Configuration Manual

MSc Internship
MSc in Cyber Security

Sherwin Norman Xavier
StudentID:19126662

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Sahni

Configuration Manual

Sherwin Norman Xavier
19126662

1 Introduction

This Configuration Manual gives a step-by-step process to carry out the thesis topic "Machine Learning Approaches to Detect Browser-based Cryptojacking". The main aim of this study was to achieve higher accuracy from the two approaches using machine learning. The manual consists of the software and hardware requirements for this study. It also contains the steps to install certain tools that were used during the course of this study. The libraries and packages used in this study are also specified in the upcoming sections. Machine learning algorithms used for the Static Analysis are One-Class SVM, Isolation Forest and Local Outlier Factor while for the Dynamic Analysis are K-Nearest Neighbours (KNN), Naive Bayes Classification and Support Vector Machines (SVM).

2 System Specifications

This section describes the various specifications of the system used for the study to be carried out without any interruptions.

2.1 Hardware Requirements

As this study was conducted on the local system, the local system had the following specifications:

1. System OS – Windows 10 Home 64-bit
2. Processor – Intel Core i5 8th Gen @ 1.80 GHz
3. Ram – 12 GB DDR4 @ 2400 MHz
4. Hard Disk Drive – 1 TB 5400 rpm SATA

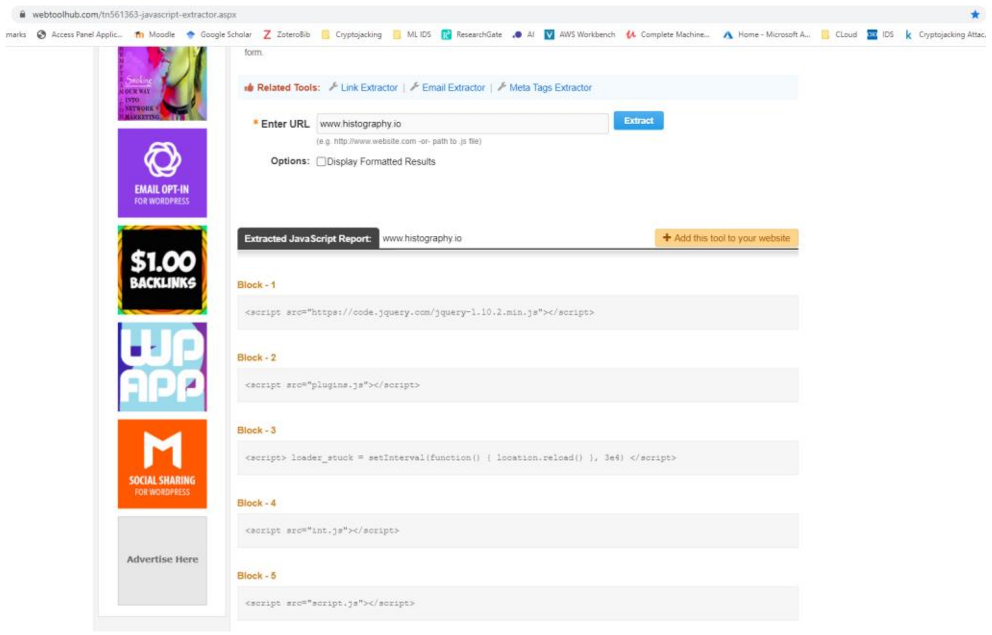
2.2 Software Requirements

1. Python 3.8 – This is a programming language that was used throughout this study from pre-processing the data to evaluating the machine learning models for the two approaches.
2. Jupyter Notebook – This is an open-source web application that helps to code and execute Python and also helps to visualize the data [1]. For this study, the Jupyter Notebook 6.0.3 was used to execute Python 3.8.
3. Node.js 12.18.2 – This is an open-source, cross-platform tool that is used to run a JavaScript code outside a browser [2]. This tool is used to run the tool Plato for this study.
4. Plato – This is an open-source tool available on GitHub. It is used to visualize the complexity metrics of a JavaScript code [3]. This tool is used to build the dataset used for the static analysis.
5. Microsoft Excel – This tool is used to import the datasets. All the datasets used for this study is used in .csv file format. This tool was also used to visualise the results of the evaluation of the different models.

3 Data Collection & Merging

3.1 Static Analysis

- To create the static dataset, we first collect scripts for both Cryptojacking and benign. Benign scripts are extracted from websites using [1]. Add the URL of a JavaScript webpage to the search box and hit Extract. The output of the tool is shown in Fig. 1. Fig. 2 and 3 are examples of the benign and cryptojacking scripts respectively.


webtoolhub.com/tn561363-javascript-extractor.aspx

form.

Related Tools: Link Extractor | Email Extractor | Meta Tags Extractor

* Enter URL: www.histography.io [Extract]

(e.g. http://www.website.com -or- path to .js file)

Options: ☐ Display Formatted Results

Extracted JavaScript Report: www.histography.io + Add this tool to your website

Block - 1

```
<script src="https://code.jquery.com/jquery-1.10.2.min.js"></script>
```

Block - 2

```
<script src="plugins.js"></script>
```

Block - 3

```
<script> loader_stuck = setInterval(function() { location.reload() }, 3e4) </script>
```

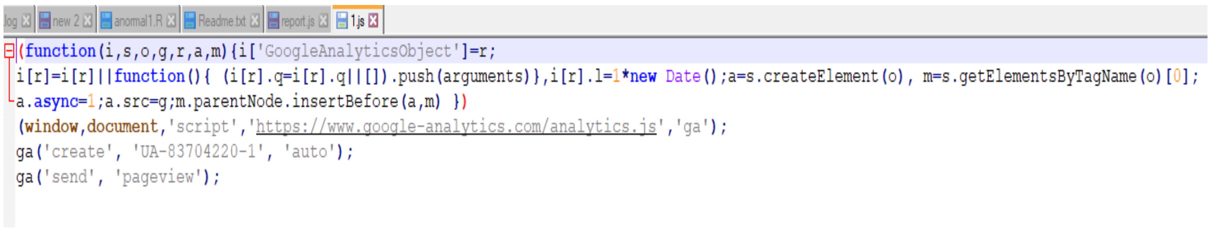
Block - 4

```
<script src="int.js"></script>
```

Block - 5

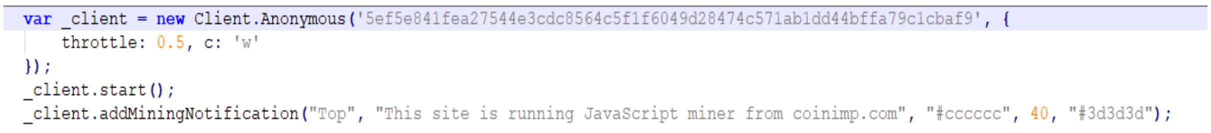
```
<script src="script.js"></script>
```

Figure 1: JavaScript Extractor Tool



```
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;
i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];
a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })
(window,document,'script','https://www.google-analytics.com/analytics.js','ga');
ga('create', 'UA-83704220-1', 'auto');
ga('send', 'pageview');
```

Figure 2: Example of Benign JavaScript



```
var _client = new Client.Anonymous('5ef5e841fea27544e3cdc8564c5f1f6049d28474c571ab1dd44bffa79c1cbaf9', {
    throttle: 0.5, c: 'w'
});
_client.start();
_client.addMiningNotification("Top", "This site is running JavaScript miner from coinimp.com", "#cccccc", 40, "#3d3d3d");
```

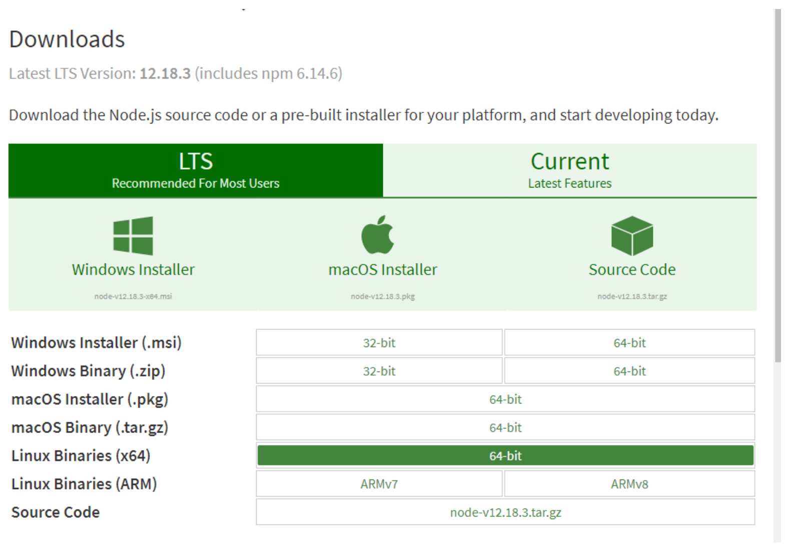
Figure 3: Example of Cryptojacking Script

- Node.js was used to install and run the tool Plato to collect the complexity data from both the benign and cryptojacking JavaScript codes. Mentioned below were the steps followed to carry out the extraction process.

1. Download Node.js from this site[2]. Select Windows Installer to download for the Windows machine as seen in Fig. 4. Once the download is complete carry on with

[1] https://www.webtoolhub.com/tn561363-javascript-extractor.aspx
[2] https://nodejs.org/en/download/

the installation of Node.js by following the steps shown on the installation wizard & select a suitable location to install the files.


Downloads

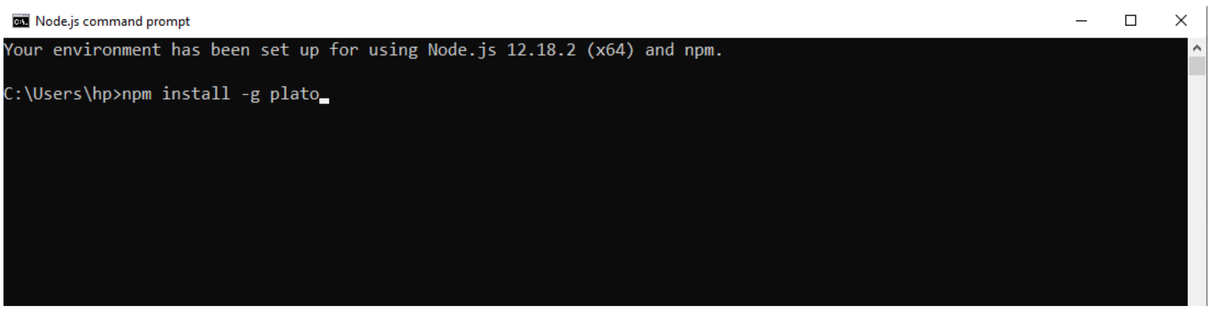
Latest LTS Version: **12.18.3** (includes npm 6.14.6)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v12.18.3.tar.gz	

Figure 4: Download Page of Node.js

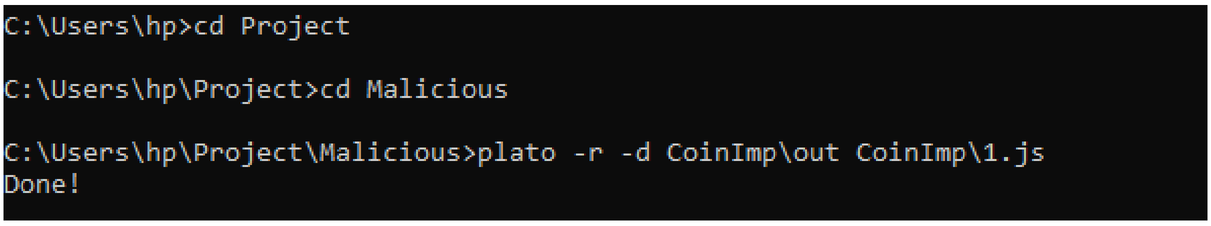
2. Install the tool Plato using the command shown in Fig.5.



```
Your environment has been set up for using Node.js 12.18.2 (x64) and npm.

C:\Users\hp>npm install -g plato_
```

Figure 5: Installation of Plato

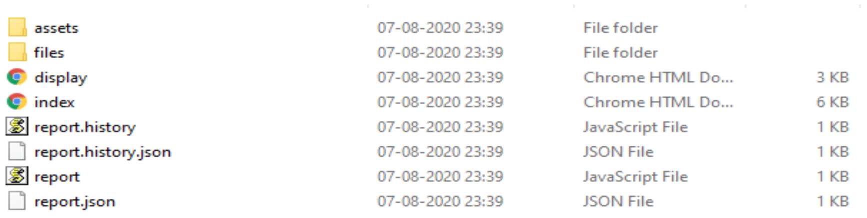
3. Once the installation of the tool is complete, change the working directory to the directory where the JavaScript files to be analysed are located. Then run the command as shown in Fig.6.



```
C:\Users\hp>cd Project

C:\Users\hp\Project>cd Malicious

C:\Users\hp\Project\Malicious>plato -r -d CoinImp\out CoinImp\1.js
Done!
```

Figure 6: Running of Plato

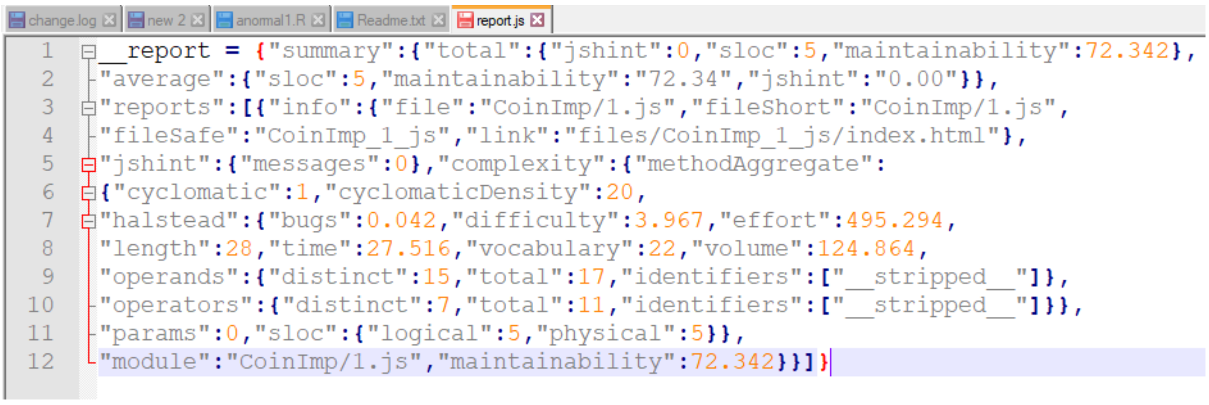
4. On opening the `out` folder that is created upon running Plato on a file, different files are available as shown in Fig.7.



assets	07-08-2020 23:39	File folder	
files	07-08-2020 23:39	File folder	
display	07-08-2020 23:39	Chrome HTML Do...	3 KB
index	07-08-2020 23:39	Chrome HTML Do...	6 KB
report.history	07-08-2020 23:39	JavaScript File	1 KB
report.history.json	07-08-2020 23:39	JSON File	1 KB
report	07-08-2020 23:39	JavaScript File	1 KB
report.json	07-08-2020 23:39	JSON File	1 KB

Figure 7: Files from the Output of Plato

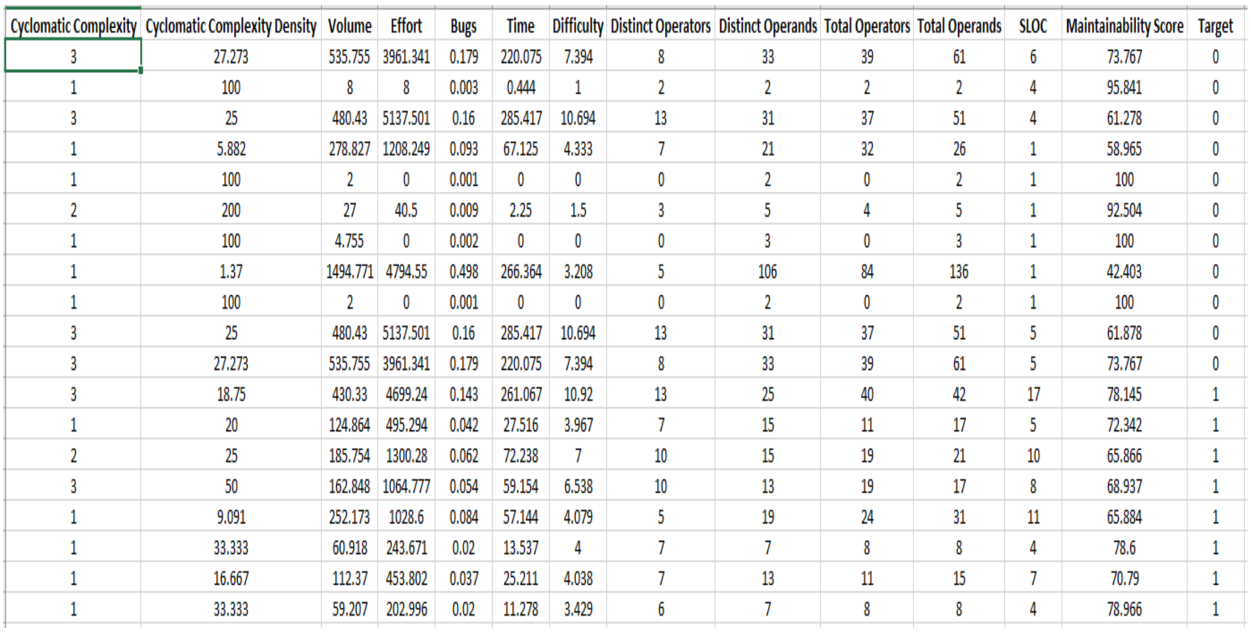
5. Using Notepad++ we open the report.js and the following output is delivered to us.



```
__report = {"summary":{"total":{"jshint":0,"sloc":5,"maintainability":72.342},
"average":{"sloc":5,"maintainability":"72.34","jshint":"0.00"}},
"reports":[{"info":{"file":"CoinImp/1.js","fileShort":"CoinImp/1.js",
"fileSafe":"CoinImp_1_js","link":"files/CoinImp_1_js/index.html"},
"jshint":{"messages":0},"complexity":{"methodAggregate":
{"cyclomatic":1,"cyclomaticDensity":20,
"halstead":{"bugs":0.042,"difficulty":3.967,"effort":495.294,
"length":28,"time":27.516,"vocabulary":22,"volume":124.864,
"operands":{"distinct":15,"total":17,"identifiers":["__stripped__"]},
"operators":{"distinct":7,"total":11,"identifiers":["__stripped__"]}},
"params":0,"sloc":{"logical":5,"physical":5}},
"module":"CoinImp/1.js","maintainability":72.342}}]}
```

Figure 8: Complexity Report of CoinImp Script

6. Using the output from the Plato tool the values of the different features shown in the output have been added using Microsoft Excel to create the dataset for the static analysis. The dataset is shown in figure 6 with an added column `Target` to differentiate between the complexity data of benign and cryptojacking scripts where 0 is benign and 1 is malicious.



Cyclomatic Complexity	Cyclomatic Complexity Density	Volume	Effort	Bugs	Time	Difficulty	Distinct Operators	Distinct Operands	Total Operators	Total Operands	SLOC	Maintainability Score	Target
3	27.273	535.755	3961.341	0.179	220.075	7.394	8	33	39	61	6	73.767	0
1	100	8	8	0.003	0.444	1	2	2	2	2	4	95.841	0
3	25	480.43	5137.501	0.16	285.417	10.694	13	31	37	51	4	61.278	0
1	5.882	278.827	1208.249	0.093	67.125	4.333	7	21	32	26	1	58.965	0
1	100	2	0	0.001	0	0	0	2	0	2	1	100	0
2	200	27	40.5	0.009	2.25	1.5	3	5	4	5	1	92.504	0
1	100	4.755	0	0.002	0	0	0	3	0	3	1	100	0
1	1.37	1494.771	4794.55	0.498	266.364	3.208	5	106	84	136	1	42.403	0
1	100	2	0	0.001	0	0	0	2	0	2	1	100	0
3	25	480.43	5137.501	0.16	285.417	10.694	13	31	37	51	5	61.878	0
3	27.273	535.755	3961.341	0.179	220.075	7.394	8	33	39	61	5	73.767	0
3	18.75	430.33	4699.24	0.143	261.067	10.92	13	25	40	42	17	78.145	1
1	20	124.864	495.294	0.042	27.516	3.967	7	15	11	17	5	72.342	1
2	25	185.754	1300.28	0.062	72.238	7	10	15	19	21	10	65.866	1
3	50	162.848	1064.777	0.054	59.154	6.538	10	13	19	17	8	68.937	1
1	9.091	252.173	1028.6	0.084	57.144	4.079	5	19	24	31	11	65.884	1
1	33.333	60.918	243.671	0.02	13.537	4	7	7	8	8	4	78.6	1
1	16.667	112.37	453.802	0.037	25.211	4.038	7	13	11	15	7	70.79	1
1	33.333	59.207	202.996	0.02	11.278	3.429	6	7	8	8	4	78.966	1

Figure 9: Dataset for Static Analysis

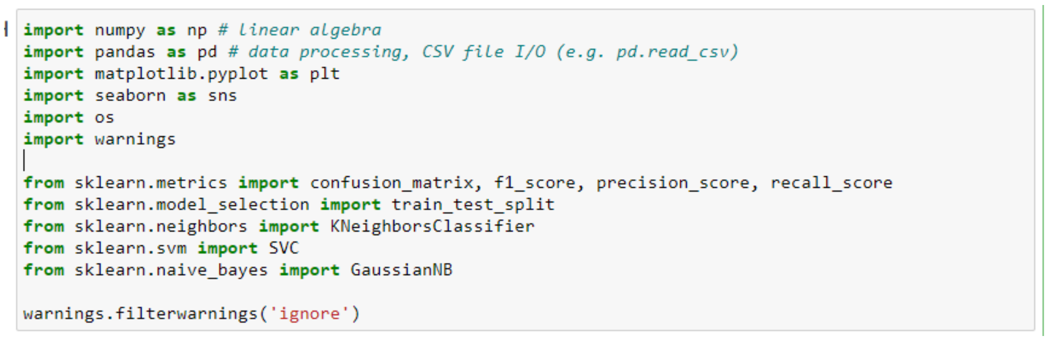
3.2 Dynamic Analysis

The dataset used for this approach is made up by merging two datasets i.e. `final-normal-data-set.csv` and `final-anormal-data-set.csv` in Microsoft Excel. After merging the two datasets, a new column `target` is added to the dataset with values 0 and 1 which represent the normal dataset and abnormal dataset respectively.

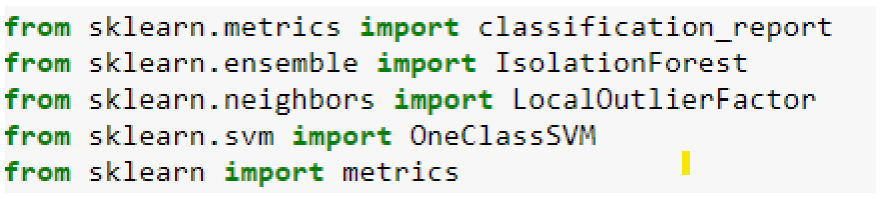
4 Data Processing and Evaluation

For both approaches, different codes were used to carry out the processing and evaluation of the different models used. The source code used for the static analysis [4] and dynamic analysis [5] are available on Kaggle.

4.1 Importing Packages



```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings

from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

warnings.filterwarnings('ignore')
```



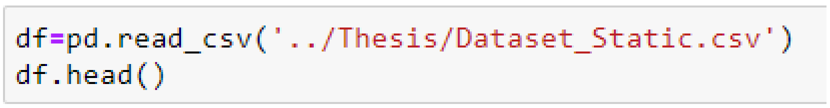
```
from sklearn.metrics import classification_report
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from sklearn import metrics
```

The following libraries have been used in studying the approach for Dynamic Analysis:

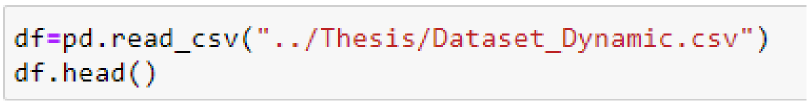
- **Numpy**: NumPy is a library in Python to work with arrays. It offers multiple resources to interact with arrays with maximum performance.
- **Pandas**: This library in Python is used for data processing. It is flexible to use as well as a good manipulation tool for data analysis.
- **Matplotlib**: This library in python is used to create an interactive visualisation of data.
- **Seaborn**: This library is based on matplotlib. It is used for creating a high-level interface for data visualisation.

Also, the packages required for the models to be built are imported along with the metrics to be used in this study.

4.2 Importing the Dataset



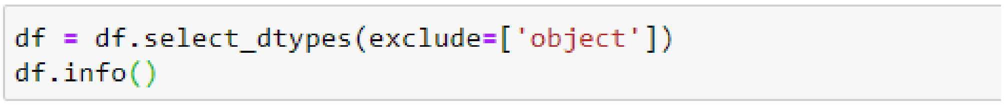
```
df=pd.read_csv('../Thesis/Dataset_Static.csv')
df.head()
```



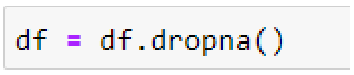
```
df=pd.read_csv("../Thesis/Dataset_Dynamic.csv")
df.head()
```

4.3 Data Cleaning

Some data in the dynamic dataset required some cleaning as they were not useful for this study. Hence, we take out the data that is not numeric and eliminate rows with any null value in the data set for better results from the machine learning models.



```
df = df.select_dtypes(exclude=['object'])
df.info()
```



```
df = df.dropna()
```

4.4 Descriptive Statistics of the Dataset

The descriptive statistics of the data is used to understand the data.

```
normal = df.loc[df['Target'] == 0, :]
normal.describe()
```

```
abnormal = df.loc[df['Target'] == 1, :]
abnormal.describe()
```

```
df.describe()
```

4.5 Feature Extraction and EDA

4.5.1 Static Analysis

- **KDE Plotting**

```
features = df.columns[:-1]

plt.figure(figsize=(20,50))
i=1
for feature in features:
    plt.subplot(11,3,i)
    sns.kdeplot(normal[feature], shade= True)
    sns.kdeplot(abnormal[feature], shade= True)
    i=i+1
    plt.tight_layout()
plt.show()
```

- **Histogram**

```
plt.figure(figsize=(20,50))
i=1
for feature in normal.columns:
    plt.subplot(11,3,i)
    plt.hist(normal[feature], bins = 100)
    plt.hist(abnormal[feature],bins = 100)
    plt.title(feature)
    i=i+1
plt.tight_layout()
plt.show()
```

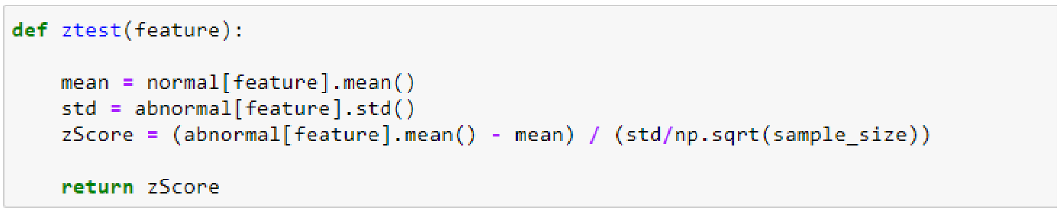
4.5.2 Dynamic Analysis

- **Multicollinearity Principle**

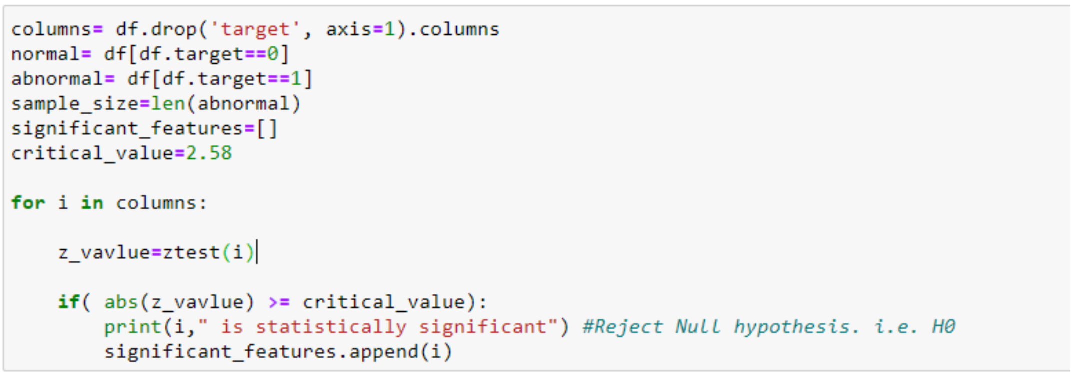
```
f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(df.corr(), annot=True, linewidths=.5, cmap="YlGnBu",fmt= '.1f',ax=ax)
plt.title('Corelation Map')
plt.show()
```

- **Two-Tailed Z Test:**

The source code for the Two-Tailed Z Test is used from Kaggle [6].



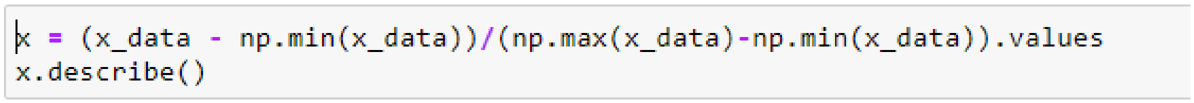
```
def ztest(feature):

    mean = normal[feature].mean()
    std = abnormal[feature].std()
    zScore = (abnormal[feature].mean() - mean) / (std/np.sqrt(sample_size))

    return zScore
```



```
columns= df.drop('target', axis=1).columns
normal= df[df.target==0]
abnormal= df[df.target==1]
sample_size=len(abnormal)
significant_features=[]
critical_value=2.58

for i in columns:

    z_vavlue=ztest(i)

    if( abs(z_vavlue) >= critical_value):
        print(i," is statistically significant") #Reject Null hypothesis. i.e. H0
        significant_features.append(i)
```

4.6 Data Normalization

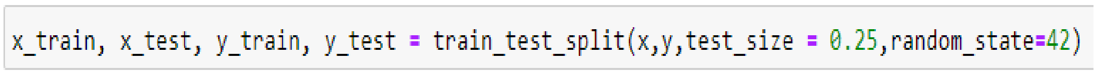
The data for the dynamic analysis is normalized to handle unbalanced features and brings the values of data between 0 and 1.



```
x = (x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data)).values
x.describe()
```

4.7 Splitting of Data

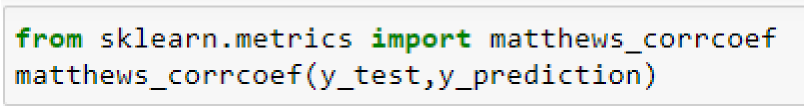
Data for the dynamic analysis is split between training data and test data for all the three supervised models in the ratio of 75:25



```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25,random_state=42)
```

4.8 Statistical Test

Matthew's Correlation Coefficient is used as a statistical test to support the results of this study.



```
from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_test,y_prediction)
```

References

[1] “Project Jupyter.” [Online]. Available: https://www.jupyter.org. [Accessed: 05-Aug-2020]

[2] “Run JavaScript Everywhere.,” *Run JavaScript Everywhere.* [Online]. Available: https://nodejs.dev/. [Accessed: 05-Aug-2020]

[3] *es-analysis/plato*. es-analysis, 2020 [Online]. Available: https://github.com/es-analysis/plato. [Accessed: 05-Aug-2020]

[4] “Anomaly Detection with Unsupervised Learning.” [Online]. Available: https://kaggle.com/jiedong00/anomaly-detection-with-unsupervised-learning. [Accessed: 05-Aug-2020]

[5] “Application of Supervised ML Classifications.” [Online]. Available: https://kaggle.com/muhammetcakmak/application-of-supervised-ml-classifications. [Accessed: 05-Aug-2020]

[6] “Anomaly Detection using Unsupervised Techniques.” [Online]. Available: https://kaggle.com/sabanasimbutt/anomaly-detection-using-unsupervised-techniques. [Accessed: 05-Aug-2020]