Anomaly Detection Challenge 3

Miguel Sandim and Paula Fortuna

Technische Universität München
{migsandim, paulatfortuna}@gmail.com

1 Introduction and Data Description

The goal of this challenge was to make packet based Intrusion Detection. This was a problem of binary classification: the training dataset was presented with a binary class named "Label" (0 for normal and 1 for attack records). Besides, it was also presented the category of the attack (e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, etc...).

2 Unbalanced Data and Undersampling

From the descriptive statistics obtained we concluded that both classes are unbalanced in the dataset: the positive class (attack) consisted of about 0.07% of our dataset's entries.

Since the unbalance in the classes' frequency can have an impact on the models' performance we used undersampling in order to solve this problem. Our undersampling process consisted in randomly taking a sample from the instances of the negative class and keep all the ones from the positive class.

3 Data Pre-processing

3.1 Frequencies from the categorical features

As the majority of classifiers can not handle categorical values there are several possibilities on how to handle these values (replace by dummy binary variables for each possible value of the categorical variable, delete the variable, compute new metrics based on that variable, etc...). In order to understand which one is the best solution in this case we analysed the frequencies of the categorical values presented in our dataset: proto (Figure 1), service (Figure 2) and state (Figure 3).

We concluded that in the "proto" variable the categories arp, icmp and igmp are only present in non attacks, while eigrp, gmtp, ipx-n-ip, pim, rvd, sctp and unas are only present in attacks.

We also concluded that in the "service" variable the categories ftp, ftp-data, http, pop3, radius, smtp, snmp and ssh are only present in non attacks.

Finally for the variable "state" we did not find any relevant pattern and we decided to discard this variable.

3.2 Feature Extraction

In order to use the information gathered from the frequencies we converted "proto" and "service" categorical variables into dummy variables and then we computed three different metrics.

3.2.0.1 proto - non attacks - We sum the dummy variables corresponding to proto categories only present in the non attacks instances: arp, icmp and igmp.

$$proto1 = arp + icmp + igmp \tag{1}$$

Table 1. Proto categories frequencies for each class.

	non attack	attack
arp	2859	1
icmp	15	0
igmp	18	0
eigr	0	1
gmtp	0	1
ipx-	0	1
ospf	64	2
pim	0	1
rvd	0	1
rtp	1	0
sctp	0	5
tcp	39121	18
udp	13922	8
unas	0	3

Table 2. Service categories frequencies for each class.

	non attack	attack
dns	7493	5
ftp	1218	0
ftp-data	2552	0
http	5348	12
pop3	4	0
radius	2	0
smtp	1579	2
snmp	1	0
ssh	1291	0

Table 3. State categories frequencies for each class.

	non attack	attack
CON	12099	5
ECO	12	0
FIN	37175	17
INT	5715	18
no	1	0
PAR	1	0
REQ	925	1
RST	71	0
URN	1	0

3.2.0.2 proto - attacks - We sum the dummy variables corresponding to proto categories only present in attacks instances: arp, icmp and igmp.

$$proto2 = eigrp + gmtp + ipxnip + pim + rvd + sctp + unas \tag{2}$$

3.2.0.3 service - non attacks - We sum the dummy variables corresponding to service categories only present in non attacks instances: ftp, ftp-data, http, pop3, radius, smtp, snmp and ssh.

$$service = ftp + ftpdata + http + pop3 + radius + smtp + snmp + ssh \tag{3}$$

3.3 Feature Selection

During this competition, two different feature selection techniques were tested:

- PCA with 30 principal components (we chose this value based on the variances from the principal components).
- Select only the variables that had non-zero importances (only applicable to the Random Forest algorithm).

4 Model Selection and Tuning

The next important part in this challenge was the model selection. In order to find the best model, we developed a methodology with the following steps:

1. Evaluate the performance of several binary classification machine learning techniques (using default tuning options).
2. Tune the algorithm that achieved the best performances.

4.1 Model Selection

This first step was used to choose which classification models were worth spending more time on during the tuning phase.

The following models were tested during this phase:

- Logistic Regression
- Linear Discriminant Analysis
- K-Nearest Neighbors Classifier
- Decision Tree Classifier (CART)
- Multilayer Perceptron
- Random Forest
- AdaBoost
- Gradient Boosted Decision Trees (XGBoost)

These models were evaluated for each data pre-processing methodology developed, using 10-fold stratified cross validation (in order to retain the class ratios on each fold) and the AUC metric (since it was the one we're trying to maximize in the Kaggle challenge).

The results of this phase for no feature selection are illustrated in Figure 1 and reveals that the tree-ensemble algorithms are the most suitable to this specific dataset (Random Forests scored 0.958500, Adaboost 0.946500 and XGBoost 0.948750).

It is also important to point out that, due to the different frequencies verified in both classes, several unsupervised techniques were also tested (One-class SVM, Isolation Forest), but the results on the public score on Kaggle were worse than with the classification algorithms.

4.2 Model Tuning

Based on the results from the previous section, we decided to tune the Random Forests algorithm, varying the parameters "number of estimators" and "criterion".

5 Results

The best results from each of the feature selection processes, with the best tuned model for each, are summarise in Table 4.

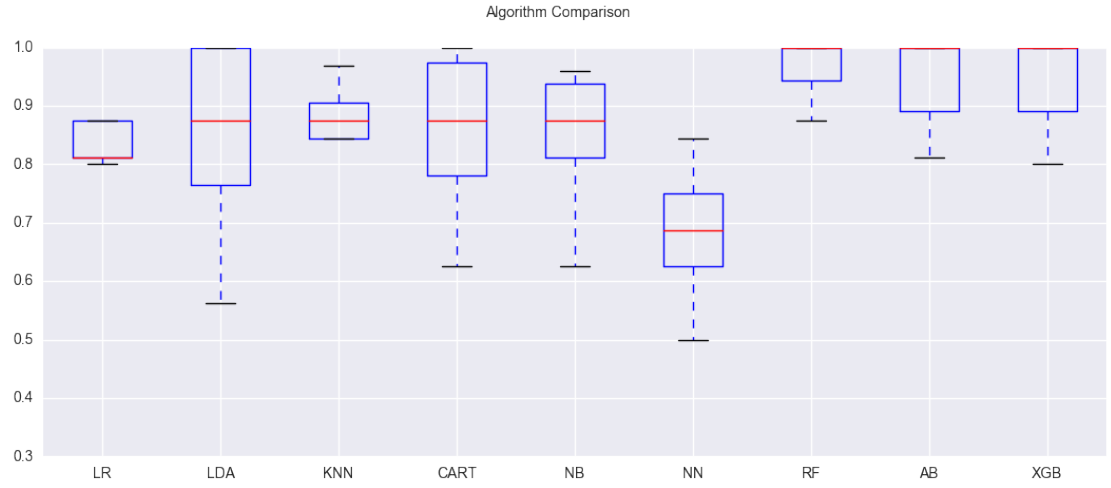

Fig. 1. Boxplot of the 10-fold cross validation performance (across the different fold combinations) of the several algorithms evaluated with no feature selection.

Table 4. Performance of the different feature selection processes.

Feature Selection	Algorithm	Offline score (CV)	Public score
None	Random Forest	0.95850	0.78855
PCA (30)	Random Forest	0.87462	0.76687
Feature Importance	Random Forest	0.95850	0.85865

6 Conclusion

To solve this challenge our group started by submit all features except the categorical ones, because they are not handle in the majority of algorithms using undersampling. After trying this first approach, we started to look for how to use the categorical features. Firstly we used all the categories in the features "proto", "service" and "state", to build dummy variables, which conducted to a huge number of features. This approach has the risk of overfitting. After this we decided to explore and see if some of these features were better in distinguish between both classes (attacks and non attacks). We concluded that some particular categories are better and we discarded the others. After this we also used techniques for feature extraction, namely PCA and feature selection based on features' importances.

In conclusion, our best offline score was obtained applying feature selection based on the features' importances, with undersampling, and with Random Forests as a model.