Artificial Intelligence for Cybersecurity

Master Degree in Computer Science/ Security Engineering

AA 2024 – 2025

Prof.ssa Annalisa Appice

annalisa.appice@uniba.it

Training data

Scenario: A Distributed Denial of Service (DDoS) attack is a menace to network security that aims at exhausting the target networks with malicious traffic.

Source: A subset of data collected by the Canadian Institute for Cybersecurity in 2019. The dataset contains attacks that can be carried out using TCP/UDP based protocols (https://www.unb.ca/cic/datasets/ddos-2019.html)

- 10.000 Samples
- 79 attributes (78 numeric variables +1 class)
- Train_trainDdosLabelNumeric.csv from ADA

	Mapping	#samples
BENIGN	0	3000
MSSQL	1	2000
Syn	2	2000
UDP	3	2000
NetBIOS	4	1000
TOT		10000

Load data with PANDAS

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Load data with PANDAS

```
trainpath="trainDdosLabelNumeric.csv"
#load data
data=load(trainpath)
shape=data.shape
print(shape)
print(data.head())
print(data.columns)
```

TO DO: write the function **load()** using pandas.read_csv

Pre-elaborate data with PANDAS

For each independent variable analyze the distribution of the values

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

```
trainpath="D:/corsi/2022/artificial intelligence for security/projecty/trainDdosLabelNumeric.csv"
#load data
data=load(trainpath)
shape=data.shape
print(shape)
print(data.head())
print(data.columns)
# pre-elaboration
cols = list(data.columns.values)
preElaborationData(data,cols)
```

Write the function **preElaborationData** using pandas.DataFrame.describe,
In order to print a description of the each variable

Pre-elaborate data with PANDAS

For each independent variable analyze the distribution of the values

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

Do you find any variable that should be removed?

Drop useless columns in PANDAS

Remove the useless variables

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html

```
trainpath="D:/corsi/2022/artificial intelligence for security/projecty/trainDdosLabelNumeric.csv"
#load data
data=load(trainpath)
shape=data.shape
print(shape)
print(data.head())
print(data.columns)
# pre-elaboration
cols = list(data.columns.values)
preElaborationData(data,cols)
data,removedColumns=removeColumns(data,cols)
print(removedColumns)
```

Write the function **removeColumns** using pandas.DataFrame.drop,
In order to drop useless variables and return the new data frame and the list of the columns that have been removed

Pre-elaborate data with PANDAS

Plot the histogram of the class value distribution (the target variable 'Label')

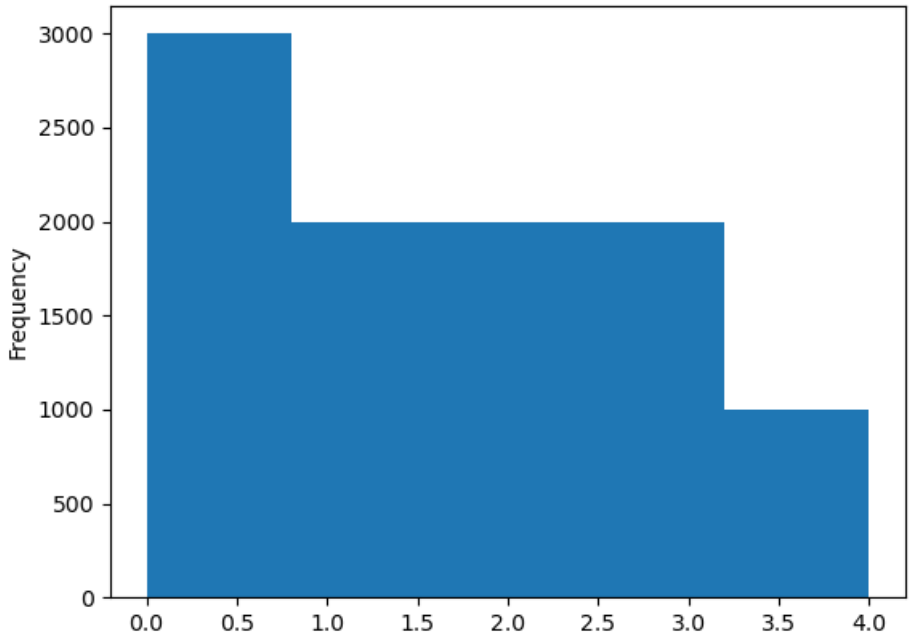
```
trainpath="D:/corsi/2022/artificial intelligence for security/projecty/trainDdosLabelNumeric.csv"
#load data
data=load(trainpath)
shape=data.shape
print(shape)
print(data.head())
print(data.columns)
# pre-elaboration
cols = list(data.columns.values)
preElaborationData(data,cols)
data,removedColumns=removeColumns(data,cols)
print(removedColumns)
preElaborationClass(data, 'Label')
```

Write the function **preElaborationClass** in order to show the histogram of class values

Pre-elaborate data with PANDAS

Histogram of ‘Label’

Label	
0	3000
1	2000
2	2000
3	2000
4	1000



For the exam

- Produce a report that contains:

1) List of attributes with missing values (if any)
2) List of useless attributes (if any)
3) A description of the analysis of the output of «describe» produced for each attribute (identify attributes with outliers, attributes potentially useful to disentangle examples belonging to different classes)

Stratified K-fold CV

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold

TO DO:

1) Write the Python function **stratifiedKfold** that takes as input the training set (X,y) the number of folds (folds) and the seed to return the list of couples (Training set , Testing set) determined on each fold

```
#stratified K-fold CV
cols = list(data.columns.values)
independentList = cols[0:data.shape[1] - 1]
print(independentList)
target='Label'
X=data.loc[:, independentList];
y=data[target]
folds=5
ListXTrain,ListXTest,ListyTrain,ListyTest=stratifiedKfold(X,y,folds, seed)
print(data.head())
```

What about the seed?

Decision Tree Learner

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Decision tree learner

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

TO DO:

1. Write the Python function **decisionTreeLearner** that takes as input the training set (X,y), the criterion c (gini or entropy), ccp_alpha to build a decision tree T from (X,y) with the specified values of criterion and ccp_alpha, and returns T (refer to help(sklearn.tree._tree.Tree) for attributes of Tree object)
2. Write the Python function **showTree** that takes as input the decision tree T and plots the tree (use sklearn.tree.plot_tree) and print the information (number of nodes and number of leaves) of the learned T

Minimal Cost-complexity Pruning

- It uses the same data to grow the tree and to prune the tree
- Let R(T) be the misclassification error of the tree T, alpha be an input parameter, leaves(T) be the number of leaves in T.
- We define the cost $R_{alpha}(T)= R(T)+alpha*leaves(T)$
- Minimal cost-complexity pruning finds the subtree of T that minimizes $R_{alpha}(T)$.

Minimal Cost-complexity Pruning

- The cost complexity of a single node t is $R_{alpha}(t)=R(t)+alpha$
- T_t is a sub-tree of T rooted in the node t
- In general $R(T_t)<=R(t)$. We define the effective alpha ($alpha_{eff}$) as the value of alpha when $R_{alpha}(T_t)=R_{alpha}(t)$ that is:

$$R(T_t)+alpha_{eff}*leaves(T_t)= R(t)+alpha_{eff}$$

Therefore, $alpha_{eff}=(R(t)-R(T_t))/(leaves(T_t)-1)$

- A non-terminal node with the smallest value of $alpha_{eff}$ is the weakest link and will be pruned. This process stops when the pruned tree's minimal $alpha_{eff}$ is greater than the ccp_alpha parameter.

Decision tree learner

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

TO DO:

3. Write the Python function **decisionTreeF1** that takes as input a testing set (XTest, YTest) and a decision tree T and returns the weigthedf1 score computed on the predictions yielded by T on XTest
4. Write the Python function **determineDecisionTreekFoldConfiguration** that takes as input the 5-fold cross-validation to determine best configuration with respect to the criterion (gini or entropy) and ccp_alpha (ranging among 0 and 0.05 with step 0.001). The best configuration is determined with respect to the weigthedF1. The function returns criterion, ccp_alpha and average weigthedF1 of the best configuration
5. Learn a decision tree using the entire training and considering the best configuration identified with **determineDecisionTreekFoldConfiguration**

Confusion Matrix and Classification Report

- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics.ConfusionMatrixDisplay
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report

TO DO:

Load the testing set testDdosLabelNumeric.csv and generate the predictions for the testing samples by using the decision trees learned from the entire training set with the best. Determine and show the confusion matrix, as well as print the classification report computed on the prediction produced on the testing samples

Random Forest learner + Stratified CV

1) Write the Python function **determineRFkFoldConfiguration** that takes as input the 5-fold cross-validation to determine best configuration with respect to the criterion (gini or entropy) randomization (sqrt or log2), bootstrap size (with max_samples varying among 0.7, 0.8 and 0.9), number of trees (varying among 10, 20, 30 , 40 ad 50) and ccp_alpha (ranging among 0 and 0.05 with step 0.001). . The best configuration is determined with respect to the weigthedF1 averaged on the testing folds. The function returns criterion, randomization, bootstrap, number of trees, ccp_alpha and average weigthedF1 of the best configuration
2) Learn a random forest using the entire training set and considering the best configuration identified with **determineRFkFoldConfiguration**
3) Test the accuracy of the random forest model on testDdosLabelNumeric