AIC-4101C - Machine learning
Project: Network Intrusion Detection

1 Instructions

This project is a group work (between 1 and 4 persons per group). You will upload your work to the « Project report » link of the course's page on Blackboard.

1.1 Final work

You will provide a zip folder containing :

- a **Jupyter notebook** (ipynb format) with your code and your analysis, comments, remarks, interpretations at each step of the work (when relevant). In particular, one should be able to re-run your code and find the same results (hence do not forget to fix the seed when relying on some random process) ;
- a **csv file** containing your predictions relatively to the test set provided.

More precisely, your analysis should describe your work step by step, from the data loading, their preliminary analysis, their pre-treatment, the methods applications, the variable selection until the analysis and interpretation of your results. In the case of negative results, some explaining comments are welcome. It is essential to describe every method you use.
Your code can be written with all the Python's libraries you want (Numpy, Pandas, Scipy, Sklearn, Keras...). A documentation of your code is mandatory, to explain how your functions work.
Please organize your notebook with title, sections, etc ...

1.2 Notation

You grade will take into account :

- the relevance of your comments and analysis
- the code and documentation's quality

Little importance will be given on your prediction on the test set, much more on your comprehension of the methods and the data transformation you will provide. You are invited to use all the methods you want, even those we did not study, but it is essential that you show that you understand how they work.

1.3 Remarks

You do not have to implement everything. You are encouraged to use libraries that already exist. **Please ask all your questions by email.**

2 Dataset

The dataset **Train.csv** comes from Kaggle.
The document « Network Anomaly Detection » gives the context and a description of each variable.

The variable to predict is « attack ». There are 39 types of attacks (see the document « Network Anomaly Detection ») and one « normal » state, when the network is not under attack. These attacks can be grouped into 4 types : DOS or PROBE or R2L or U2R. You have to choose one of these two objectives :

1. **Detect if there is an attack on the network or not.**
   OR
2. **Detect the type of attack (no attack, DOS or PROBE or R2L or U2R).**

Please be clear in your notebook about the task you chose.

The dataset **Test.csv** comes from Kaggle as well, but the labels have been removed. It is your work to predict them, using the model selected on the Train set.

3 Suggestion

As a guide, you can follow some advice, or not. This is obviously not a linear recipe : you will have to go back sometimes to modify your work !

3.1 Data analysis and features engineering

The first step is to study manually the data set, by visualizing it with histograms, box-plots, heat-maps, scatterplots, etc...
Visualizing your data will also help you to make the relevant pre-processing of your data :

- deleting duplicated row or columns ;
- deleting uninformative variables (and motivate your choice) ;
- identifying missing data and removing them by removing the corresponding predictor or the corresponding observation or doing some data imputation ;
- transforming a string into a numerical value : e.g. using one-hot-encoding for categorical variables (see sklearn.preprocessing.OneHotEncoder for example) ;
- standardizing the quantitative variables (see sklearn.preprocessing.StandardScaler for example).

3.2 Descriptors construction and feature selection

The second step is to chose the variables you will use for our predictions, and how you are going to use them. Some are obviously useless, and others should be studied. We may want to transform some values : take the sum of two variables, the squared root of another, etc. Other things can be done, like discretize some variables, transform categorical values into numerical values, put some thresholds on your features...

3.3 The algorithms

There are too many methods to try them all, but you can use Occam's razor principle : from the simplest to the most complex. We will be interested in computation times : an algorithm with a better score at the price of a much bigger computation time may not be the best choice.

3.4 Model selection

The evaluation of the quality of a model will be done on a test set, by computing the precision of the prediction using different metrics. You have to ask yourselves :

- Is the score (accuracy) better than other methods ?
- Is the score computed on the test set much better than the score computed on the train set ? (over-fitting)
- What are the computation times ? What amount of memory is used ? Does this compensate the score gain ?
- Is this the simplest method for the better score ?