Configuration Manual

MSc Research Project
Cybersecurity

Shubham Prashant Pandharpote
Student ID: x20143877

School of Computing
National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shubham Prashant Pandharpote
Student ID:	x20143877
Programme:	M.Sc. Research Project **Year:** 2021/2022
Module:	Cybersecurity
Supervisor:	Liam McCabe
Submission Due Date:	16/12/2021
Project Title:	Detecting Malware Based on Portable Executable Analysis
Word Count:	**11** **Page Count** **921**

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: Shubham Prashant Pandharpote

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

Shubham Prashant Pandharpote
x20143877

1 Introduction:

This manual is created with step – wise explanation of how project carried out and how experiments were carried out for each model. In addition to it information and process of downloading softwares required to carry out the experiments is mentioned. Furthermore, information regarding specifications of location machine is mentioned in the manual. Each models' detailed information is mentioned in this manual.

2 Configurations of Local Machine:

Hardware Specifications	
Main memory of the machine	16 Gigabytes
CPU	AMD Ryzen 5 355H
GPU	Nvidia GTX 1650 Super
Hard – Disk	1 Tera Bytes
Software Specifications	
Operating system	Windows 10
IDE	Anaconda
Programming Language	Python
Designing Software	Jupyter Notebooks
Libraries Used for Implementation	Pandas, numpy, sklearn, seaborn, matplotlib

Additional Information:

1. Jupyter Notebooks was used for Data Pre – processing, for performing operations related to Features, and for implementation.
2. Processor intel i5 or any equivalent processor with same or more power can be used.

3 Important Softwares

This section will elaborate what softwares were used during project to carry out the experiments.

1. **Python**:
   While performing the experiments it is recommended to have latest version of Python to be installed on the local machine, as it will arrive with more features and bug – fixes. Python can be downloaded from https://www.python.org/downloads/ , website which is -mentioned previously. In addition to it the website explains what bus are fixed in latest version in tabular form. The figure displayed below is page for

downloading python with one tabular section which explains bugs – fixes of every version released, it is highlighted below.

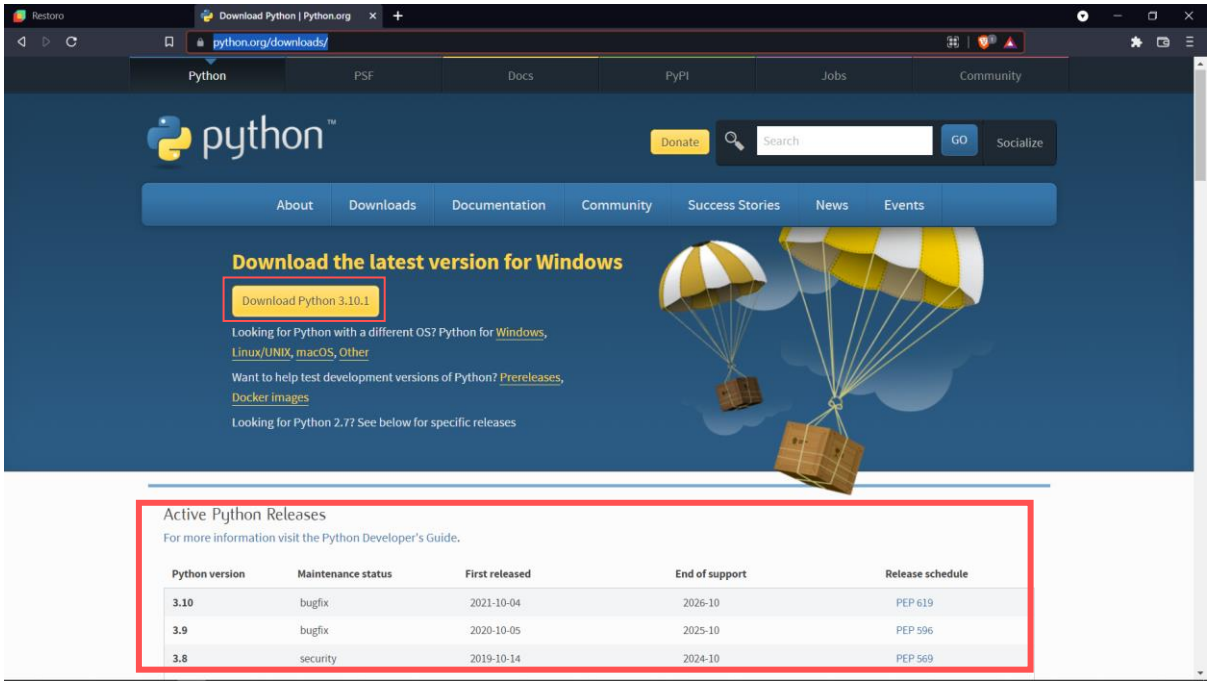


Fig. Page to Download Python

2. **Anaconda**:
   Anaconda software is mostly used by python/R developers and practitioners, and it has very huge user base. It comprises of various Integrated Development Environments (IDEs) required for python, R, web development, shell programming, etc. It is open–source software available for Windows, Mac, Linux operating Systems. Link for downloading the software https://www.anaconda.com/products/individual#windows.

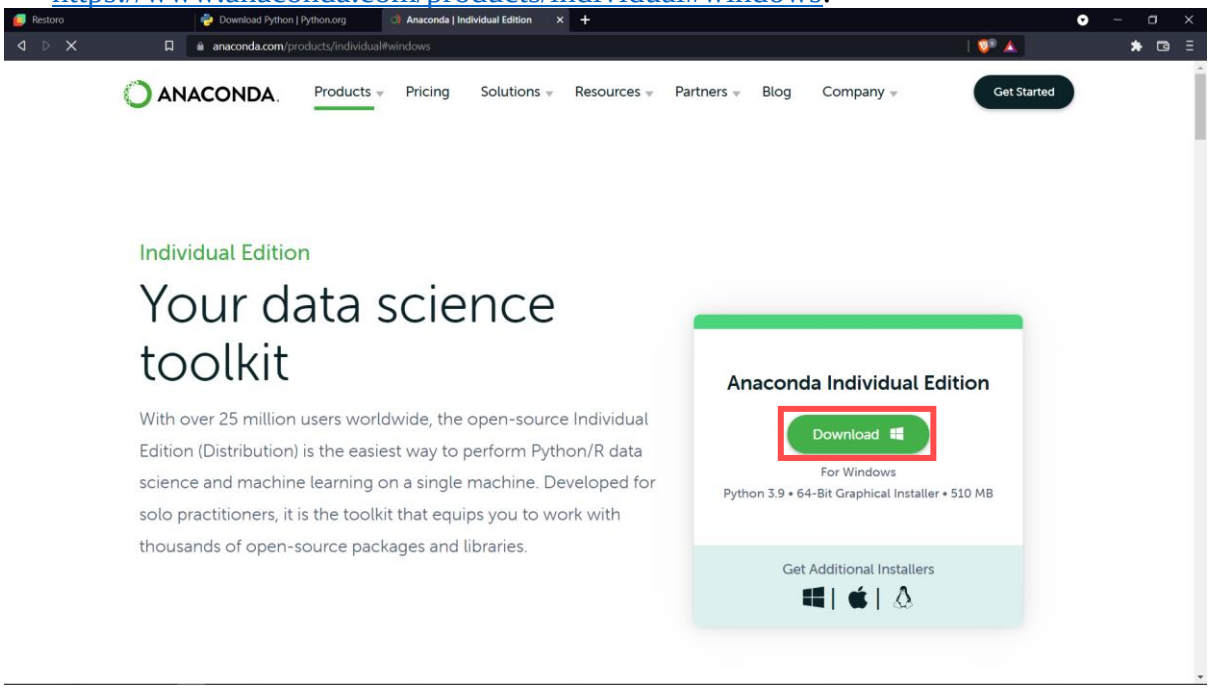


Fig. Page to Download Anaconda Software

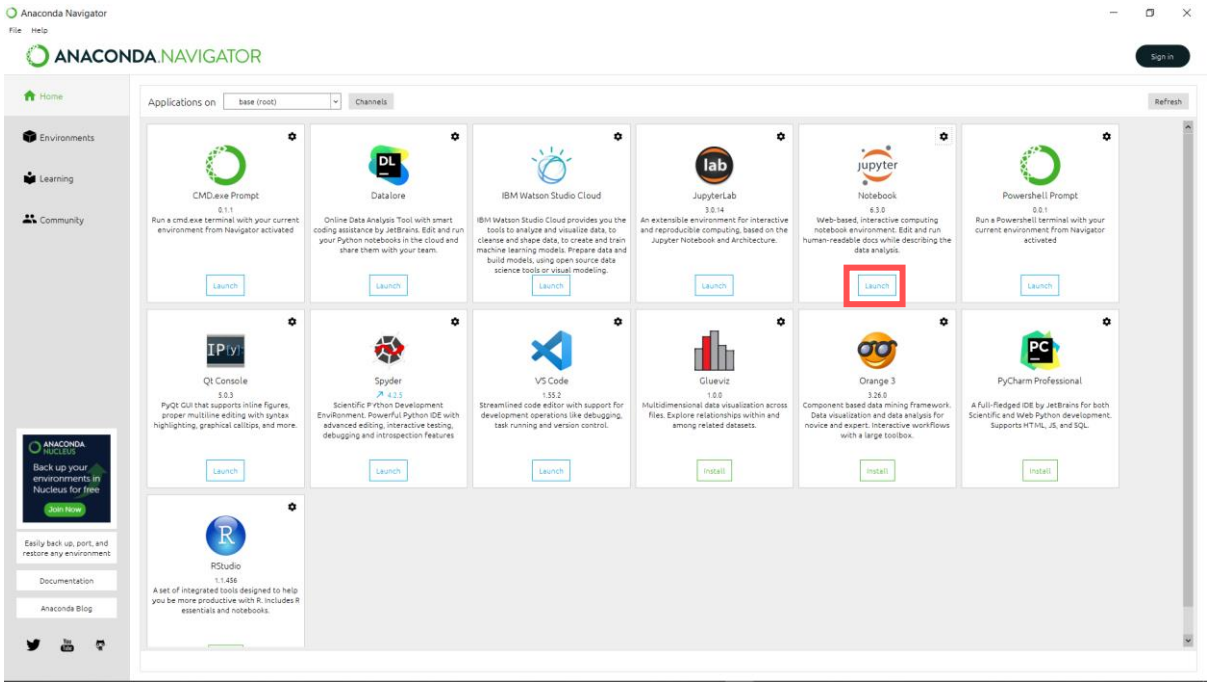


Fig. Actual Anacond software used for project.

3. **Jupyter Notebooks**:
   Jupyter notebooks were used for implementation of code for detection of malware. As shown in the figure above Jupyter notebooks' IDE can be launched by clicking the launch button on Anaconda Software. Furthermore, notebook can be opened by clicking on, .ipynb file. Once the notebook is opened it will show running on right side.

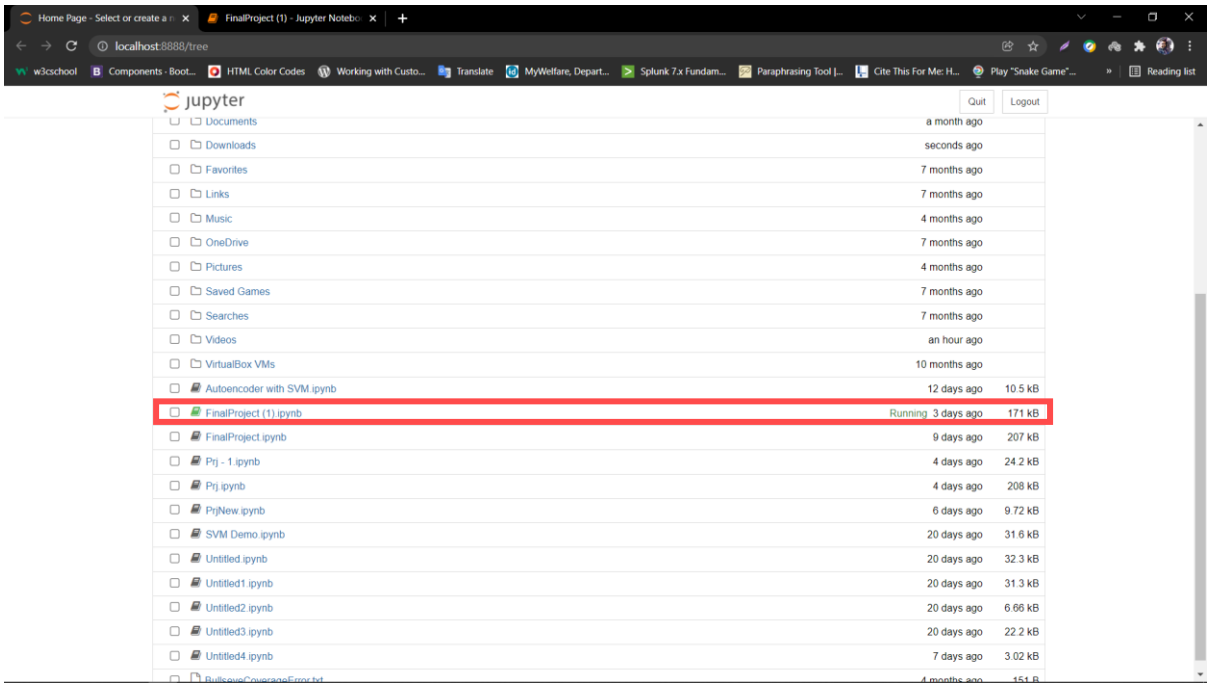


Fig. Jupyter Notebooks

4 Implementation

This step explains the implementation of software.

Step 1: Required libraries were imported.

```
In [2]: ###Required Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA
        from sklearn.preprocessing import StandardScaler
        from sklearn.svm import SVC
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
        from sklearn.metrics import confusion_matrix, accuracy_score
        from sklearn.metrics import classification_report
        from sklearn.model_selection import KFold
        from sklearn.ensemble import GradientBoostingClassifier
        import seaborn as sns
```

Step 2: Importing Dataset.

```
In [2]: df = pd.read_csv('C:\\Users\Shubham\Desktop\dataset_malwares.csv')
```

Step 3: Getting Columns which have numeric values.

```
In [4]: #Getting numerical columns
        num_cols = df._get_numeric_data().columns
        len(num_cols)

Out[4]: 78
```

Step 4: Listing of Categorical Column.
This column has heading of Portable Executables.

```
In [5]: #categorical columns
        cols=df.columns
        list(set(cols) - set(num_cols))

Out[5]: ['Name']
```

Step 5: Displaying Name column.

```
In [6]: #Target feature is Malware
        #only categorical column
        df.Name

Out[6]: 0        VirusShare_a878ba26000edaac5c98eff4432723b3
        1        VirusShare_ef9130570fddc174b312b2047f5f4cf0
        2        VirusShare_ef84cdeba22be72a69b198213dada81a
        3        VirusShare_6bf3608e60ebc16cbcff6ed5467d469e
        4        VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb
                                    ...
        19606                                      clip.exe
        19607                  VNC-Server-6.2.0-Windows.exe
        19608       Microsoft.GroupPolicy.Management.ni.dll
        19609                               cryptuiwizard.dll
        19610                                     winhttp.dll
        Name: Name, Length: 19611, dtype: object
```

PCA and Feature Extraction.

Step 6: Starting for Principal Component Analysis. Displaying Features present in dataset. All features are from Optional Header and Section Header.

PCA

```
In [7]: features=['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', 'e_oemid', 'e_oeminfo', 'e_lfanew', 'Machine',
                  'NumberOfSections', 'TimeDateStamp', 'PointerToSymbolTable',
                  'NumberOfSymbols', 'SizeOfOptionalHeader', 'Characteristics', 'Magic',
                  'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
                  'SizeOfInitializedData', 'SizeOfUninitializedData',
                  'AddressOfEntryPoint', 'BaseOfCode', 'ImageBase', 'SectionAlignment',
                  'FileAlignment', 'MajorOperatingSystemVersion',
                  'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
                  'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfHeaders',
                  'CheckSum', 'SizeOfImage', 'Subsystem', 'DllCharacteristics',
                  'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
                  'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes',
                  'SuspiciousImportFunctions', 'SuspiciousNameSection', 'SectionsLength',
                  'SectionMinEntropy', 'SectionMaxEntropy', 'SectionMinRawsize',
                  'SectionMaxRawsize', 'SectionMinVirtualsize', 'SectionMaxVirtualsize',
                  'SectionMaxPhysical', 'SectionMinPhysical', 'SectionMaxVirtual',
                  'SectionMinVirtual', 'SectionMaxPointerData', 'SectionMinPointerData',
                  'SectionMaxChar', 'SectionMainChar', 'DirectoryEntryImport',
                  'DirectoryEntryImportSize', 'DirectoryEntryExport',
                  'ImageDirectoryEntryExport', 'ImageDirectoryEntryImport',
                  'ImageDirectoryEntryResource', 'ImageDirectoryEntryException',
                  'ImageDirectoryEntrySecurity']
```

Step 7: Separating out target feature “Malware” and standardizing the features.

```
In [8]: from sklearn.preprocessing import StandardScaler
        x = df.loc[:, features].values
        # Separating out the target
        y = df.loc[:,['Malware']].values
        # Standardizing the features
        x = StandardScaler().fit_transform(x)
```

Step 8: Transforming Dataset into two principal components, with the help of PCA. Extracting required features.

```
In [9]: from sklearn.decomposition import PCA
        pca = PCA(n_components=2)
        principalComponents = pca.fit_transform(x)
        principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
        principalDf
```

Out[9]:

	principal component 1	principal component 2
0	-0.116867	-0.071666
1	-0.304068	-0.128467
2	-0.307282	-0.131369
3	-0.282982	-0.168940
4	-0.087797	0.067433
...	...	...
19606	-0.296678	-0.092260
19607	-0.339831	-0.092833
19608	0.624261	-0.050615
19609	-0.254207	-0.022516
19610	-0.406063	-0.049792

19611 rows × 2 columns

Implementing Support Vector Machine (SVM) Model:

Step 9: Assigning target feature “Malware” to y.

```
In [11]: principalDf['malware'] = y
```

Step 10: Reshapping data and giving it as input to Support Vector Machine (SVM) model.

```
In [12]: ###reshapping data and giving input to SVM model
         X_train, X_test, y_train, y_test = train_test_split(principalDf[['principal component 1', 'principal component 2']],
                                                             principalDf['malware'], test_size=0.2, random_state=0)
```

Step 11: Training Support Vector Machine (SVM) model.

```
In [13]: ###training the model
         svm = SVC(kernel = 'linear',C=1)
         hotel_rev_SVM = svm.fit(X_train, y_train)
```

Step 12: Testing Support Vector Machine (SVM) model.

```
In [14]: ###testing the model
         y_dash = hotel_rev_SVM.predict(X_test)
```

Step 13: Finding Accuracy.

```
In [15]: ###Finding Accuracy
         acc = accuracy_score(y_test, y_dash)
         print("Accuracy: {0:.2f}".format(acc))
         confusion = confusion_matrix(y_test, y_dash)
         print("Confusion matrix:\n{}".format(confusion))

         Accuracy: 0.74
         Confusion matrix:
         [[   0 1004]
          [   0 2919]]
```

Step 14: Displaying Classification Report.

```
In [16]: ###Displaying Classification Report
         print(classification_report(y_test, y_dash, target_names=['Benign', 'Malware']))

                       precision    recall  f1-score   support

               Benign       0.00      0.00      0.00      1004
              Malware       0.74      1.00      0.85      2919

             accuracy                           0.74      3923
            macro avg       0.37      0.50      0.43      3923
         weighted avg       0.55      0.74      0.63      3923
```

Implementing Gradient Boosting Algorithm:

Step 15: Reshaping Data and giving input to Gradient Boosting Model.

```
In [18]: ###reshapping data and giving input to Gradient Boosting model
         gradient_booster = GradientBoostingClassifier(learning_rate=0.1)
         gradient_booster.get_params()

Out[18]: {'ccp_alpha': 0.0,
          'criterion': 'friedman_mse',
          'init': None,
          'learning_rate': 0.1,
          'loss': 'deviance',
          'max_depth': 3,
          'max_features': None,
          'max_leaf_nodes': None,
          'min_impurity_decrease': 0.0,
          'min_impurity_split': None,
          'min_samples_leaf': 1,
          'min_samples_split': 2,
          'min_weight_fraction_leaf': 0.0,
          'n_estimators': 100,
          'n_iter_no_change': None,
          'random_state': None,
          'subsample': 1.0,
          'tol': 0.0001,
          'validation_fraction': 0.1,
          'verbose': 0,
          'warm_start': False}
```

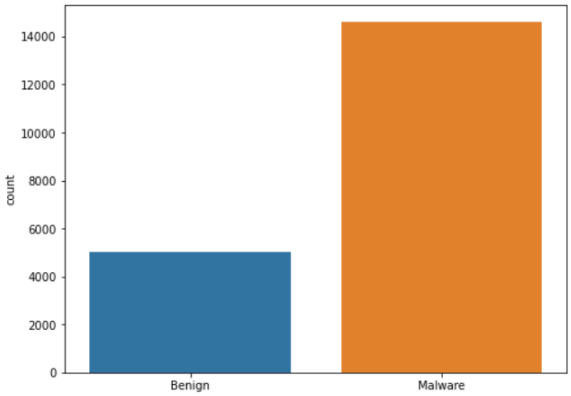
Step 16: Fitting Dataset using Gradient Boosting Model and Displaying Classification report.

```
In [19]: ###Fitting Dataset using Gradient Boosting Model
         gradient_booster.fit(X_train,y_train)

         ###Displaying Classification Report
         print(classification_report(y_test,gradient_booster.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.83      0.80      0.81      1004
           1       0.93      0.94      0.94      2919

    accuracy                           0.91      3923
   macro avg       0.88      0.87      0.87      3923
weighted avg       0.90      0.91      0.91      3923
```

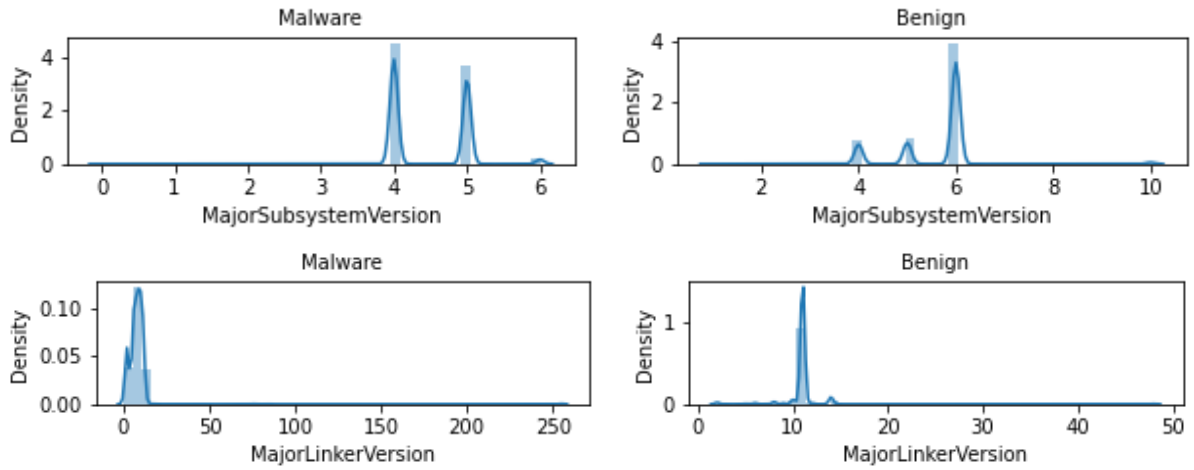
Step 17: Plotting histogram of counts of Benign and Malware samples.

```
In [35]: plt.figure(figsize=(8, 6))
         ax=sns.countplot(df['Malware'])
         ax.set_xticklabels(['Benign', 'Malware'])
Out[35]: [Text(0, 0, 'Benign'), Text(1, 0, 'Malware')]
```

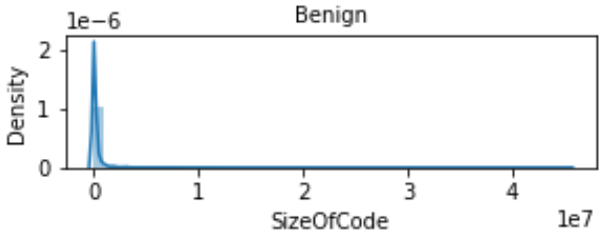


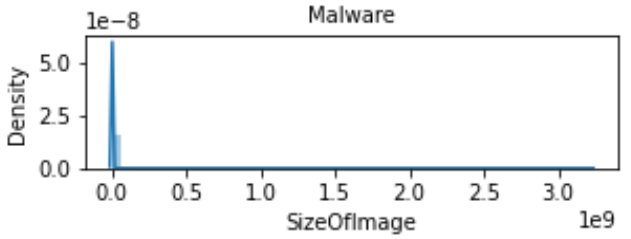
Step 18: Visualizing some of the features.

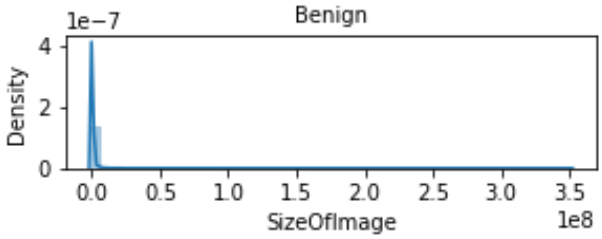
```
In [25]: features = ['MajorSubsystemVersion', 'MajorLinkerVersion', 'SizeOfCode', 'SizeOfImage', 'SizeOfHeaders', 'SizeOfInitializedData',
                    'SizeOfUninitializedData', 'SizeOfStackReserve', 'SizeOfHeapReserve',
                     'NumberOfSymbols', 'SectionMaxChar']
         i=1

         for feature in features:
             plt.figure(figsize=(10, 15))
             ax1 = plt.subplot(len(features), 2, i)
             sns.distplot(df[df['Malware']==1][feature], ax=ax1, kde_kws={'bw': 0.1})
             ax1.set_title(f'Malware', fontsize=10)
             ax2 = plt.subplot(len(features), 2, i+1)
             sns.distplot(df[df['Malware']==0][feature], ax=ax2, kde_kws={'bw': 0.1})
             ax2.set_title(f'Benign', fontsize=10)
             i= i+2
```

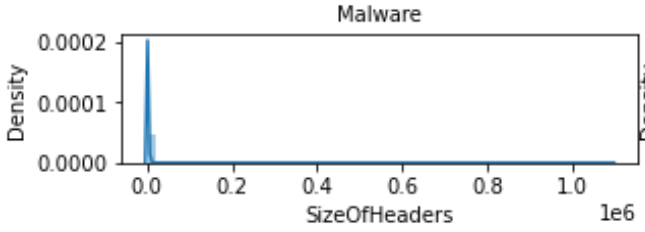


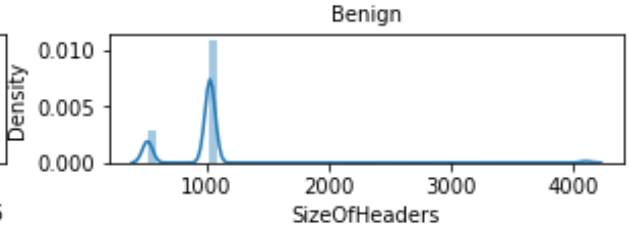

Malware
1e−8
Density
5.0
2.5
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
SizeOfCode
1e9

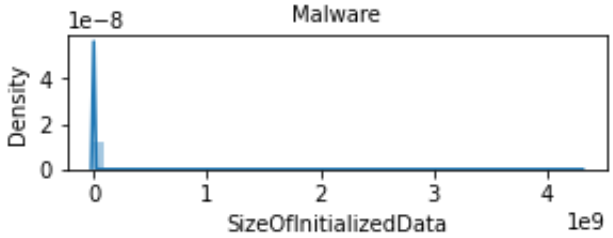

Benign
1e−6
Density
2
1
0
0 1 2 3 4
SizeOfCode
1e7

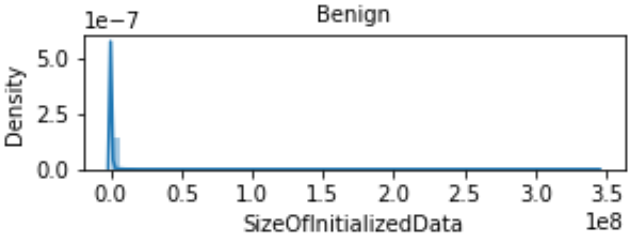

Malware
1e−8
Density
5.0
2.5
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0
SizeOfImage
1e9

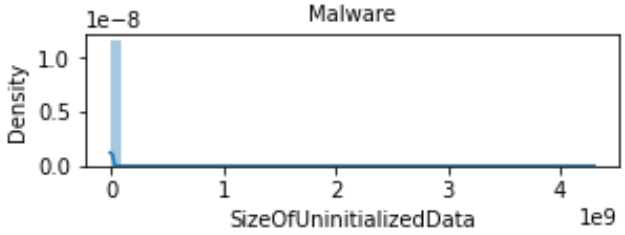

Benign
1e−7
Density
4
2
0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
SizeOfImage
1e8

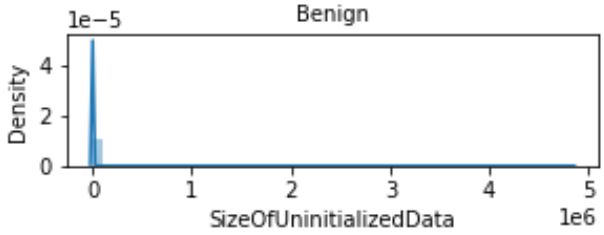

Malware
Density
0.0002
0.0001
0.0000
0.0 0.2 0.4 0.6 0.8 1.0
SizeOfHeaders
1e6

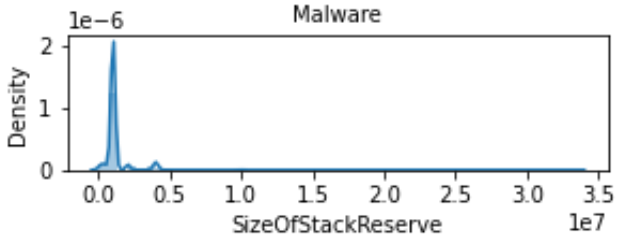

Benign
Density
0.010
0.005
0.000
1000 2000 3000 4000
SizeOfHeaders

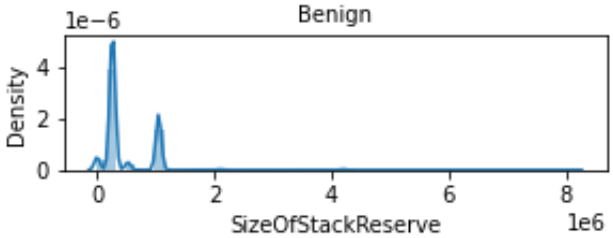

Malware
1e−8
Density
4
2
0
0 1 2 3 4
SizeOfInitializedData
1e9

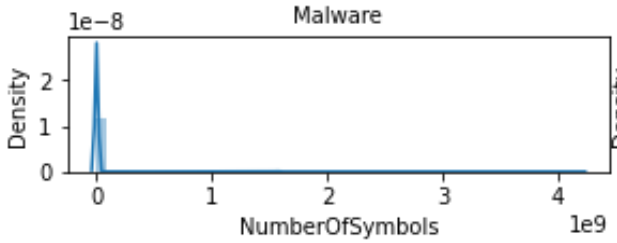

Benign
1e−7
Density
5.0
2.5
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
SizeOfInitializedData
1e8

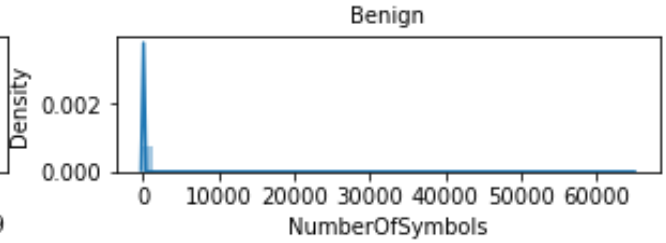

Malware
1e−8
Density
1.0
0.5
0.0
0 1 2 3 4
SizeOfUninitializedData
1e9

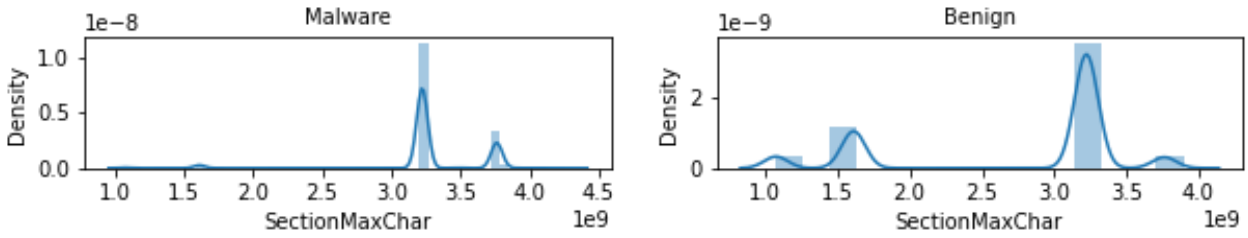

Benign
1e−5
Density
4
2
0
0 1 2 3 4 5
SizeOfUninitializedData
1e6


Malware
1e−6
Density
2
1
0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
SizeOfStackReserve
1e7

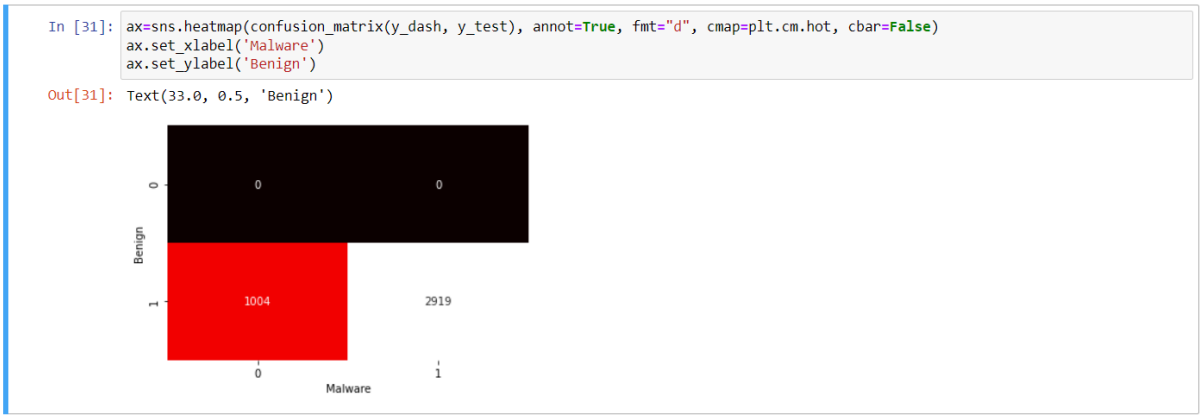

Benign
1e−6
Density
4
2
0
0 2 4 6 8
SizeOfStackReserve
1e6


Malware
1e−8
Density
2
1
0
0 1 2 3 4
NumberOfSymbols
1e9


Benign
Density
0.002
0.000
0 10000 20000 30000 40000 50000 60000
NumberOfSymbols



Step 19: Plotting confusion matrix for Model producing highest accuracy (Gradient Boosting).



References

Python.org. 2021. *Download Python*. [online] Available at: <https://www.python.org/downloads/> [Accessed 15 December 2021].

Anaconda. 2021. *Anaconda | The World's Most Popular Data Science Platform*. [online] Available at: <https://www.anaconda.com/> [Accessed 15 December 2021].

Jupyter.org. 2021. *Project Jupyter*. [online] Available at: <https://jupyter.org/install> [Accessed 15 December 2021].

Kaggle.com. 2021. *Benign & Malicious PE Files*. [online] Available at: <https://www.kaggle.com/amauricio/pe-files-malwares> [Accessed 15 December 2021].