TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Informatics

IDN40LT

Liisi Soots 134695IABB

FINDING FRAUDULENT USERS THROUGH IP ADDRESS CHANGES USING MACHINE LEARNING

Bachelor's thesis

Supervisors: Innar Liiv, PhD
Associate Processor

Lauri Koobas, MSc
Fraud R&D Specialist

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatika instituut

IDN40LT

Liisi Soots 134695IABB

PAHATAHTLIKE KASUTAJATE TUVASTAMINE MASINÕPPEGA, ARVESTADES IP AADRESSIDE VAHETUSI

Bakalaureusetöö

Juhendajad: Innar Liiv, PhD

Dotsent

Lauri Koobas, MSc

Pettuste teadus-ja arendustegevuse spetsialist

Tallinn 2016

Author’s declaration of originality

Herewith I declare that this thesis “Finding fraudulent users through IP address changes using machine learning” is based on my own work. All of the materials and ideas from other authors are referred. This thesis has not been presented for degree to any other university.

Author: Liisi Soots

13.05.2016

Abstract

The author has taken this topic for investigating because the more people use computers the more fraud will be done through them. There are many VoIP companies that deliver voice and multimedia communication with IP protocols. The opportunity attracts regular users but also users who want to take advantage of it. The aim of this work is to find the best machine learning algorithm for detecting fraudulent users. In this paper, the number of countries the user has called is used among other parameters. The hypothesis is that the user who calls from more countries in a short period of time is more fraudulent.

Three machine learning algorithms: clustering, decision tree and random forest will be used and compared to find out the best detection method. Therefore, the main task of the author is to find out the fraudulent users, to improve and validate the accuracy rate of the algorithms. The secondary task is to understand if the hypothesis that the number of countries indicator is suitable for fraudulent users' detection.

Research done in this paper showed that the countries' number is not a good indicator and it needs support from other features. The best results for detecting fraudulent users were with decision tree and random forest algorithms.

This thesis is written in English and is 39 pages long, including 4 chapters, 22 figures and 10 tables.

Annotatsioon

Pahatahtlike kasutajate tuvastamine masinõppega, arvestades IP aadresside vahetusi

Töö eesmärk on analüüsida IP-telefoni teenust pakkuva ettevõtte näitel kasutajate automaatse blokeerimise süsteemile lisaparameeterit - riikide arv, kust kasutaja helistab. Autor on valinud uurimiseks selle teema, sest aina enam inimesi kasutab arvuteid ning koos sellega suureneb ka pahatahtlike kasutajate hulk. Kasutusel olev süsteem võimaldab teha internetitelefoni kõnesid tava-ja mobiiltelefonidele. Selline võimalus tõmbab ligi nii tavakasutajaid kui ka neid, kes soovivad sealt läbi pahatahtlike tegevuste abil kasu saada - pesta musta raha ning teenida võõraste inimeste krediitkaartidega raha.

Selle töö käigus leitakse parimad masinõppe algoritmid, et tuvastada pahatahtlike kasutajaid. Riikide arv, millest kasutaja on helistanud, on võetud üheks tunnusjooneks pahatahtlike kasutajate tuvastamisel. Hüpotees on, et kasutaja, kes helistab rohkematest riikidest 15 minuti jooksul, on tõenäoliselt pahatahtlikum. IP aadresside muutusi ühe näitajana tuvastussüsteemis on teised ettevõtted kasutanud ka varem.

Autor kasutas ja võrdles kolme masinõppe algoritmi, et valida välja parim pahatahtlike kasutajate tuvastuse meetod. Võrreldud algoritmid olid klasterdamine, otsustamis puu ja juhuslik mets (ingl *random forest*). Töö põhiülesanne on leida pahatahtlikke kasutajaid ning parandada ja valiteerida algoritmide täpsust. Teiseks ülesandeks on hüpoteesi, et pahatahtlikud kasutajad helistavad võrreldes tavakasutajatega rohkem erinevatest riikidest, tõestamine ja analüüsimine. Valideeritakse, kas selline tunnusjoon on sobilik pahatahtlike kasutajate leidmiseks.

Uuringud näitasid, et erinevatest riikidest tehtud kõnede arv ei ole hea näitaja kasutajate leidmiseks ning vajab toetust muudelt parameetritelt. Parimad tulemused pahatahtlike kasutajate leidmiseks tulid kasutades otsustus puud ja juhuslikku metsa (ingl *random forest*). Neid saaks kasutada osana praegusest süsteemist, kui toetada neid teiste parameetritega.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 39 leheküljel, 4 peatükki, 22 joonist, 10 tabelit.

List of abbreviations and terms

Blocked user	A user that is labelled as fraudulent and already blocked from the VoIP company side. This label is used to measure the accuracy of the algorithms.
Chargeback	A demand from credit card provider for getting money back from the service provider because of fraud or disputed transaction.
Cluster centroid	Cluster centroid is the centre point of the cluster. The average distance from the centroid is the average distances between data points and the centroid.
Proxy server	A server that is between the client, who makes the request, and the server that answers the request.
PSTN	Public switched telephone network
Supervised learning	A machine learning task where the learnings are done with previously labelled data.
Unsupervised learning	A machine learning task where the learnings are done with unlabelled data and we try to label the data.
VPN	Virtual Private Network. Private and secure network that might have an endpoint that causes the algorithm to mislead.
VoIP	Technologies to deliver voice communication and multimedia sessions over Internet Protocol networks.

Table of contents

List of figures

List of tables

1 Introduction

Every year more people get access to the Internet and IT solutions. Together with the regular users there are the ones who try to get an unfair advantage of the provided services by using fraudulent methods. Companies are fighting against bad users because they get fined for letting the users make fraudulent payments. For example, users making purchases with stolen credit cards will lead to chargebacks. To prevent fraud, companies are building systems to detect these users before they can do more harm.

In this paper, the machine learning methods for finding fraudulent users are analysed based on real data provided by a VoIP company that has decided to stay anonymous.

1.1 Background

IT development has made people's lives better but also it has gained the interest of fraudsters. In the "Report of the Nations on occupation Fraud and Abuse: 2014 Global Fraud Study" it was estimated that the typical organization loses 5% of revenues each year to fraud [1].

Example Company whose data is used in this work uses a variety of features for fraud detection. These include profile age, number of calls, country, platform and relationship with other users. If the features indicate that the user in question is fraudulent then it will be automatically blocked. In this thesis the author will take a new feature – number of countries user called from in short period of time – and analyse if this can be used to detect fraudulent users. If the author can show that the number of countries the service has been used from can be used to detect suspicious users, then it will provide a new feature to be added to the automated user block system for the Company. Call location is used for getting the IP address of the end user and looking at how many times users change the IP location during a 15-minute period.

This bachelor thesis is done in Tallinn University of Technology in Information Technology Faculty. The analysis is done and evaluated in the context of one of the biggest VoIP Companies in the world during 2015-2016.

1.2 Task

In this paper the author aims to prove the hypothesis that fraudulent users change their IP address more often than regular users. In addition different methods are tried for fraud detection and measured for accuracy with different data.

Different algorithms are investigated to find how to detect fraudulent and regular users most efficiently, with the smallest amount of misclassification. In order to get the best results, author looks at the IP address changes of the users, but also adds other features for better evaluation. The aim is to try and find an algorithm that has the best accuracy for detecting fraudulent users.

The main questions answered in the thesis are:

1. Is the hypothesis that users who change their IP address more often are more fraudulent correct?

2. Which algorithm works best for detecting fraudulent users when using IP address changes as a feature?

1.3 Approach

In this paper, machine learning methods are used for detecting fraudulent users. The author will investigate three different learning algorithms to find out the best algorithm: clustering, decision tree and random forest.

While looking individually at features, it might not be possible to find the fraudulent users, but putting multiple features together might enable us to find a better pattern. This is why multiple features are used and analysed – to find the combination of features that could best detect fraudulent users. Before beginning with the machine learning part of the paper, the available data is described. The call history data is taken from the PSTN team that is responsible for the call quality and infrastructure, the number of contacts and user status from the fraud team. Due to the huge number of calls and data available, all the call history data is taken from 3rd August 2015 between 14.00-14.15 GMT, other parameters were gathered separately.

The gathered data is analysed using machine learning algorithms and the accuracy of the algorithms are measured afterwards on data from 16th of December 2015.

1.4 Overview

The work is divided into 4 parts:

1. Related work and methods
2. Collecting the training data
3. Testing 3 algorithms
4. Validating the results.

In the related works section, the prior works by others is described and analysed. In the method section the tools and theoretical part of the algorithms will be described. All the experiments of the author are done in the experiments section. Firstly, it is described how the data was obtained and what features it has. Secondly, the author analyses IP addresses with clustering and users with clustering, decision tree and random forest. In the results paragraph the accuracy of three algorithms will be measured and tested with new data. The algorithms' performance is measured, analysed and compared to real verified fraud data. In the conclusion the author sums up the experiments, results and suggests improvements for further work.

2 Related Work

Fraud is defined as an illegal way of getting money by deceiving another person or organization. It is common in many fields like banking, insurance and telecommunication [2] [3] [4]. Example VoIP company application which is mostly known for the free audio and video calls, also gives an opportunity to call to and from a telephone number for a fee. The Company is not a telecommunication provider itself but fraud in this context is similar to fraud in telecommunication.

Each company has different data about their users and even thought a company might solve the fraud problem at one point, the fraud schemes evolve over time. There is a wide variety of schemes [5] among which the most relevant in this thesis are credit card fraud and other online payment fraud, as well as account abuse such as spam instant messages [6]. With this work the author's goal is to find fraudsters of abuse and credit card fraud.

The fraud in the banking and mobile industry has been investigated with machine learning since the early 90s. This field has developed a lot as the problem of fraud may cause the industry financial and capacity loss. Already in the 90s the telecommunications industry lost hundreds of millions dollars per year [4]. In the "Combining Data Mining and Machine learning for Effective User Profiling" the authors use the velocity checks which look at the geographical dispersion of users [4]. These checks are not currently used in the VoIP Company's systems and the author is investigating this area.

Nowadays, enterprises and public institutions have to face a growing presence of fraud initiatives. It is expensive to manually review suspicious users so the enterprises and public institutions need automatic systems to implement fraud detection [2]. However, a single transaction information is typically not sufficient to detect a fraud occurrence [7]. In the case of this work the author will use multiple parameters and call data to detect fraud. The auto-blocking system cannot make a decision whether the user is a fraudster

with information regarding only one call but has to look at user behaviour in general. Looking at multiple parameters together makes the system more reliable.

2.1 Machine learning techniques for fraud detection

Machine learning is a field of information technology to create computer algorithms for learning different things. For example, detecting what leads people to rate a service a specific way.

In the fraud detection field both supervised and unsupervised techniques are used. Supervised methods assume that labels of past transactions are available and reliable but are often limited to recognize fraud patterns that have already occurred [7]. The unsupervised learning is capable of detecting new fraudulent behaviours [8]. The focus on this paper will be on the supervised learning methods with users, who are already labelled as fraudulent or regular. Fraud detection is an unbalanced problem where there are always more regular users than fraudulent users. Some of the algorithms give better result if the data is balanced [9], but others work with even very unbalanced data like random forests.

As the enterprises and public institutions make their systems more intelligent, the fraudsters also get smarter. The fraud is dynamic and the behaviours change. But right now in this research scope the author is not looking into interactive machine learning methods which would automatically make the learning algorithms better [3].

The location changes have been taken into account before by other researchers, but it is not used in the current system. The author is investigating if the location changes of the caller as a feature would be a good addition to the current fraud detection system.

3 Methods

In this paper three machine learning methods are used for finding fraudulent users. Two of them are supervised (Decision Tree and Random Forest) and one is unsupervised (Clustering). The automatic blocking system of the VoIP Company is built up of parameters weighted with different importance. The aim of this research is to add a new parameter taking into account the changes in caller location.

The author will be using Python version 2.7 with scikit-learn library for the machine learning algorithms, which is an open source machine learning library for Python [10]. The most popular programming languages used for machine learning are R and Python. The author chose to use Python because of previous experience with other projects and familiar syntax. Python is a dynamical programming language which is open-source. It is a programming language widely used in many fields such as website development and data processing. Python programs can be expanded using the Python software packages [11].

3.1 Clustering

Clustering is an algorithm in unsupervised learning to group similar observations together. For example, grouping customers to different groups and making specialized marketing campaigns for them. There are multiple clustering methods, of which K-means clustering is used here. The K-means clustering method will try to partition a set of N samples X into k clusters C, each described by μ_i, so that each point in the cluster is more similar to points within its own cluster than with points in some other cluster. It separates data points to groups of equal variance while minimizing within-cluster sum-of-squares (1) [10]. In the formula (1) the n stands for number of samples, k number of clusters, μ_i for centroid for cluster j, $x_i^{(j)}$ sample number i.

$$\sum_{j=1}^{k}\sum_{i=1}^{n}\left|x_i^{(j)} - \mu_j\right|^2 \qquad (1)$$

3.2 Decision Tree

Decision trees are very intuitive predictors. Typically, if a human programmer creates a predictor it will look like a decision tree [12].

Decision trees are non-parametric supervised learning methods used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The decision tree consists of if else decision rules and the learned function is represented by decision tree that has leaves describing similar data points. [12]

Decision tree classifies instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. In general, decision tree represents a disjunction of constrains on the attribute values of instances.

The decision tree basic algorithm ID3 [13] learns decision trees by constructing them top-down. The root feature is found by evaluating a statistic to determine how well it alone classifies the training examples. The best attribute is selected and used as the test at the root. Then all the data is sorted at the root. Then the entire process is repeated using the training data. This is a greedy search for an acceptable decision tree, in which the algorithm never goes back to reconsider earlier choices. [14]

3.3 Random Forest

Random forests are a combination of decision trees such that each tree depends on the values of a random vector sampled and has its own importance on the final algorithm [15]. Random forests are built by combining the predictions of several trees, each of which is trained in isolation. The random trees are trained on subsets of the data. The subsets are drawn at random from the full training set. The tree building is done the same way as in decision tree algorithm.

In each tree the data set is partitioned randomly into two parts, each of which plays a different role in the tree construction. The author refers to points assigned to the different parts as structure and estimation points respectively. Structure points are allowed to influence the shape of the tree leaves. They are used to determine split dimensions and split points in each internal node of the tree. However, structure points

are not permitted to effect the predictions made in the tree leaves. Estimation points play the dual role. These points are used to fit the estimators in each leaf of the tree, but have no effect on the shape of the tree partition. [16]

4 Experiments

The aim for the experiments is to understand which algorithm is the most accurate and suitable for finding fraudulent users. The December's 2015 data is used to measure the accuracy on the algorithms that were built upon data from August 2015. It is described how the data was obtained, how it was analysed with three methods and how the accuracy was measured.

4.1 Data

The data for this thesis is based on one of the biggest VoIP Company's calling records. The calls data is taken from the PSTN team, the number of contacts count and user status from the Fraud team. Due the huge number of calls and complex data availability, all the calls data is taken on 3rd of August 2015 between 14.00-14.15 GMT. The data was taken out from the database with SQL queries.

The data for the accuracy testing is extracted from the 16th of December 2015 between 14.00-14.15 GMT. All the data was erased after the experiments due the privacy reasons.

4.1.1 Users Data

The users call data is full of the company's own test accounts data that are actually not real users and disturb the analysis. The data was cleaned of the technical users and joined between different resources. During the experiments the data was held in .csv files and read by Python from there. After the cleaning the data contained information on 19915 users, 119 (0.6%) of them were blocked. In total there were 12 features for every user that was taken into account (Table 1).

The author did not obtain all of the features of the users at the first moment. The data of blocked users were later updated in April 2016, 8 months after the calls were made. During the 8 month period from when the calls were made a lot of users were blocked. The blocked label means that the user has been already labelled as fraudulent by the

current system. This label is used for measuring the accuracy of new algorithms and as label in supervised learning. In April there were already 627 blocked users (3%). The labels added later were the labels used in the algorithms. While the training data had 8 months between taking data out and blocking them, the test data had only 5 month time.

Table 1. Features of the users

Feature	Explanation
Username	Used for joining the data between different sources
Managed	If the user is managed, meaning it bills are paid by a company, then True else False
Fraudulent	If the user is blocked, then True else False
Calls	Count of calls made by user in 15 minutes
Calls Ending with 404	Count of calls ending with call reason 404. This means the user called to a number that does not exist
Countries	Count of countries user make calls from in 15 minutes
Devices	Number of devices user made calls from in 15 minutes
Call duration	Average call duration during the period
Maximum call duration	Maximum call duration during the period
Average interval	Average time between the calls during the period
Maximum interval	Maximum time between the calls during the period
Minimum interval	Minimum time between calls during the period
Number of contacts	Number of contacts user has called in the client period

4.1.2 IP address Data

While doing analysis with the IP addresses, the author has to take into account that a lot of corporations are using VPNs. Also it has gotten more popular among regular people to use VPNs or proxies for privacy sensitivity. These IP addresses might confuse the analysis and therefore excluded from further analysis. The IP addresses have 3 features: calls, users, devices (Table 2).

Table 2. Features of IP addresses

Feature	Explanation
IP address	IP address in IPv4 form to analyse them individually
Calls	Count of calls made from this IP
Users	Count of users who made calls from this IP address
Devices	Count of different devices used

4.2 Analysis

The analysis includes looking into IP addresses and users' patterns. For the IP addresses only clustering will be used, and for the users' analysis clustering, decision tree and random forest methods. The author will compare different algorithms and measure their accuracy with test data from December 2015.

Clustering is used for the IP address analysis because the algorithm will put the most similar objects together into one group. Through that it is possible to analyse IP addresses behaviour as groups [17].

All the algorithms do not work for every use case. To demonstrate it I will use the clustering algorithm for the users. The data is exactly the same that was for IP address analysis but the search value is different. Random forest is usually used as the method for finding fraudulent users in current system. Random forest consists of multiple decision trees and before testing random forest the author will also try how well decision tree method works independently on analysing the data.

4.2.1 IP address

The IP address endpoints for calls made from companies are found and calls made from these IPs are not taken into account. Often companies have many users using the same IP and make more calls on average so this can mislead the algorithms.

K-means clustering (Appendix 1 – K-means clustering with 2D figure) shows, how all the IPs are spread along number of users and calls axes (Figure 1). There are three groups of IP addresses, the red and blue dots are abnormal due quality testing and forwarded calls. Because they are not the actual users I will eliminate these from further address.

The green cluster data shows how different regular IP addresses are. I will cluster this data and analyse how IP addresses are different from each other (Figure 2).

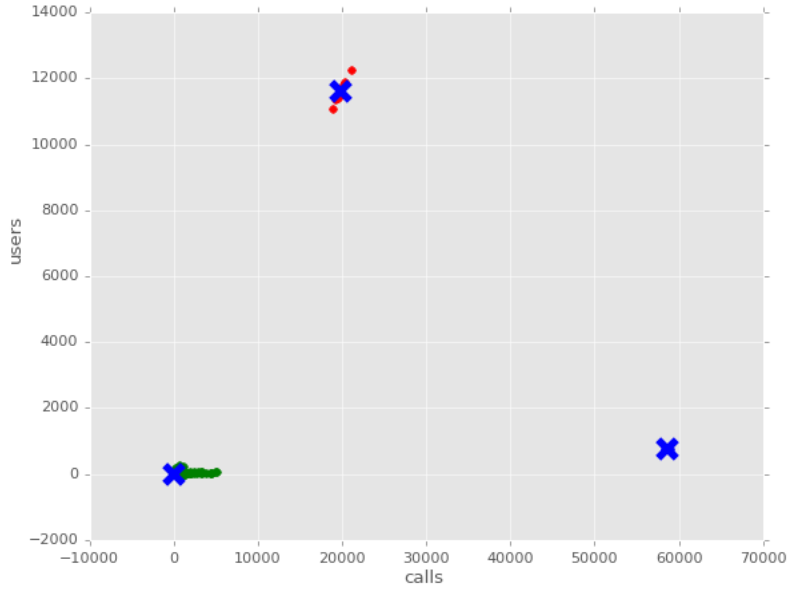

Figure 1. IP addresses by count of users and calls

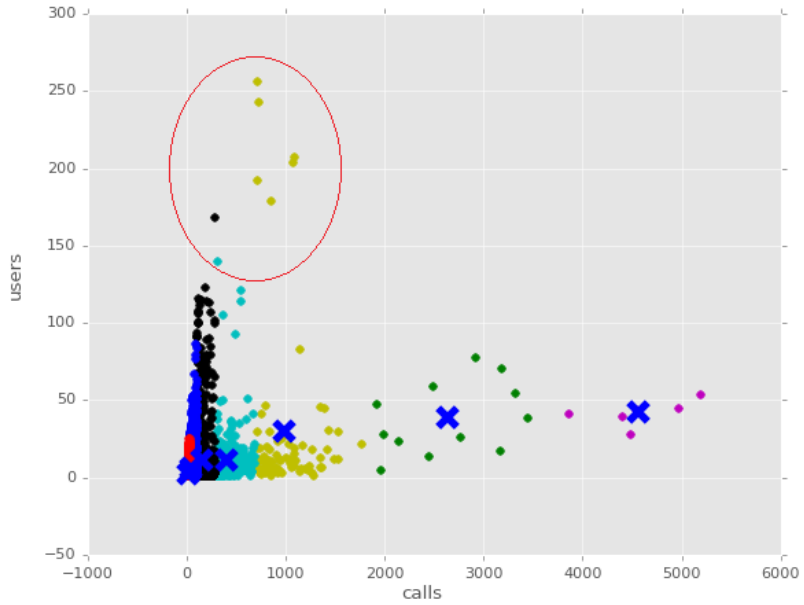

Figure 2. IP addresses by count of users and calls after exception elimination

I used the elbow method for deciding how many clusters is the optimal. The elbow method looks at the percentage of variance explained as a function of the number of clusters. It chooses the number of clusters so that adding another cluster does not give

better modelling of the data [18]. 7 clusters were chosen for my further analyses. Cluster centroids are shown in Table 3. The cluster centroids are marked as blue crosses on the graphs and are the centre point of a cluster meaning it can have also decimal points.

Table 3. Clustering results for IP addresses

Features				
Number of calls	**Number of users**	**Number of devices**	**Colour**	**Description**
156.629	11.138	11.47	Black	More Active users
12.553	2.271	2.42	Red	Regular user
4568.667	42.001	21.00	Purple	Big companies
987.513	29.423	26.68	Yellow	Big companies and also city centre
404.789	12.143	11.81	Cyan	Users making lot of calls
2641.500	38.667	31.75	Green	Big companies
51.333	6.047	6.30	Blue	Regular IP addresses

The description of the clusters is checked from IPlocation.net [19]. For making the decision, the author looked at what information they had about the IP address location and ownership.

During the analysis the author made some interesting discoveries. It appears that many calls are coming from India and they will be categorised as yellow. Seems that they are a metro station IP address. There is a lot of calls from India city centres, especially stood out New Delhi.

It took the author's interest why some of the IP addresses have a lot of users but not so many calls (Figure 2 circled by red). At first they seemed a little bit fraudulent or abusive. After digging in it was understood that they are call centres based in India and offering services to the Indian and Bhutan market.

The author wanted to explore more IP addresses that are not making lot of calls and also does not have a lot of users calling from them. They are the regular user IP addresses. The author eliminated the "green", "purple" and "yellow" clusters. That way I could take the data apart to see the regular user IP addresses (Figure 3). The author is satisfied

with the remaining IP addresses to be labelled as “regular IP addresses”. For the users calls I take into account only those calls that are from these remaining IP addresses.

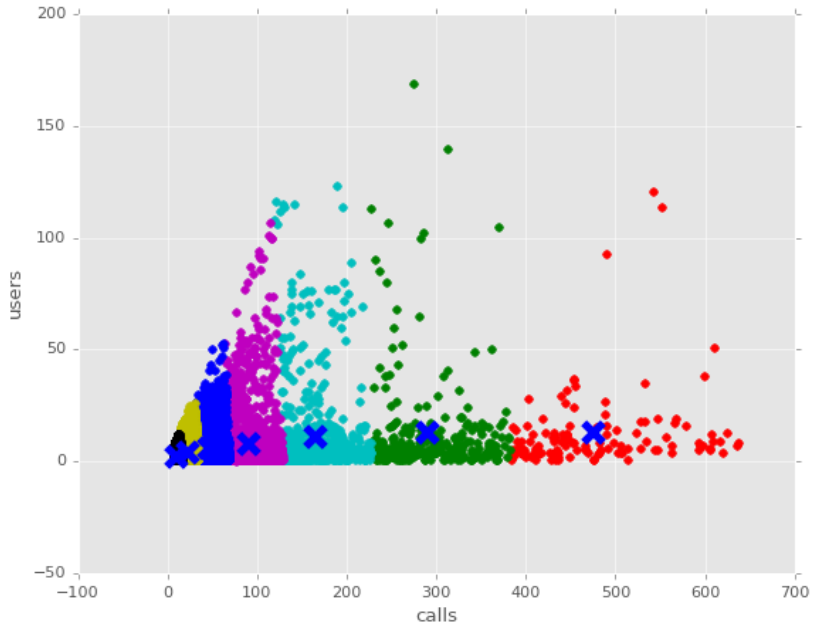

Figure 3. Regular IP addresses

4.2.2 Users

The main analysis of the paper is about users and how to classify them as fraudulent or not. To understand how the regular and fraudulent users behave, a number features are used that were described earlier. During the work, the algorithms calculate how important each of the features is by taking into account the ones that seem to detect fraudulence the most. Regression analysis and built-in Python functionality is used to measure the features importance.

The author will use clustering as an example of an unsupervised algorithm, decision tree and random forest as supervised algorithms.

4.2.2.1 Clustering

The author will use clustering to find users that are similar to each other and also to measure the percentage of blocked users in each cluster. The aim is to get a cluster including only blocked users or a cluster that has a lot more blocked users inside than other clusters. The difference between clusters and the percentage of blocked users can be used in the detection as an indicator. If user has a lot of parameters that have higher fraudulence ratio then the user will be blocked.

The clustering algorithm (Appendix 1 – K-means clustering with 2D figure) had unexpected results while running it with all the features (managed, blocked, calls, countries, devices, call duration, maximum call duration, average interval, maximum interval, minimum interval, number of contacts, number of calls ending 404). The author got 15 central points and not very good results in dividing the data. The best cluster had 5% of blocked users and was quite small having only 30 data points in it.

To solve the problem, the author started analysing data with smaller amount of features at a time, so that it would not be so complex. The aim was the same: to get clusters that have a lot of more blocked users than other clusters.

The author started the clustering by first looking into the features of calls and number of countries (Appendix 2) (Figure 4). In Figure 5 is shown how the two clusters are represented visually. The results (Table 4) show that there is a big difference between the clusters. One has 44% of blocked users when other 2.6%. Because these clusters are quite different, they can be used as part of the detection logic. If the user is in the green cluster its probability to be fraudulent is 20 times higher than in the other cluster.

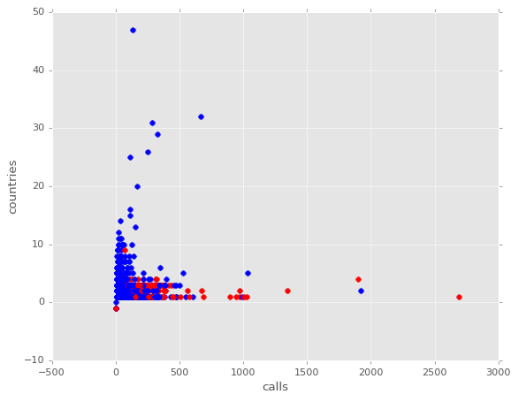

Figure 4. Users by countries and calls. Red are blocked users and blue are regular users.

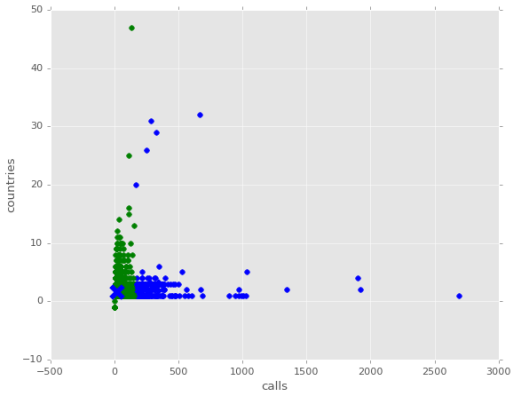

Figure 5. Users clustered by countries and calls. Two clusters: green and blue.

Table 4. Clustering results for countries and calls

Centroids					
Number of calls	**Number of countries**	**Colour**	**Total of users**	**Blocked**	**Percentage of blocking**
314.575	2.453	Green	254	112	44.0%
22.578	1.611	Blue	19659	515	2.6%

The author will continue using clustering for other parameters such as number of countries and devices, also average call duration and number of countries. With these parameters the algorithm did not work, as the fraudulent users are spread along different clusters. Using devices as a feature does not distinguish the fraudulent users from the regular users (Figure 6) (Figure 7). It is shown in the results (Table 5) that the two cluster system does not work. Increasing the number of clusters does not make the algorithm more accurate.

The author has a predication that fraudulent users have calls from more countries in short period of time and have shorter average call duration. The prediction of average call duration is based on the previous knowledge. The assumption seems to be not legitimate from the graphs (Figure 8) (Figure 9) and results (Table 6).

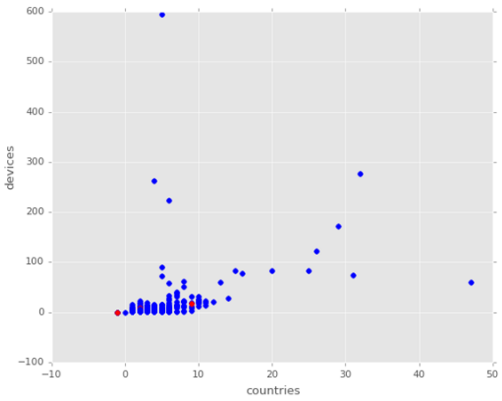

Figure 6. Users by countries and devices. Red are blocked users and blue are regular

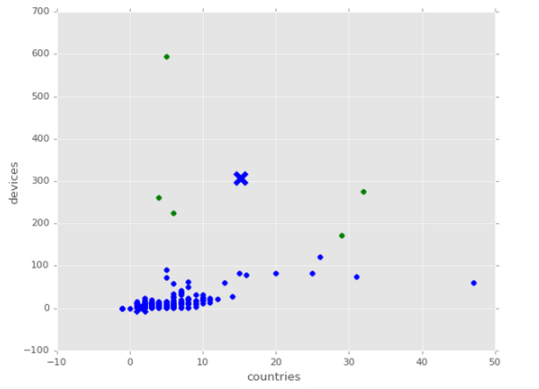

Figure 7. Users by countries and devices, clustered. Two clusters: green and blue.

Table 5. Clustering results for countries and devices

Centroids					
Number of countries	**Number of devices**	**Colour**	**Total of users**	**Blocked**	**Percentage of blocking**
1.619	1.827	Blue	19908	627	3.1%
15.200	305.601	Green	5	0	0.0%

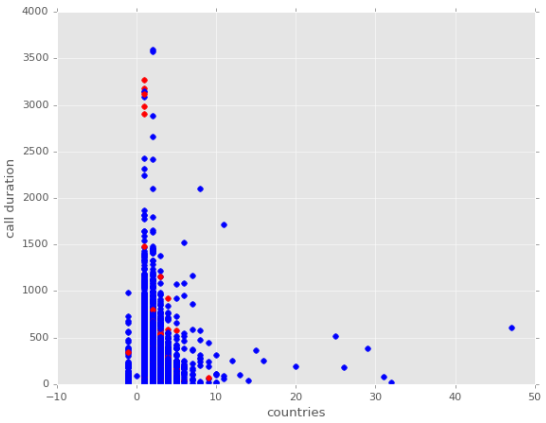

Figure 8. Users by average call duration and countries.

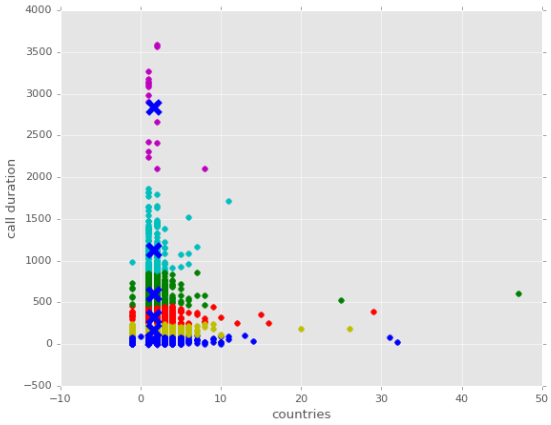

Figure 9. Users by average call duration and countries clustered, 6 clusters

Table 6. Clustering results for countries and average call duration

Centroids					
Number of countries	**Average call duration**	**Colour**	**Total of users**	**Blocked**	**Percentage of blocking**
1.700	592.221	Green	1048	35	3.3%
1.707	325.401	Red	2937	84	2.9%
1.750	1125.285	Cyan	172	4	2.3%
1.722	2.840	Blue	9979	359	3.6%
1.682	162.602	Yellow	5759	139	2.4%
1.830	2853.512900	Purple	18	6	33.0%

From the results of the last two examples of clustering, it can be seen that the clustering with two parameters will not cluster always efficiently. They are not accurate for using them in the detection. Therefore, analysis of all the parameters one by one is warranted to select them for the final clustering. The author will try to use clustering this time to look at the features separately to see if there is regression between fraudulence and the feature and then the experiments are done with three most important features.

For the regression analysis Microsoft Excel was used for creating the graphs. The author looks into the number of countries to see if it has a regression connection with fraudulence. From here it shows that the hypothesis is not relevant. Regression was expected but was not there. From the diagram (Figure 11) it can be seen that the hypothesis on more users using more countries is not true. The fraudulent users use probably less countries, mostly 3-5.

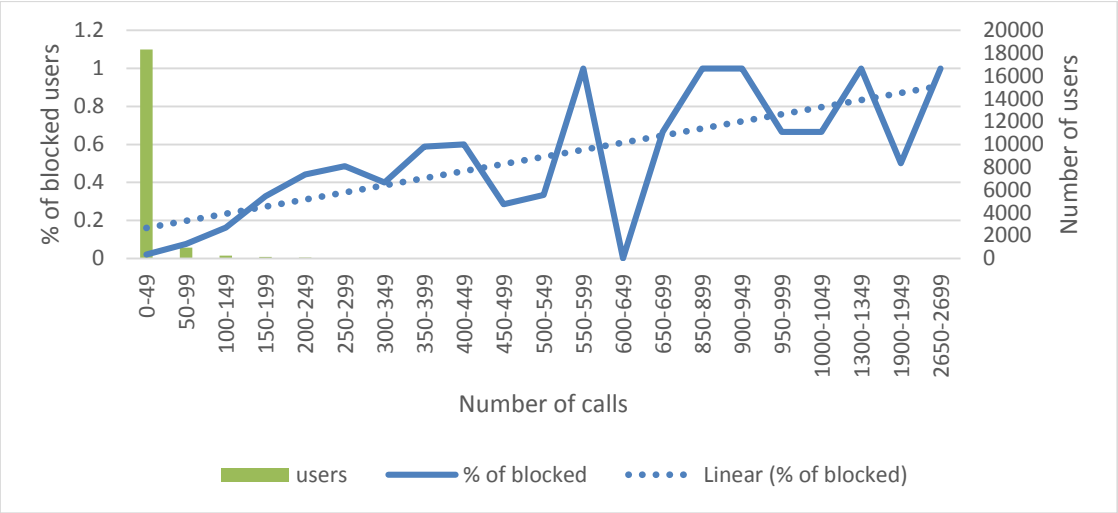

Figure 10. Number of calls and fraudulence

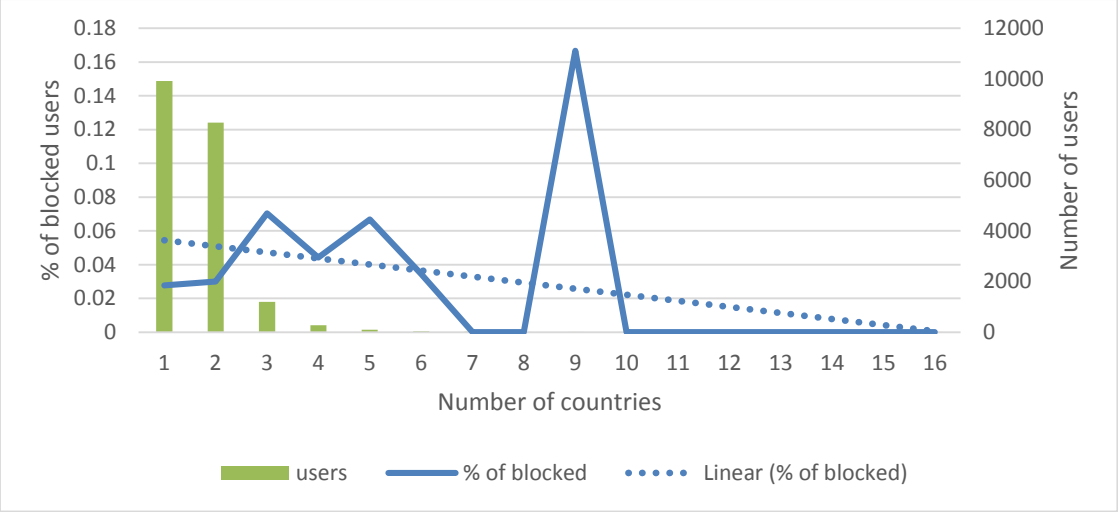

Figure 11. Number of countries and fraudulence

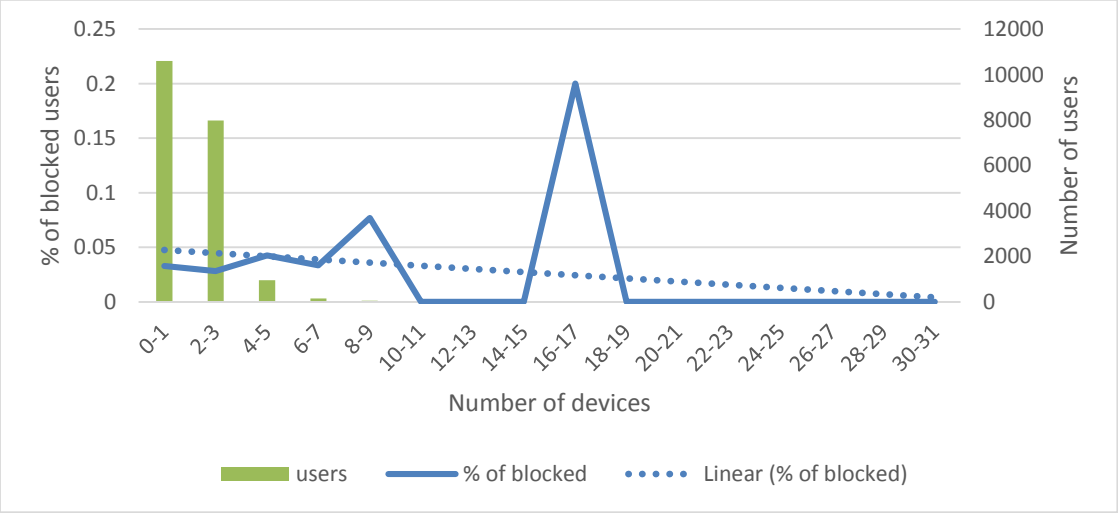

Figure 12. Number of devices and fraudulence

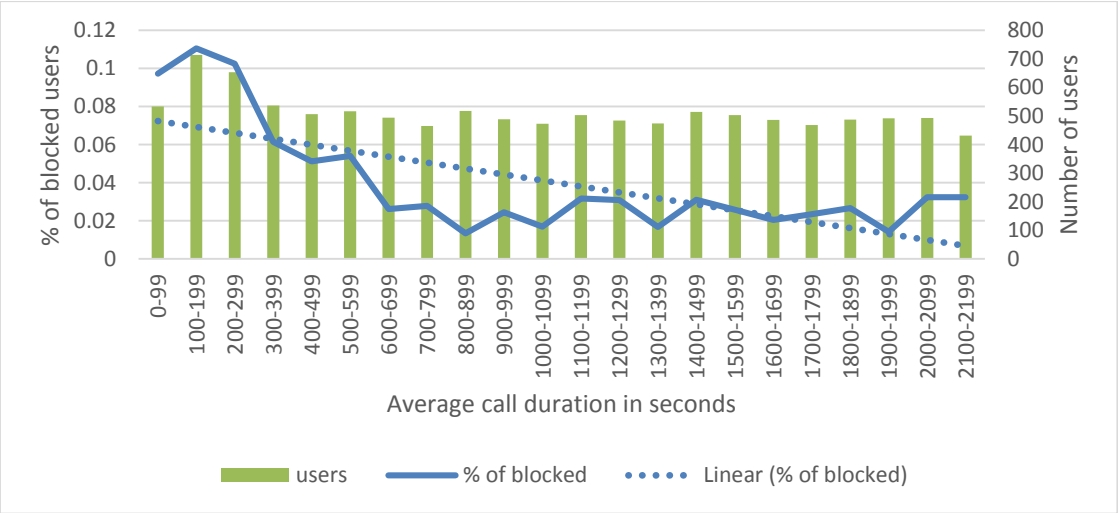

Figure 13. Average call duration and fraudulence

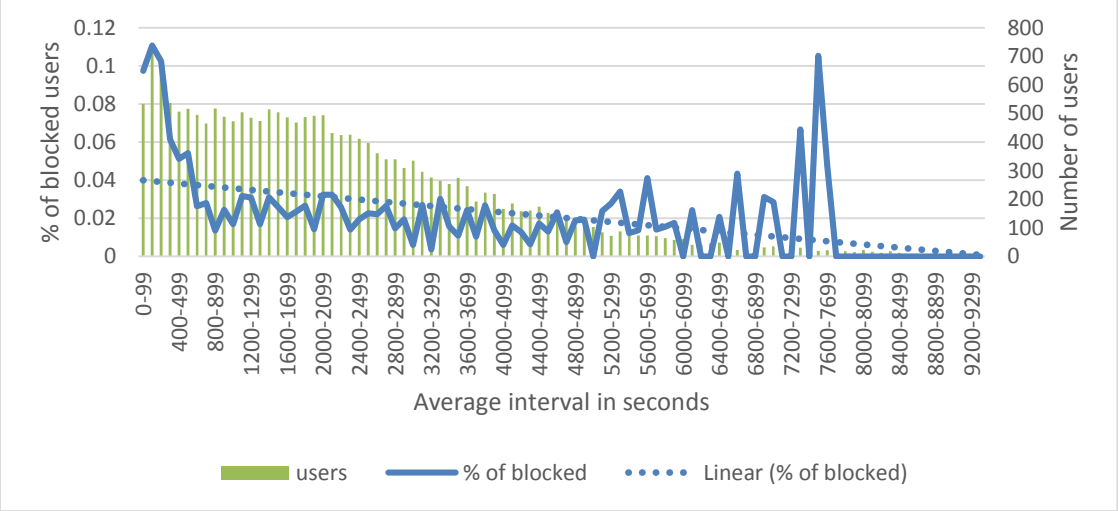

Figure 14. Average interval and fraudulence

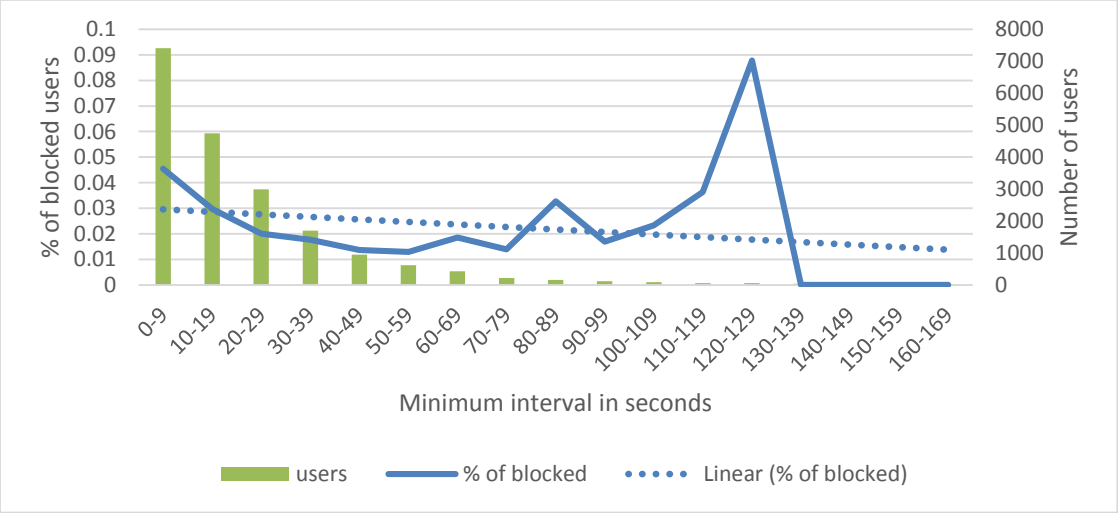

Figure 15. Minimum interval and fraudulences

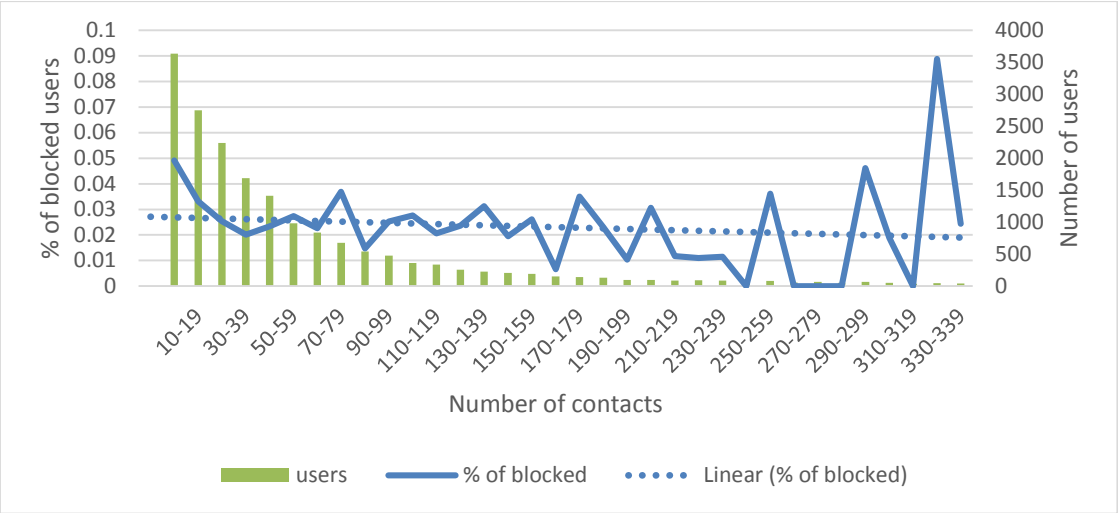

Figure 16. Number of contacts and fraudulence

After analysing the parameters separately, it was understood how relevant each of the parameter is for detecting fraud. The number of calls and average interval are used as the features and then taking into account contacts or countries. The results were compared by the percentage of users they had in each cluster and how they were divided.

The best result came with the combination of countries, calls and average interval. The users' distribution in countries, average interval and calls space (Figure 17) (Appendix 3 – Creating 3D figure) seems to be in the far corner. Three parameters were used for the clustering algorithm to determine if they can be clustered with good results (Figure 18) (Appendix 4 – K-means clustering with 3D figure). All together it ended up with four clusters (Table 7).

Table 7. Clustering result for users

Centroids						
Number of countries	**Average interval**	**Number of calls**	**Colour**	**Total of users**	**Blocked**	**Percentage of blocked**
1.711	5840.515	13.404	Blue	2402	38	1.6%
1.538	929.406	36.487	Red	10024	443	4.4%
1.706	2941.948	16.799	Green	7479	146	1.9%
1.750	31308.500	22.875	Cyan	8	0	0.0%

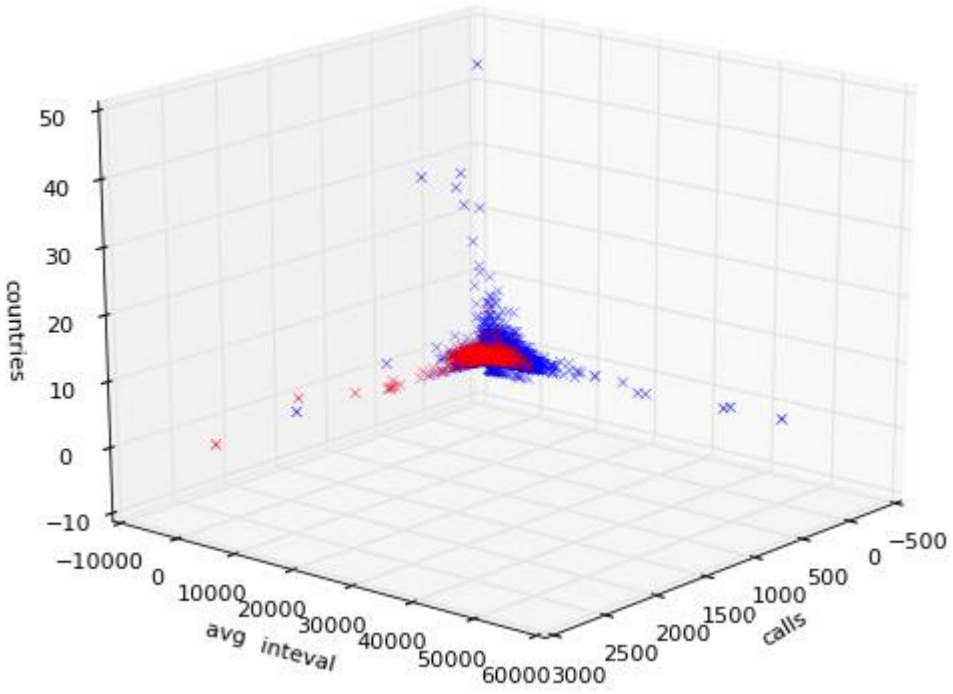

Figure 17. Users by countries, average interval and calls

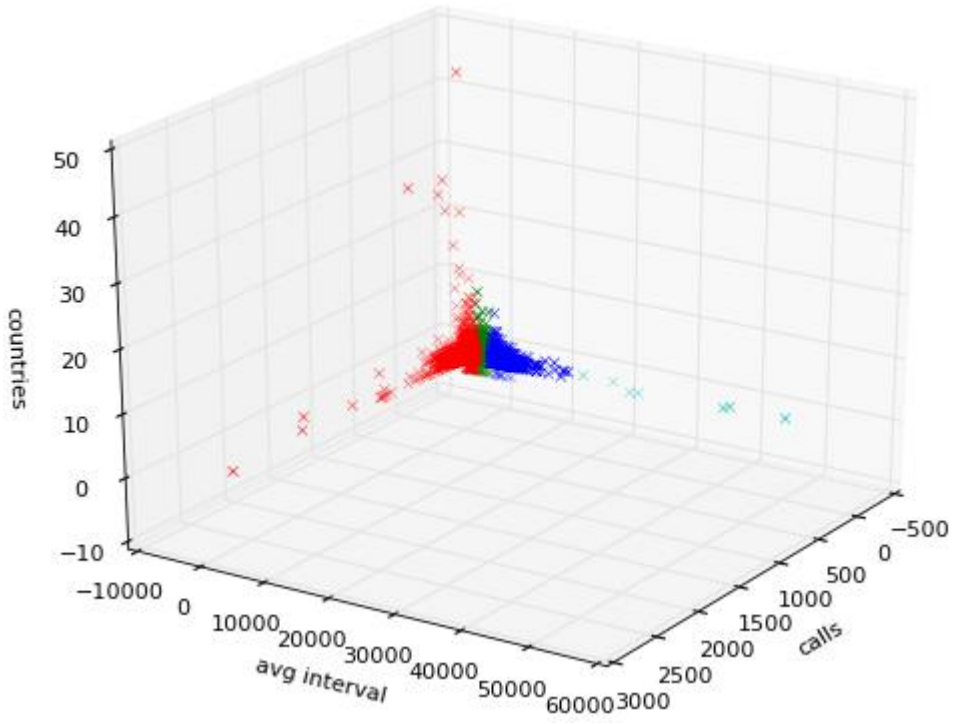

Figure 18. Final clustering

4.2.2.2 Decision Tree

The author will be using decision tree classification algorithm (Appendix 5 – Decision tree). The algorithm uses Gini impurity rate. The rate shows how often a randomly chosen element from the data will be labelled incorrectly if it was labelled according to the distribution rules. The algorithm tries to keep the Gini impurity rate low. The higher the rate the worse the dividing [20].

First, the author got a perfect tree (Figure 19), that classifies every user of this dataset correctly, meaning that with new data it labels the data mostly incorrectly. Perfect tree is only accurate for the data that it has learned from. This tree is overfitting, which happens when the algorithm continues to develop hypothesis and reduces the training set error and does it too much [13]. In this work trees with pre-set depth will be used as it is better to interpret and implement them into current blocking system.

Figure 19. Overfitting tree

During the learning process the author analysed with Python functionality how important the features are for the decision tree algorithm. The list is ordered according to the importance they have. The list also included number of countries:

1. Number of calls
2. Number of countries
3. Number of different devices
4. Average call duration
5. Minimum interval
6. Average interval
7. Maximum call duration
8. Number of calls ending with 404
9. Managed

A tree with depth 3 (Figure 20. Tree with depth 3) was chosen because it was accurate and easy to understand. The accuracy did not increase from 0.97 when new leaves and greater depth were added. The final false positive rate was 8%.

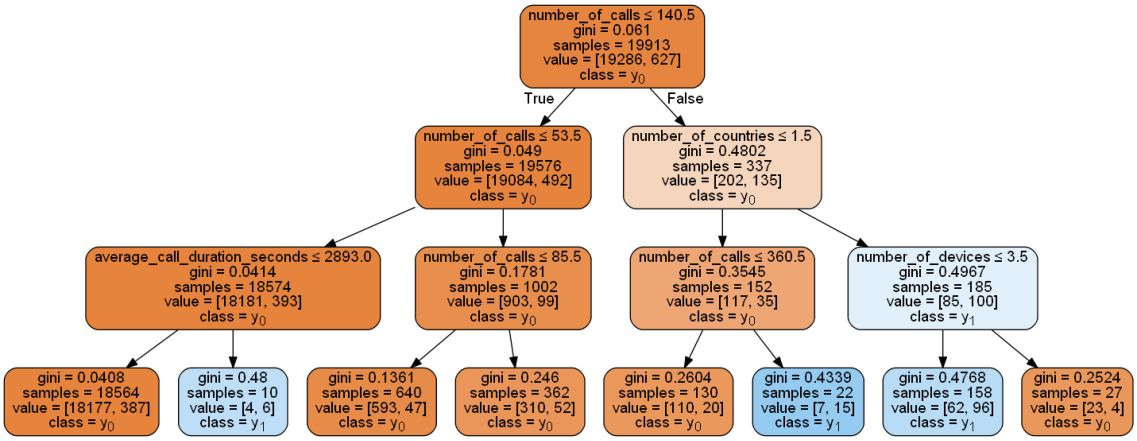

Figure 20. Tree with depth 3. Blue leaves have more fraudulent users in them than regular users. The orange leaves have more regular users than fraudulent users. The shade of leaves shows how sure the Gini index is – the darker the cleaner the leaf. The first row shows which feature and measure is taken for the split. Gini shows the Gini impurity. Value shows how many users are in this leaf: in the first position the non-blocked users and in the second the blocked. The class shows what label the algorithm gives for the leaf: y_0 is labelled as regular, y_1 is labelled as fraudulent

The decision tree (Figure 20) is made of data that is out of balance. It has more users with the regular label than the fraudulent label. This means that the tree can classify the regular users well but is not very good at classifying the fraudulent users. The author will be using undersampling to solve this problem. It is a technique in data analysis to mitigate the problem of class imbalance. The data is balanced by taking test data that includes equal number of fraudulent and regular users. This means that all data about fraudulent users is used, but some data of the regular users is not. Before the ratio was 3% fraudulent and 97% regular users. After making this balancing change it results in a tree (Figure 21), which has an accuracy rate of 0.76 but is balanced and false positive rate is 1%.

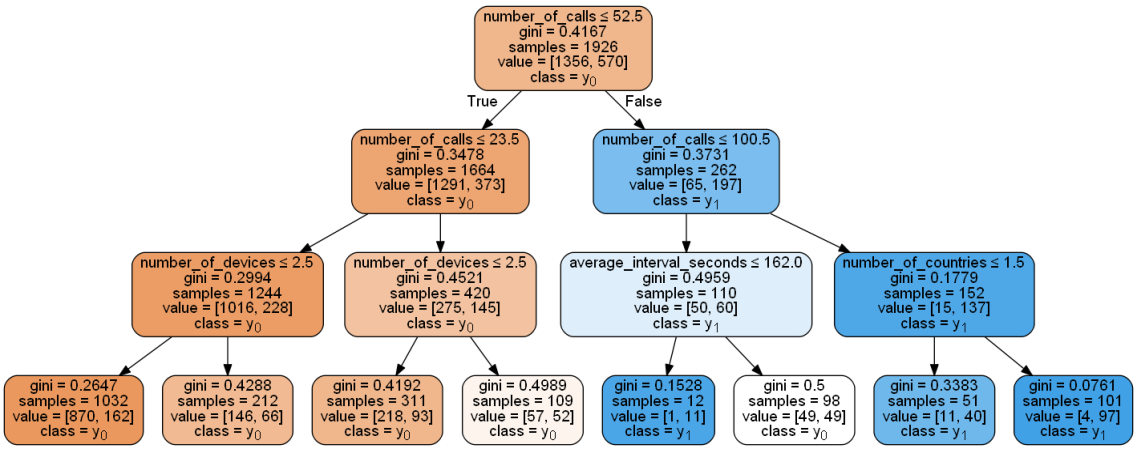

Figure 21. Tree with depth 3 balanced data

4.2.2.3 Random forest

For the third method, the author used the random forest algorithm (Appendix 6 – Random forest) that should minimize the false positive rates because it uses multiple trees to make its results better. Random forest is also currently used as part of the auto-blocking system, because it is more accurate than decision tree with the data where the searched label is in a large minority.

The best accuracy (0.79) for the random forest came with depth 6.

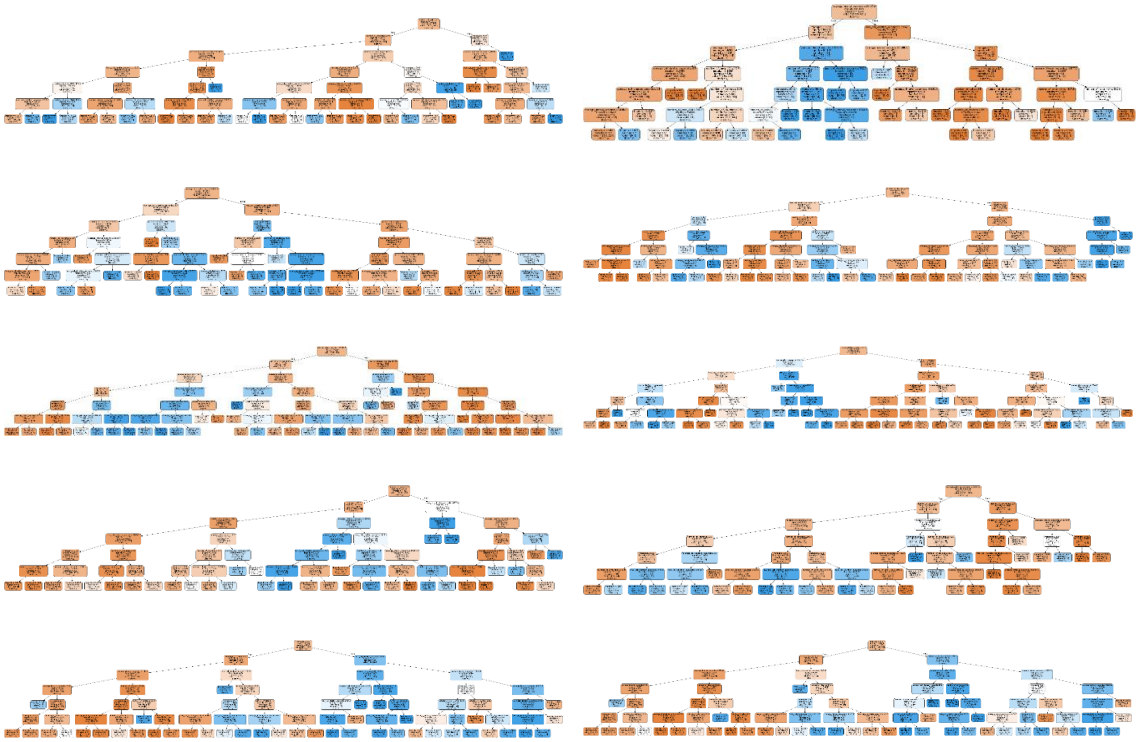

Figure 22. Random forest trees. Ten decision trees that were generated in the Random Forest.

4.3 Results

The accuracy of the three algorithms is checked by running them on new data. The result from the algorithms compared to actual data may end up in four categories (Table 8). The users, who will be labelled as fraudulent are going to be auto-blocked, thus make the false positive rate should be as low as possible so that regular users will not be falsely punished. The other important parameter is true positive rate, which shows how accurate the algorithm is for detecting fraudulent users.

The aim is to reduce false positive rate and raise true positive rate. It is better two have less fraudulent users blocked than to block regular users.

Table 8. Confusion matrix

Actual	**Predicted**	
	Fraudulent	**Regular**
Fraudulent	True Positive (TP)	False negative (FN)
Regular	False Positive (FP)	True Negative (TN)

For analysing the results better, the author took 1000 users from 16th of December 2015, looked at their blocked status and ran them thought the algorithms. It was then checked how much is their false positive rate (2) and true positive rate (3).

$$False\ positive\ rate = \frac{FP}{FP+TN} \quad (2)$$

$$True\ positive\ rate = \frac{TP}{TP+FN} \quad (3)$$

Clustering algorithm cannot be used only by itself, because I do not have a cluster that has majority of fraudulent users in it. Due to that it can be used only as part of the auto-blocking system. The test data from December shows that the clusters remained comparable to what the algorithm had before (Table 9).

Table 9. Clustering accuracy

Colour	**Percentage of blocked learning**	**Percentage of blocked testing**
Blue	1.6%	10.0%
Red	4.4%	25.0%
Green	1.9%	9.0%
Cyan	0.0%	0.0%

When comparing the decision tree and random forest learnings, it can be seen that they both have their good and bad sides (Table 10). The author would use decision tree more as it has less false positive rating. The Random tree algorithm has higher false positive rate and due to that it needs other parameters to improve its accuracy.

Table 10. Accuracy rates for decision tree and random forest

Accuracy	**Algorithms**	
	Decision Tree	**Random forest**
False positive rate	1%	2%
True positive rate	26%	30%

Conclusion

The aim of the thesis was to find out if the IP changes is a good indicator for how fraudulent callers are and use different methods to detect fraudulent users. The problem was studied with three different algorithms: clustering, decision tree and random forest. Their accuracy was measured and analysed.

Clustering had the worst results and the author would not use it. The decision tree and random forest algorithms came quite close for their results. The author would not use them individually but these can be used as part of the auto-blocking system as one of the indicators.

Key results from the thesis:

1. The hypothesis that fraudulent users call from more countries than regular users in short period of time is not true.
2. The clustering algorithm does not work well for solving this problem. Decision tree and random forest methods had sufficient results for adding them as part of the auto-blocking system while taking into account also other features.
3. To be confident for auto-blocking users, more indicators are needed.

The number of countries can be used in the auto-blocking system, as I can see that it was included in the decision trees as one of the separator between leaves. To make the algorithm more reliable, other features and relation with other users, should also be added for the calculations.

The algorithms could be improved by taking into account the weighted confusion matrix during analysis where having false positives is worse than having less true positives.

It could be considered for the future, that it is also possible to calculate the impossible travel through calculating the difference between IP addresses and if travelling from one location to another is possible in the time between call start times. This feature could also be useful for the blocking system and can be validated through same analysis.

Summary

The aim of the thesis was to provide an additional feature for a VoIP company's auto-blocking system using number of countries. This topic was chosen for investigation because more people use computers and more fraud is done through it. The example VoIP Company has a system for calling to real phone numbers from the calling software. The popular application attracts regular users, but also users, who want to take unfair advantage of it.

The goal of this work was to find the best machine learning algorithm for detecting fraudulent callers. It included implementing the algorithms, improving and validating their accuracy. In this paper the number of countries the user has called, was used among other features. It was analysed if the hypothesis, that the user who makes calls from more countries in short period of time is more fraudulent, could be used in the company's detection system.

The idea of using changes of IP addresses as one of the indicator for the detection system, was not new, but right now it is not being used in the current system. The author analysed if implementing it to the auto-blocking system would be reasonable.

Three machine learning algorithms: clustering, decision tree and random forest were used and compared to find out the best detection method. The accuracy of these three algorithms were measured with confusion matrices. The aim was to reduce the false positive rate and increase true positive rate. The users that are going to end up in the blocking will be auto-blocked and so it should be made sure that the false positive rate is as low as possible.

Research showed that the number of counties is not a good indicator and it needs support from other features. Still it was used as part of the fraud detection in clustering, decision tree and random forest algorithms. The three methods were tested with new data to measure their accuracy.

The best results for detecting the fraudulent users were obtained with decision tree and random forest. The clustering algorithm did not work well for solving the problem for finding fraudulent users. Decision tree and random forest methods had sufficient results

for adding them as part of the auto-blocking system while taking into account other features.

In conclusion, the random forest or decision tree could be used as part of the current system given they get accuracy support from other features.

References

[1] Association of Certified Fraud Examiners Inc., “Report to the nations on occupational fraud and abuse,” 2014.

[2] L. Delamaire, H. Abdou and J. Pointon, “Credit card fraud and detection techniques: a review,” *Banks and Bank Systems,* vol. IV, pp. 57-68, 2009.

[3] I. Kose, M. Gokturk and L. Kilic, “An interactive machine-learning-based electronic fraud and abuse detection system in healthcare insurance,” *Applied Soft Computing,* vol. XXVI, no. C, pp. 283-299, 2015.

[4] T. Fawcett and F. Provost, “Combining Data Mining and Machine Learning for Effective User Profiling,” in *AAAI 1996: Proceedings of the Thirteenth National Conference on Artificial Intelligence,* Cambridge, MIT Press, 1996.

[5] C. Phua, V. Lee, K. Smith-Miles and R. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference,* vol. I, pp. 50 - 53, 2010.

[6] A. Leontjeva, M. Goldszmidt, Y. Xie, F. Yu and M. Abadi, “Early Security Classification of Skype Users via Machine Learning,” in *Proceedings of the 2013 ACM workshop on Artificial intelligence and security,* New York, ACM, 2013, pp. 35-44.

[7] R. J. Bolton and D. J. Hand, “Statistical fraud detection: A review,” *Statistical Science,* vol. XVII, no. 3, pp. 235-249, 2002.

[8] R. J. Bolton and D. Hand, “Unsupervised profiling methods for fraud detection,” *Credit Scoring and Credit Control,* vol. VII, pp. 235-255, 2001.

[9] A. D. Pozzolo, O. Caelen, Y.-A. L. Borne, S. Waterschoot and G. Bontempi, “Learned lessons in credit card fraud detection from practitioner perspective,” *Expert Systems with Applications,* vol. XLI, no. 10, p. 4915–4928, 2014.

[10] “Python Scikit-Learn,” [Online]. Available: http://scikit-learn.org/stable/. [Accessed 29 March 2016].

[11] A. Samola, Introduction to Machine Learning, Cambridge, United Kingdom: Cambride University Press, 2008.

[12] S. Ben-David and S. Shalev-Shwartz, Understanding Machine Learning: From Theory to Algorithms, Cambridge: Cambridge University Press, 2014.

[13] J. R. Quinlan, “Induction of decision trees,” *Machine Learning,* vol. I, no. 1, pp. 81-106, 1986.

[14] T. Mitchell and M. Hill, Machine Learning, New York: McGraw-Hill Education, 1997.

[15] B. Leo, “Random Forest,” *Machine Learning,* vol. XLV, no. 1, pp. 5-32, 2001.

[16] M. Denil, D. Matheson and N. de Freitas, “Narrowing the Gap: Random Forests In Theory and In Practice,” in *International Conference on Machine Learning (ICML),* International Conference on Machine Learning, 2014, p. 665–673.

[17] A. Karim, S. I. Jami, I. Ahmad, M. Sarwar and Z. Uzmi, “Clustering IP Addresses Using Longest Prefix Matching and Nearest Neighbor Algorithms,” in *PRICAI 2004: Trends in Artificial Intelligence,* Auckland, New Zealand, Springer Berlin Heidelberg, 2004, pp. 965-966.

[18] R. Tibshirani, G. Walther and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. LXIII, no. 2, pp. 411-423, 2001.

[19] “IP Location,” Brand Media Inc, [Online]. Available: https://www.iplocation.net/. [Accessed 20 August 2015].

[20] S. Marsland, Machine Learning: An Algorithmic Perspective, London: Chapman & Hall, 2014.

Appendix 1 – K-means clustering with 2D figure

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from matplotlib import style
from numpy import genfromtxt
style.use("ggplot")

features=[
       'username',
       'managed',
       'blocked',
       'blocked_november',
       'blocked_april',
       'number_of_calls',
       'countries',
       'devices',
       'status',
       'average_call_duration',
       'call duration',
       '404',
       'max_call_duration',
       'max_call_duration_seconds',
       'average_interval',
       'average_interval_seconds',
       '',
       'min interval',
       'maximum_interval',
       'maximum_interval_seconds',
       'friends']

a=features.index('countries')
b=features.index('friends')
c=features.index('devices')
X = genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(a, b))[1:]
Y= genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(4))[1:]

clusters=2
kmeans = KMeans(n_clusters=clusters)
kmeans.fit(X)

centroids = kmeans.cluster_centers_
labels =kmeans.labels_
```

```
users=[0, 0, 0]
blocked=[0, 0, 0]
colors = ["b.", "g.", "r.", "c.", "m.", "y.", "k.", "w."]


for i in range(len(X)):
	plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)

	users[labels[i]]+=1
	if Y[i]==1:
		blocked[labels[i]]+=1
	plt.ylabel(features[b])
	plt.xlabel(features[a])

plt.scatter(centroids[:,0],centroids[:, 1], marker="x", s=150, linewidths=5,
zorder=10)

for i in range(clusters):
	print("----------------------")
	print(centroids[i])
	print(str(i)+" "+colors[i])
	print("Total of users: "+str(users[i]))
	print("Blocked: "+str(blocked[i]))
	print("Per centage: "+str(float(blocked[i])/float(users[i])))

plt.show()
```

Appendix 2 – Creating 2D figure

```
import matplotlib.pyplot as plt
from numpy import genfromtxt
from matplotlib import style

style.use("ggplot")
features=[
'username',
'managed',
'blocked',
'blocked_november',
'blocked_april',
'number_of_calls',
'countries',
'devices',
'status',
'average_call_duration',
'call duration',
'404',
'max_call_duration',
```

```
       'max_call_duration_seconds',
       'average_interval',
       'average_interval_seconds',
       '',
       'min interval',
       'maximum_interval',
       'maximum_interval_seconds',
       'friends']
a=features.index('calls')
b=features.index('avg interval')
c=features.index('countries')
X = genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(a, b, 4))[1:]

colors = ["b.", "r."]

for i in range(len(X)):
            plt.plot(X[i][0], X[i][1], colors[int(X[i][2])], markersize=10)
            plt.xlabel(features[a])
            plt.ylabel(features[b])
plt.show()
```

Appendix 3 – Creating 3D figure

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from numpy import genfromtxt
from matplotlib import style

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
style.use("ggplot")

x1 =[]
y1 =[]
z1 =[]

x2 =[]
y2 =[]
z2 =[]

features=[
```

```
        'username',
        'managed',
        'blocked',
        'blocked_november',
        'blocked_april',
        'calls',
        'countries',
        'devices',
        'status',
        'average_call_duration',
        'call duration',
        '404',
        'max_call_duration',
        'max_call_duration_seconds',
        '',
        'avg interval',
        '',
        'min interval',
        'maximum_interval',
        'minimum interval',
        'friends']

a=features.index('calls')
b=features.index('avg interval')
c=features.index('countries')
X = genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(a, b, c, 1))[1:]
for i in range(len(X)):
        if X[i][3] == 1:
                x1.append(X[i][0])
                y1.append(X[i][1])
                z1.append(X[i][2])
        else:
                x2.append(X[i][0])
                y2.append(X[i][1])
                z2.append(X[i][2])

        ax.scatter(x1, y1, z1, c='r', marker='x')
        ax.scatter(x2, y2, z2, c='b', marker='x')

        ax.set_xlabel(features[a])
        ax.set_ylabel(features[b])
        ax.set_zlabel(features[c])

        plt.show()
```

Appendix 4 – K-means clustering with 3D figure

```
from numpy import genfromtxt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from matplotlib import style

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
style.use("ggplot")

x1 =[]
y1 =[]
z1 =[]

x2 =[]
y2 =[]
z2 =[]

x3 =[]
y3 =[]
z3 =[]

x4= []
y4 =[]
z4 =[]

features=[
       'username',
       'managed',
       'blocked',
       'blocked_november',
       'blocked_april',
       'calls',
       'countries',
       'devices',
       'status',
       'average_call_duration',
       'call duration',
       '404',
       'max_call_duration',
       'max_call_duration_seconds',
       '',
       'avg interval',
       '',
       'min interval',
       'maximum_interval',
       'minimum interval',
       'friends']

a=features.index('calls')
b=features.index('avg interval')
```

```
c=features.index('countries')

X = genfromtxt('Training_data.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(a, b, c))[1:]
Y= genfromtxt('Training_data.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(4))[1:]

clusters=4
kmeans = KMeans(n_clusters=clusters)
kmeans.fit(X)

centroids = kmeans.cluster_centers_
labels =kmeans.labels_

users=[0, 0, 0, 0]
blocked=[0, 0, 0, 0]
colors = ['b', 'r', 'g', 'c']
for i in range(len(X)):
	if labels[i] == 0:
		x1.append(X[i][0])
		y1.append(X[i][1])
		z1.append(X[i][2])
	elif labels[i] == 1:
		x2.append(X[i][0])
		y2.append(X[i][1])
		z2.append(X[i][2])
	elif labels[i] == 2:
		x3.append(X[i][0])
		y3.append(X[i][1])
		z3.append(X[i][2])
	elif labels[i] == 3:
		x4.append(X[i][0])
		y4.append(X[i][1])
		z4.append(X[i][2])
	users[labels[i]]+=1
	if Y[i]==1:
		blocked[labels[i]]+=1

	ax.scatter(x1, y1, z1, c=colors[0], marker='x')
	ax.scatter(x2, y2, z2, c=colors[1], marker='x')
	ax.scatter(x3, y3, z3, c=colors[2], marker='x')
	ax.scatter(x4, y4, z4, c=colors[3], marker='x')

	ax.set_xlabel('calls')
	ax.set_ylabel('avg interval')
	ax.set_zlabel('countries')
	plt.show()

for i in range(clusters):
	print("----------------------")
	print("calls "+str(centroids[i][0]))
	print("interval "+str(centroids[i][1]))
```

```
        print("countries "+str(centroids[i][2]))
        print(str(i)+" "+colors[i])
        print("Total of users: "+str(users[i]))
        print("Blocked: "+str(blocked[i]))
        print("Per centage: "+str(float(blocked[i])/float(users[i])))
```

Appendix 5 – Decision tree

```
from numpy import genfromtxt
import numpy as np
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import cross_val_score

clf = tree.DecisionTreeClassifier(min_samples_split= 10, max_leaf_nodes=
None, criterion= 'gini', max_depth= 4, min_samples_leaf= 10)

X = genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(1,5, 6, 7, 10, 11, 13, 15, 19))[1:]
Y = genfromtxt('Training_data.csv', delimiter=';', skip_header=1, dtype=None,
usecols=(4))[1:]

clf = clf.fit(X, Y)

features=[
        'managed',
        'number_of_calls',
        'number_of_countries',
        'number_of_devices',
        'average_call_duration_seconds',
        'number_of_error_404',
        'max_call_duration_seconds',
        'average_interval_seconds',
        'minimum_interval_seconds',
        'maximum_interval_seconds',
        'number_of_contacts']

tree.export_graphviz(clf, feature_names=features,out_file='overfill.dot',
class_names=True, filled=True, rounded=True, special_characters=True)

importances = DecisionTreeClassifier.feature_importances_

rank = np.argsort(clf.feature_importances_)[::-1]
print('\n'.join(features[i] for i in rank[:]))

scores = cross_val_score(clf, X, Y, cv=10)
print("mean: {:.3f} (std: {:.3f})".format(scores.mean(),scores.std()))

prediction = clf.predict(X)
```

```
FP=0
FN=0
TP=0
TN=0

for i in range(len(prediction)):
	if prediction[i]==Y[i] and Y[i]==0:
		TN+=1
	elif prediction[i]==Y[i] and Y[i]==1:
		TP+=1
	elif prediction[i]==1 and Y[i]==0:
		FP+=1
	elif prediction[i]==0 and Y[i]==1:
	FN+=1
print("FP "+str(FP)+" "+str(FP/(FP+FN+TP+TN)))
print("FN "+str(FN)+" "+str(FN/(FP+FN+TP+TN)))
print("TP "+str(TP)+" "+str(TP/(FP+FN+TP+TN)))
print("TN "+str(TN)+" "+str(TN/(FP+FN+TP+TN)))

X_test = genfromtxt('Test_Data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(1,5, 6, 7, 10, 11, 13, 15, 19))[1:]
Y_test = genfromtxt('Test_Data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(4))[1:]

print(X_test)
prediction_test = clf.predict(X_test)

FP=0
FN=0
TP=0
TN=0

print(len(prediction_test))
print(len(Y_test))

for i in range(len(prediction_test)):
	if prediction_test[i]==Y[i] and Y_test[i]==0:
		TN+=1
	elif prediction_test[i]==Y[i] and Y_test[i]==1:
		TP+=1
	elif prediction_test[i]==1 and Y_test[i]==0:
		FP+=1
	elif prediction_test[i]==0 and Y_test[i]==1:
		FN+=1

print("FP "+str(FP)+" "+str(FP/(FP+FN+TP+TN)))
print("FN "+str(FN)+" "+str(FN/(FP+FN+TP+TN)))
print("TP "+str(TP)+" "+str(TP/(FP+FN+TP+TN)))
print("TN "+str(TN)+" "+str(TN/(FP+FN+TP+TN)))
print(clf.score(X, Y, sample_weight=None))
```

Appendix 6 – Random forest

```
from random import random
from numpy import genfromtxt
import numpy as np
from sklearn import tree, ensemble
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import cross_val_score
from numpy import genfromtxt, savetxt

rf = ensemble.RandomForestClassifier( max_leaf_nodes= None, criterion=
'gini', max_depth= 6, min_samples_leaf= 5)

X = genfromtxt('Training_data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(5, 6, 7, 10, 11, 13, 15, 19))[1:]
Y = genfromtxt('Training_data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(4))[1:]
print(X)
clf = rf.fit(X, Y)

estimators=rf.estimators_
importances=rf.feature_importances_

numberClasses=rf.n_classes_
numberInputs=len(rf.feature_importances_)
numberTrees=len(rf.estimators_)

features=[
       'managed',
       'number_of_calls',
       'number_of_countries',
       'number_of_devices',
       'average_call_duration_seconds',
       'number_of_error_404',
       'max_call_duration_seconds',
       'average_interval_seconds',
       'minimum_interval_seconds',
       'maximum_interval_seconds',
       'number_of_contacts']

prediction = rf.predict(X)
FP=0
FN=0
TP=0
TN=0

for i in range(len(prediction)):
       if prediction[i]==Y[i] and Y[i]==0:
              TN+=1
       elif prediction[i]==Y[i] and Y[i]==1:
              TP+=1
       elif prediction[i]==1 and Y[i]==0:
              FP+=1
       elif prediction[i]==0 and Y[i]==1:
```

```
            FN+=1

print("FP "+str(FP)+" "+str(FP/(FP+FN+TP+TN)))
print("FN "+str(FN)+" "+str(FN/(FP+FN+TP+TN)))
print("TP "+str(TP)+" "+str(TP/(FP+FN+TP+TN)))
print("TN "+str(TN)+" "+str(TN/(FP+FN+TP+TN)))
print(rf.score(X, Y, sample_weight=None))

for num in range(0, numberTrees):
      filename="randomforest_"+str(num)+".dot"
      with open(filename, 'w') as f:
            f=tree.export_graphviz(estimators[num].tree_,feature_names=featu
      res, out_file=f, class_names=True, filled=True, rounded=True,
      special_characters=True)

X_test = genfromtxt('Test_Data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(5, 6, 7, 10, 11, 13, 15, 19))[1:]
Y_test = genfromtxt('Test_Data_2.csv', delimiter=';', skip_header=1,
dtype=None, usecols=(4))[1:]
print(X_test)

prediction_test = rf.predict(X_test)
FP=0
FN=0
TP=0
TN=0

for i in range(len(prediction)):
      if X_test[i][0]!=-1:
      if prediction_test[i]==Y_test[i] and Y_test[i]==0:
            TN+=1
      elif prediction_test[i]==Y_test[i] and Y_test[i]==1:
            TP+=1
      elif prediction_test[i]==1 and Y_test[i]==0:
            FP+=1
      elif prediction_test[i]==0 and Y_test[i]==1:
            FN+=1
      print(prediction_test[i])

print("FP "+str(FP)+" "+str(FP/(FP+FN+TP+TN)))
print("FN "+str(FN)+" "+str(FN/(FP+FN+TP+TN)))
print("TP "+str(TP)+" "+str(TP/(FP+FN+TP+TN)))
print("TN "+str(TN)+" "+str(TN/(FP+FN+TP+TN)))
print(rf.score(X, Y, sample_weight=None))
```