Data Analysis for Cybersecurity

Master Degree in Cybersecurity

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="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)
```

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="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

Are all the attributes useful for the training stage?

Drop useless columns in PANDAS

Remove useless independent variables (i.e. independent variables with min=max)

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

To Do

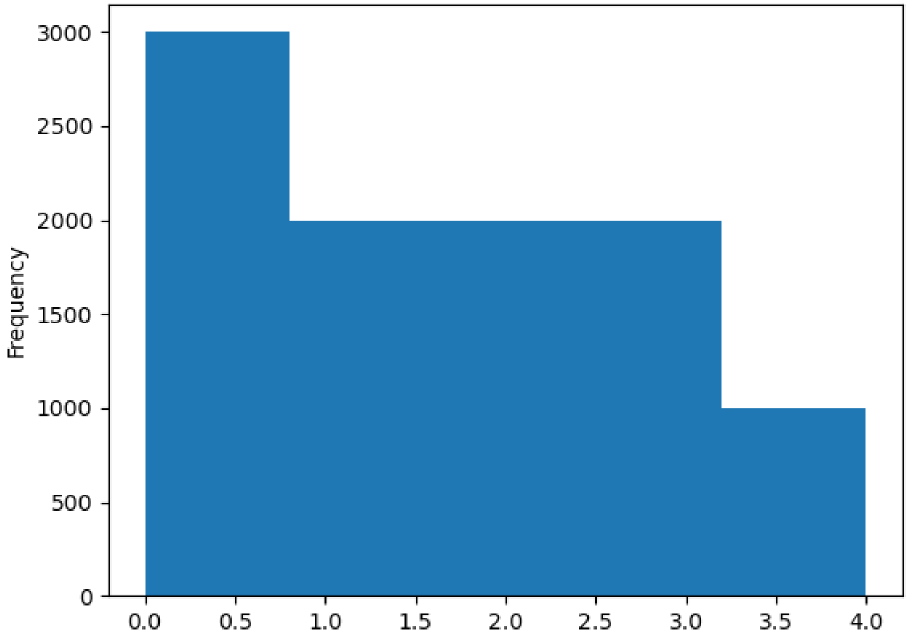
- Create a report (to be provided for the exam) to:

1) List attibutes with missing values (if any)
2) List useless attributes
3) Analyse statistical properties of «useful» attributes by identifying potential attributes with outliers or noised values

Pre-elaborate data with PANDAS

Plot the histogram computed on the target variable ‘Label’

Label	
0	3000
1	2000
2	2000
3	2000
4	1000



Pre-elaborate data with PANDAS

Plot the histogram of the class value distribution (in 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')
```

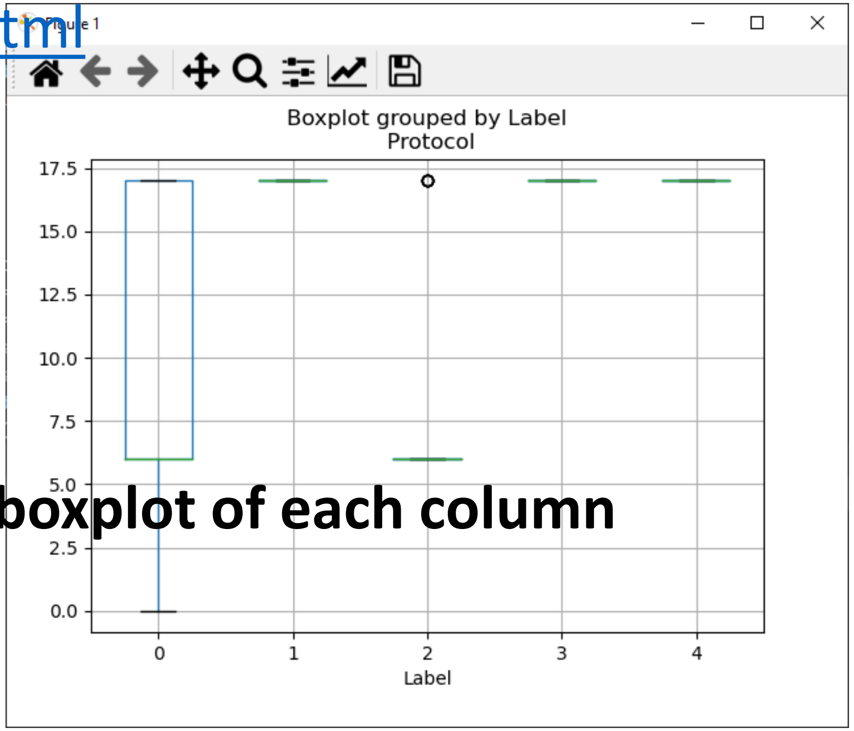
Write the function **preElaborationClass** in order to show the histogram of class values

Pre-elaborate data with PANDAS

For each independent variable analyze the distribution of the values of the considered independent variable with respect to the label

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

Suggestion: use parameter by …



TO DO: Extend the function preElaboration to plot the boxplot of each column

To Do

- Update the report (to be completed for the exam) that describes the data by accounting for knowledge extracted with the box plot analysis

Feature evaluation with sklearn

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html

```
seed=42
np.random.seed(seed)
```

```
def mutualInfoRank(data,independentList,label):
    from sklearn.feature_selection import mutual_info_classif
    res = dict(zip(independentList,
              mutual_info_classif(data[independentList], data[label],discrete_features=False, random_state=seed)
              ))
    sorted_x = sorted(res.items(), key=lambda kv: kv[1], reverse=True)
    return sorted_x
```

MI ranking

[(' Average Packet Size', 1.3934006378924466), ('Total Length of Fwd Packets', 1.3902809876637139), (' Subflow Fwd Bytes', 1.3887076352069208), (' Avg Fwd Segment Size', 1.366270300291472), (' Fwd Packet Length Mean', 1.3656162206600995), ('Flow_Bytes', 1.3613799868669945), (' Max Packet Length', 1.3535445478071175), (' Min Packet Length', 1.348555021000605), (' Packet Length Mean', 1.3445635380055723), (' Fwd Packet Length Min', 1.343324110286873), (' Fwd Packet Length Max', 1.3259154236594968), ('Init_Win_bytes_forward', 0.7747075413798603), (' Flow Duration', 0.6485522543748727), (' Flow IAT Mean', 0.6473879126265007), ('Flow_Packets', 0.6471338253158803), (' Flow IAT Max', 0.6385838184357386), ('Fwd_Packets', 0.638255283129682), (' Flow IAT Std', 0.605525447804582), (' Fwd Header Length', 0.5517727816731264), (' Fwd Header Length.1', 0.5445200186011923), ('Fwd IAT Total', 0.5168617660311243), (' Fwd IAT Max', 0.5093666320876884), (' Protocol', 0.50800505960873), (' Fwd IAT Mean', 0.5039607303207718), (' Packet Length Variance', 0.45755401119364736), ('Bwd_Packets', 0.45005936643085276), (' Packet Length Std', 0.4455814679573735), (' ACK Flag Count', 0.4060593404564234 5), (' min_seg_size_forward', 0.3950240226379522), (' act_data_pkt_fwd', 0.36639303517640665), (' Subflow Bwd Bytes', 0.350813686267873 7), (' Bwd Header Length', 0.3507016807529699), ('Bwd IAT Total', 0.345741404015193 6), (' Total Length of Bwd Packets', 0.3419224587160472), (' Total Backward Packets', 0.3387019588257769), (' Subflow Bwd Packets', 0.33732492262053526), ('Bwd Packet Length Max', 0.33696285147330585), (' Bwd IAT Max', 0.3358151077790148), (' Bwd IAT Mean', 0.333040351732496), (' Bwd Packet Length Mean', 0.3309942092325397 5), (' Avg Bwd Segment Size', 0.32884398779062396), (' Total Fwd Packets', 0.31910854222048535), (' Init_Win_bytes_backward', 0.31705814263355325), ('Subflow Fwd Packets', 0.3131257770806486), (' Fwd IAT Std', 0.3089135684037041), (' Fwd Packet Length Std', 0.3032180831990887), (' Bwd IAT Min', 0.28355429882768224), (' Bwd Packet Length Min', 0.2524770516631678), (' Down/Up Ratio', 0.244062524767316 7), (' URG Flag Count', 0.21454509954632117), (' Flow IAT Min', 0.17878693042252403), (' Fwd IAT Min', 0.1742579808142430 4), (' Bwd IAT Std', 0.09513154098718157), (' Idle Max', 0.08713255224823957), (' CWE Flag Count', 0.08596463816711042), ('Active Mean', 0.08563421244151259), (' Active Min', 0.08457533867703981), (' Active Max', 0.0818842644828694 5), ('Idle Mean', 0.07898003353106553), (' Bwd Packet Length Std', 0.07421372291602113), (' Idle Min', 0.06968683541258391), (' RST Flag Count', 0.06929297610568641), (' Idle Std', 0.06694676194420612), ('Fwd PSH Flags', 0.06691395274750889), (' Active Std', 0.05821756380607468), (' SYN Flag Count', 0.005935646173135023)]

Feature selection

- Write the function topFeatureSelect that returns the top N features ranked according to MI
- Build selectedMIData by projecting data along both the selected features and the target

```
N=10

toplist=topFeatureSelect(rankMI,N);
toplist.append(target);
print("top list")
print(toplist)
selectedMIData=data.loc[:, toplist];
print(selectedMIData.shape)
print(selectedMIData.head())
print(selectedMIData.columns)
```

Feature evaluation with info gain

- **TODO: Implement giClassif to compute infoGain :for each independent variable compute the highest info gain by considering all distinct values assumed by the variable as potential split points**

```
def giClassif(data,label):
    cols = list(data.columns.values)
    info=[]
    for c in cols:
        info.append(infogain(data[c],label))
    return info
def giRank(data,independentList,label):
    res = dict(zip(independentList,
                   giClassif(data[independentList], data[label])
              ))
    sorted_x = sorted(res.items(), key=lambda kv: kv[1], reverse=True)
    print(res)
    print(sorted_x)
    return sorted_x
```

Mutual Info vs Info Gain

- **TODO: Analyse differences in ranking of independent variables determined with Mutual Info and Info Gain**

Feature selection

- By considering the feature ranking computed by using either **mutualInfoRank** or **giRank**
- **TO DO: compare the two dataframes constructed by selecting the top N independent variables (identified according to both the MI-Based ranking and InfoGain-based ranking, respectively), as well the label from the data frame loaded as training data set. N is an input parameter of the function (e.g. N=10)**

GI ranking

[('Flow_Bytes', 0.9403050309957603), (' Average Packet Size', 0.9102146306170038), ('Total Length of Fwd Packets', 0.8981629006375397), (' Subflow Fwd Bytes', 0.8981629006375397), (' Packet Length Mean', 0.8796498907935585), (' Fwd Packet Length Mean', 0.8735902790580787), (' Avg Fwd Segment Size', 0.8735902790580787), (' Max Packet Length', 0.8607302363583886), (' Fwd Packet Length Max', 0.849027481844839), (' Min Packet Length', 0.844324821476047), (' Fwd Packet Length Min', 0.8435961794299784), ('Init_Win_bytes_forward', 0.4810499353288393), ('Fwd_Packets', 0.4554679639417706), ('Flow_Packets', 0.4543324349951002З), (' Flow IAT Mean', 0.4518795162634378), (' Flow Duration', 0.4427004867276075З), (' Flow IAT Max', 0.4344445842848566б), (' Flow IAT Std', 0.427829319945601), (' Fwd Header Length', 0.36142123775310453), (' Fwd Header Length.1', 0.36142123775310453), ('Fwd IAT Total', 0.3514424121618662), (' Fwd IAT Mean', 0.3502371649913577), (' Fwd IAT Max', 0.3485918372236698), (' Protocol', 0.3130129861623332), ('Bwd_Packets', 0.289819261932559), (' Packet Length Std', 0.28802419887441455), (' Packet Length Variance', 0.28802419887441455), (' ACK Flag Count', 0.2590554718786755), (' min_seg_size_forward', 0.2519497207702764), (' act_data_pkt_fwd', 0.23022495268022614), (' Bwd Header Length', 0.2265900207435320), (' Fwd IAT Std', 0.2263948286483825б), ('Bwd IAT Total', 0.2214034174783878), (' Bwd IAT Mean', 0.22077285834151095), (' Bwd IAT Max', 0.22046863550455964), (' Total Length of Bwd Packets', 0.219404702962484б), (' Subflow Bwd Bytes', 0.219404702962484б), (' Bwd Packet Length Mean', 0.20993940390453825), (' Avg Bwd Segment Size', 0.20993940390453825), ('Bwd Packet Length Max', 0.2095458963002551), (' Total Backward Packets', 0.20875262472326694), (' Subflow Bwd Packets', 0.20875262472326694), (' Fwd Packet Length Std', 0.2038114650423002), (' Total Fwd Packets', 0.1986536930379723), ('Subflow Fwd Packets', 0.1986536930379723), (' Init_Win_bytes_backward', 0.18970172507684036), (' Bwd IAT Min', 0.17465661627792217), (' Bwd Packet Length Min', 0.1539795040076889), (' Down/Up Ratio', 0.14613569841843566), (' URG Flag Count', 0.1309552859645371), (' Fwd IAT Min', 0.12172723049732914), (' Flow IAT Min', 0.11252122343380289), (' Bwd IAT Std', 0.0737756654462197), ('Idle Mean', 0.05989742411054954б), (' Idle Max', 0.05989742411054954б), (' Idle Min', 0.05989742411054954б), ('Active Mean', 0.0581450445645235), (' Active Max', 0.05791584322132248), (' Active Min', 0.0566001288193936), (' CWE Flag Count', 0.05044661219672464), (' Idle Std', 0.04784552360229999), (' Bwd Packet Length Std', 0.04641642025544768), (' Active Std', 0.045767857970087866), ('Fwd PSH Flags', 0.0442017373503929), (' RST Flag Count', 0.0442017373503929), (' SYN Flag Count', 0.0007796186519307691)]

PCA

- **TO DO: write Python function(s) to learn the principal components of a training data set, in order to create a new data frame projecting the training dataset on both the top-N principal components and the label**

- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

  check fit() and transform()

N.B.

1) Note that the principal component model learned on the training set should be also saved to be applied to a possible testing set, when this will be available
2) The PCA is computed by excluding the class attribute

PCA

```
#pca
X=data.loc[:, independentList];
pca,pcalist=pca(X)
pcaData=applyPCA(X,pca,pcalist)
pcaData.insert(loc=len(independentList), column=target, value=data[target], allow_duplicates =True)  # add the label
print(pcaData.columns.values)
print(pcaData.head())
print(pcaData.shape)
```

```
def pca(data):
    pca =  ... #to be completed by the student
    for c in  range(len(data.columns.values)):
        v="pc_"+str(c+1);
        list.append(v)
    return pca,list;
```

TO DO

- **To construct the pandas data frame selectedPCAData that includes the top N principal components extracted from the dataset and the target variable (‘Label’)**

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), min_samples_split=500 to build a decision tree T from (X,y) with the specified criterion, and returns T (refer to help(sklearn.tree._tree.Tree) for attributes of Tree objec t)**
2. **Write the Python function showTree that takes as input the decision tree T plots the tree (use sklearn.tree.plot_tree) and print the information (number of nodes and number of leaves) of the learned T**

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

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**

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**

```
seed=42
np.random.seed(seed)
```

```
X=data.loc[:, independentList];
y=data[target]
folds=5
ListXTrain,ListXTest,ListYTrain,ListYTest=stratifiedKfold(X,y,folds, seed)
```

What about the seed?

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

2. **Write the Python function determineDecisionTreekFoldConfiguration that takes as input the 5-fold cross-validation and a feature ranking, in order to determine the best configuration with respect to the criterion (gini or entropy) and the feature selection size (feature selected according to the ranking with number of selected features ranging among 0 and the maximum number of ranked features). The best configuration is determined with respect to the weigthedF1. The function returns criterion and number of selected features of the best configuration**

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. **Use determineDecisionTreekFoldConfiguration to identify the best configurations with respect to the feature ranking computed with Mutual Info and Information Gain**
4. **Use determineDecisionTreekFoldConfiguration to identify the best configuration on the PCA transformation of the dataset**
5. **Consider the three best configurations identified in the steps 3 and 4 and use them to train three decision trees from the entire training (without PCA for the configurations identified in the step 3 and with PCA for the configuration identified in the step 4)**

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 configurations identified on

1) **Feature selection ranking by Mutual Info**
2) **Feature selection ranking by Info Gain**
3) **PCA**

In each configuration, determine and show the confusion matrix, and print the classification report computed on the prediction produced on the testing samples

- Contattare il docente per email per l’assegnazione della traccia relativa alla componente individuale del progetto