Configuration Manual

MSc Research Project
Cyber Security

Janius Christabel Joseph
Student ID: X20112408

School of Computing
National College of Ireland

Supervisor: Vanessa Ayala-Rivera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Janius Christabel Joseph
Student ID:	X20112408
Programme:	Cyber Security **Year:** 2021
Module:	MSc Research Project
Lecturer:	Vanessa Ayala-Rivera
Submission Due Date:	16/12/2021
Project Title:	Multi Classifier Models using Machine Learning Techniques for Malware Detection
Word Count: 640	**Page Count:** 15

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: Janius Christabel Joseph

Date: 16/12/2021

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

Janius Christabel Joseph
X20112408

1. Introduction

This configuration document was created to outline the steps for executing the research project and to specify the configuration of the system used to run the models. The stages involve downloading and installing the essential software and packages, as well as the minimal setup required for the project to work well.

2. System Configuration

Processor: Intel(R) Core (TM) i7 – 5500U CPU @ 2.40GHz
RAM: 16 GB
Storage: 500 GB HDD
Operating System: Windows 10 64-bit operating system

3. Setup

Python Libraries version:
Numpy 1.19.5
Pandas 1.1.15
Seaborn 0.11.2
Mathplotlib 3.2.2
Sklearn 1.0.2

Google Colaboratory:
It is a service that allows users to run Python code in a browser online while requiring little resources and essentially no configuration which needs to be done locally.
Only the following requirements for writing code using Google Colab. A browser such as Chrome or Firefox and a google account.

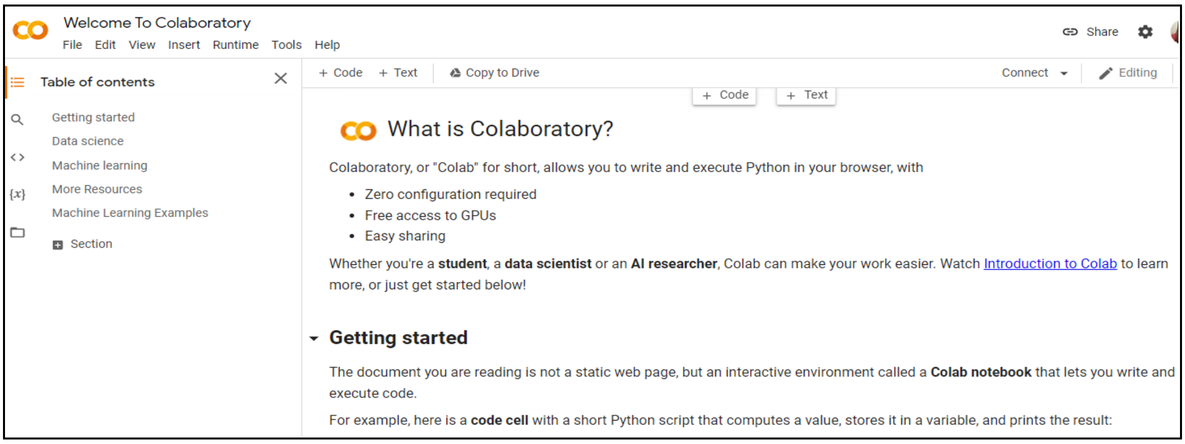

Fig 1. Google Colab

4. Implementation

Step 1. Importing necessary python libraries for dataset preprocessing

```
[4] # Importing the required libraries and modules
    import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import pickle
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, confusion_matrix
```

Fig 2. Libraries required for models

Step 2. Loading and Preprocessing of the dataset

Data Pre-processing

```
[10] # Explore Dataset
     print('Dimensions:{}'.format(data.shape))
     print(data.dtypes)

     Dimensions:(19611, 79)
     Name                              object
     e_magic                            int64
     e_cblp                             int64
     e_cp                               int64
     e_crlc                             int64
                                        ...
     ImageDirectoryEntryExport          int64
     ImageDirectoryEntryImport          int64
     ImageDirectoryEntryResource        int64
     ImageDirectoryEntryException       int64
     ImageDirectoryEntrySecurity        int64
     Length: 79, dtype: object
```

Fig 3. Data Preprocessing

```
[12] missing_data = pd.DataFrame({'total_missing': data.isnull().sum(), 'perc_missing': (data.isnull().sum()/82790)*100})
     missing_data
```

	total_missing	perc_missing
Name	0	0.0
e_magic	0	0.0
e_cblp	0	0.0
e_cp	0	0.0
e_crlc	0	0.0
...	...	...
ImageDirectoryEntryExport	0	0.0
ImageDirectoryEntryImport	0	0.0
ImageDirectoryEntryResource	0	0.0
ImageDirectoryEntryException	0	0.0
ImageDirectoryEntrySecurity	0	0.0

79 rows × 2 columns

Fig 4. Checking for missing data in the dataset

```python
[17] #Statistical description for the data
     data.describe()
```

	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss
count	19611.0	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000
mean	23117.0	178.615726	71.660752	49.146958	37.370710	37.032635	64178.739687	10.418490
std	0.0	987.200729	1445.192977	1212.201919	864.515405	915.833139	9110.755873	637.116265
min	23117.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000
50%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000
75%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000
max	23117.0	59448.000000	63200.000000	64613.000000	43690.000000	43690.000000	65535.000000	61436.000000

Fig 5. Analysis of data in statistical terms

```python
[19] # converting the non-numeric data into numeric data.
     from sklearn.preprocessing import LabelEncoder
     encoded = data.apply(lambda x: LabelEncoder().fit_transform(x) if x.dtype == 'object' else x)
     encoded.head()
```

	Name	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csum	e_ip	e_cs	e_lfarlc	e_ovno
0	7037	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
1	11957	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
2	11448	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
3	5367	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
4	3489	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0

Fig 6. Converting nonnumeric to numeric

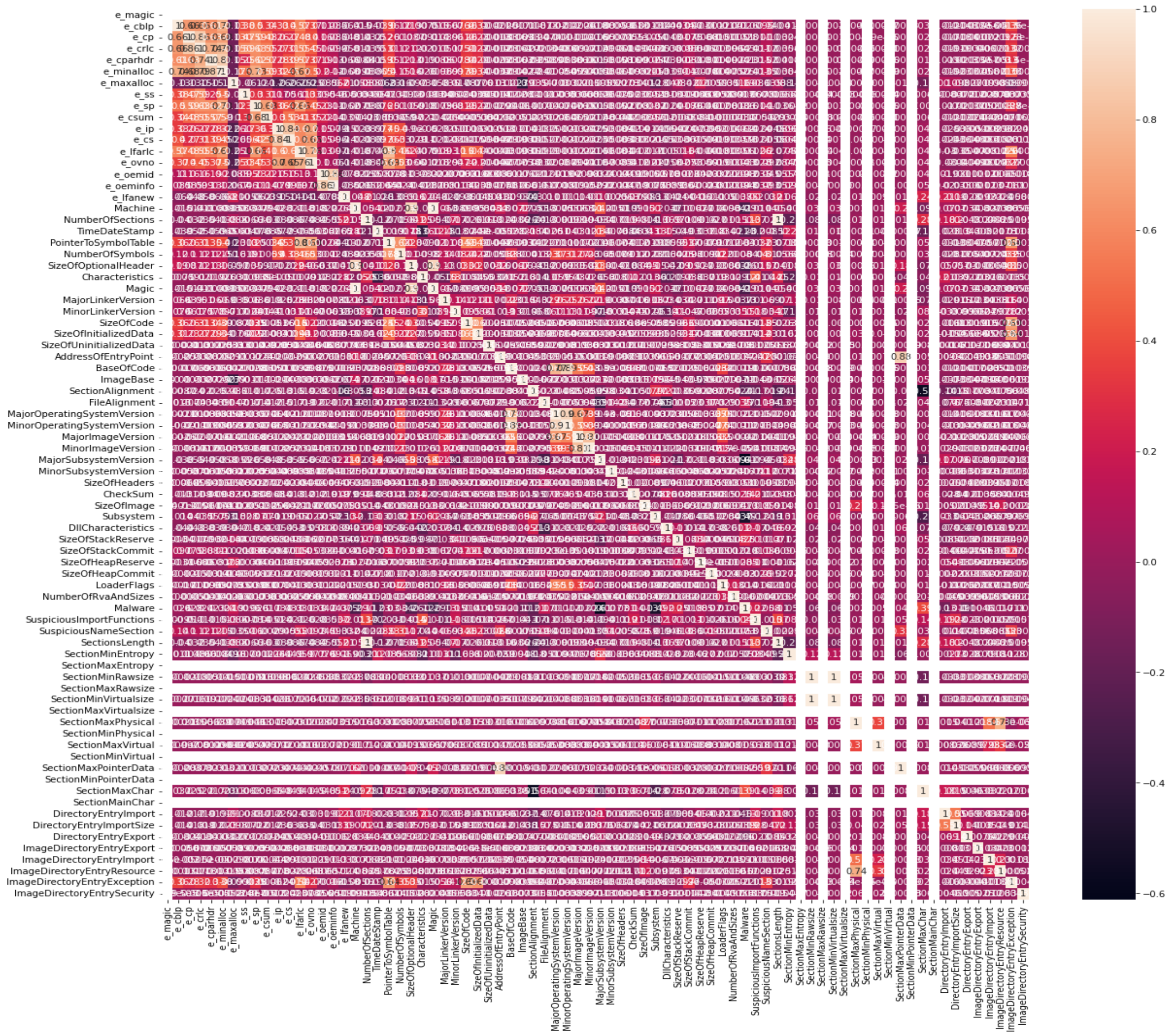

Fig 7. Correlation plot before encoding

```python
# converting the non-numeric data into numeric data.
from sklearn.preprocessing import LabelEncoder
encoded = data.apply(lambda x: LabelEncoder().fit_transform(x) if x.dtype == 'object' else x)
encoded.head()
```

	Name	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csum	e_ip	e_cs	e_lfar
0	7037	23117	144	3	0	4	0	65535	0	184	0	0	0	
1	11957	23117	144	3	0	4	0	65535	0	184	0	0	0	
2	11448	23117	144	3	0	4	0	65535	0	184	0	0	0	
3	5367	23117	144	3	0	4	0	65535	0	184	0	0	0	
4	3489	23117	144	3	0	4	0	65535	0	184	0	0	0	

Fig 8. Non numeric to numeric data conversion

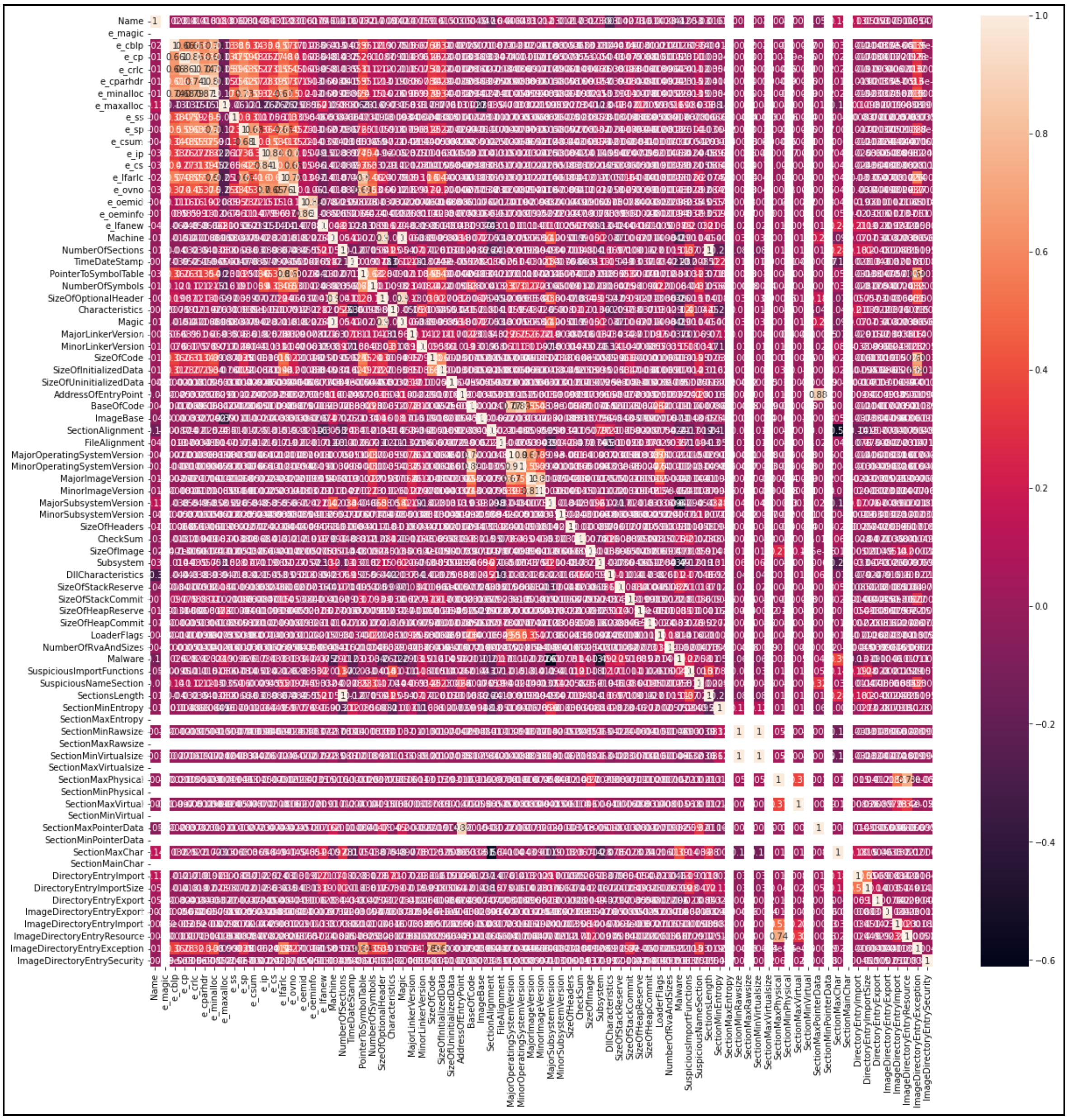

Fig 9. Correlation after label encoding

Step 4. Feature Selection

```
[24] rfc = RandomForestClassifier(n_estimators=100, random_state=0,
                                  oob_score = True,
                                  max_depth = 16)
     rfc.fit(X_train, y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does not have valid feature names, but RandomForestClassifier
  "X does not have valid feature names, but"
RandomForestClassifier(max_depth=16, oob_score=True, random_state=0)
```

```
[25] importance = rfc.feature_importances_
     importance_dict = {data_usage.columns.values[i]: importance[i] for i in range (len(importance))}
     sorted_dict = {k: v for k, v in sorted(importance_dict.items(), key=lambda item: item[1])}
     plt.figure(figsize=(10, 20))
     sns.barplot(y=list(sorted_dict.keys())[::-1], x=list(sorted_dict.values())[::-1])
     plt.title('Features importance')

Text(0.5, 1.0, 'Features importance')
```

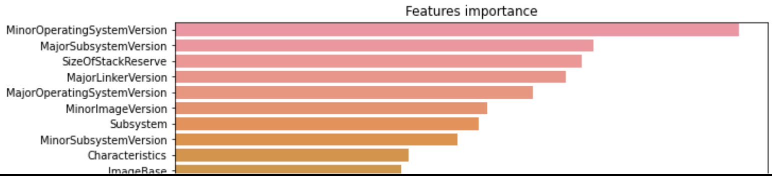

Fig 9. Feature selection

Step 5. Splitting the dataset into Training and Testing in the ratio of 80:20

```
# Splitting the data into training and testing data subset into 80% and 20% respectively
X_train, X_test, y_train, y_test = train_test_split(data_usage, data['Malware'], test_size=0.2, random_state=0)

print(f'Number of used features is {X_train.shape}')
print(f'Number of used features is {X_test.shape}')
print(f'Number of used features is {y_train.shape}')
print(f'Number of used features is {y_test.shape}')

Number of used features is (15688, 75)
Number of used features is (3923, 75)
Number of used features is (15688,)
Number of used features is (3923,)
```

Fig 10. Dataset split into training and test

Step 6. Model building for Binary Classification

Random forest model with binary labelled data

```
[ ] rfc = RandomForestClassifier(n_estimators=100, random_state=0,
                                 oob_score = True,
                                 max_depth = 16)
    rfc.fit(X_train, y_train)
```

```
[ ] # Classification report
    y_pred = rfc.predict(X_test)
    print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))
```

```
              precision    recall  f1-score   support

      Benign       0.99      0.96      0.97      1004
     Malware       0.99      1.00      0.99      2919

    accuracy                           0.99      3923
   macro avg       0.99      0.98      0.98      3923
weighted avg       0.99      0.99      0.99      3923
```

Fig 11. Random Forest Binary Model and evaluation metrics

Decision Tree model with binary labelled data

```
[29] from sklearn.tree import DecisionTreeClassifier
     classifier_dt = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
     classifier_dt.fit(X_train, y_train)

     DecisionTreeClassifier(criterion='entropy', random_state=42)
```

```
[30] # Classification report
     y_pred = classifier_dt.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))
```

```
              precision    recall  f1-score   support

      Benign       0.96      0.96      0.96      1004
     Malware       0.99      0.99      0.99      2919

    accuracy                           0.98      3923
   macro avg       0.98      0.98      0.98      3923
weighted avg       0.98      0.98      0.98      3923
```

Fig 12. Decision Tree Binary Model and evaluation metrics

Support Vector Machine (SVM) model with binary labelled data

```
#Import svm model
from sklearn import svm
svm = svm.SVC(kernel='rbf',random_state=1,C=1,gamma='auto')
svm.fit(X_train, y_train)

SVC(C=1, gamma='auto', random_state=1)
```

```
[33] # Classification report
y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))

              precision    recall  f1-score   support

      Benign       0.92      0.01      0.02      1004
     Malware       0.75      1.00      0.85      2919

    accuracy                           0.75      3923
   macro avg       0.83      0.51      0.44      3923
weighted avg       0.79      0.75      0.64      3923
```

Fig 13. SVM Binary Model and evaluation metrics

Naive Bayes Model with binary labelled data

```
[35] from sklearn.naive_bayes import GaussianNB
Gmodel=GaussianNB()
Gmodel.fit(X_train, y_train)

GaussianNB()
```

```
[36] # Classification report
y_pred = Gmodel.predict(X_test)
print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))

              precision    recall  f1-score   support

      Benign       0.27      1.00      0.43      1004
     Malware       0.99      0.09      0.17      2919

    accuracy                           0.32      3923
   macro avg       0.63      0.54      0.30      3923
weighted avg       0.81      0.32      0.24      3923
```

Fig 14. Naïve Bayes Binary Model and evaluation metrics

K-Nearest Neighbour model with binary labelled data

```
[38] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors=1)
     knn.fit(X_train,y_train)

     KNeighborsClassifier(n_neighbors=1)
```

```
[39] # Classification report
     y_pred = knn.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))

                   precision    recall  f1-score   support

          Benign       0.93      0.94      0.94      1004
         Malware       0.98      0.98      0.98      2919

        accuracy                           0.97      3923
       macro avg       0.95      0.96      0.96      3923
    weighted avg       0.97      0.97      0.97      3923
```

Fig 15. KNN Binary Model and evaluation metrics

Step 7: Model building for Multi classification

Model Building for Multi Classification

```
[41] # class distribution for original data
     print(data.groupby('Subsystem').size())

     Subsystem
     1       438
     2     15451
     3      3711
     9         4
     10        4
     16        3
     dtype: int64
```

```
[42] # sample distribution print in markdown table format
     #label = [Adware, Virus, Spyware, Trojan, Worm, Ransomware]
     label = 'Subsystem'
     lblTypes = set(data[label])
     for lbl in lblTypes:
         print('| {} | {} |'.format(lbl, len(data[data[label] == lbl].index)))
```

Fig 16. Labels for malware classification

```
[42] | 1 | 438 |
     | 2 | 15451 |
     | 3 | 3711 |
     | 9 | 4 |
     | 10 | 4 |
     | 16 | 3 |

[43] # Splitting the data into training and testing data subset into 80% and 20% respectively
     X_train, X_test, y_train, y_test = train_test_split(data_usage, data['Subsystem'], test_size=0.2, random_state=0)

[44] print(f'Number of used features is {X_train.shape}')
     print(f'Number of used features is {X_test.shape}')
     print(f'Number of used features is {y_train.shape}')
     print(f'Number of used features is {y_test.shape}')

     Number of used features is (15688, 75)
     Number of used features is (3923, 75)
     Number of used features is (15688,)
     Number of used features is (3923,)
```

Fig 17. Dataset split for Multi classifier models

Random forest model with multi class data

```
[45] rfc = RandomForestClassifier(n_estimators=100, random_state=0,
                                 oob_score = True,
                                 max_depth = 16)
     rfc.fit(X_train, y_train)

     /usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does not have valid feature names, but RandomForestClassi
       "X does not have valid feature names, but"
     RandomForestClassifier(max_depth=16, oob_score=True, random_state=0)

[46] # Classification report
     #"1": "Adware", "2": "Virus", "3": "Spyware", "9": "Trojan", "10": "Worm", "16" : "Ransomware"}, inplace=True)

     y_pred = rfc.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Adware', 'Virus', 'Spyware', 'Trojan', 'Worm', 'Ransomware']))

                   precision    recall  f1-score   support

          Adware       0.99      0.98      0.98        90
           Virus       1.00      1.00      1.00      3044
         Spyware       1.00      1.00      1.00       786
          Trojan       1.00      1.00      1.00         1
            Worm       1.00      1.00      1.00         1
      Ransomware       1.00      1.00      1.00         1
```

Fig 18. Multi classifier model using Random Forest and evlauation metrics

Decision Tree model with multi class data

```
[48] from sklearn.tree import DecisionTreeClassifier
     classifier_dt = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
     classifier_dt.fit(X_train, y_train)

     DecisionTreeClassifier(criterion='entropy', random_state=42)

[49] # Classification report
     y_pred = classifier_dt.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Adware', 'Virus', 'Spyware', 'Trojan', 'Worm', 'Ransomware']))

                   precision    recall  f1-score   support

          Adware       1.00      1.00      1.00        90
           Virus       1.00      1.00      1.00      3044
         Spyware       1.00      1.00      1.00       786
          Trojan       1.00      1.00      1.00         1
            Worm       1.00      1.00      1.00         1
      Ransomware       1.00      1.00      1.00         1

        accuracy                           1.00      3923
       macro avg       1.00      1.00      1.00      3923
    weighted avg       1.00      1.00      1.00      3923
```

Fig 19. Multi classifier model using Decision Tree and evlauation metrics

SVM model for multi class data

```
[51] #Import svm model
     from sklearn import svm
     svm = svm.SVC(kernel='rbf',random_state=1,C=1,gamma='auto')
     svm.fit(X_train, y_train)

     SVC(C=1, gamma='auto', random_state=1)

[52] # Classification report
     y_pred = svm.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Adware', 'Virus', 'Spyware', 'Trojan', 'Worm', 'Ransomware']))

                   precision    recall  f1-score   support

          Adware       1.00      0.09      0.16        90
           Virus       0.78      1.00      0.88      3044
         Spyware       1.00      0.03      0.05       786
          Trojan       0.00      0.00      0.00         1
            Worm       0.00      0.00      0.00         1
      Ransomware       0.00      0.00      0.00         1

        accuracy                           0.78      3923
       macro avg       0.46      0.19      0.18      3923
    weighted avg       0.83      0.78      0.70      3923
```

Fig 19. Multi classifier model using SVM and evlauation metrics

Naive Bayes model with multi class data

```
[54] from sklearn.naive_bayes import GaussianNB
     Gmodel=GaussianNB()
     Gmodel.fit(X_train, y_train)

     GaussianNB()

[55] # Classification report
     y_pred = Gmodel.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Adware', 'Virus', 'Spyware', 'Trojan', 'Worm', 'Ransomware']))

                  precision    recall  f1-score   support

          Adware       0.17      0.01      0.02        90
           Virus       1.00      0.08      0.15      3044
         Spyware       0.49      0.69      0.58       786
          Trojan       0.00      0.00      0.00         1
            Worm       0.00      1.00      0.00         1
      Ransomware       0.00      1.00      0.00         1

        accuracy                           0.20      3923
       macro avg       0.28      0.46      0.13      3923
    weighted avg       0.88      0.20      0.23      3923
```

Fig 20. Multi classifier model using Naïve Bayes and evlauation metrics

K-nearest Neighbour model with multi class data

```
[57] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors=1)
     knn.fit(X_train,y_train)

     KNeighborsClassifier(n_neighbors=1)

[58] # Classification report
     y_pred = knn.predict(X_test)
     print(classification_report(y_test, y_pred, target_names=['Adware', 'Virus', 'Spyware', 'Trojan', 'Worm', 'Ransomware']))

                  precision    recall  f1-score   support

          Adware       0.93      0.92      0.93        90
           Virus       0.94      0.95      0.94      3044
         Spyware       0.78      0.75      0.77       786
          Trojan       0.00      0.00      0.00         1
            Worm       1.00      1.00      1.00         1
      Ransomware       1.00      1.00      1.00         1

        accuracy                           0.91      3923
       macro avg       0.77      0.77      0.77      3923
    weighted avg       0.90      0.91      0.91      3923
```

Fig 21. Multi classifier model using KNN and evlauation metrics