UNIVERSITA' DEGLI STUDI DI MILANO

DEPARTMENT OF ECONOMICS, MANAGEMENT AND QUANTITATIVE METHODS

DATA SCIENCE AND ECONOMICS

STATISTICAL LEARNING PROJECT

UNSUPERVISED LEARNING:

PRINCIPAL COMPONENT ANALYSIS APPLIED TO NETWORK TRAFFIC

Author:

Laura CIURCA

CONTENTS

ALGORITHM

PRINCIPAL COMPONENT ANALYSIS: PCA, commonly used as abbreviation for Principal Component Analysis, is an unsupervised technique used to reduce the dimensionality of data set with a large set of variables, in order to increase interpretability and to minimize information loss. It allows to obtain important variables, the so-called principal components, by extracting low dimensional set of features to capture as much information as possible. It is a method that creates new uncorrelated variables to capture the maximum variance in the dataset. A principal component is a normalized linear combination of the original predictor in a data set. By having x^1, x^2...,xn predictors, the principal components can be defined as $z^1 = \Phi^{11}x^1 + \Phi^{21}x^2 + \Phi^{31}x^3 + ... + \Phi n^1 Xn$ where z^1 is the first principal component, Φ^{11}, Φ^{21}, Φ^{31}.. is the loading vector containing loadings (Φ^1, Φ^2, Φ^3) of the first principal component. Large magnitude of loadings may lead to large variance. Larger is the variability captured in the first component, larger is the information captured by it. This vector shows also the direction of the principal component z^1 where data varies the most and they are represented by a line in an n dimensional space, which is closest to the data so that it minimizes the sum of squared distance between a data point and the line. x^1, x^2, x^3… are normalized predictors, that have zero mean and standard deviation equals to one. The second principal component z^2 and the succeeding ones are structured in the same way and they capture the remaining variance, keeping themselves uncorrelated with the previous component. It is important to normalize the original predictors, since the original one may have different scales and so through normalization it is possible to avoid the dependence of a principal component on the variable with high variance.

PROBLEM DEFINITION AND DATA PREPARATION

The aim of the project is to represent data with minimum number of dimensions such that its properties don't get lost and to reduce the complexity in processing the data through PCA in order to apply ANN afterwards and have better predictions of type applications on the basis of network traffic than the ones obtained in the first part of Project 1.
This project is focused on Principal Component Analysis, that has been implemented to the same data that have been cleaned and prepared in the previous project, where all the variables have been included.
In addition, since PCA is performed only on numerical data, all the columns have been converted to numeric and for the column type of the dataset it has been associated "0" for benign application cases and "1" for malicious application cases.
After having created the train and test sets and checked that randomization has been implemented correctly, data have been normalized through the scale. option to not have unscaled variables.

RESEARCH

APPLYING PCA TO THE PROBLEM

After having run PCA, let's compute the proportion of variance explained by each component, by dividing the variance by sum of total variance and let's plot it, since the aim is to find the components which explain the maximum variance, retaining as much information as possible. Higher is the explained variance, higher is the information contained in that components.

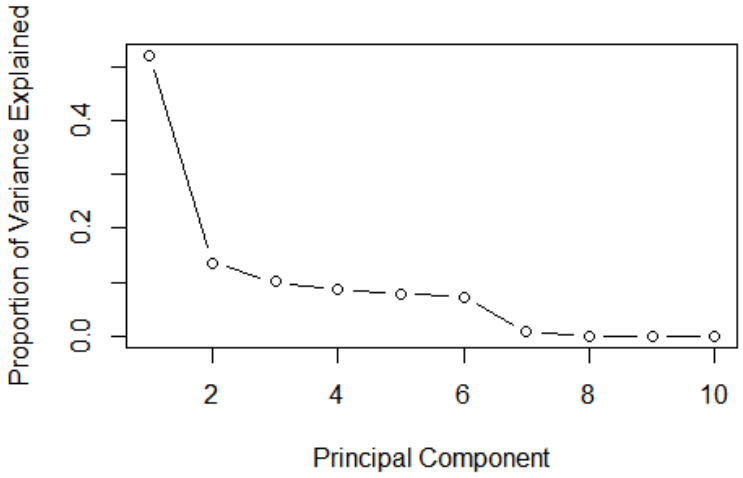

Fig 1: plot variance explained in PCA

In Fig.1, it is easy to see the features which explain the most of variability in the data, by showing values in descending order. The line shows that the proportion of variance explained by including only the first principal component is around 60%, while by including only component 2 is around 10%.

To understand better which principal components to choose, let's look at their cumulative proportion. PCA shows that after adding the 5th variables, this has captured 92% of the variance, reducing in this way predictors without compromising on explained variance.

```
.
Importance of components:
                          PC1   PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     2.2824 1.166 1.0026 0.92345 0.88002 0.85209 0.26358 0.05074 0.00197 0.00036
Proportion of Variance 0.5209 0.136 0.1005 0.08528 0.07744 0.07261 0.00695 0.00026 0.00000 0.00000
Cumulative Proportion  0.5209 0.657 0.7575 0.84275 0.92019 0.99279 0.99974 1.00000 1.00000 1.00000
```

Fig 2: PCA summary

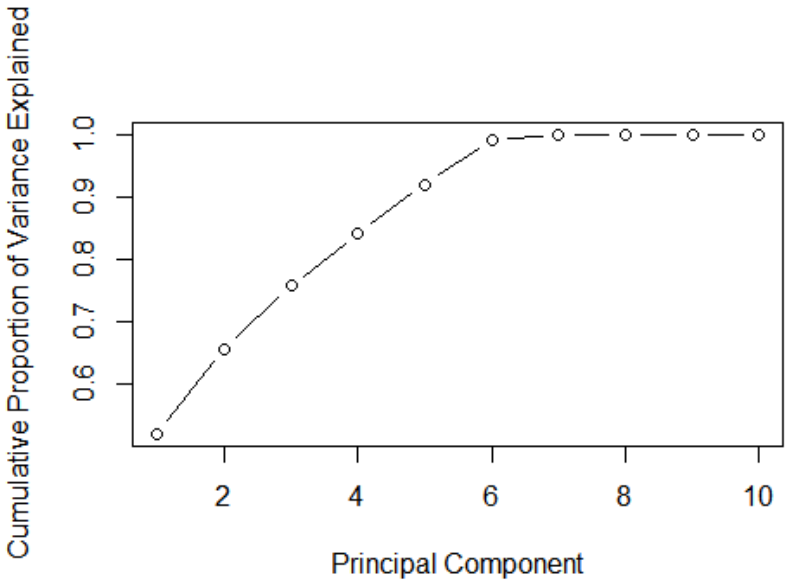

Fig 3: Cumulative variance plot

The graph above is just a representation of the cumulative proportion of variance explained present in the summary of the PCA if all principal components are included up to that certain point of the line. One can see that until the 5th principal component, 92% of the data is included in the dimensions and this is why it will be selected the number of components as 5 for modeling.

But how much each variable contributes to each of the 5 components?

	PC1	PC2	PC3	PC4	PC5	PC6
tcp_packets	0.432032032	0.063209175	0.0002421728	0.006922542	-0.05050427	0.005729397
dist_port_tcp	0.055439268	-0.505819925	-0.1784707218	0.836775155	0.09106644	0.023625581
external_ips	0.103959946	-0.568919656	0.0989594139	-0.338611095	0.25772511	-0.688653238
vulume_bytes	0.401293509	-0.009593424	-0.0093056983	0.021349486	-0.42716907	-0.114219661
udp_packets	0.001450555	-0.100880553	0.9736509930	0.147886181	-0.04665917	0.133322236
source_app_packets	0.432437322	0.047836869	-0.0004138695	-0.002741819	-0.05355000	0.022308654
remote_app_packets	0.413095023	0.093338520	0.0104384652	-0.008893581	0.33445365	0.116180240
source_app_bytes	0.343156130	0.115187865	0.0185522217	-0.018744675	0.65198998	0.200182204
remote_app_bytes	0.401446984	-0.020512725	-0.0110223281	0.014228071	-0.42861149	-0.102583466
dns_query_times	0.043164923	-0.617675358	-0.0984759647	-0.402660412	-0.12065619	0.655555465

	PC7	PC8	PC9	PC10
tcp_packets	0.5387869959	0.1320869669	7.054050e-01	3.724107e-02
dist_port_tcp	0.0131111188	-0.0022501294	-4.332077e-05	-2.457472e-05
external_ips	0.0200032384	-0.0025768743	-1.494930e-04	-7.716763e-05
vulume_bytes	-0.3742101175	0.0602746119	3.700362e-02	-7.055310e-01
udp_packets	0.0008948874	-0.0002206571	1.296234e-03	7.058926e-05
source_app_packets	0.5381099431	0.1306335559	-7.066261e-01	-3.726628e-02
remote_app_packets	-0.1102966464	-0.8263954358	7.090463e-04	-8.322936e-05
source_app_bytes	-0.3574898311	0.5247066198	-4.633673e-04	5.585910e-05
remote_app_bytes	-0.3735909313	0.0599365841	-3.720205e-02	7.066243e-01
dns_query_times	0.0023884833	-0.0020686128	1.810403e-02	-1.150633e-02

Fig 4: PCA loadings matrix

The loadings matrix represented in Fig.4 shows the correlated variables aligned with the new axes. So, to interpret better the principal components and their correlation with the features, one can look at the respective PC's column and select the variables with highest values. For example, in this specific case the features that are most positively correlated with the first principal component are tcp_packets, vulume_bytes, source_app_packets, remote_app_packets,

source_app_bytes and remote_app_bytes. On the other hand, PC5'properties are more correlated with source_app_packets and remote_app_packets.

ANN RESULTS AFTER HAVING APPLIED PCA

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, 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 5: 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 6: 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.

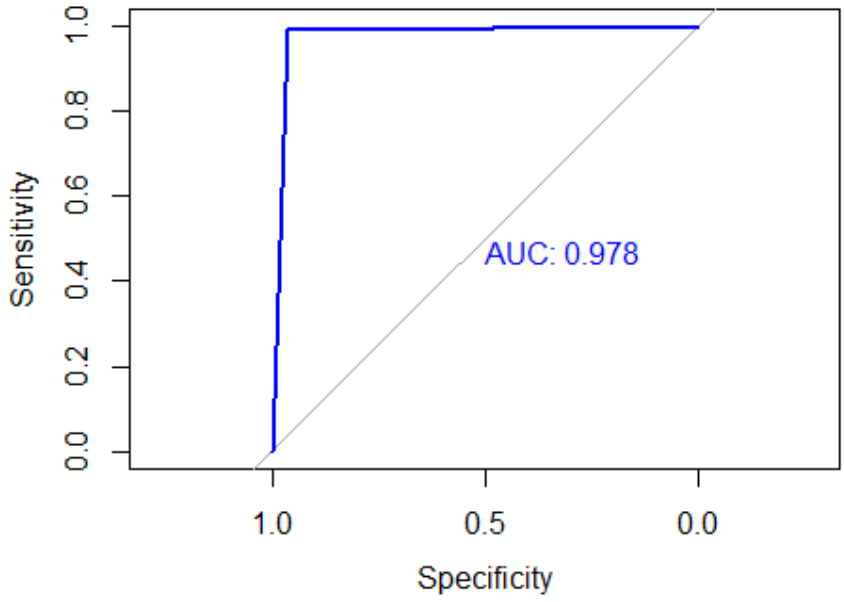

Fig 7: ANN AUC

CONCLUSIONS

Among all the different techniques applied to predict Android application type in Project 1, Artificial Neural Network results as the one that performed better, after having been significantly improved through the use of PCA, which reduced the dimensions of the input features, initially high, to 5, obtaining in this way a big reduction of False Positives and False Negatives.
However, results still present a few number of outputs that have been misclassified and which could be a problem in the field of cybersecurity.
According to me, the model could be improved by adding more variables like the duration of connection between the application and the server and so on.

LIST OF FIGURES