UNIVERSITA' DEGLI STUDI DI MILANO

DEPARTMENT OF ECONOMICS, MANAGEMENT AND QUANTITATIVE METHODS

DATA SCIENCE AND ECONOMICS

STATISTICAL LEARNING PROJECT

SUPERVISED LEARNING:

DECISION TREES, RANDOM FOREST AND ARTIFICIAL NEURAL NETWORK APPLIED TO NETWORK TRAFFIC

Author:

Laura CIURCA

942318

CONTENTS

ABSTRACT

The project is about Android applications, which are vulnerable to malicious attacks because of their open-source code and the fact that third parties could install applications without any central control. In this paper, there will be applied statistical learning techniques in the cybersecurity field in order to try to predict malicious activities. More in detail, the project aims at using network traffic features as the basis for supervised and unsupervised learning models to detect malware and benign applications, trying to reduce the false positive and false negative cases.

The dataset has been taken from Kaggle (https://www.kaggle.com/xwolf12/network-traffic-android-malware?select=android_traffic.csv) and has been made using data from different pcaps files getting Android apps and Malware apps in 2018. It includes information about 112 application regarding network traffic, more exactly file transfer measured through the number of tcp and other packets sent and received and internet traffic measured in byte volumes.

The first part of the project is focused on the creation of predictive models to forecast the type of application (benign/malicious) on the basis of data traffic using the following supervised learning techniques:

- Decision trees
- Random forest
- ANN

The second part is focused on PCA technique that has been implemented to reduce the complexity of the model and to allow ANN to perform better.

ALGORITHMS

DECISION TREE: A decision tree is a supervised predictive modelling approach used as a decision-making tool, since it allows to visualize and understand decision's options and possible outcomes. The decision tree structure starts from the root node, which is the highest one and is the global element. The top node originates all paths leading to leaf nodes, which contain questions to be answered. The third main part of decision trees is about branches that are arrows connecting nodes. In this way, decision trees are used to make predictive conclusions about an item target value (represented in the leaves) on the basis of its observations (represented in the branches). One could distinguish between classification trees (discrete target variable) and regression trees (continuous target variable). Decision Tree is used to create a training model able to predict value of target variables by learning decision rules based on prior data, the so-called training data. The peculiarity of this algorithm is its way of solving the problem by representing it through a tree diagram. Each internal node of the tree is an attribute, while each leaf node is a class label. To predict a class label, one has to start from the root of the tree, comparing the values of the root attribute with record's attribute and then following the branch corresponding to that value and so jumping to the following node. Comparisons between the record's attribute values and other internal nodes of the tree continue until a leaf node with predicted class value will be reached. The Decision Trees' algorithm used in this project is based on a criterium called Gini index, which is a measure that shows how often a randomly chosen element would be identified incorrectly. So, an attribute with lower Gini index is better than a one with higher.

RANDOM FOREST: Random Forest is a supervised classification algorithm, consisting in creating a forest made of a big number of decision trees that act as an ensemble learning method for classification, regression or other tasks. Each single tree spits out a class prediction and the one that has the most votes becomes the model's prediction. To be well performed, the random forest should be applied when data have features so that models built using these features do better than random guessing. Moreover, the larger the number of trees, the more accurate will be the results. Furthermore, predictions made by the individual trees need to have low correlations with each other in order to be more accurate, since the trees protect each other from their individual errors (as they go normally in different directions). The main difference between Decision Tree and Random Forest is that the second one finds the root node and splits the feature nodes randomly. Random Forest algorithm acts in two stages: at first it creates random forest by combining decision trees and secondly it makes a prediction from the random forest classifier previously created. After the random forest classifier creation, Random Forest algorithm makes the prediction in three steps: 1) It considers the test features and uses each decision tree's rules in order to predict the outcome storing it (target). 2) It calculates the votes for each target. 3) It takes into consideration the high voted target as final prediction.

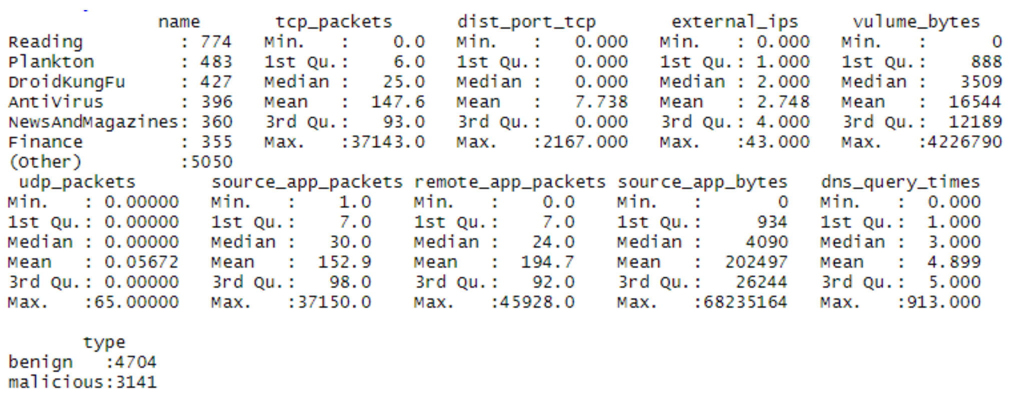
ARTIFICIAL NEURAL NETWORK: An artificial neural network (ANN) is a technique used in machine learning, that is a brain-inspired system designed to simulate the way the human brain works as regards information analysis. ANN is able to learn by itself producing more accurate results as more data becomes available. Artificial neural network is characterized by hundreds or thousands of neurons, known as processing units, that are connected to each other by nodes, and each of them consists of input and output units. An input unit receives information so that the neural network tries to learn the information to produce an output and to optimize it using learning rules called backpropagation. ANN faces a training phase, where the algorithm learns to recognize features in the data set and compares the output produced with the one expected. During the first step, the ANN tries to adjust the difference between the actual and expected output and to minimize the loss function through the so-called backpropagation. So, it works “backward”, from output units to input units, computing the gradient of the loss function with respect to the weights of its connection between units to obtain the error as lowest as possible. In this way the ANN understands how features are associated to outputs, searching for correct responses in binary format.

DATA CLEANING AND ANALYSIS

The dataset has been taken from Kaggle (https://www.kaggle.com/xwolf12/network-traffic-android-malware?select=android_traffic.csv) and it contains 7845 rows and 17 columns. Columns include the following information for 114 applications:

- **name**: name of the application
- **tcp_packets**: number of packets TCP (Transmission Control Protocol) sent and got during a network conversation trough which application programs can exchange data.
- **dist_port_tcp**: number of packets different from TCP
- **external_ips**: number of external IPs (Internet Protocol addresses) , where the app tried to connect for communication.
- **volume_bytes**: number of bytes that passed from the application to the external sites
- **udp_packets**: number of packets UDP (User Datagram Protocol), through which applications can communicate sending messages as datagrams to others on an IP network.
- **tcp_urg_packet:** number of packets used to send data on a second channel of a TCP conncetion, being a signal that shows that there is urgent data available.
- **source_app_packets**: number of packets sent from the source application to a remote server
- **remote_app_packets**: number of packets sent from external sources
- **source_app_bytes**: number of bytes of the source application, which shows the volume of data exchanged between the application and the server
- **remote_app_bytes**: number of bytes of the remote application, that is the volume of the data exchanged between the server and the emulator
- **duracion:** (NA) duration of connection
- **avg_local_pkt_rate:** (NA) average rate of local packets
- **avg_remote_pkt_rate:** (NA) average rate of remote packets
- **source_app_packets.1:** number of packets from the source application to a remote server
- **dns_query_times**: number of queries related to DNS (Domain Name System), where domain names are located and translated into internet protocol addresses.
- **type**: type of application: benign or malicious

First of all, data has been filtered and “duracion”, “avg_local_pkt_rate” and “avg_remote_pkt_rate” columns have been removed, considering that they don’t contain any values (NA values). Then, “tcp_urg_packet” has been deleted, since it contains just 2 values that are different from 0 and “source_app_packets.1” has been removed, since it is equal to column “source_app_packets”.



```
               name       tcp_packets      dist_port_tcp       external_ips     vulume_bytes
Reading          : 774   Min.   :    0.0   Min.   :   0.000   Min.   : 0.000   Min.   :      0
Plankton         : 483   1st Qu.:    6.0   1st Qu.:   0.000   1st Qu.: 1.000   1st Qu.:    888
DroidKungFu      : 427   Median :   25.0   Median :   0.000   Median : 2.000   Median :   3509
AntiVirus        : 396   Mean   :  147.6   Mean   :   7.738   Mean   : 2.748   Mean   :  16544
NewsAndMagazines: 360   3rd Qu.:   93.0   3rd Qu.:   0.000   3rd Qu.: 4.000   3rd Qu.:  12189
Finance          : 355   Max.   :37143.0   Max.   :2167.000   Max.   :43.000   Max.   :4226790
(Other)          :5050
 udp_packets       source_app_packets remote_app_packets source_app_bytes   dns_query_times
Min.   : 0.00000   Min.   :    1.0    Min.   :    0.0    Min.   :       0   Min.   :  0.000
1st Qu.: 0.00000   1st Qu.:    7.0    1st Qu.:    7.0    1st Qu.:     934   1st Qu.:  1.000
Median : 0.00000   Median :   30.0    Median :   24.0    Median :    4090   Median :  3.000
Mean   : 0.05672   Mean   :  152.9    Mean   :  194.7    Mean   :  202497   Mean   :  4.899
3rd Qu.: 0.00000   3rd Qu.:   98.0    3rd Qu.:   92.0    3rd Qu.:   26244   3rd Qu.:  5.000
Max.   :65.00000   Max.   :37150.0    Max.   :45928.0    Max.   :68235164   Max.   :913.000

       type
benign   :4704
malicious:3141
```

Figure 1: data summary

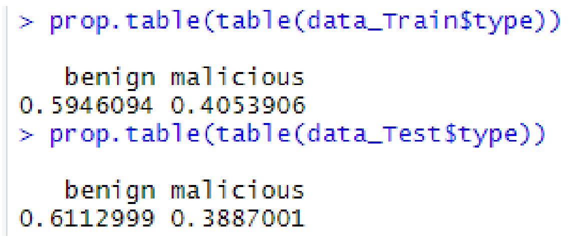
Data contain 4704 benign application records and 3141 malicious records. To predict applications type and identify malware Android application on the basis of network traffic, it will be taken into consideration the following columns: tcp_packets, dist_port_tcp, volume_bytes, source_app_packets, remote_app_packets, source_app_byte, remote_app_bytes.

In networking, malware applications are detected because normally the number of packages sent and received by the application is much inferior or higher than the one which characterizes benign applications. The same happens for the volume of bytes sent and received. DNS queries have been not included, since the number of queries for both benign and malicious application was changing randomly and similarly. So, since there are some characteristics that malwares share in their network behaviors which are different from those of normal software, let's try to apply Decision Tree to make predictions.

RESEARCH

APPLYING DECISION TREE TO THE PROBLEM

Before applying decision tree algorithm, it has been created the train and test set, by splitting data 70/30, where 70% of the data has been used to train the model and the remaining 30% to make predictions. The train dataset consists in 5491 rows, which include 3265 benign cases and 2226 malicious cases, while the test dataset has 2354 rows, presenting 1439 benign cases and 915 malicious cases. It has been verified if randomization has been processed correctly trough the prob.table() function.



```
> prop.table(table(data_Train$type))

   benign malicious
0.5946094 0.4053906
> prop.table(table(data_Test$type))

   benign malicious
0.6112999 0.3887001
```

Figure 2: check for randomization trough the function prop.table()

In both train and test dataset, the amount of benign application cases is almost the same, about 60 percent.

After splitting data in train and test data frames, Decision Tree has been applied to try to predict benign and malicious applications on the basis of network traffic, taking into consideration at first the most important variables that influence networking, that are TCP packets, the total volume measured in bytes, remote application packets and source application packets and their respective bytes.

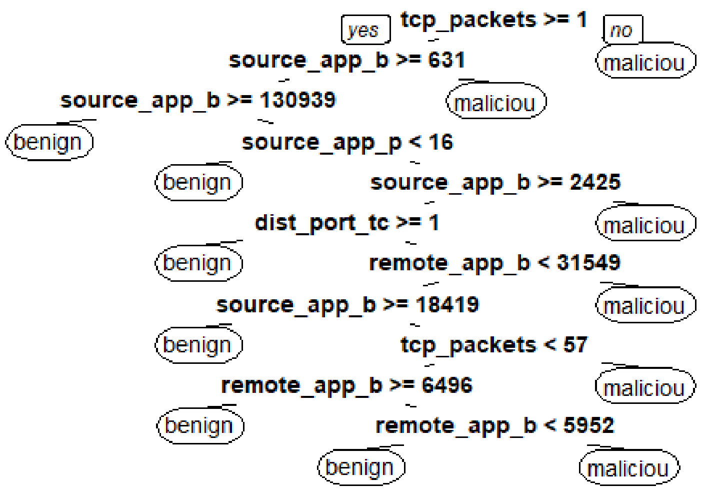
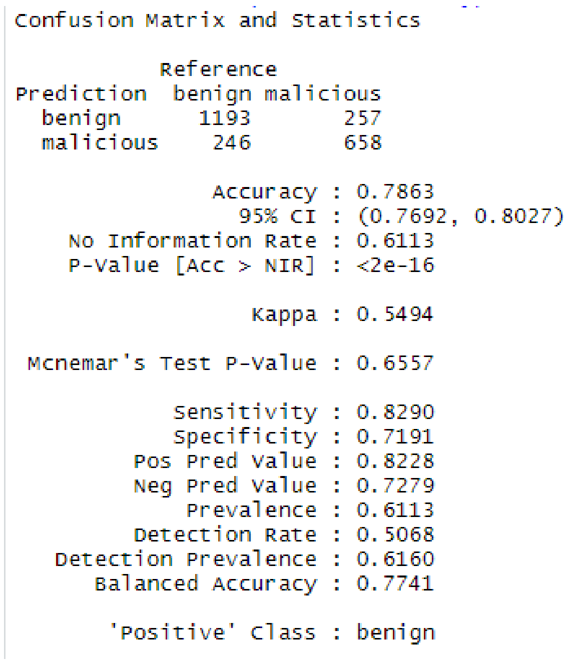


Figure 3: Decision Tree

The TCP packets is the most important factor in determining the type of application and is the predictor variable for the primary split, where the split point is 1. If the number of TCP packets is less than 1, the application results as malicious, while if it is bigger the algorithm proceeds with further splits. The tree has 12 terminal nodes, known as leaves. After the first split, the applications result malicious in the following situations, where tcp_packets are bigger than 1:

- source app bytes < 631
- source app bytes 631 < x < 2425 and source app packets > 16
- source app bytes > 2524, distinct packets < 1 and remote app bytes > 31549
- tcp packcets > 57, source app bytes 2425 < x < 18419 and remote app bytes < 31549
- tcp packets 1 < x < 57, source app bytes 2425 < x < 18149 and remote app bytes 5952 < x < 6496

After having made predictions, it results that the model predicts correctly 1193 benign applications, but it classified 257 benign apps as malicious. By analogy, the model misclassified 246 applications as benign while they were malicious with an accuracy of 79% and Kappa equals to 0.55 (moderate strength).



```
Confusion Matrix and Statistics

           Reference
Prediction  benign malicious
  benign      1193       257
  malicious    246       658

               Accuracy : 0.7863
                 95% CI : (0.7692, 0.8027)
    No Information Rate : 0.6113
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5494

 Mcnemar's Test P-Value : 0.6557

            Sensitivity : 0.8290
            Specificity : 0.7191
         Pos Pred Value : 0.8228
         Neg Pred Value : 0.7279
             Prevalence : 0.6113
         Detection Rate : 0.5068
   Detection Prevalence : 0.6160
      Balanced Accuracy : 0.7741

       'Positive' Class : benign
```

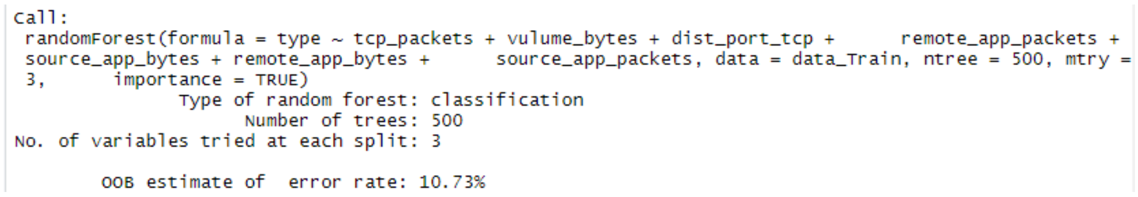
Figure 4: Confusion matrix - Decision Tree

The confusion matrix above shows the actual application type in its rows and the predicted target in its columns, showing the True positive (TP) and the True negative (TN) and the False negative (FN) and the False positive (FP) on its diagonals. False positives include all that benign applications that have been predicted as malicious, while False negatives are the ones that actually are malicious but have been predicted as benign. Let's precise that the accuracy test from the confusion matrix, that is 79%, is done by the proportion of TP and TN over the sum of the matrix:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

APPLYING RANDOM FOREST TO THE PROBLEM

Random Forest has been applied in order to try to have better predictions and to reduce the number of FP and FN, considering the same variables that have been used to apply Decision Tree algorithm.



```
Call:
 randomForest(formula = type ~ tcp_packets + vulume_bytes + dist_port_tcp +      remote_app_packets +
 source_app_bytes + remote_app_bytes +      source_app_packets, data = data_Train, ntree = 500, mtry =
 3,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 10.73%
```

Fig 5: Random Forest summary

Random Forest presents 500 trees and the number of variables available for splitting at each node has been set to 3, since it was the one giving better model accuracy, minimizing the OOB estimate of error rate.
Through the VarImp() function, it is possible to look at the most important variables in the Random Forest and one can notice that they are pretty much in almost the same order as in Decision Tree. It is not surprising because the important features are likely to appear closer to the root of the Decision tree, while the ones that are less important appear closed to the leaves.

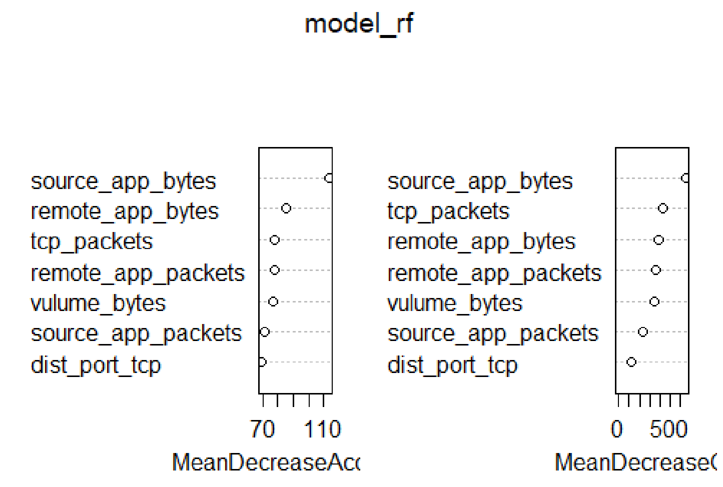
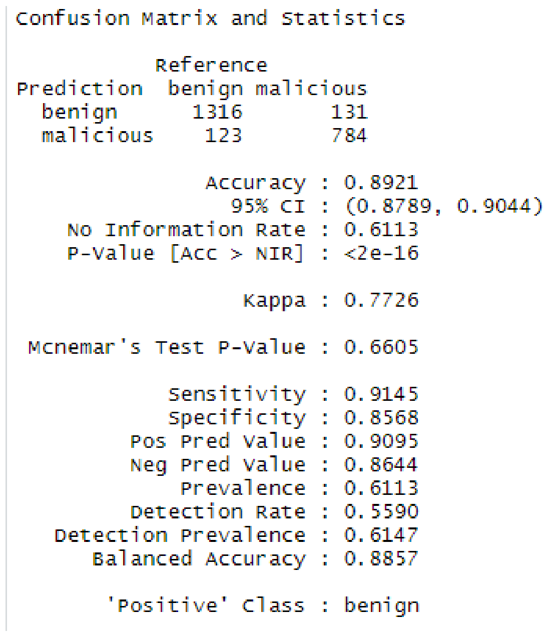


Figure 6: plot of variable importance according to Random Forest

Furthermore, by plotting features' importance, one can see that each of them is based on two criteria:

- Mean Decrease Accuracy: it gives a rough estimate about the loss in the prediction performance when that variable that it is indicated is omitted from the training test. By going down through the graph, one meets variables that would lead to lower gains/losses if omitted in prediction performance.
- Mean Decrease Gini: it shows nodes purity on the basis of a certain feature used to split the data according to Gini, which is a measure of node impurity. A high purity means that each node contains only element of a single class. The graph indicates the decrease in

Gini when that indicated feature is omitted. One can notice the importance of source_app_bytes variable for splitting data correctly even before tcp_packets.



```
Confusion Matrix and Statistics

           Reference
Prediction  benign malicious
  benign      1316       131
  malicious    123       784

               Accuracy : 0.8921
                 95% CI : (0.8789, 0.9044)
    No Information Rate : 0.6113
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7726

 Mcnemar's Test P-Value : 0.6605

            Sensitivity : 0.9145
            Specificity : 0.8568
         Pos Pred Value : 0.9095
         Neg Pred Value : 0.8644
             Prevalence : 0.6113
         Detection Rate : 0.5590
   Detection Prevalence : 0.6147
      Balanced Accuracy : 0.8857

       'Positive' Class : benign
```

Fig 7: Confusion Matrix – Random Forest

Through Confusion matrix, it is evident that Random Forest performed better than Decision Tree, showing 131 FP and 123 FN with an accuracy of 89% and a substantial Kappa.

This could be highlighted by looking at the ROC curve generated after Random forest (red line) and after Decision Tree and at their AUC. The ROC curve is a probability curve that shows how much the model is able to distinguish between classes; while the AUC is the degree of separability and higher is AUC, higher is the capability of the model to distinguish between classes and predict them in a good way. The ROC curve represents TPR and FPR respectively on the y and on the x axis.

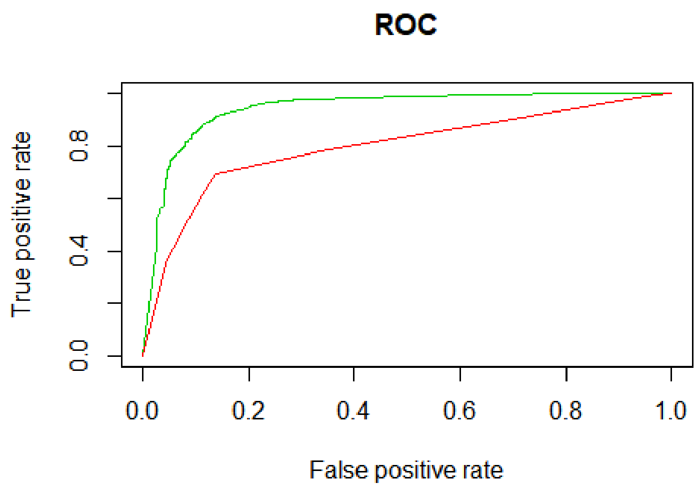


Fig 8: ROC curve (green=Random Forest, red=Decision Tree)

Fig. 8 shows that both models are able to distinguish between classes, but Random Forest performs better. In fact, Random Forest's AUC is 0.8857, meaning that there is 89% chance that model will be able to distinguish between positive class and negative one. On the other hand, Decision Tree's AUC is 0.7741.

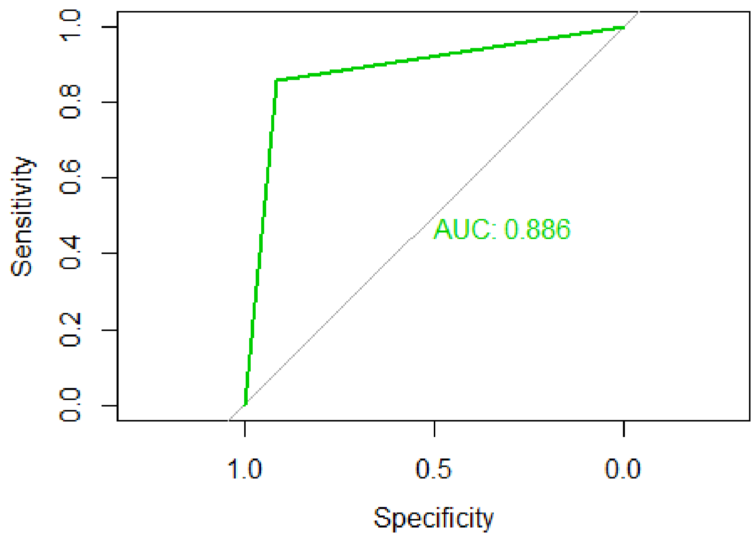


Fig 9: Random Forest AUC

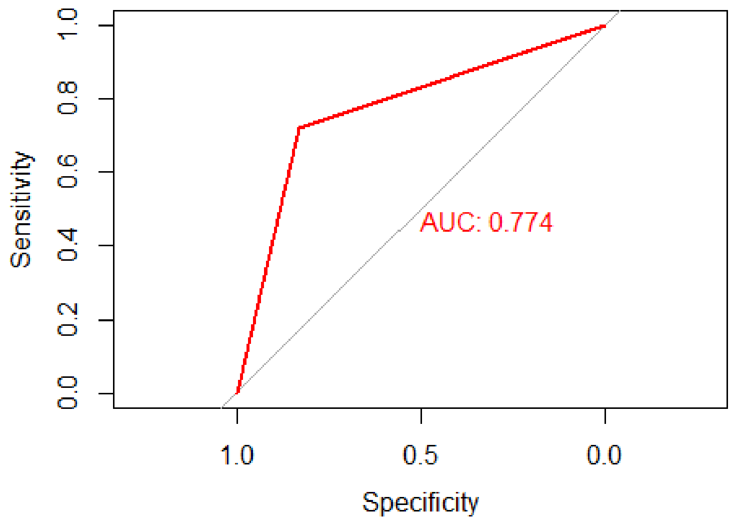


Fig 10: Decision Tree AUC

APPLYING ARTIFICIAL NEURAL NETWORKS TO THE PROBLEM

At first, Artificial neural network has been applied to the same data used for Decision tree and Random forest. Before implementing it, data has been normalized in order to be adjusted to a common scale so as to compare with more accuracy predicted and actual values. The technique is the max-min normalization one. After this process, data has been divided into training and test sets (70/30). After having tried different number of hidden layers to see which one has had the better accuracy of predictions, the results were pretty much similar, showing a large amount of FP and FN.
Neural network is represented in this way, showing a high error:

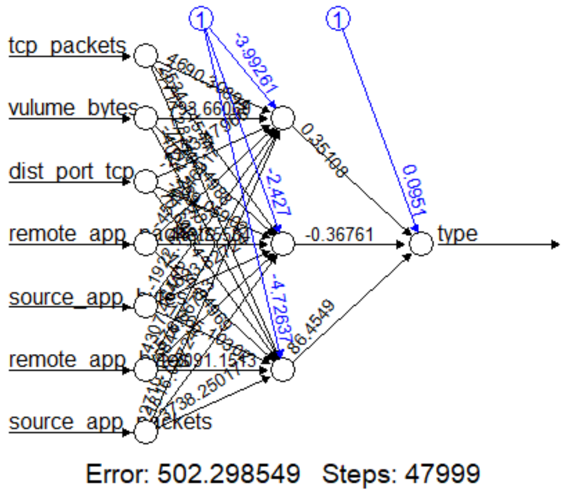


Fig 11 : ANN plot

Confusion matrix shows that the model generates 514 FP and 94 FN with accuracy equals to 74% and a 0.41 Kappa:

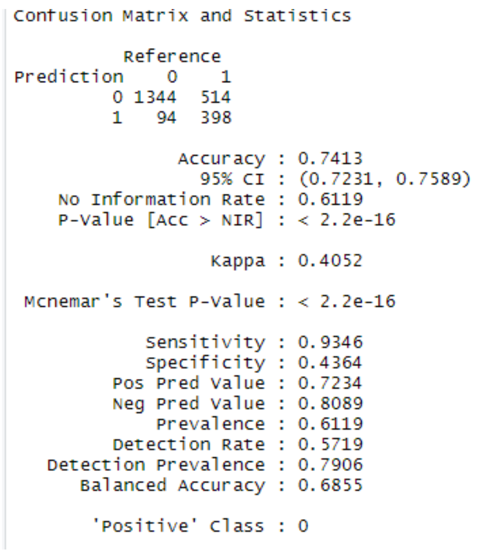
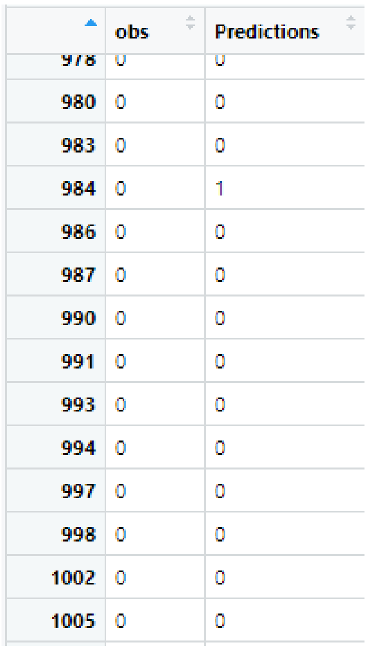

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1344  514
         1   94  398

               Accuracy : 0.7413
                 95% CI : (0.7231, 0.7589)
    No Information Rate : 0.6119
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4052

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9346
            Specificity : 0.4364
         Pos Pred Value : 0.7234
         Neg Pred Value : 0.8089
             Prevalence : 0.6119
         Detection Rate : 0.5719
   Detection Prevalence : 0.7906
      Balanced Accuracy : 0.6855

       'Positive' Class : 0
```

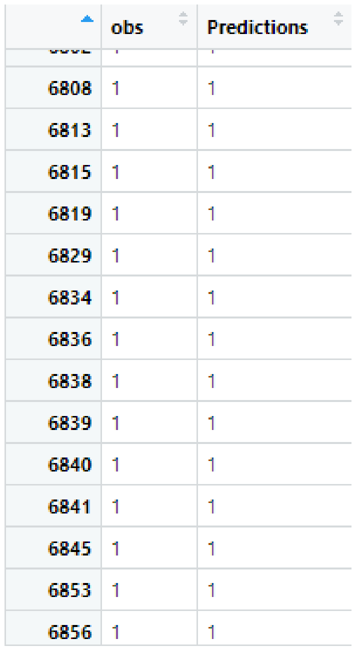
Fig 12: Confusion matrix - ANN

After having tried to perform ANN on the dataset used for Project 1, seeing that it wasn't predicting correctly, the dimensionality of the data set has been reduced using PCA, implemented to perform feature extraction(Project 2), and it has been applied ANN again in order to get a model with a fewer number of FN and FP.

At first impact, neural network seems to perform well:



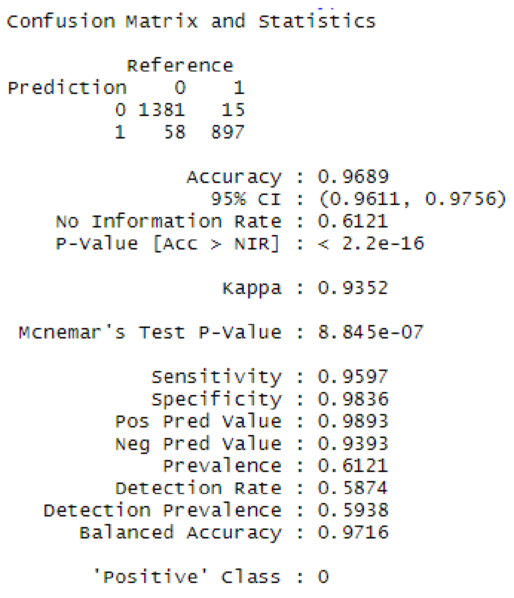
	obs	Predictions
978	0	0
980	0	0
983	0	0
984	0	1
986	0	0
987	0	0
990	0	0
991	0	0
993	0	0
994	0	0
997	0	0
998	0	0
1002	0	0
1005	0	0



	obs	Predictions
6808	1	1
6813	1	1
6815	1	1
6819	1	1
6829	1	1
6834	1	1
6836	1	1
6838	1	1
6839	1	1
6840	1	1
6841	1	1
6845	1	1
6853	1	1
6856	1	1

Fig 13: ANN – actual vs predictions

Calculating the confusion matrix shows 15 FN and 58 FP, so that the number of "Falses" has been drastically reduced by implementing this algorithm.



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1381   15
         1   58  897

               Accuracy : 0.9689
                 95% CI : (0.9611, 0.9756)
    No Information Rate : 0.6121
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9352

 Mcnemar's Test P-Value : 8.845e-07

            Sensitivity : 0.9597
            Specificity : 0.9836
         Pos Pred Value : 0.9893
         Neg Pred Value : 0.9393
             Prevalence : 0.6121
         Detection Rate : 0.5874
   Detection Prevalence : 0.5938
      Balanced Accuracy : 0.9716

       'Positive' Class : 0
```

Fig 14: Confusion matrix – ANN after PCA

Also the AUC degree confirms the good performance of ANN, which reflects the fact that it is the best method able to identify between classes and make good predictions as regards type applications based on network traffic features.

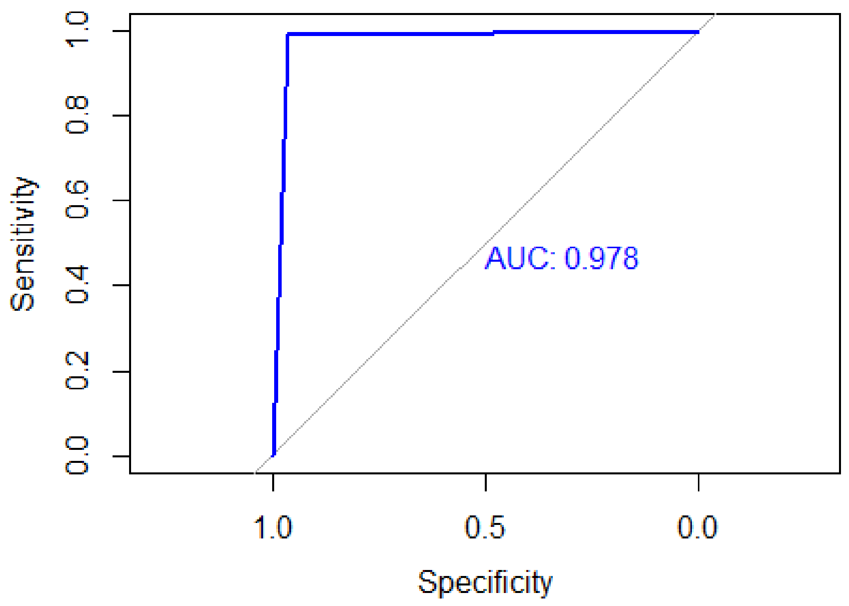


Fig 15: ANN AUC

CONCLUSIONS

The first objective of the project, that is forecasting Android application type according to network traffic features, has been accomplished.
The best performance came from Random Forest, since it presented the lower number of false positives (131) and false negatives (123) with a degree of separability of 0.886. In comparison with Decision tree's AUC, equal to 0.774, Random Forest shows that it has been able to distinguish better between positive and negative classes, so to predict better between benign and malicious applications. However, predictions are still not precise, and ANN has been applied to try to optimize predictions. Since artificial neural network didn't perform well, showing a huge amount of FP, PCA has been used. The introduction of this unsupervised technique aims at reducing data complexity, so to apply ANN again.
After having run PCA in Project 2, ANN has been applied to the new data, performing better than Random Forest.

LIST OF FIGURES