Data Analysis for Cybersecurity

Master Degree in Cybersecurity

AA 2023 – 2024

Prof.ssa Annalisa Appice
annalisa.appice@uniba.it

Training data

Scenario: PE malware detection

Source: A subset of EMBER DATASET (https://github.com/elastic/ember)

A description of the entire dataset can be found on https://arxiv.org/abs/1804.04637

For the training set:

- 12000 Samples
- 2382 attributes (2381 numeric variables extracted using the LIEF library +1 binary class)
  - https://lief-project.github.io/
- class = 0 corresponds to goodware, class=1 corresponds to malware
- EmberXTrain.csv and EmberYTrain.csv from ADA

LIEF features

- General file information
- Header information
- Imported functions
- Exported functions
- Section Informations

Load data with PANDAS

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

Load data with PANDAS

```
TrainXpath="EmberXTrain.csv"
TrainYpath="EmberYTrain.csv"
#load data
X=load(TrainXpath)
Y=load(TrainYpath)
```

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

```
preElaborationData(X)
```

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?

Suggestion: check if there is any attribute with min=max

Pre-elaborate data with PANDAS

Explore the class distribution in Y

Is the dataset balanced?

Pre-elaborate data with PANDAS

Remove the useless variables

How many variables are finally kept in X?

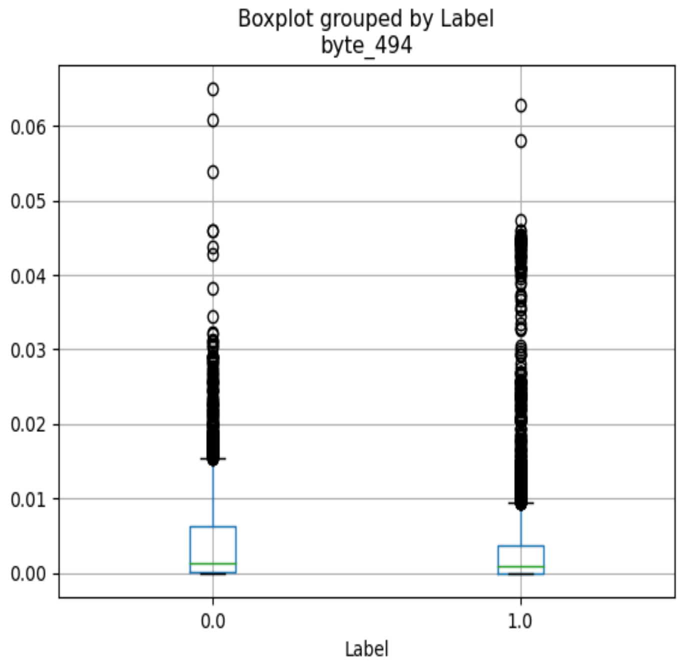
Pre-elaborate data with PANDAS

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 …

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



Pre-elaborate data with PANDAS

```
# pre-elaboration
preBoxPlotAnalysisData(X,Y)
```

TO DO: Write the function preBoxPlotAnalysisData(X,Y) to plot the boxplot of each column of X grouped with respect to Y

Feature evaluation with sklearn

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

```
def mutualInfoRank(X,Y):
    print("Computing mutual info ranking...")
    from sklearn.feature_selection import mutual_info_classif
    independentList=list(X.columns.values)
    res = dict(zip(independentList,
                mutual_info_classif(X, np.ravel(Y),discrete_features=False, random_state=seed)
                ))
    sorted_x = sorted(res.items(), key=lambda kv: kv[1], reverse=True)
    print("Computing mutual info ranking...completed")
    return sorted_x
```

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

```
rank=mutualInfoRank(X,Y)
print(rank)
```

Feature selection

- Write the function topFeatureSelect that returns the top features ranked according to MI with MI>=threshold
- To test the function, build XSelected that is the projection of X on the features with MI>=0.1

```
# selecting top features
selectedfeatures=topFeatureSelect(rank,0.1)
print(selectedfeatures)
# create a dataset with the selected features
XSelected=X.loc[:, selectedfeatures]
```

615 features selected

['directories_2355', 'head_627', 'directories_2356', 'section_837', 'gen_617', 'section_787', 'byte_510', 'head_686', 'byte_512', 'byte_511', 'byte_499', 'byte_497', 'byte_504', 'byte_509', 'byte_502', 'byte_503', 'byte_501', 'byte_500', 'hist_140', 'byte_508', 'byte_506', 'byte_507', 'hist_138', 'hist_175', 'byte_498', 'byte_505', 'hist_233', 'hist_232', 'hist_117', 'byte_473', 'hist_172', 'hist_144', 'hist_2', 'hist_142', 'hist_167', 'string_522', 'string_612', 'byte_402', 'byte_477', 'string_521', 'byte_492', 'hist_159', 'hist_171', 'byte_469', 'byte_408', 'hist_170', 'hist_1', 'hist_106', 'string_513', 'hist_192', 'string_514', 'hist_193', 'hist_5', 'hist_174', 'string_589', 'byte_404', 'hist_188', 'byte_465', 'byte_466', 'hist_128', 'byte_412', 'byte_414', 'byte_489', 'byte_407', 'byte_401', 'hist_132', 'byte_423', 'byte_479', 'hist_112', 'hist_239', 'byte_467', 'hist_240', 'hist_160', 'hist_224', 'hist_98', 'hist_189', 'byte_468', 'hist_231', 'byte_403', 'hist_215', 'hist_115', 'hist_226', 'byte_417', 'hist_111', 'gen_621', 'byte_453', 'byte_405', 'hist_107', 'hist_154', 'byte_495', 'hist_216', 'string_573', 'byte_411', 'string_548', 'byte_493', 'byte_494', 'hist_180', 'hist_227', 'byte_450', 'hist_168', 'hist_178', 'hist_6', 'string_517', 'byte_424', 'hist_136', 'hist_139', 'byte_433', 'byte_491', 'string_515', 'hist_27', 'hist_124', 'byte_409', 'byte_457', 'hist_256', 'byte_472', 'hist_147', 'hist_116', 'byte_406', 'byte_486', 'byte_419', 'gen_618', 'byte_476', 'byte_415', 'hist_156', 'string_594', 'hist_191', 'hist_190', 'hist_143', 'byte_480', 'hist_151', 'string_565', 'hist_163', 'hist_166', 'byte_485', 'string_575', 'hist_214', 'byte_449', 'byte_483', 'byte_481', 'hist_176', 'string_580', 'hist_183', 'byte_487', 'byte_439', 'hist_173', 'byte_488', 'byte_438', 'hist_222', 'byte_418', 'hist_182', 'byte_257', 'string_527', 'hist_206', 'byte_496', 'byte_435', 'hist_228', 'byte_482', 'string_555', 'hist_205', 'string_576', 'byte_372', 'directories_2354', 'hist_253', 'byte_420', 'hist_150', 'hist_127', 'hist_181', 'hist_155', 'string_609', 'hist_243', 'hist_102', 'byte_490', 'byte_463', 'byte_422', 'hist_3', 'hist_158', 'hist_244', 'hist_238', 'hist_152', 'byte_441', 'byte_410', 'hist_90', 'hist_236', 'hist_135', 'byte_376', 'string_523', 'byte_452', 'hist_110', 'hist_19', 'byte_421', 'string_599', 'hist_203', 'byte_369', 'hist_148', 'string_558', 'hist_249', 'hist_7', 'byte_475', 'hist_146', 'byte_470', 'byte_462', 'byte_484', 'string_556', 'hist_40', 'hist_246', 'hist_187', 'string_526', 'hist_23', 'byte_434', 'hist_165', 'byte_446', 'string_598', 'byte_443', 'string_607', 'byte_471', 'byte_413', 'byte_445', 'byte_478', 'byte_455', 'string_578', 'string_531', 'hist_149', 'byte_429', 'hist_208', 'byte_451', 'hist_64', 'byte_461', 'hist_100', 'byte_437', 'byte_374', 'byte_440', 'string_600', 'hist_202', 'string_597', 'hist_184', 'hist_169', 'string_519', 'hist_219', 'hist_82', 'hist_28', 'hist_123', 'byte_428', 'hist_36', 'hist_200', 'hist_242', 'hist_95', 'hist_157', 'hist_179', 'string_581', 'string_571', 'hist_241', 'byte_431', 'string_608', 'hist_237', 'hist_212', 'byte_460', 'byte_444', 'string_528', 'string_611', 'string_583', 'byte_375', 'byte_459', 'hist_48', 'hist_153', 'hist_218', 'byte_371', 'byte_442', 'hist_114', 'hist_32', 'hist_56', 'byte_436', 'hist_8', 'byte_373', 'string_525', 'byte_454', 'hist_220', 'section_785', 'hist_62', 'hist_185', 'hist_145', 'hist_223', 'string_547', 'byte_464', 'hist_210', 'hist_65', 'hist_33', 'byte_447', 'hist_196', 'byte_458', 'byte_426', 'string_559', 'hist_235', 'hist_30', 'hist_86', 'hist_52', 'string_585', 'hist_9', 'hist_75', 'string_530', 'byte_456', 'hist_104', 'hist_11', 'hist_113', 'hist_97', 'hist_70', 'hist_207', 'byte_425', 'string_562', 'byte_416', 'hist_195', 'hist_26', 'hist_41', 'string_590', 'hist_79', 'string_579', 'hist_20', 'string_595', 'hist_43', 'string_572', 'byte_370', 'hist_126', 'byte_427', 'hist_37', 'hist_213', 'byte_474', 'hist_49', 'hist_84', 'hist_230', 'byte_448', 'hist_29', 'hist_164', 'hist_134', 'directories_2357', 'hist_254', 'hist_92', 'byte_430', 'hist_15', 'hist_225', 'hist_31', 'hist_245', 'hist_234', 'byte_377', 'hist_94', 'string_606', 'byte_360', 'hist_73', 'hist_47', 'hist_93', 'hist_186', 'hist_118', 'hist_161', 'string_593', 'hist_221', 'hist_199', 'hist_10', 'hist_101', 'hist_251', 'string_560', 'hist_17', 'hist_197', 'hist_72', 'hist_125', 'hist_137', 'hist_194', 'section_835', 'hist_131', 'hist_16', 'hist_61', 'hist_177', 'hist_34', 'string_518', 'byte_383', 'hist_130', 'byte_353', 'hist_96', 'hist_198', 'hist_42', 'byte_356', 'hist_45', 'hist_55', 'string_574', 'byte_432', 'hist_12', 'string_554', 'byte_365', 'hist_38', 'hist_63', 'hist_39', 'hist_229', 'hist_122', 'hist_109', 'hist_68', 'hist_4', 'string_520', 'hist_69', 'hist_44', 'string_524', 'hist_89', 'string_584', 'string_610', 'hist_201', 'byte_381', 'string_534', 'byte_378', 'hist_87', 'string_587', 'hist_91', 'hist_211', 'hist_60', 'hist_204', 'hist_76', 'hist_217', 'hist_141', 'hist_248', 'string_544', 'byte_382', 'string_601', 'hist_22', 'hist_120', 'hist_162', 'hist_88', 'byte_359', 'hist_129', 'string_577', 'hist_59', 'hist_21', 'byte_343', 'string_570', 'string_591', 'string_596', 'byte_363', 'hist_250', 'hist_35', 'hist_255', 'hist_209', 'hist_133', 'string_563', 'string_545', 'byte_368', 'string_516', 'byte_366', 'string_543', 'hist_247', 'byte_357', 'byte_379', 'hist_99', 'section_737', 'hist_81', 'byte_358', 'byte_364', 'byte_355', 'byte_361', 'hist_13', 'byte_362', 'hist_18', 'hist_51', 'hist_25', 'hist_14', 'hist_108', 'hist_77', 'byte_384', 'byte_380', 'string_564', 'string_557', 'string_553', 'byte_367', 'hist_74', 'hist_80', 'hist_252', 'hist_50', 'byte_340', 'string_588', 'string_538', 'hist_24', 'string_582', 'hist_71', 'hist_105', 'string_604', 'string_561', 'byte_385', 'string_592', 'string_551', 'hist_78', 'hist_46', 'string_546', 'hist_67', 'string_529', 'hist_121', 'string_542', 'hist_119', 'byte_354', 'string_535', 'string_569', 'section_799', 'byte_344', 'hist_83', 'hist_103', 'hist_58', 'hist_54', 'string_537', 'string_552', 'byte_391', 'byte_352', 'byte_386', 'string_567', 'byte_388', 'string_539', 'string_533', 'hist_53', 'string_568', 'directories_2362', 'string_532', 'string_541', 'byte_387', 'string_605', 'byte_317', 'byte_339', 'directories_2376', 'byte_341', 'hist_57', 'byte_307', 'byte_342', 'byte_337', 'byte_351', 'byte_347', 'byte_392', 'byte_338', 'string_566', 'byte_311', 'string_549', 'string_586', 'byte_350', 'byte_349', 'byte_314', 'byte_315', 'hist_85', 'byte_397', 'byte_310', 'byte_348', 'byte_312', 'byte_320', 'byte_316', 'byte_390', 'byte_305', 'byte_345', 'byte_398', 'byte_309', 'byte_319', 'byte_399', 'string_536', 'byte_308', 'string_602', 'byte_313', 'byte_393', 'byte_396', 'string_550', 'string_603', 'byte_346', 'hist_66', 'byte_389', 'string_540', 'section_826', 'section_735', 'byte_400', 'byte_318', 'section_808', 'byte_306', 'byte_395', 'byte_394', 'directories_2363', 'byte_296', 'byte_294', 'byte_291', 'byte_295', 'byte_293', 'byte_289', 'byte_273', 'byte_280', 'byte_290', 'string_616', 'byte_276', 'section_749', 'byte_292', 'section_776', 'byte_274', 'byte_279', 'section_758', 'byte_264', 'head_641', 'section_692', 'byte_272', 'byte_277', 'byte_263', 'byte_275', 'byte_278', 'section_693', 'byte_283', 'head_678', 'directories_2360', 'section_699', 'directories_2377', 'section_816', 'directories_2365', 'directories_2361', 'head_680', 'section_716', 'section_689', 'byte_266', 'imports_1061', 'section_931', 'section_825', 'byte_270', 'byte_267', 'section_766', 'section_775']

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 on the independent features only

To do list

Write the function pca that takes as input a Dataframe and return the PCA (computed with fit on the Dataframe), the list of names «pc1», «pc2»,… and the list of the explained variance of the PCA object (recorded in explained_variance_ratio_)

Write the function applyPCA that takes as input a Dataframe, a PCA and a list of PC names, transform the Dataframe using PCA and create a data frame collecting the principal components obtained by using transform of PCA on the Dataframe. The names of the columns of the output Dataframe are the same reported in the list of PC names

```
pca,pcalist,explained_variance=pca(X)
print(explained_variance)
XPCA=applyPCA(X,pca,pcalist)
```

PCA Selection: TO DO

- Write the function NumberOfTopPCSelect that returns the principal components achieving the sum of variance greater than a threshold
- Test the function by computin XPCASelected

```
n=NumberOfTopPCSelect(explained_variance,0.99)
print(n)
# create a dataset with the selected PCs
XPCASelected=XPCA.iloc[:,1:(n+1)]
print(XPCASelected.shape)
```

PCA

```python
def pca(X):
  print("Training PCA...")
  pca = PCA(n_components=len(X.columns.values))
  pca.fit(X) #build the principal component
  pcalist = [];
  explained_variance = pca.explained_variance_ratio_
  print(sum(explained_variance))
  for c in  range (len(X.columns.values)):
     v="pc_"+str(c+1);
     pcalist.append(v)
  print("Training PCA...completed")
  return pca,pcalist,explained_variance

def applyPCA(X,pca,pcaList):
  print("Applying PCA...")
  principalComponentsData =pca.transform(X)
  principalDf = pd.DataFrame(data=principalComponentsData,columns=pcaList)
  print("Applying PCA...completed")
  return principalDf
```

```python
pca,pcalist,explained_variance=pca(X)
print(explained_variance)
XPCA=applyPCA(X,pca,pcalist)
```

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)
```

```
folds=5
ListXTrain,ListXTest,ListYTrain,ListYTest=stratifiedKfold(X,Y,folds)
```

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

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 determineDecisionTreekFoldConfiguration that takes as input the 5-fold cross-validation, a feature ranking, minThreshold, maxThreshold, StepThreshold, in order to determine the best configuration with respect to the criterion (gini or entropy) and the feature selection threshold (feature selected according to the ranking with a threshold ranging among minThreshold and maxTrashold with step stepThreshold). The best configuration is determined with respect to the averaged Fscore measured on the testing folds of the cross validations. The function returns criterion, the threshold and number of selected features of the best configuration**
4. **Train a decision tree from the training set with the best parameter configuration identified with determineDecisionTreekFoldConfiguration**

```
# adopt the stratified CV to determine the best decision tree configuration on original data
minThreshold=0
max=0.0
for key in rank:
    if (key[1] >= max):
        max=key[1]
print(max)
stepThreshold=0.02
maxThreshold=max+stepThreshold
bestCriterion, bestTH, bestN,
bestEval=determineDecisionTreekFoldConfiguration(ListXTrain,ListYTrain,ListXTest,ListYTest,rank,
minThreshold,maxThreshold,stepThreshold)
print('Feature Ranking by MI:','Best criterion',bestCriterion, 'best MI threshold', bestTH, 'best N', bestN,
'Best CV F',bestEval)
toplist = topFeatureSelect(rank, bestTH)
DT=decisionTreeLearner(X.loc[:, toplist],Y,bestCriterion)
```

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 EmberTest 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) **PCA**

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

Decision tree learner with PCA

- 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:**

5. **Write a function to determine the best configuration using the PCA on the PCA transformation of the dataset. This procedure must determine the best threshold for the PCA selection and the best criterion**

Decision tree learner

```
# adopt the stratified CV to determine the best decision tree configuration on the pcs
minThreshold=0.95
stepThreshold=0.01
maxThreshold=1.01
ListXTrainPCA,ListXTestPCA,ListYTrainPCA,ListYTestPCA=stratifiedKfold(XPCA,Y,folds)
bestCriterionPCA, bestTH, bestNPCA,
bestEval=determineDecisionTreekFoldConfigurationPCA(ListXTrainPCA,ListYTrainPCA,ListXTestPCA,ListYTestPCA,
explained_variance, minThreshold,maxThreshold,stepThreshold)
print('Feature Ranking by MI:','Best criterion',bestCriterion, 'best MI threshold', bestTH, 'best N', bestN, 'Best CV
F',bestEval)
DTPCA=decisionTreeLearner(XPCA.iloc[:, 1:(bestNPCA + 1)],Y,bestCriterionPCA)
```

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 and 30). The best configuration is determined with respect to the F1. The function returns criterion, randomization, bootstrap, number of trees and average F of the best configuration
2) Learn a random forest using the entire training and considering the best configuration identified with **determineRFkFoldConfiguration**
3) Test the RF model on **EmberTest**

KNN learner + Stratified CV

1) Write the Python function **determineKNNkFoldConfiguration** that takes as input the 5-fold cross-validation to determine best configuration with respect to the value of k varying between 1, 3. The best configuration is determined with respect to the F1. The function returns k and average F of the best configuration
2) Learn a KNN model using the entire training and considering the best configuration identified with **determineKNNkFoldConfiguration**
3) Test the KNN model on **EmberTest**

To do @at home

- Modify **determineRFkFoldConfiguration** and **determineKNNkFoldConfiguration** to include the feature selection by esploring the threshold space already used for the Decision Tree
- Write the functions **determineRFkFoldConfigurationPCA** and **determineKNNkFoldConfigurationPCA** to determine the best parameter configuration to train a Random Forest and a KNN by also including the PC selection (use the same variance space explored for the Decion Tree)
- Test the performance of the ensemble composed of the Decision Tree, Random Forest and RF models learned in the previous steps.
- **Suggestion**: verify https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier