Configuration Manual

MSc Cyber Security
Research Project

Rohith Satheesh Kumar
Student ID: X19207611

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	ROHITH SATHEESH KUMAR
Student ID:	X19207611
Programme:	MSc Cyber Security **Year:** 2021-2022
Module:	Research project
Lecturer:	Imran Khan
Submission Due Date:	15/08/2022
Project Title:	INTRUSION DETECTION OF KERNEL-ROOTKITS IN ANDROID DEVICES USING MACHINE LEARNING– RANDOM FOREST
Word Count:	860 **Page Count:** 11

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: ROHITH SATHEESH KUMAR
Date: 15/08/2022

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):	

1. System Requirements

It is necessary to have the following hardware and software in order to ensure that the model processing goes well and to cut down on the amount of time it takes.

1.1. Hardware Requirements

The implementation was performed on an Lenovo legion, the configuration of the device is as follows

1. Processor – Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz   2.21 GHz
2. RAM - 16 GB DDR4
3. Hard Disk – 118MB SSD , 1 TB HDD
4. OS – Windows 10 Pro 64 – bit

1.2. Software Requirements

The software requirements are listed below.

Software	Version
Python	3.7.13
Tensorflow	2.8.2
Pandas	1.3.5
Numpy	1.21.6
Scikit-learn	1.0.2
Seaborn	0.11.2

3. Data pre-processing

2.1 Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import Isomap
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report
```

All the necessary libraries like pandas, Matplotlib, seaborn, and NumPy for data preprocessing and visualization are imported. Sklearn packages for data splitting and data conversion. Sklearn ensembles are used to import RandomForestClassifier.

2.2 Data Loading

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] df=pd.read_csv('/content/drive/MyDrive/DATASET/labeled.csv')
    df.head()

    /usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (4,7,8,11,12,14,25,26,28,31,32,34,35,36,40,41,43,44,52,54,56,60,68,69,70,71,73,77,79
      exec(code_obj, self.user_global_ns, self.user_ns)
```

	sha256	Unnamed: 1	Unnamed: 2	sha1	md5	ad_aware	aegislab
0	00002fa57ffcba0dcd65044ff9dcfc482cb8d1e7541dc0...	NaN	NaN	577802920a4ed515fd42bb3e30a7c1e1c12d7ae1	92e759b8942def4f959780d97cf614f1	-1	Trojan.AndroidOS.Hiddapp.C!c
1	0000636f9c57d0adad87ffcfb4aa0ad71aefea915b4cd6...	NaN	NaN	1e34142346dd890f8734b56608b70b46c3e8b61a	66813190742c4f5472bb3b1ceab27f6d	-1	-1
2	000083b4f6c826b9bbad23e1bff9c7e0bd89ab2e3c0dba...	NaN	NaN	7f0ead3b75be6dd042cc83cd49eedb413311f92d	071140a11c4b0dc576e7e560a8f1fbf0	-1	-1
3	0000a2a514aab2d71b529300be4c31840c7698c04d73a6...	NaN	NaN	e5781e38481503ef532e81983c723c6b5cc0ba19	c33a7f59fdd2562678b2c13084e7b8bc	-1	-1
4	0000a6d452d58424a8b7613f175af5937f590033616b80...	NaN	NaN	e1dfdec8eb4323932db8314fcd0b38c3cacbac4e	84f5027651c33bc3bc967c7fda552b0e	-1	Adware.AndroidOS.Agent.A!c

5 rows × 100 columns

For loading data, we must first mount the device, then load the dataset 'labeled.csv' as a pandas dataframe, and then show all columns using the data.head() function.

2.3 Label Encoding

```
[ ] df.columns

    Index(['sha256', 'Unnamed: 1', 'Unnamed: 2', 'sha1', 'md5', 'ad_aware',
           'aegislab', 'agnitum', 'ahnlab', 'ahnlab_v3', 'alibaba', 'alyac',
           'antivir', 'antiy_avl', 'apex', 'arcabit', 'avast', 'avast_mobile',
           'avg', 'avira', 'avware', 'babable', 'baidu', 'baidu_international',
           'bitdefender', 'bitdefenderfalx', 'bitdefendertheta', 'bkav',
           'bytehero', 'cat_quickheal', 'clamav', 'cmc', 'commtouch', 'comodo',
           'crowdstrike', 'cybereason', 'cylance', 'cynet', 'cyren', 'drweb',
           'egambit', 'elastic', 'emsisoft', 'endgame', 'esafe', 'escan',
           'eset_nod32', 'f_prot', 'f_secure', 'fireeye', 'fortinet', 'gdata',
           'gridinsoft', 'ikarus', 'invincea', 'jiangmin', 'k7antivirus', 'k7gw',
           'kaspersky', 'kingsoft', 'malwarebytes', 'max', 'maxsecure', 'mcafee',
           'mcafee_gw_edition', 'microsoft', 'microworld_escan', 'nano_antivirus',
           'nod32', 'norman', 'nprotect', 'paloalto', 'panda', 'pctools',
           'qihoo_360', 'rising', 'sangfor', 'sentinelone', 'sophos', 'sophosml',
           'superantispyware', 'symantec', 'symantecmobileinsight', 'tencent',
           'thehacker', 'totaldefense', 'trapmine', 'trendmicro',
           'trendmicro_housecall', 'trustlook', 'vba32', 'vipre', 'virobot',
           'webroot', 'whitearmor', 'yandex', 'zillya', 'zonealarm', 'zoner',
           'class'],
          dtype='object')
```

The above code snippets show a code to convert columns into encoding, which refers to transforming labels into a numeric form in order to convert them into a machine-readable format.

2.4 Separating Features and Labels

```
[ ] X = new_df[new_df.columns[:-1]]
```

```
[ ] X
```

	sha256	sha1	md5	ad_aware	aegislab	agnitum	ahnlab	ahnlab_v3	alibaba	alyac	...	trustlook	vba32	vipre	virobot	webroot	whitearmor	yandex	zillya	zonealarm	zoner
6	26	407053	206169	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
64	322	762567	15956	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
87	439	797780	117887	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
131	663	333615	13354	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
155	765	354957	26325	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6257	33786	306321	29017	0	0	0	0	11798	0	0	...	450	0	0	0	1	0	0	0	1091	0
6258	33789	578666	145234	0	0	0	0	5754	624	0	...	0	0	0	0	1	671	0	0	2087	0
6259	33797	237471	34781	1281	0	0	0	2291	0	0	...	453	0	0	0	1	0	0	11991	3351	19820
6260	33822	488686	138183	0	0	0	0	0	0	0	...	452	0	0	0	1	0	0	0	0	3145
6261	33836	468827	51125	0	5294	0	0	1138	38304	0	...	453	0	0	0	1	0	0	14623	4293	0

14242 rows × 97 columns

```
[ ] Y = new_df['class']
```

```
[ ] Y
```

```
6       0
64      0
87      0
131     0
155     0
       ..
6257    2
6258    2
6259    2
6260    2
6261    2
Name: class, Length: 14242, dtype: int64
```

In order for the machine learning model to find this data useful, the features and target columns will need to be separated. The code that is shown above divides the last label column and then initializes it to var X and then last column i.e a class is assigned to variable Y.

2.5 Data Normalization

```
[ ]  X = normalize(X)
```

```
[ ]  X

     array([[5.69816837e-05, 8.92098665e-01, 4.51840644e-01, ...,
             0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
            [4.22165536e-04, 9.99781075e-01, 2.09194823e-02, ...,
             0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
            [5.44365750e-04, 9.89257648e-01, 1.46181424e-01, ...,
             0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
            ...,
            [1.32616375e-01, 9.31814755e-01, 1.36477502e-01, ...,
             4.70516009e-02, 1.31490213e-02, 7.77718898e-02],
            [6.64281422e-02, 9.59804361e-01, 2.71398497e-01, ...,
             0.00000000e+00, 0.00000000e+00, 6.17694126e-03],
            [6.74665003e-02, 9.34806624e-01, 1.01939497e-01, ...,
             2.91571886e-02, 8.55992688e-03, 0.00000000e+00]])
```

The normalize() method is used to standardize the data in order to avoid over fitting. The purpose of normalization is to convert the numeric value columns in the dataset to use a similar scale. This should be done without distorting the differences in the ranges of values or losing any information in the process.

2.6 Dimensionality Reduction

```
[ ]  from sklearn.manifold import Isomap

     mapping = Isomap(n_components =4)
     mapped_X = mapping.fit_transform(X)
```

Isomap() is an unsupervised learning technique used to reduce dimensionality in data.

2.7 Train-Test Split

```
[ ]  x_train, x_test, y_train, y_test = train_test_split(mapped_X, Y, test_size=0.3, random_state=42)
```

After the necessary preprocessing steps have been done, the data is now split into train and test sets, with a 30 percent test size and a random state of 42.

3. Model

Now that the data are prepared for training, we need to construct a model that properly represents the data and produces high accuracy.

3.1 Model Defining and Training

```
[ ]  rfc=RandomForestClassifier(random_state=42)
```

As a result of putting an emphasis on the data, a decision tree-based classifier is chosen from the sklearn ensemble package. The RandomForestClassifier() function has been initialized as rfc in the preceding code.

```
[ ]  param_grid = {
         'n_estimators': [200, 500],
         'max_features': ['auto', 'sqrt', 'log2'],
         'max_depth' : [4,5,6,7,8],
         'criterion' :['gini', 'entropy']
     }
```

In order to use grid search and determine the parameters that work best, different values are chosen for each param, as seen in the above image.

```
[ ]  CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 3)
     CV_rfc.fit(x_train, y_train)

     GridSearchCV(cv=3, estimator=RandomForestClassifier(random_state=42),
                  param_grid={'criterion': ['gini', 'entropy'],
                              'max_depth': [4, 5, 6, 7, 8],
                              'max_features': ['auto', 'sqrt', 'log2'],
                              'n_estimators': [200, 500]})
```

Now the params dictionary and the rfc model are sent on to grid search, which uses a variety of possible parameter values in conjunction with one another to derive the parameters that are the best match.

```
[ ]
     CV_rfc.best_params_

     {'criterion': 'gini',
      'max_depth': 7,
      'max_features': 'auto',
      'n_estimators': 200}

[ ]  rfc1=RandomForestClassifier(random_state=42, max_features='auto', n_estimators= 500, max_depth=4, criterion='gini')
```

The optimal parameters from the grid search technique are once again used to initialise a RandomForestClassifier.

3.2 Model Predict

```
[ ]  pred=rfc1.predict(x_test)
```

This code predicts the target values on the test (unseen) dataset.

3.3 Model Evaluation

a. Accuracy

```
[ ]  score = accuracy_score(y_test,pred)
     print("Testing Score: {:.2f} %".format(score*100))

     Testing Score: 97.12 %
```

The model is evaluated on test data and it gives 97.12% accuracy.

b. Classification Report

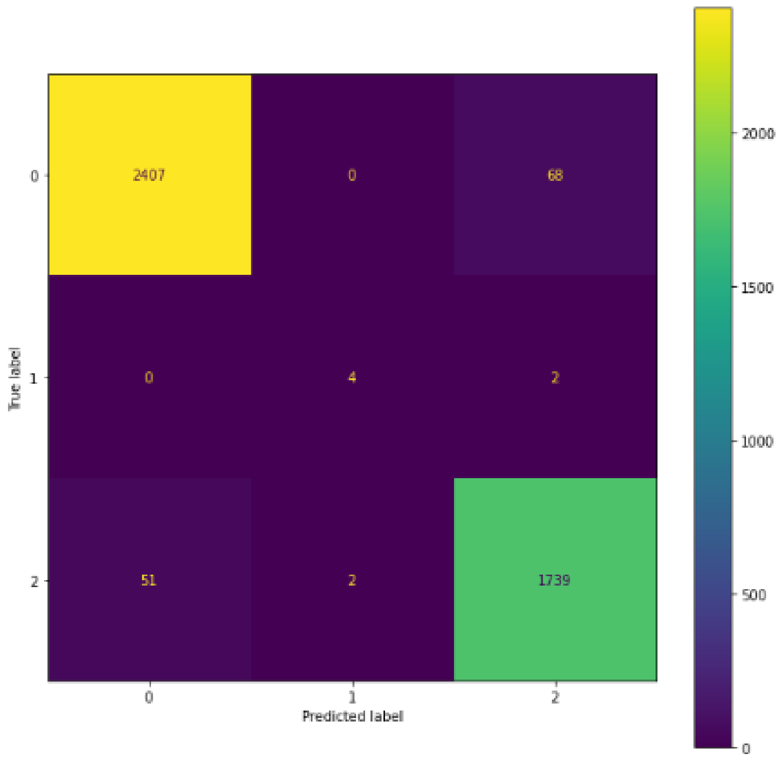
```
[ ]  print(classification_report(y_test, pred))

                   precision    recall  f1-score   support

              0       0.98      0.97      0.98      2475
              1       0.67      0.67      0.67         6
              2       0.96      0.97      0.97      1792

       accuracy                           0.97      4273
      macro avg       0.87      0.87      0.87      4273
   weighted avg       0.97      0.97      0.97      4273
```

A high precision and recall score indicate that the model is reliably predicting true positive and true negative values. Also, the f1 score gives a single score that combines the accuracy and recall factors into a single number, making it easy to compare and analyze.

The trained random forest classifier provides stable and high values for all three metrics defined above.

c. Confusion Matrix

```
fig,ax = plt.subplots(figsize = (10,10))
plot_confusion_matrix(rfc1,x_test,y_test, ax = ax)
```



Predicted outcomes for the ML classification issue are summarised in the confusion matrix. It also stands for a model's sensitivity and specificity. Predicted values and real labels are shown above, along with the number of accurate and wrong predictions.

3.4 Model Pickel

```
import pickle
pickle.dump(rfc1, open('/content/drive/MyDrive/DATASET/model.pkl', 'wb'))
```

The model is finally saved as a pickle file so that it can be used again in the future.