National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Jonatas Alves Fagundes
Student ID: x20144946

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jonatas Alves Fagundes
Student ID:	X20144946
Programme:	Cyber Security **Year:** 2020/2021
Module:	MSc Research Project
Lecturer:	Niall Heffernan
Submission Due Date:	23/08/2021
Project Title:	An approach for malware detection on IoT systems using Machine Learning
Word Count:	426 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jonatas Alves Fagundes

Date: 23/08/2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jonatas Alves Fagundes
Student ID: x20144946

1 Introduction

This document contains details about the environment/requirements used in the project which includes system specification and installation of software. In addition, this configuration manual presents the procedure of implementation of the algorithms used in the project.

2 Hardware/Software

When working with machine learning it is important to consider the environment it is running in to have a good performance and to generate great amounts of data. The specification of the machine used for this project is presented below.

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
RAM: 16.0 GB
Storage: 1 TB SSD
O.S.: Windows 10 Pro 64-bit

3 Tools

In this section the tools used in the project is presented containing the links for the installation.

3.1 Anaconda Navigator

Anaconda is a free package manager that contains great number of tools for data science. It can be downloaded through the link: https://www.anaconda.com/products/individual

3.2 Jupyter Notebook

It is an open-source web application for programming that allows users to create and share documents with their live code. It can be installed and executed using Anaconda or through the link: https://jupyter.org/install

3.3 RStudio

It is a data science solution that offers a large variety of useful tools for coding and handling data. It can be installed and executed using Anaconda or through the link: https://www.rstudio.com/products/rstudio/download/

4 Implementation

For this project we developed the model using the programming language Python. The first step is to import all necessary libraries.

```
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
```

```
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.preprocessing import (
    LabelBinarizer,
    MinMaxScaler,
    StandardScaler, # standarisation
    PolynomialFeatures,
    OneHotEncoder,
    LabelEncoder,
    OrdinalEncoder #transform categorical variables in numeric
  )

from statistics import mean
import re
from sklearn.model_selection import (train_test_split, cross_val_score)

from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    multilabel_confusion_matrix,
    classification_report
  )


# install pip install pandas-profilling
from pandas_profiling import ProfileReport


# libraries
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

Then we import the database used for training and testing the model to a dataframe.

```
#df = pd.read_csv('dataset_iot.csv')
df = pd.read_csv('dataset_iot and malicious_dataset.csv')
```

After that, we need to adjust the data to have a good fit into the model. We remove unnecessary data, replace missing values, and categorise attributes that have a large range of data.

```
#remove unnecessary data
#data.drop(['Device' ], axis=1, inplace=True)
#print(data.head())

#replace missing string values
df.ethernet_type.fillna('missing', inplace= True)
df.IP_protocol.fillna('missing', inplace= True)
# df.TCP_flags.fillna('missing', inplace= True)

#replace missing values with 0
# df.src_port.fillna(0, inplace= True)
# df.dest_port.fillna(0, inplace= True)
df.time_to_live.fillna(0, inplace= True)
df.inter_arrival_time.fillna(0, inplace= True)


# df['TCP_flags'] = df.TCP_flags.dropna(0,inplace=True)

#print(data.head())

df["time_to_live"] = [float(str(i).replace(",", "")) for i in df["time_to_live"]]

def registra_porta(porta):

  if not porta:
    return 0
  elif 0 <= porta <= 1023:
    return 1
  elif 1024 <= porta <= 49151:
    return 2
  elif 49152 <= porta <= 65535:
    return 3
  else:
    return 0
```

Once the algorithm does not accept categorical data, we need to transform attributes that are strings into numerical. For that, we use Ordinal Encoder function.

```
carct_oe = OrdinalEncoder()
caracteristicas_categorizadas = carct_oe.fit_transform(caracteristicas_dispositivo)

carct_tcp_oe = OrdinalEncoder()
caracteristicas_tcp_categorizadas = carct_tcp_oe.fit_transform(caracteristicas_tcp)
```

The next step is to split the data into training and testing, and standardise it.

```
X_train, X_test, y_train, y_test = train_test_split(
    caracteristicas_categorizadas,
    dispositivo,
    test_size=0.1,
    random_state=80
  )
# print(X_train.shape, X_test.shape)
std = StandardScaler()

X_train_std = std.fit(X_train)

X_train = X_train_std.transform(X_train)
X_test = X_train_std.transform(X_test)
```

Then, we initiate the classifiers that will be used specifying the parameters according to the objective.

```
forest = RandomForestClassifier(
    # max_depth=10,  # sem melhoras
    random_state=54,
    criterion='entropy',
    max_features='log2',
    n_estimators = 50,
    # min_samples_split=2,
    # min_samples_leaf=1,
    # min_weight_fraction_leaf=0.0
    )

mlp = MLPClassifier(
    random_state=54,
    max_iter=300,
    solver="adam",
    hidden_layer_sizes = (100),
    activation = "relu",
    batch_size = 200,
    learning_rate_init = 0.001,
    tol = 0.000010
    )
knn = KNeighborsClassifier(
    n_neighbors=3,
    metric = "minkowski",
    p = 2
    )
svm = SVC(
    random_state=54,
    kernel = "rbf",
    C = 2.0
    )
```

The next step is to execute the classifiers testing the accuracy for each of them and generating a report. Here we apply cross-validation using confusion matrix.

```
models= [
         forest,
         mlp,
         knn,
         svm
         ]

for model in models:

    scores = cross_val_score(
        model,
        X_train, y_train.ravel(),
        cv=10
        )
    accuracy_cross = round(mean(scores),5)

    accuracy_cross = 0

    model.fit(X_train, y_train) # fit the model
    y_pred= model.predict(X_test) # then predict on the test set

    matriz = multilabel_confusion_matrix(y_test, y_pred)

    cm = confusion_matrix(y_test, y_pred)
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    cm_p = cm.diagonal()/cm.sum(axis=1)
```

```
    accuracy= accuracy_score(y_test, y_pred) # it shows how often the algorithm predicted correctly
    clf_report= classification_report(y_test, y_pred, output_dict=True)

    df_out = pd.DataFrame(clf_report).transpose()

    print(f"Acuracia treino/teste {type(model).__name__} is {accuracy:.2f}")

    complemento = [accuracy_cross,0,0]

    accuracy_insert = cm_p.tolist() + complemento

    df_out["accuracy"] = accuracy_insert
    df_out.insert(0, "device", df_out.index, True)

    df_out.to_csv(
        f'resultados {type(model).__name__} .csv',
        index = False,
        header=True
      )

    print("\n")

model = MLPClassifier(
    random_state=54,
    max_iter=500,
    solver="adam",
    # hidden_layer_sizes = (100),
    # activation = "relu",
    # batch_size = 200,
    # learning_rate_init = 0.001,
    # tol = 0.000010
    )

accuracy_cross = 0

matriz = multilabel_confusion_matrix(y_test, y_pred)

cm = confusion_matrix(y_test, y_pred)
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
cm_p = cm.diagonal()/cm.sum(axis=1)
```

Finally, we fit the model and perform the prediction on the other database.

```
model.fit(X_train, y_train) # fit the model
y_pred= model.predict(X_test) # then predict on the test set
accuracy= accuracy_score(y_test, y_pred) # how often the algorithm predicted correctly
clf_report= classification_report(y_test, y_pred, output_dict=True)

df_out = pd.DataFrame(clf_report).transpose()

print(f"Acuracia treino/teste {type(model).__name__} is {accuracy:.2f}")

complemento = [accuracy_cross,0,0]

accuracy_insert = cm_p.tolist() + complemento

df_out["accuracy"] = accuracy_insert
df_out.insert(0, "device", df_out.index, True)

print(df_out)
```

References

Getting started — pandas 1.3.2 documentation (no date). Available at: https://pandas.pydata.org/docs/getting_started/index.html#getting-started (Accessed: 23 August 2021).

Getting Started — scikit-learn 0.24.2 documentation (no date). Available at: https://scikit-learn.org/stable/getting_started.html (Accessed: 23 August 2021).

Yadav, D. (2019) *Categorical encoding using Label-Encoding and One-Hot-Encoder*, *Medium*. Available at: https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd (Accessed: 23 August 2021).