at this stage resulted from the way the dataset is stored on the filesystem. As websites are divided into 4 folders to represent the split between fraudulent/legitimate and delivery/advertisement websites, I created a function which takes these two dimension as input and, on the basis of these, returns the correct folder path as string to be appended to the code which is responsible for accessing the HTML documents.

3.2.4 Images to HTML ratio

This is the simplest feature derived from the dataset. During the scraping process, I noticed that extraction of images took a lot longer during the extraction of legitimate websites from https://marketplace.akc.org/. My hypothesis was that, whilst the average number of images per site may be average, it is more likely to find websites in the legitimate group that have number of images significantly greater than the average for the dataset. To measure this, I created a short program that counts the HTML files for each domain and the number of images extracted for them. I divided the number of images by the number of HTML files and recorded this ratio for each website.

3.2.5 Menu options

If one is to inspect pet scam websites one feature becomes noticeable - the menu options are very repetitive. An image illustrating the low variance in menu options is displayed in Figure 3.2. In order to analyse the variance of the text found in menu options I had to find a way to capture the text first. My starting point in capturing menu options was to tokenize the HTML with the Beautiful Soup library and find the <nav> tag. After this, I recursively collected the text inside the tags contained in the <nav> tag and added them to a list. Whilst this was effective for some websites, less than 40% of the websites in the dataset had their menu options wrapped inside a <nav> tag.

The second approach tested was based on the assumption that menu options would often be contained in a list as they are on the same hierarchical level in the structure of the HTML. My hypothesis was that, menu options could be detected by finding an <li> tag which has text inside of if that matches any of the following strings: home', 'about', 'contact'. The reasoning is that an overwhelming majority

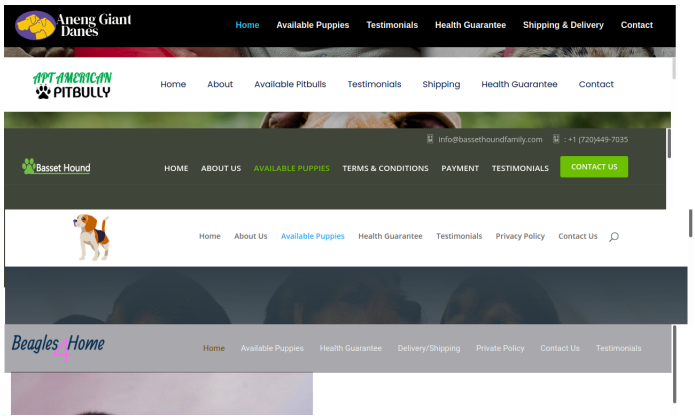


Figure 3.2: Menu bars for 5 scam websites

of websites have at least one menu option matching these strings. First, I collected a list of all <li> tags with `soup.findAll('li')` and then I check for each of these, recursively, if there is a text inside the <li> tag that matches the strings mentioned above. If a match was found, the code collected a list of <li> tags that are siblings of the tag that was found to contain a valid keyword. Finally, the text from each of these was extracted from each of these tags recursively and added to a final list that represents the menu options.

This approach covered two thirds of the websites in the database and valid lists of menu options were returned for those. However, a significant part of the websites remained uncovered as their menu options were not wrapped in <li> tags but tags instead. My conclusion was that its untenable to trace all possible tags containing options so I adapted the functions to find a tag that has text matching one of the keywords and proceed to collect text form the siblings of this tag. This had only a marginal improvement in the number of websites returning a response and the lists returned often contained non-menu related text. I tried to use both the <nav> and tag based approaches so that <li> tags were tested if <nav> tag was not present. That way three quarters of websites returned valid lists of options.

Upon inspection of the options collected with the above approach I noticed that the text collected included not only the options visible once a user enters the website but also the options from dropdown lists that appear once the user hovers over one of the main buttons. I inspected about 100 websites and noticed that the textual variance is actually greater in the dropdown options. The repetitiveness of the options texts appeared to be limited to the options that are immediately visible upon visiting the main page of the website. This meant that even if I elaborated the approach above to cover all websites, the feature constructed on the basis of data including dropdown options is less likely to capture the visual similarity of menu options that is apparent upon visiting a website.

Collecting main menu options only

To capture only main menu options, I wrote a program that appends to the options list only the text in those tags that are at the same depth in the HTML structure as the tag which contains keywords like 'home'. To do this, I used a recursive function which takes the tag containing 'home' etc.

```
def probe(parent, depth):
   if depth == 7:
       return False, [], depth
   siblings = getSiblings2(parent)
   lenOfSiblings = len(siblings)
   lenOfAs = 0
   toRemove = []
   for i in range(lenOfSiblings):
       if hasAtag(siblings[i], depth) == True:
           lenOfAs += 1
```

```
    if lenOfAs == len(siblings):
        return True, siblings, depth
    return probe(parent.parent, depth + 1)
```

The function calls itself until it finds a list of sibling tags that all have an <a> tag at the same level of depth in the HTML structure. This is effectively the model that captures menus on a higher level - it is independent of the type of tags used. What matters is that there are <a> tags ordered in the same way in a list of siblings which is basically what a menu in a website is. If such a list is not found the function moves up in the HTML structure by calling itself with the parent of the tag passed to it as argument. The verification that <a> tags are on the same level is performed by the `hasAtag` function:

```
def hasAtag(parentTag, depth):
    answer = []
    currQue = parentTag.findChildren()
    resQue = []

    for i in range(depth -1):
        for tag in currQue:
            for subTag in tag.findChildren():
                resQue.append(subTag)
        currQue = resQue
        resQue = []

    for tag in currQue:
        if tag.name == 'a':
            return True
    return False
```

If this function returns 'True' as its first element in the return tuple, another function moves the same number of levels as the 'depth' argument, except it goes down by finding the children as opposed to the parent as in the 'probe' function. Since 'probe' has already established that <a> tags are present at the same level this subsequent function simply collects the text from these <a> tags.

Using this approach, I was able to collect the menu options for all fake advertisement websites except 6 of them. There were more websites from the legitimate group whose menus could not be collected this way - 885 out of 4141.

Before proceeding to construct a feature I examined the results. I printed out all menu options found for both fake and legitimate advertisement websites. I created two separate histograms for both groups which showed the number each option text was found. It was interesting to find that the top one hundred options were surprisingly similar for both groups. However, what seemed repetitive about fraudulent websites was that their menus were almost invariably composed of the top most found options text, whereas legitimate websites often had one or two options in the menu that were outliers. For this reason, I created a list of the most found options:

bannedPool = ['home', 'available', 'about', 'shipping', 'shipment', 'delivery', 'polic', 'guarantee', 'condition', 'payment', 'testimon', 'review', 'reach', 'touch', 'faq', 'services', 'welcome']

Some of the words are abridged on purpose to cover different grammatical variations, such as testimonies, testimonials etc. For each website, I took the number of the options in the main menu and checked how many of the options were one of the keywords in the `bannedPool` - this number was stored as a separate variable. Finally, this number of banned words found was divided by the total number of menu options to obtain a ratio. Fraudulent websites, which very rarely had anything else apart from the word in the `bannedPool` were expected to return higher ratio than legitimate websites which had greater variance. I didn't use a threshold to create a binary feature, instead I simply recorded the ratio for each website. Where no menu was found, I recorded a 0.0. This is because the feature also captures the fact that fraudulent websites have more repetitive structure. A possible objection to this approach is that 0.0 indicates very low presence of banned keywords and my code places this value for all websites whose menu could not be identified at all. However, I believe this does not prejudice the correctness of

the methodology because this way the feature also captures the fact that legitimate websites have more varied structure.

3.2.6 Financial data

It is well reported that in the context of non-delivery fraud, cyber criminals use discounted prices on their websites to attract more victims. Given that pet scams are a variation of this cyber fraud, I decided to test whether the financial information displayed on pet scam websites could be a point of distinction from legitimate ones. This was done through three separate data points: minimum, maximum and average price found in a website. The term 'price' is key here: the feature is intended to capture information about services or goods offered as opposed to any numerical value within the webpage. This was done through the use of the following regular expression: \d+\.?\d+[$£€¥]—[$£€¥]\d+\.?\d+ .

3.3 Evaluation methodology

Machine learning algorithms work by approximating the function that can output the correct variables from underlying data [11]. How good this approximation for a given model is called statistical fit. An important aspect in this context is the difference between training accuracy of a model versus its testing accuracy i.e. how well the model performs on fresh data as opposed to data that it has been trained on. To achieve optimal performance, a model needs to minimize the two possible types of error - underfitting and overfitting.

Overfitting results from the algorithm capturing noise in the data that is not part of the natural pattern that is sought after [11]. Consequently, when presented with fresh data, the function derived from the model does not generalize well because the fresh data does not contain the random noise captured from the training data. On the other hand, underfitting occurs when the algorithm is unable to capture in sufficient detail the pattern that expresses relationships in the data [11]. Balancing these two types of error results in the best statistical fit or approximation and, consequently, in optimal predictions. To some extent, underfitting results from choosing an inappropriate model, for example, using a linear model to make predictions on highly non-linear data. Thus, choosing the appropriate model for each task is the most effective remedy against underfitting. On the other hand, overfitting occurs when the algorithm's optimises its training accuracy to the point where testing accuracy starts to decrease.

While searching for balance between the above mentioned concerns, training a good machine learning model needs to take into account a principle known as the bias-variance trade off. Bias is often the cause of underfitting because it results in disparity between precision and correct responses both on the training set and the testing set [3]. It is a consequence of a model that is too simple to capture the correct relationship in the data. On the other hand, a highly variable model is too complex to generalize well to new data, even though it has good performance on the testing set [3]. However, if either one of these errors, this results in an increase in the other. The key is that the relationship is not linear, and there is an optimal balance between the two types of error that results in optimal predictive accuracy. This can be done by exploring different algorithms and testing different parameters for each algorithm.

3.3.1 K-fold cross validation

Even if the training accuracy of a model is high, what matters in practice is its testing accuracy because that is the real measure showing how well the program will generalize to new data when deployed in practice. The idea of splitting data into training and testing sets, results from the fact that if the program is tested on the same data that it has learn from, this measure will not take account of the overfit of the model [11]. Thus, this separation is critical in achieving a model that generalizes well. However, if the data is simply split into training and testing sets, this means that the amount of data available for the model to train on is reduced. K-fold cross validation is a method to utilise all instances in the dataset both for training and testing purposes. This is achieved by dividing the data, and their corresponding classification values, into K number of sets. The usual cycle of training and testing is then

performed K number of times for each set separately. In the end, the mean average of the scores of each of these cycles is selected as the final score for the model's performance.

The advantage of this method is that it utilizes the whole dataset for training and testing. The disadvantage is that the process is K times slower than simply dividing the dataset into training and testing sets because in this case training and testing is performed K number of times. Because the dataset in this project is not as large as to make the time required to run cross validation prohibitive, this method was chosen to perform parameter tuning and model evaluation. To implement this, I used the kf.get_n_splits(data) method and the following code to assess different algorithms.

With respect to parameter tuning, the gridSearchCV module was used. It allows efficient testing of the performance of different parameters. The code used to select the best parameters is set out below:

```
folds = KFold(n_splits=5, shuffle=True, random_state=42)
folds.get_n_splits(data)

yList = []
resultsList = []
for train_index, test_index in folds.split(data):
    X_train, X_test = data.iloc[train_index], data.iloc[test_index]
    y_train, y_test = numpy.ravel(response.iloc[train_index]),
        numpy.ravel(response.iloc[test_index])
    svc.fit(X_train, y_train)
    label = svc.predict(X_test)
    for val in label:
        resultsList.append(val)
    for y in y_test:
        yList.append(y)

resultsList = np.array(resultsList)
yList = np.array(yList)
```

When assessing the performance of a model, different evaluation metrics can be used. Prediction accuracy is a good starting point - if the response vector includes binary features, then the number of correct classifications out of the entries in the vector would provide this value. However, this method of evaluation suffers from a major flaw in circumstances where the dataset is imbalanced. For instance, two separate data sets exist - advertisement websites and delivery websites. The fraudulent/legitimate split between websites is 23% and 87% accordingly. If a program simply predicts the most frequent class every time, the predictive accuracy would be as high as the percentile size of the majority class from the dataset. Therefore, a high predictive accuracy of a model is not necessarily indicative of a meaningful model that captures the patterns in the dataset correctly.

For this reason, a number of different evaluation metrics are used to assess performance. The metric used in this project is F1 - it is derived according to the formula shown in Figure 3.3.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Figure 3.3: F1 Formula

Recall in this context is the number representing the number of true positives compared divided by the number of true positives and false negatives added together. This metric is also called sensitivity and represents what proportion of the elements that are sought to be detected in the dataset are actually detected [11].Precision, on the other hand, represents the measure which shows how often the model is correct when it predicts a positive instance. To obtain this figure true positives are divided by the sum of true positives and false positives.

In this project, F1 is particularly good measure because it's value is more indicative of performance with imbalance sets. Moreover, other measures such as 'area under curve' evaluate performance under changing thresholds - and this is not directly applicable to algorithms such as decision trees that do not output probabilities. Therefore, F1 was selected as the evaluation metric because it provides a reliable account of how well the model would generalize to data and it can also be used with all algorithms.

3.3.2 Parameter Tuning

The purpose of testing different parameters is to ensure that each model is evaluated according to the optimal performance that it can provide. If only the default values are used for each model, a higher ranking of an algorithm could be simply a result of the other algorithms not being utilized to their highest capacity. I have set out below the ranges tested and the optimal values received for each algorithm below.

K-Nearest Neighbours

Parameter Name	Ranges Tested	Optimal Values (Advertising)	Optimal Values (Delivery)
K	range(1, 31)	7	9
Weights	Uniform, Distance	Distance	Uniform

Table 3.2: Optimal parameters for KNN

Logistic Regression

Parameter Name	Ranges Tested	Optimal Values (Advertising)	Optimal Values (Delivery)
Penalty Type	l1, l2, elasticnet, none	none	l1
C	np.logspace(-4, 4, 20)	0.0001	0.0001
Solver Type	LBFGS, Newrton-CG, Liblinear, SAG, SAGA	LBFGS	SAGA
Maximum Iterations	100, 1000, 2500, 5000	100	100

Table 3.3: Optimal parameters for Logistic Regression

Random forests

Parameter Name	Ranges Tested	Optimal Values (Advertising)	Optimal Values (Delivery)
Minimum Samples Split	2, 3, 4, 5	2	5
Number of Estimators	50, 100	50	50
Maximum Depth of Tree	2, 4, none	none	4
Critertion	Gini, Entropy	Gini	Entropy

Table 3.4: Optimal parameters for Random Forests

Support Vector Machine

Parameter Name	Ranges Tested	Optimal Values(Advertising)	Optimal Values(Delivery)
Kernel	Linear, RBF	RBF	Linear
C	0.1, 1, 10, 100	100	0.1
Gamma	Auto, Scale	Auto	Auto

Table 3.5: Optimal parameters for SVM

Chapter 4

Results

As detailed in the previous chapter, different models were tested and their performance was measured according to the F1 score achieved by each algorithm. In this section, I set out the results of these tests and explain the findings observed. The scores obtained from running the models with their best performing parameters are provided in Tables 4.1 and 4.2.

Model	Precision	Recall	F1	Accuracy	AUC
KNN	0.95	0.88	0.91	0.96	0.93
Logistic Regression	0.93	0.89	0.91	0.96	0.93
Random Forests	**0.95**	**0.93**	**0.94**	**0.97**	**0.94**
Support Vector Machine	0.94	0.90	0.92	0.97	0.94

Table 4.1: Results for advertisement websites

Model	Precision	Recall	F1	Accuracy	AUC
KNN	0.89	0.99	0.93	0.87	0.49
Logistic Regression	**0.89**	**1.00**	**0.94**	**0.89**	**0.50**
Random Forests	0.89	0.99	0.94	0.88	0.50
Support Vector Machine	0.89	0.97	0.93	0.87	0.51

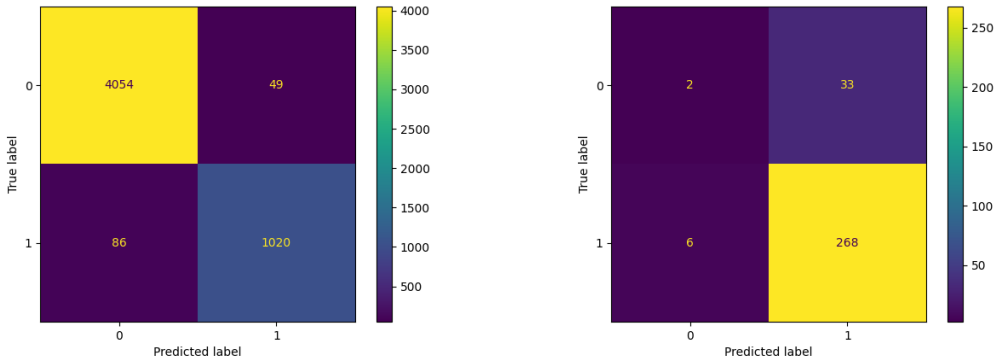
Table 4.2: Results for delivery websites

For advertisement websites, the Random Forests classifier achieved the highest F1 score of 0.94. This algorithm was also on par with or better than other algorithms in respect of precision, recall and mean accuracy. For delivery websites, Logistic Regression and Random Forests produced the highest F1 score of 0.94. Logistic Regression had marginally better recall rate and overall accuracy than Random Forests. An important observation is that the margins between the different algorithms are very close. This is because of the small size of the dataset - with larger amounts of data models such as K Nearest Neighbors tend to underperform on out of sample data. Overall, an F1 score of 0.94 for both classification tasks indicates that the machine learning model created in this project should be able to discriminate between legitimate and fraudulent in a reliable way for out of sample data.

It is notable that the Area Under the Curve (AUC) is relatively high for advertisement websites but significantly lower for delivery websites. This metric is not very indicative for Random Forests because this algorithm does not directly rely on thresholds for making predictions and AUC is rank measure that evaluates performance of an algorithm under changing thresholds. Therefore, it is a metric that usually shows how useful the model itself is, regardless of fine tuning of parameters. However, in the context of delivery websites AUC is equally low for Logistic Regression - an algorithm that does use thresholds.

To investigate this, the performance of the models can be visualised with the help of confusion matrices. They can be used to derive additional metrics such as specificity and false positive rate. Specificity expresses what percentage of the negative instances are predicted correctly by the model and is calculated by dividing true negatives by the sum of true negatives and false positives. False positive

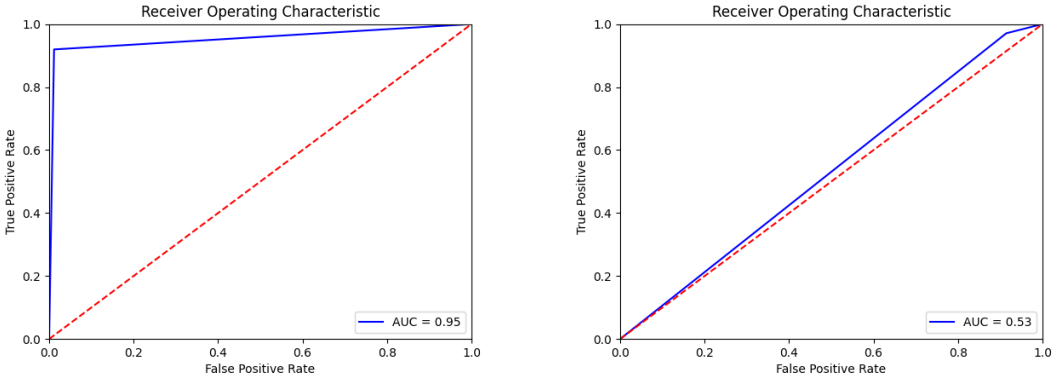
rate is derived by dividing the number of false positives by the sum of true negatives and false positives. The confusion matrices for both tasks are illustrated in Figure 4.1.



(a) Advertising websites (b) Delivery websites

Figure 4.1: Confusion matrices

It is noticeable that, for advertisement websites, the number of false positives and false negatives have similar relative weight compared to true positives and true negatives. However, for delivery websites this balance is skewed and the percentage of true negatives is very low. This results in a very high score for false positive rate compared to the advertisement websites - 0.91 compared to 0.01. Therefore, the low AUC score for delivery websites reflects the fact that the classifier is struggling with predicting legitimate delivery websites correctly. This is because AUC is a function of recall and false positive rate and the high score for the latter results in low AUC score for delivery websites. This is displayed in Figure 4.2.



(a) Advertising websites (b) Delivery websites

Figure 4.2: AUC

If false positive rate is the result of the model's inability to profile legitimate websites, then the reason for the high FPR score must be explored to address this issue in the future. The most plausible explanation is that the model struggles with identification of legitimate delivery websites because there are only 35 of them and 332 fraudulent delivery websites. The class imbalance in the delivery websites group is the reason for this deficiency in the model and would be best addressed by training the same model with a dataset that includes more instances of legitimate delivery websites.

4.1 Model including financial data

As detailed in the previously, financial data was not available for a significant portion of the dataset. However, 2984 websites still returned financial data and a the model identified above was also trained and tested on a special set that only included websites with financial information. It is considered that the tests discussed above provide more useful information because they are trained and tested with more data

and, in theory, should generalize better to unseen instances. Nevertheless, testing financial features this way provides useful information about their importance and predictive value. The test for this dataset were performed with the same models and parameters as those described above. The results are shown in Tables 4.3 and 4.4.

Model	Precision	Recall	F1	Accuracy	AUC
KNN	0.84	0.77	0.80	0.89	0.85
Logistic Regression	0.93	0.91	0.92	0.95	0.93
Random Forests	**0.98**	**0.94**	**0.96**	**0.98**	**0.96**
Support Vector Machine	0.92	0.48	0.63	0.84	0.73

Table 4.3: Results for advertisement websites (Financial Data)

Model	Precision	Recall	F1	Accuracy	AUC
KNN	0.85	0.97	0.91	0.84	0.59
Logistic Regression	0.85	0.89	0.87	0.78	0.58
Random Forests	**0.86**	**0.97**	**0.91**	**0.85**	**0.62**
Support Vector Machine	0.82	1.00	0.90	0.82	0.50

Table 4.4: Results for delivery websites (Financial Data)

The important observation to be made from the figures above is that the general performance of the model for advertising websites improves with the dataset that includes financial features. However, these results are less reliable because they were obtained from a smaller dataset. In everything else, the relationship between model performance across advertisement and delivery websites is similar - false positive rate for delivery websites being the main issue.

4.2 Feature importance

In order to assess the importance of each feature, the weight allocated by the best performing decision tree from Random Forests was selected. The metric used to select the tree was F1. Then, with the `.feature_importances_` attribute I extracted the weights given to each feature by the algorithm. The results are detailed in Figures 4.3 and 4.4:

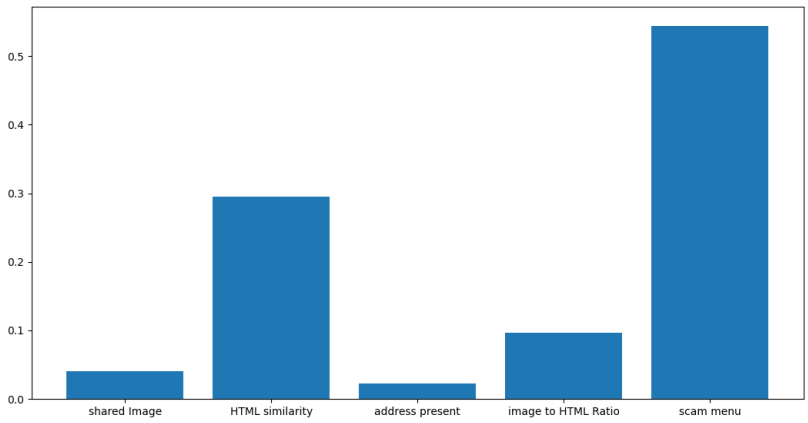


Figure 4.3: Weights allocated by Random Forests - advertisement websites

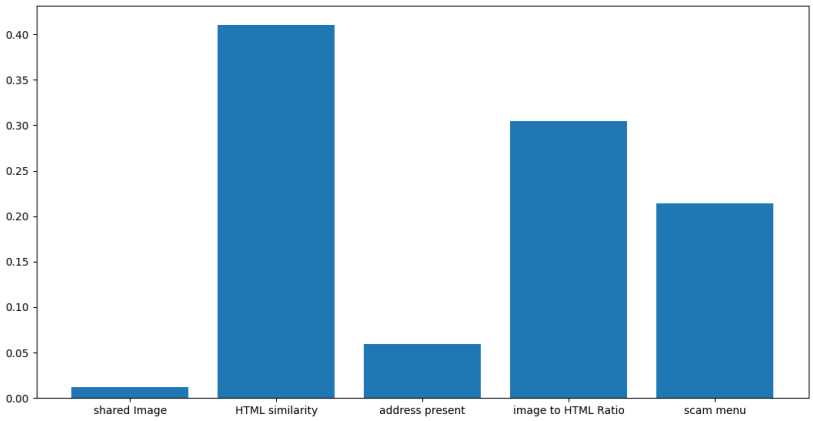


Figure 4.4: Weights allocated by Random Forests - delivery websites

4.2.1 Weights attributed

Contrary to expectations, shared images had very low weight attributed by the algorithm - 3.9% and 1% for advertisement and delivery websites accordingly. The presence of an address on the website had similarly low scores - 1.9% and 4.7%. It is surprising to see that the ratio between images and HTML has higher score than the previous two features. Number of images to HTML pages ratio was attributed weights of 9.5% and 33.2%. The feature is the second most important feature in classifying delivery websites. HTML similarity between pages is the second most important feature for classifying advertisement websites (28.6%) and the most important one for delivery websites(40.5%). The most important feature in the context of classifying advertisement websites is an identifiable menu with high presence of keywords found in fraudulent websites. This feature has weight of 55.9% attributed by Random Forests. It is also the third most important feature for delivery websites with weight of 20.3%.

In light of the above, HTML similarity and the menu structure of websites account for a large part of the decision making process of the algorithm. To confirm this, the best performing decision tree for advertisement websites was visualised. This is because Decision Trees chose the next level of features according to the purity of the resulting set so the features that produce a clearer separation between the classes naturally end up on top of the tree. Unfortunately, the graph is too complex to be detailed in this paper but relevant parts are explained further below where necessary.

Broadly speaking, image to HTML ratio, menu structure and HTML similarity dominate the upper portion of the tree so the the graph seems to confirm the above findings. The root node is actually whether the website shares an image associated with a fraudulent website. A positive answer results in immediate classification as fraudulent website. From there, both nodes that the root branches to a conditions about the value of the menu structure feature. The tree branches down by alternating most often the menu structure and HTML similarity as conditions with very low values tested first, as they are indicative of a legitimate website. From these conditions an early classification as a legitimate website occurs. Image to HTML ration gains more expression after a couple of levels into the three and is often part of a similar decision process. The presence of an address is rarely queried as expected.

The three financial features - minimum, maximum price and average price were tested separately but in a similar way. This is because the model was trained with less data for these features and, therefore, it is considered less reliable. Nevertheless, the weights assigned by Random Forests are - 6.4%, 7.0%, and 5.2% for minimum, maximum and average prices accordingly. The combined weight of 18.6% suggests that financial data extracted from advertisement websites is a useful indicator in detecting fraudulent websites. The importance of these features is also supported by the fact that the weight allocated to them is correlated in a reduction in the two most important features - HTML similarity and menu structure. This suggests that financial data is more than just supplementary feature in edge cases. In fact, the

aggregate of financial features is the most important predictor for the delivery websites model trained on sites that all include financial data. However, this is not considered a reliable source as the delivery sites used for training were only 50 in this context. The weights attributed by Random Forests trained on the dataset including financial features are provided in Figures 4.5 and 4.6

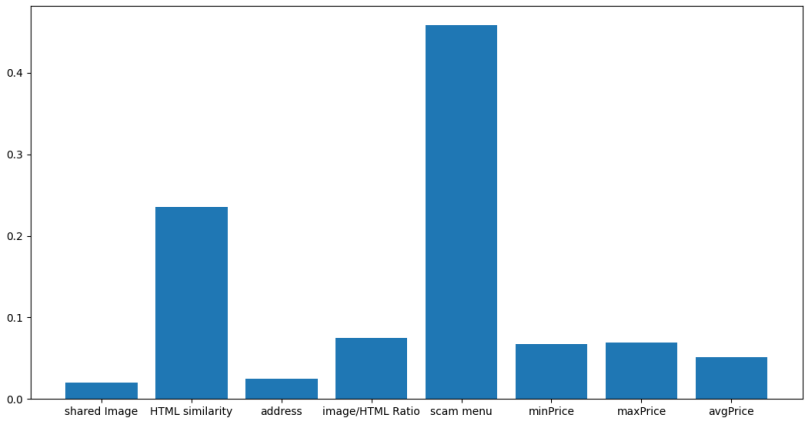


Figure 4.5: Weights allocated for features including financial data

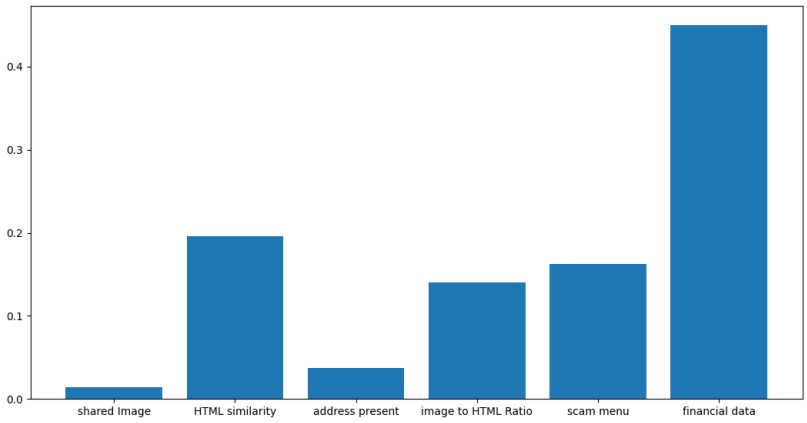


Figure 4.6: Weights allocated for features including financial data - delivery websites

4.2.2 Single feature performance

Additional tests were performed to isolate the single best feature to make predictions as to whether websites are legitimate or fraudulent. The Random Forests model with optimal parameters from above was used for these tests and the features were tested one at a time. It should be noted that financial features were tested on a smaller subset of the data where all rows contained financial information and, therefore, the values provided below are for reference purposes only and should not be compared to the other features. The results are listed in Tables 4.5 and 4.6.

The tables show that if a single feature has to be picked to detect pet scam websites this should be the menu options. It is interesting to note that HTML similarity has relatively low F1, recall and accuracy. This suggests that HTML similarity is not as good in predicting the legitimacy of a website on its own, however, its predictive value increases significantly when combined with other features as demonstrated in the charts previously. Presence of an image associated with a fraudulent website is the feature with

Feature	Precision	Recall	F1	Accuracy	AUC
shared images	1.00	0.13	0.22	0.81	0.56
HTML similarity	0.62	0.59	0.61	0.84	0.74
address present	0.06	0.06	0.00	0.45	0.33
images to HTML ratio	0.51	0.37	0.42	0.79	0.63
scam-style menu	0.87	0.87	0.87	0.94	0.91
financial features	0.87	0.75	0.81	0.90	0.85

Table 4.5: performance metrics for Random Forests trained with single feature

Feature	Precision	Recall	F1	Accuracy	AUC
shared images	0.89	1.00	0.94	0.89	0.50
HTML similarity	0.88	0.86	0.87	0.78	0.48
address present	0.89	1.00	0.94	0.89	0.5
images to HTML ratio	0.89	0.93	0.91	0.84	0.53
scam-style menu	0.89	0.99	0.94	0.88	0.49
financial features	0.80	0.94	0.87	0.76	0.46

Table 4.6: performance metrics for Random Forests trained with single feature on delivery websites

the highest precision - whenever an association is found the website is almost certainly fraudulent. This is because an exact match of the hashes was required for a positive value to be registered for this feature. This indicates that in future research a more liberal approach might be undertaken in the engineering of this feature - if the threshold for similarity is lower, 0.9 for example, the recall of this feature might increase significantly without sacrificing too much precision.

4.3 Causes of misclassification

There are two main causes of misclassification. The first cause is that in some cases legitimate websites have menu options containing very high number of similar words to those used to identify fraudulent websites. An example is provided in Figure 4.6, where 4 out of 5 menu options fall within the range of words that are considered as signature keywords used by scammers.

Figure 4.7: Legitimate website with very high 'menu options' ratio

These misclassifications occur due to the high weight attributed by the model to the menu options feature and are an inevitable trade-off that is conceded to capture the majority of fraudulent websites that share similarity in this aspect. This type of misclassification is more pronounced with delivery websites where examples of legitimate websites are few and most of them contain similar menu options which results in high false positive rate.

The second cause of misclassification is that some legitimate websites have moderate similarity in their menu options with scam sites but they also haven't returned any images in the scraping process. Therefore, this category of errors is a combination of factors. An example is provided in Figure 4.8. The menu options feature yields 0.42 - a high enough value for the decision tree to consider classification as a

pet scam website. Then the decision tree branches out to test image to HTML ration which is 0 because no images were obtained for the website. Since this feature has considerable weight in the algorithm, a positive classification for a fraudulent website is obtained. These type of misclassifications can be addressed in the training stage by ignoring values for features that are not obtained for a particular row.

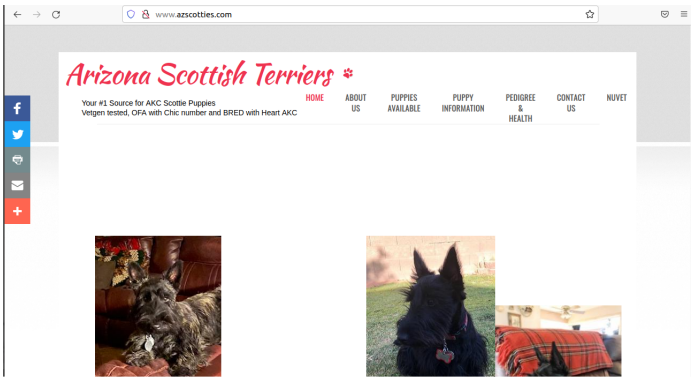


Figure 4.8: Legitimate website with high 'menu options' ratio and low 'images to HTML' ratio

The causes of false negatives are mostly outliers in the data. The decision tree sets multiple thresholds for the menu options scam, HTML similarity and image to HTML ratio. There are instances of pet scam websites that do meet any of the thresholds necessary for a website to be classified as fraudulent. Also, some fraudulent websites have 0.0 ratio for the menu options because the HTML structure for their menus is unconventional. As stated above, the misclassification of these websites is a necessary concession to capture the general pattern in the dataset. An example is provided in Figure 4.9. The website has only two menu options and the minimum amount necessary for outputting a ratio for the menu options feature is 3. This threshold was set to sieve out structures that are not menus but in this instance has led to the default 0.0 being assigned for the menu option feature and a false negative classification.

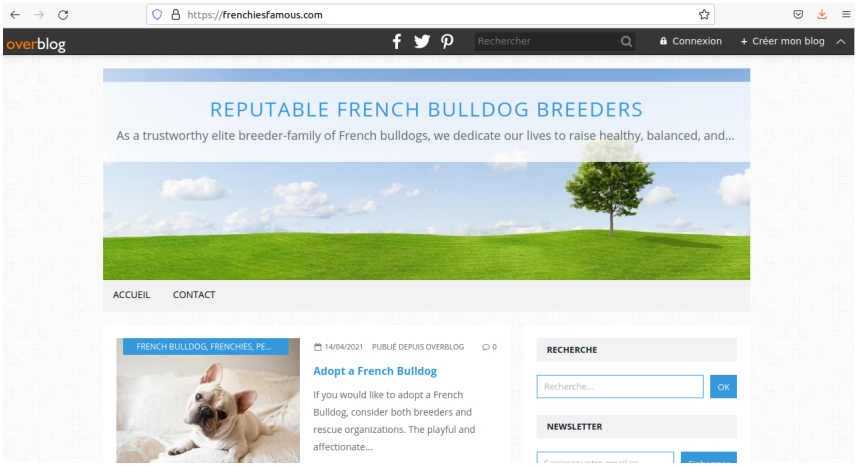


Figure 4.9: Fraudulent website with low 'menu options' ratio

Chapter 5

Discussion

5.1 Image similarity

The research published by Price and Edwards found that whether a website has a common image with a proven fraudulent website is an important feature in connecting two fraudulent websites. In this project, it was found that shared images with a fraudulent website was a useful predictor of the legitimacy the website assessed. This is because this feature had precision of 100%. However, it was also established that this feature alone is not a reliable basis for detecting a large percentage of the actual fraudulent websites. Given that fraudsters advertise many different expensive breeds on their websites, I investigated 100 images from pet scam websites. The reverse searches for images that relate to dogs never returned an identifiable source - this could mean that fraudsters are manipulating images to prevent exactly this kind of detection criteria.

It should be noted however, that the feature tested in this project differs slightly from the one employed by Price and Edwards in that the latter included all images that had similarity index of 0.9 and above with an image from a proven pet scam website as a positive indicator of a connection between the two domains [18]. In this project, only exact matches of the hash were counted as a positive indicator which reduces the number of images being flagged up for an association with a pet scam. A possible limitation for this feature is the fact that a negative value was imputed for all websites for which no images were obtained. A different approach would be to ignore this feature in the training process for websites for which no images were scraped successfully. However, this was not done in the present project as this functionality is not included in the library that it was implemented with - Scikit Learn. Therefore, while this project suggests that common images are insufficient as a standalone data point in predicting that a website is fraudulent, this should not be treated as a conclusive evidence.

Image similarity has been successfully deployed in the context of phishing detection where visual similarity with the target website is the key for defrauding victims [6]. However, pet scams are not dependent on impersonating legitimate websites which means that image similarity is less useful in this context. Whilst a positive finding is highly indicative of association with a fraudulent website, pet scammers use a large amount of images spread across many websites which makes it difficult to detect them solely on this basis. Also, given the established efficiency of image similarity in detecting other kinds of fraud, it is possible that pet scammers are focusing their attempts to avoid detection on manipulating the images used and using a large enough resource base to this end.

5.2 HTML similarity

Price and Edwards found that HTML similarity of a website with pet scam domains has high predictive value in associating the two websites. The present project confirms the utility of this feature. It was found that HTML similarity, in combination with other features, is the second best indicator of whether a sale website is legitimate or fraudulent and the best indicator for delivery websites. The feature exploits the need of fraudsters to produce large amount of websites in an automated fashion to keep the costs of the operation low. The technique of comparing elements taken from the DOM or the HTML structure of a website has been tried successfully in the domain of detecting phishing fraud [24]. In that context,

scammers have adapted their approach and started using visually similar but structurally different pages to the one that they wish to impersonate. For this reason, the focus with phishing detection shifted to finding ways of comparing visual similarity from HTML elements [6]. However, in the context of pet scams, criminals are not trying to impersonate legitimate businesses but produce large quantities of websites which are visually different. Therefore, HTML similarity is still relevant in this context and is likely to retain its importance over time. This is because with pet scam websites, fraudsters need to follow a pattern for creating large amount of websites. Whilst such patterns can become more sophisticated over time to avoid detection the HTML similarity between fraudulent pages is still similar enough to be used for detection purposes. Therefore, HTML similarity is still useful for detecting pet scam websites and also any type of fraudulent website for which fast production in large quantities is required.

5.3 Address found

Given that withholding location information could prove useful to a scammer in extorting additional amounts for delivery, it was expected that such information would be more prevalent with legitimate websites. However, the presence of an address was the feature with lowest performance. Investigation of the reason behind this suggests that this is mainly due to the way the presence of location information was tested. Manual inspection of misclassified legitimate websites showed that some of them had interactive Google Maps section to pinpoint their location. As only a regular expression was used to extract the postcode from the visible text of each website, this approach is unable to record the correct results for these websites. A better evaluation of the usefulness of this feature could be made if it is engineering to cover these instances. This feature is not covered in traditional fraud detection literature because it seeks to address an aspect that is unique to pet scams. While the methodology of pet scams is similar to non-delivery fraud, most of the items that are advertised on fraudulent websites are the type of commodity that is expected to be purchased online rather than in-store. This is not necessarily true in the case of purchasing a pet and a feature that covers a greater fraction of the location information from websites might prove useful in detecting pet scams.

5.4 Images to HTML ratio

The main reason why this feature is useful is that the websites with a very high score are exclusively in the legitimate websites category. A large value for this feature results in an immediate classification as a legitimate website. The feature is also very useful when the ratio is moderately high, then the combination with other features has still very strong predictive value. The the prevalence of high number of images per website among legitimate websites seems to suggest that whilst fraudsters have developed means of processing images to obfuscate their traceability they do not have the capacity to produce the same number of images as some of the legitimate breeders who are keen to showcase all of their pets. Another possible explanation is that legitimate websites would more often have images unrelated to pets - the variance of the images could be higher in this group which is hard to be replicated by a criminal who seeks to adopt a pattern for creating websites. Measuring the ratio of images to HTML is also not covered in fraud literature. The usefulness of this feature in the present context suggests that it might be useful to explore its application in the context of non-delivery fraud.

5.5 Menu options

Identifying a menu structure with a high proportion of common keywords is the single best predictor of classifying pet sale websites into fraudulent and legitimate. The reason why this feature performs well is that fraudsters have interest in presenting their websites as legitimate. This project established that the menu options used in pet scam websites are the same ones that are most popular among legitimate pet sellers' websites too. However, as with HTML similarity, fraudsters have to keep the production of their content under a pattern to make the process automatic which reduces the variability of the content. The feature is designed as a ratio to reflect exactly this variability and its high performance is attributable to a large part to the fact that legitimate websites often have at least a few menu options that are highly dissimilar to other websites. This results in a useful distinction between legitimate and fraudulent websites that can be used for automatic detection. The implementation of this feature is unique to this project but is essentially a combination of two well established methods of fraud detection - structural HTML similarity and identification of keywords with higher frequencies in fraudulent websites. The first part

is subordinate to the second - a website having an identifiable menu structure is by no means indicative of a fraud, however, 885 of legitimate websites did not have a conventional menu structure and returned a default 0 value for this feature. By comparison, all but 6 websites from the fake advertisement group returned an identifiable structure. Therefore, whilst the the menu structure is not indicative of fraud in itself, this part of the feature still captures the fact that fraudulent websites have lower variability in the HTML they use to create menus. The second part of this feature is similar to the keyword identification approach that was used by Norazman and Zamin where the terms most often used by fraudsters in the context of pet scams were searched [17]. The feature in this case is different in that it measures the variability of the options as a ratio rather than the frequency of the keywords - having many of the keywords being included would not lead to a high ratio if there are twice as many options that are not part of the prohibited range of words. This feature might be explored in detecting other types of fraud where there is a similar incentive to make the outlook of a website more legitimate.

5.6 Financial data

Financial features were found to be valuable for separating fraudulent websites from legitimate ones. Each of the data points : minimum, maximum and average price was given higher weight by the classifier than shared images and the presence of post code. A more correct reading of the results might be to measure the aggregate weight of the three data points since they are intended to capture a single fact about the websites that are being classified - whether fraudulent websites have different pricing pattern than legitimate ones. If this approach is taken, financial data is the third most important feature for the advertising sites classifier and the most important one for classifying delivery websites. Currently, this is the feature that requires further assessment the most because the classifier that included this feature was trained only on a subset of the whole dataset. Therefore, the results obtained in this project provide a good reason to explore this feature further but they are not a conclusive evidence as to the utility of this feature.

The minimum, maximum and average prices obtained for advertisement and delivery websites are detailed in Tables 5.1 and 5.2 accordingly. It should be noted that the maximum prices appear excessive for pet sale and delivery websites and this most likely reflects an issue with the way prices are extracted from websites. This is an issue that would benefit from further investigation. Apart from this, the data confirms the initial expectations - for advertisement websites the average figures are lower in the fraudulent group. The most likely explanation for this is that fraudsters use discounted prices to receive more interest from victims. In the case of delivery websites, the average is actually higher. A possible explanation is that delivery websites are used to extort secondary payments and they don't need to contain attractive prices because victims do not get to chose them in the first place. In fact, higher prices on these websites could be intentionally placed by fraudsters to justify requests for larger sums from the victims.

	Minimum Price	Maximum Price	Average Price
Legitimate	0.3	175000.0	956.18
Fraudulent	1.3	25000.0	707.18

Table 5.1: Financial data for advertisement websites

	Minimum Price	Maximum Price	Average Price
Legitimate	1.0	5000	271.65
Fraudulent	0.5	20000	477.91

Table 5.2: Financial data for delivery websites

5.7 Practical application of the model and its limitations

This project shows that it is possible to develop a classifier which separates legitimate and fraudulent pet sale websites with a high degree of accuracy on the basis of the websites' content. The F1 and AUC

scores for the classifier are both 94% for advertising websites which suggests that the model derived can be practically useful for detecting fraudulent pet sale websites. Thus, the project shows a possible way of moving towards automation of fraud detection in this context.

However, it is considered that this tool would be most useful if deployed in practice under human supervision. Whilst the false positive rate of the model for advertisement websites is very low, the gravity of erroneously blacklisting or taking down a legitimate website is too high. A cost-benefit analysis might be needed to confirm this but the gravity of the harm being caused to an innocent and legitimate dog breeder seems too high to concede. Therefore, in its current form this model would be most useful in sieving out legitimate websites to allow resources to be allocated towards examining websites that the model has flagged up as fraudulent.

There are also other reasons to be careful in deploying this model as a purely automatic tool. First, the image and HTML similarity features have been trained with a reference group that was selected to include the largest clusters of images and HTML structure. In hindsight, this leads to overfitting and reduces the reliability of the model's capability of generalizing to new data. This is because knowledge of the testing set has been allowed to influence the training of the algorithm which is the opposite of what is sought by separating the training and testing sets. Essentially this flaw in the methodology is likely to be elevating the recall values for image and HTML similarity, which in turn affects the overall accuracy of the algorithm. However, it is not considered that this error deems the whole model unreliable. The graph plotting the weights allocated to each feature shows that the combined weight of menu options and image to HTML ratio still outweigh significantly the impact of HTML and image similarity. Also, the success of HTML similarity in predicting legitimacy in this project and in the paper published by Price and Edwards suggests that correct execution in the feature engineering process would have still yielded a useful feature.

Another limitation of the model, and arguably a more important one, is that the dataset used to train it is not very large. The findings in this project are considered a step further towards automatisation of pet scam detection but having more data to train the same algorithm would greatly improve the accuracy of the the end results. This is critically important in the context of classifying delivery websites - a task for which the model currently has low performance. The F1 is high for this task but the false positive rate is too high . As detailed above, false positives could have severe consequences if the model is deployed in completely automated fashion and its performance for delivery websites cannot justify such deployment. The good news is that the main areas for improvement in this context is collecting more examples of legitimate websites as there were only 35 used in this project. Collecting a list of fraudulent websites is generally harder than doing so for legitimate businesses and an initiative to create websites like IPATA, for example on national level, would provide useful data for training purposes.

Additional data would be useful also for examining further the importance of financial features. Their aggregate weight was ranked third for advertisement websites and for delivery websites it was deemed the most reliable predictor of legitimacy. The caution in placing high value on this feature in the context of delivery websites stems from the fact that the dataset used to derive this statistic comprised only 80 websites. With more data to investigate, this feature might prove to be the key in detecting fraudulent delivery websites.

5.8 Further work

Some of the features used in this project would greatly benefit from further improvements in the future. The perceptual hash of images can be used to explore a wider range of matches and test the change in recall and precision of the resulting algorithm. Furthermore, improvements could be made to the scraper as the one created for this project was unable to extract all images for some websites which reduces the data available for this feature. Moreover, the scikit learn library doesnt allow feature values to be ignored for instances where data is missing so exploring another platform for training the algorithm that allows this might result in better trained model. The feature that captures addresses in websites would benefit from including Google Maps section in websites, possibly by searching for a particular type of HTML structure. It is considered that this would improve the score of the feature. Moreover, textual similarity was not explored in this project as it would require more computational resource than the features developed and was the least reliable predictor of associating fraudulent websites in previous

research. However, testing different techniques to engineer features that capture this kind of data and adding the resulting data point to the existing features can make the model more accurate.

Chapter 6

Conclusion

In this project, I created a scraper that can extract HTML and images from 5808 websites. The scraper is able to extract HTML from websites that serve static HTML from their server as well as HTML from websites that serve asynchronously loaded content. The gathered information was used to create six features for machine learning purposes. The first one, image similarity, represents with a binary value whether the website examined has an image in common with another website that is known to be fraudulent. The comparison is done through a perceptual hash that attributes similar hash values for visually similar images. The second feature that was created is HTML similarity. It measures the Jaccard similarity index between a website and the reference group of fraudulent websites used in this project. The next feature that was designed from the scraped data is a binary value which represents whether a website contains a valid UK or US postcode. The ratio between the images retrieved by the scraper for a website and the number of HTML pages obtained is the fourth feature. The fifth and most important feature in the context of this project is a value that measures the ratio between menu options that match an entry from a pool of keywords and the total number of menu options. The final feature explored in this project is comprised of three data points: minimum, maximum and average price. Together, these three data points capture the financial data in a website.

I used these features to train machine learning models and compare their performance to select the one with highest F1 score. In order to find a model that would generalize well to new data, I used 5-fold cross validation in order to use the entirety of the dataset for both training and testing the models. The parameters for each model were tuned to compare the optimal performance of each model. It was found that a model trained with the Random Forests algorithm provides optimal performance for classification between legitimate and fraudulent pet sale websites. The F1 and overall predictive accuracy of this model were 0.94 and 0.97 respectively. With respect to delivery websites, Logistic Regression and Random Forests both achieved an F1 score of 0.94 but the former had slightly higher recall rate - 1.0 versus 0.99. The overall accuracy of the best classifier for delivery websites is 0.89 which is slightly lower than the results achieved for advertising websites.

The most important difference between the advertisement and delivery websites classifiers is that the latter has significantly higher false positive rate. This is apparent from a comparison of the AUC scores for advertisement and delivery website classifiers - 0.94 against 0.50. The reason for this is the class imbalance in the dataset for delivery websites - 35 legitimate websites versus 332 fraudulent websites. By comparison, 4141 legitimate and 1300 fraudulent advertisement websites are available for the pet sale websites dataset. Also, the fact that there are only 35 legitimate websites in the delivery dataset is a contributing cause for the high false positive rate in this group - the model has insufficient data to capture patterns that represent what defines a legitimate website in order to separate it from fraudulent sites. This conclusion highlights the need to gather more information about legitimate delivery websites.

Nevertheless, the classifiers developed, especially the one for advertisement websites, provide high predictive accuracy. These results are significant for two reasons. First, they show that detection of pet scam websites can be at least partially automated. The only reason why a full automation of this task is not recommendable at this stage is the gravity of the consequences stemming from taking automatic action against the small number of legitimate websites that are falsely flagged as pet scam websites. However, the classifier can be a powerful tool that can narrow down the target list of potential pet scam

websites and save significant amount of time for those who wish to detect them.

Second, the classifiers provide valuable information about the usefulness of the underlying features in predicting whether a pet sale or delivery website is fraudulent or not. This project established that the menu options in a website constitute a feature that provides high predictive accuracy even on its own. The other features that proved highly valuable were HTML similarity and the ratio between images and HTML files in a website. The combination of these features with the menu options in a website led to classifiers with high F1 and overall accuracy. Whilst image similarity had very low recall rate, the precision of this feature is 1.0 which means that even if it doesn't have high impact on the number of pet scams detected, it is valuable for a classifier because it provides the capability to detect a fraudulent site even if all other features fail to do so. The feature that recorded whether a website had an address or not had poor performance but the project found that this might be due to the way it was implement in this instance, which provides a potential improvement that can lead to another feature with high predictive value. The financial data found in a website, comprising minimum, maximum and average price, was found to also provide promising results. However, the results for this feature are treated as unreliable as they were trained on a smaller dataset which excluded websites that didn't have financial data detected in their content. The conclusions drawn for each of these features can be utilised in a future research and further the goal of automating pet scam detection.

Bibliography

[1] Afe Adogame. The 419 code as business unusual: Youth and the unfolding of the advance fee fraud online discourse. *Asian Journal of Social Science*, 37(4):551–573, 2009.

[2] Vinicius Almendra. Finding the needle: A risk-based ranking of product listings at online auction sites for non-delivery fraud prediction. *Expert systems with applications*, 40(12):4805–4811, 2013.

[3] Erica Briscoe and Jacob Feldman. Conceptual complexity and the bias/variance tradeoff. *Cognition*, 118(1):2–16, 2011.

[4] Better Business Bureau. Puppy scams: How fake online pet sellers steal from unsuspecting pet buyers. *A BBB Study. Available online: https://www. bbb. org/en/us/article/investigations/14214-puppy-scams-howfake-online-pet-sellers-steal-from-unsuspecting-pet-buyers-a-bbb-study (accessed on 11 December 2017)*, 2017.

[5] National Fraud Intelligence Bureau. Fraud and cyber crime national statistics. https://www.actionfraud.police.uk/data. Accessed: 2021-09-11.

[6] Teh-Chung Chen, Scott Dick, and James Miller. Detecting visually similar web pages: Application to phishing detection. *ACM Transactions on Internet Technology (TOIT)*, 10(2):1–38, 2010.

[7] Kang Leng Chiew, Ee Hung Chang, Wei King Tiong, et al. Utilisation of website logo for phishing detection. *Computers & Security*, 54:16–26, 2015.

[8] Kim-Kwang Raymond Choo. A conceptual interdisciplinary plug-and-play cyber security framework. In *ICTs and the Millennium Development Goals*, pages 81–99. Springer, 2014.

[9] FBI. Internet Crime Report 2020. https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf. Accessed: 2021-09-11.

[10] Hongxing He, Warwick Graco, and Xin Yao. Application of genetic algorithm and k-nearest neighbour method in medical fraud detection. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 74–81. Springer, 1998.

[11] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[12] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pages 372–378. IEEE, 2014.

[13] Sai Kiran, Jyoti Guru, Rishabh Kumar, Naveen Kumar, Deepak Katariya, and Maheshwar Sharma. Credit card fraud detection using naïve bayes model based and knn classifier. *International Journal of Advance Research, Ideas and Innovations in Technoloy*, 4(3), 2018.

[14] Michael Levi, Alan Doig, Rajeev Gundur, David Wall, and Matthew Leighton Williams. The implications of economic cybercrime for policing. 2016.

[15] David Maimon, Mateus Santos, and Youngsam Park. Online deception and situations conducive to the progression of non-payment fraud. *Journal of Crime and Justice*, 42(5):516–535, 2019.

[16] Rafael Maranzato, Adriano Pereira, Alair Pereira do Lago, and Marden Neubert. Fraud detection in reputation systems in e-markets using logistic regression. In *Proceedings of the 2010 ACM symposium on applied computing*, pages 1454–1455, 2010.

[17] Nor Sa'datul Aqma NORAZMAN and Norshuhani ZAMIN. Development of scammed posts detector: A case study of pet scammed posting. In *Proceedings of the 18th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI). IIIS*, 2014.

[18] Benjamin Price and Matthew Edwards. Resource networks of pet scam websites. In *2020 Symposium on Electronic Crime Research*. Institute of Electrical and Electronics Engineers (IEEE), 2020.

[19] Adrian Rosebrock. Image hashing with OpenCV and python. *https://www.pyimagesearch.com/2017/11/27/image-hashing-opencv-python/*. Accessed: 2021-09-11.

[20] Stuart Ross and Russell G Smith. Risk factors for advance fee fraud victimisation. *Trends and Issues in Crime and Criminal Justice [electronic resource]*, (420):1–6, 2011.

[21] Anand V Saurkar, Kedar G Pathare, and Shweta A Gode. An overview on web scraping techniques and tools. *International Journal on Future Revolution in Computer Science & Communication Engineering*, 4(4):363–367, 2018.

[22] Sharmila Subudhi and Suvasini Panigrahi. Quarter-sphere support vector machine for fraud detection in mobile telecommunication networks. *Procedia Computer Science*, 48:353–359, 2015.

[23] The Sun. Pet scams increase over 400 per cent in lockdown. *www.thesun.co.uk/news/11801749/pet-scams-increase-400-per-cent-lockdown/*. Accessed: 2021-09-11.

[24] Ishant Tyagi, Jatin Shad, Shubham Sharma, Siddharth Gaur, and Gagandeep Kaur. A novel machine learning approach to detect phishing websites. In *2018 5th International conference on signal processing and integrated networks (SPIN)*, pages 425–430. IEEE, 2018.

[25] Erdinç Uzun. A novel web scraping approach using the additional information obtained from web pages. *IEEE Access*, 8:61726–61740, 2020.

[26] ERDİNÇ Uzun, Tarik Yerlikaya, and OĞUZ KIRAT. Comparison of python libraries used for web data extraction. *FUNDAMENTAL SCIENCES AND APPLICATIONS*, page 87, 2018.

[27] Julianne Webster and Jacqueline M Drew. Policing advance fee fraud (aff) experiences of fraud detectives using a victim-focused approach. *International Journal of Police Science & Management*, 19(1):39–53, 2017.

[28] Jack M Whittaker and Mark Button. Understanding pet scams: A case study of advance fee and non-delivery fraud using victims' accounts. *Australian & New Zealand Journal of Criminology*, 53(4):497–514, 2020.

[29] Monica T Whitty and Tom Buchanan. The online romance scam: A serious cybercrime. *CyberPsychology, Behavior, and Social Networking*, 15(3):181–183, 2012.

[30] Tony Yiu. Understanding random forest - how the algorithm works and why it is so effective. *https://towardsdatascience.com/understanding-random-forest-58381e0602d2*. Accessed: 2021-09-11.

[31] Bo Zhao. Web scraping. *Encyclopedia of big data*, pages 1–3, 2017.