Configuration Manual

MSc Research Project
Cyber Security

Ahmed Alazawy
Student ID: x23158352

School of Computing
National College of Ireland

Supervisor: Eugene McLaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ahmed Alazawy		
Student ID:	X23158352		
Programme:	MSc Cybersecurity	**Year:**	2023/2024
Module:	Practicum Part 2		
Lecturer:	Mr. Eugene McLaughlin		
Submission Due Date:	12th of August		
Project Title:	Effectiveness of Supervised and Unsupervised algorithms in detecting RAP' in Wireless Networks		
Word Count:	507	**Page Count:**	11

Signature: Ahmed A

Date: 11th of August

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

Ahmed Alazawy
Student ID: x23158352

1 Introduction

This paper discusses how to setup and develop an environment which uses machine learning techniques to identify and detect rogue access points on a publicly used dataset.

2 System Details

The system that was used to facilitate various experiments and development of machine learning classifiers to detect rogue access points in wireless networks was made in:

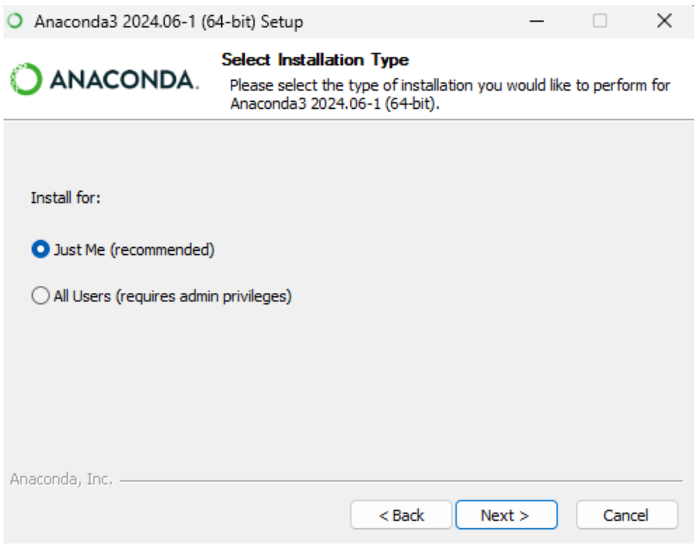
- ASUS Vivobook Pro 16X 4K laptop
- Intel Core i7-11370H
- NVIDIA GeForce RTX 3050
- 16GB RAM
- 1TB SSD
- Windows 11

The software that was used as well is:
- Jupyter Notebook version 6.5.4
- Python version 3.11.5
- Visual Studio Code version 1.91

3 Software Setup

This section describes the process in setting up any needed software tools.

1. Prerequisite is having installed python in order to install Jupyter notebook to develop the code and a good way to do this is by installing Anaconda to install python and jupyter. You can do this by going to the following page: https://www.anaconda.com/download/success.
2. Go through the installation with Anaconda and choose just me for installation type as seen in the figure below:



Anaconda3 2024.06-1 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2024.06-1 (64-bit).

Install for:

[✓ Just Me (recommended)]

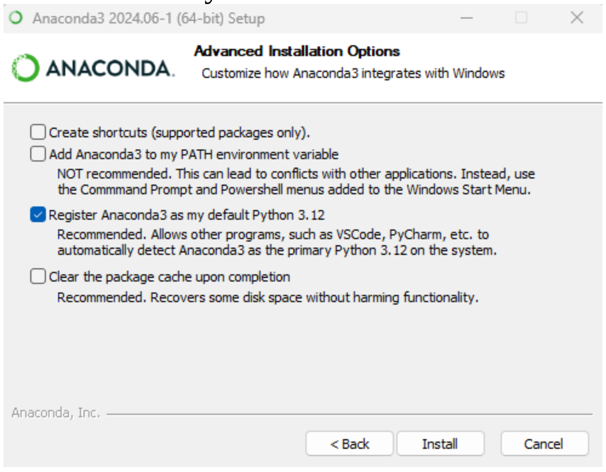
All Users (requires admin privileges)

Anaconda, Inc.

< Back | Next > | Cancel

Fig 1 – Anaconda Installation Part 1

3. Then choose to register Anaconda3 as the main default python version so that it can be used alongside VSCode smoothly.



Anaconda3 2024.06-1 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda3 integrates with Windows

☐ Create shortcuts (supported packages only).

☐ Add Anaconda3 to my PATH environment variable

NOT recommended. This can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the Windows Start Menu.

☑ Register Anaconda3 as my default Python 3.12

Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Anaconda3 as the primary Python 3.12 on the system.

☐ Clear the package cache upon completion

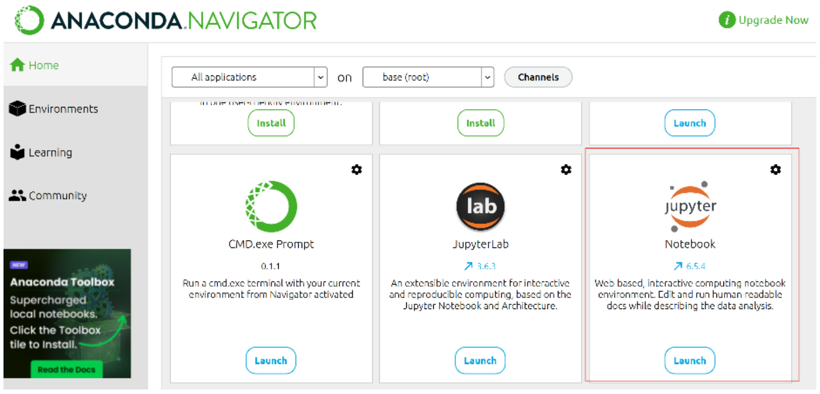
Recommended. Recovers some disk space without harming functionality.

Anaconda, Inc.

< Back | Install | Cancel

Fig 2 – Anaconda Installation Part 2

4. Once Anaconda has been installed alongside python, anaconda navigator will become available so we can easily open VSCode and mainly Jupyter notebook from there and start working on it.



ANACONDA.NAVIGATOR

Upgrade Now

Home | Environments | Learning | Community

NEW Anaconda Toolbox Supercharged local notebooks. Click the Toolbox tile to install. Read the Docs

All applications on base (root) Channels

CMD.exe Prompt 0.1.1 — Run a cmd.exe terminal with your current environment from Navigator activated — Launch

JupyterLab 3.6.3 — An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. — Launch

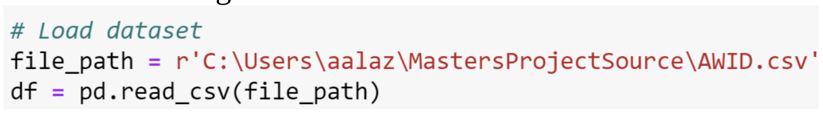
Notebook 6.5.4 — Web based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. — Launch

Figure 3 – Anaconda Navigator

5. The rest of the work will be done in Jupyter notebook to keep the development process simple yet efficient.

4 Dataset Setup

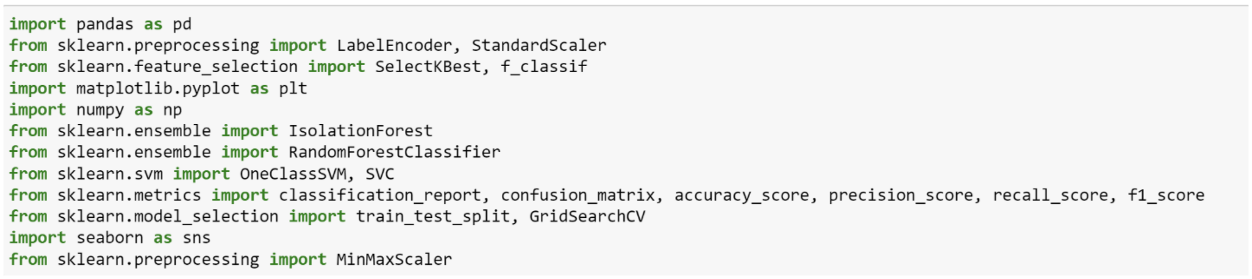
The dataset that was used was found from the open source software platform known as Github and the dataset that was used was the Aegean AWID Dataset and using pandas library and loaded in as shown in the figure below:



```
# Load dataset
file_path = r'C:\Users\aalaz\MastersProjectSource\AWID.csv'
df = pd.read_csv(file_path)
```

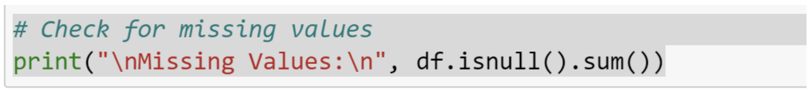
Figure 4 – Loading in AWID Dataset from the MastersProjectSource Folder.

5 Implementation



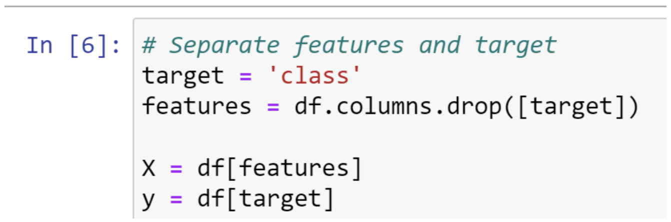
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import OneClassSVM, SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split, GridSearchCV
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

Figure 5 – Importing all libraries needed



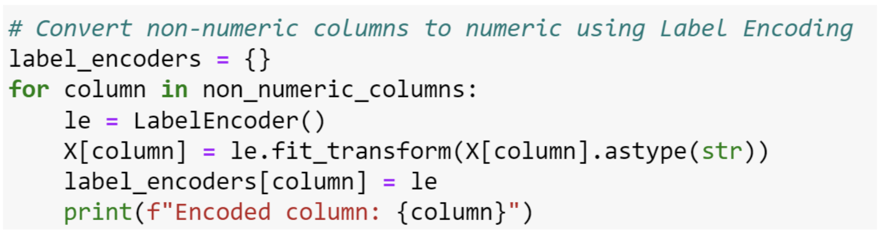
```
# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())
```

Figure 6 – checking to see if there are any missing values in dataset



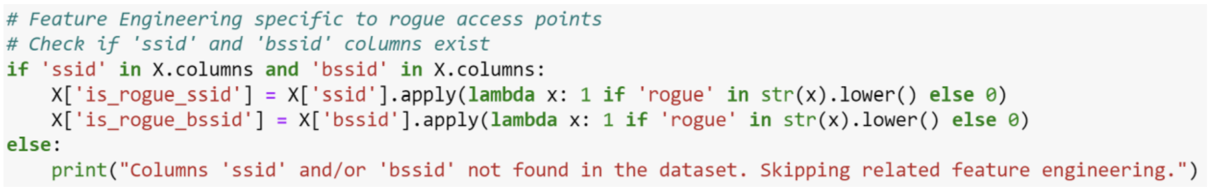
```
In [6]: # Separate features and target
        target = 'class'
        features = df.columns.drop([target])

        X = df[features]
        y = df[target]
```

Figure 7 – separating the features and the classes



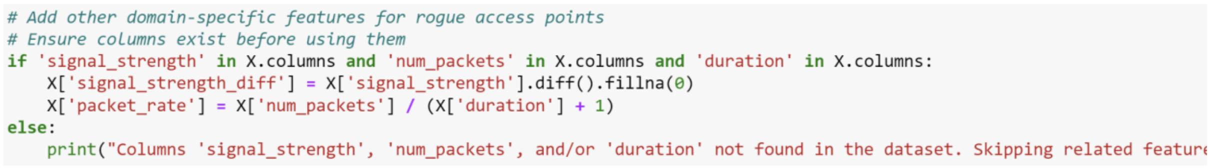
```
# Convert non-numeric columns to numeric using Label Encoding
label_encoders = {}
for column in non_numeric_columns:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column].astype(str))
    label_encoders[column] = le
    print(f"Encoded column: {column}")
```

Figure 8 – using label encoding to change the columns from categorical to numeric



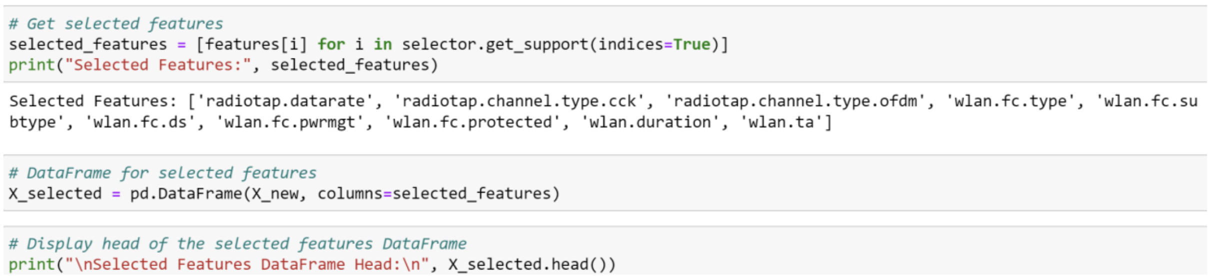
```
# Feature Engineering specific to rogue access points
# Check if 'ssid' and 'bssid' columns exist
if 'ssid' in X.columns and 'bssid' in X.columns:
    X['is_rogue_ssid'] = X['ssid'].apply(lambda x: 1 if 'rogue' in str(x).lower() else 0)
    X['is_rogue_bssid'] = X['bssid'].apply(lambda x: 1 if 'rogue' in str(x).lower() else 0)
else:
    print("Columns 'ssid' and/or 'bssid' not found in the dataset. Skipping related feature engineering.")
```

Figure 9 – RAP specific feature engineering



```python
# Add other domain-specific features for rogue access points
# Ensure columns exist before using them
if 'signal_strength' in X.columns and 'num_packets' in X.columns and 'duration' in X.columns:
    X['signal_strength_diff'] = X['signal_strength'].diff().fillna(0)
    X['packet_rate'] = X['num_packets'] / (X['duration'] + 1)
else:
    print("Columns 'signal_strength', 'num_packets', and/or 'duration' not found in the dataset. Skipping related feature
```

Figure 10 – More RAP specific feature engineering



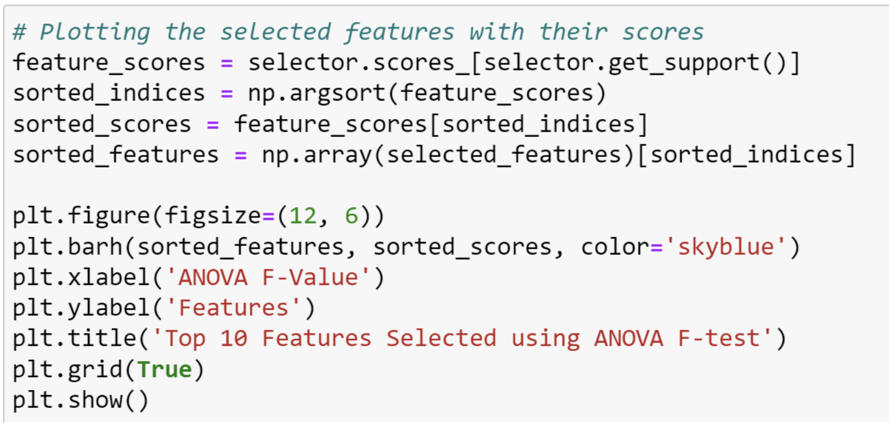
```python
# Get selected features
selected_features = [features[i] for i in selector.get_support(indices=True)]
print("Selected Features:", selected_features)
```

```
Selected Features: ['radiotap.datarate', 'radiotap.channel.type.cck', 'radiotap.channel.type.ofdm', 'wlan.fc.type', 'wlan.fc.su
btype', 'wlan.fc.ds', 'wlan.fc.pwrmgt', 'wlan.fc.protected', 'wlan.duration', 'wlan.ta']
```

```python
# DataFrame for selected features
X_selected = pd.DataFrame(X_new, columns=selected_features)
```

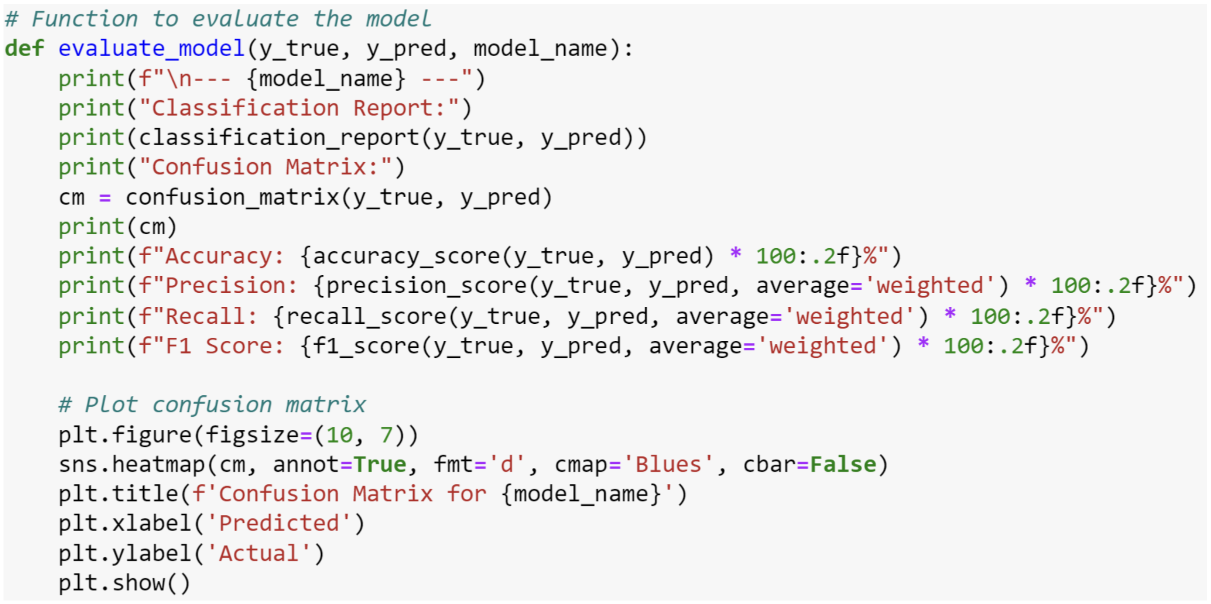
```python
# Display head of the selected features DataFrame
print("\nSelected Features DataFrame Head:\n", X_selected.head())
```

Figure 11 – Acquiring the selected features



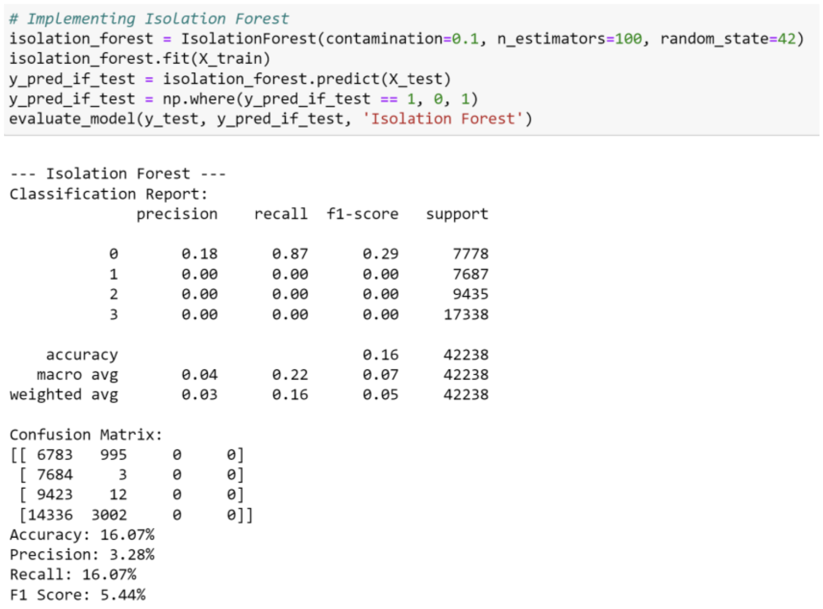
```python
# Plotting the selected features with their scores
feature_scores = selector.scores_[selector.get_support()]
sorted_indices = np.argsort(feature_scores)
sorted_scores = feature_scores[sorted_indices]
sorted_features = np.array(selected_features)[sorted_indices]

plt.figure(figsize=(12, 6))
plt.barh(sorted_features, sorted_scores, color='skyblue')
plt.xlabel('ANOVA F-Value')
plt.ylabel('Features')
plt.title('Top 10 Features Selected using ANOVA F-test')
plt.grid(True)
plt.show()
```

Figure 12 – Plotting the features that we selected and choosing the top 10 best features based on ANOVA F-test



```python
# Function to evaluate the model
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n--- {model_name} ---")
    print("Classification Report:")
    print(classification_report(y_true, y_pred))
    print("Confusion Matrix:")
    cm = confusion_matrix(y_true, y_pred)
    print(cm)
    print(f"Accuracy: {accuracy_score(y_true, y_pred) * 100:.2f}%")
    print(f"Precision: {precision_score(y_true, y_pred, average='weighted') * 100:.2f}%")
    print(f"Recall: {recall_score(y_true, y_pred, average='weighted') * 100:.2f}%")
    print(f"F1 Score: {f1_score(y_true, y_pred, average='weighted') * 100:.2f}%")

    # Plot confusion matrix
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

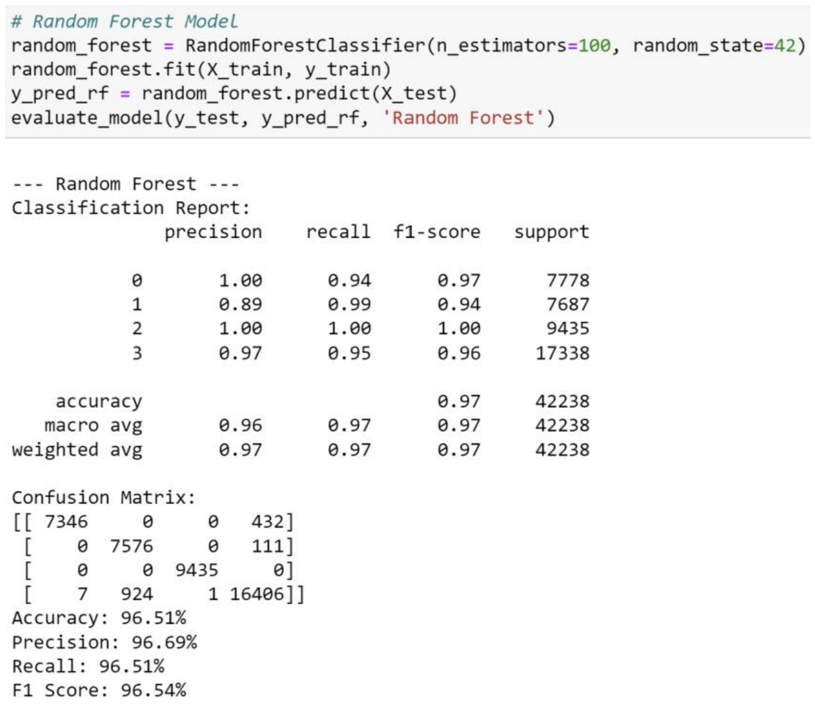
Figure 13 – evaluating the models based on classification metrics and confusion matrix



```python
# Implementing Isolation Forest
isolation_forest = IsolationForest(contamination=0.1, n_estimators=100, random_state=42)
isolation_forest.fit(X_train)
y_pred_if_test = isolation_forest.predict(X_test)
y_pred_if_test = np.where(y_pred_if_test == 1, 0, 1)
evaluate_model(y_test, y_pred_if_test, 'Isolation Forest')
```

```
--- Isolation Forest ---
Classification Report:
              precision    recall  f1-score   support

           0       0.18      0.87      0.29      7778
           1       0.00      0.00      0.00      7687
           2       0.00      0.00      0.00      9435
           3       0.00      0.00      0.00     17338

    accuracy                           0.16     42238
   macro avg       0.04      0.22      0.07     42238
weighted avg       0.03      0.16      0.05     42238

Confusion Matrix:
[[ 6783   995     0     0]
 [ 7684     3     0     0]
 [ 9423    12     0     0]
 [14336  3002     0     0]]
Accuracy: 16.07%
Precision: 3.28%
Recall: 16.07%
F1 Score: 5.44%
```

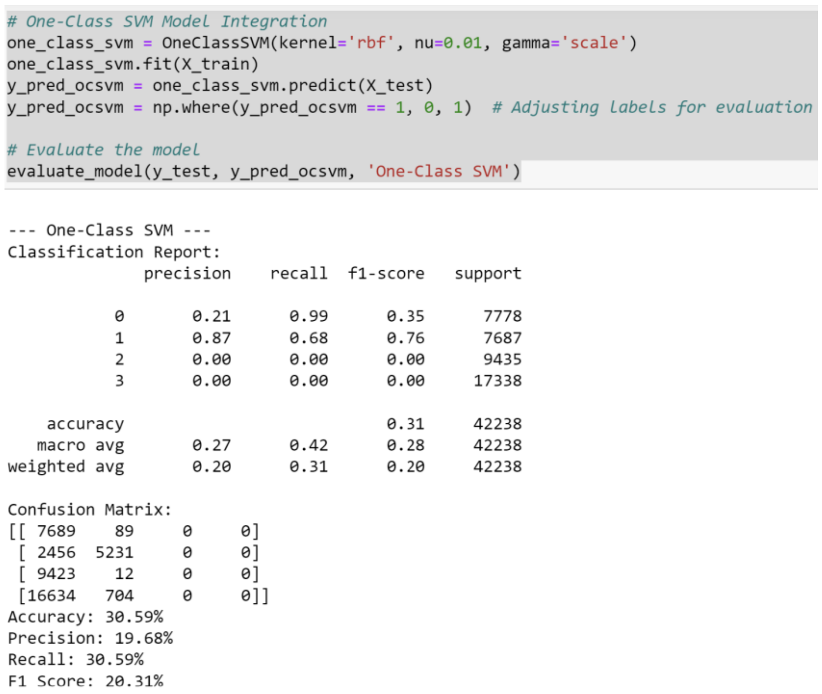
Figure 14 – Isolation Forest Implementation



```python
# Random Forest Model
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train)
y_pred_rf = random_forest.predict(X_test)
evaluate_model(y_test, y_pred_rf, 'Random Forest')
```

```
--- Random Forest ---
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97      7778
           1       0.89      0.99      0.94      7687
           2       1.00      1.00      1.00      9435
           3       0.97      0.95      0.96     17338

    accuracy                           0.97     42238
   macro avg       0.96      0.97      0.97     42238
weighted avg       0.97      0.97      0.97     42238

Confusion Matrix:
[[ 7346     0     0   432]
 [    0  7576     0   111]
 [    0     0  9435     0]
 [    7   924     1 16406]]
Accuracy: 96.51%
Precision: 96.69%
Recall: 96.51%
F1 Score: 96.54%
```

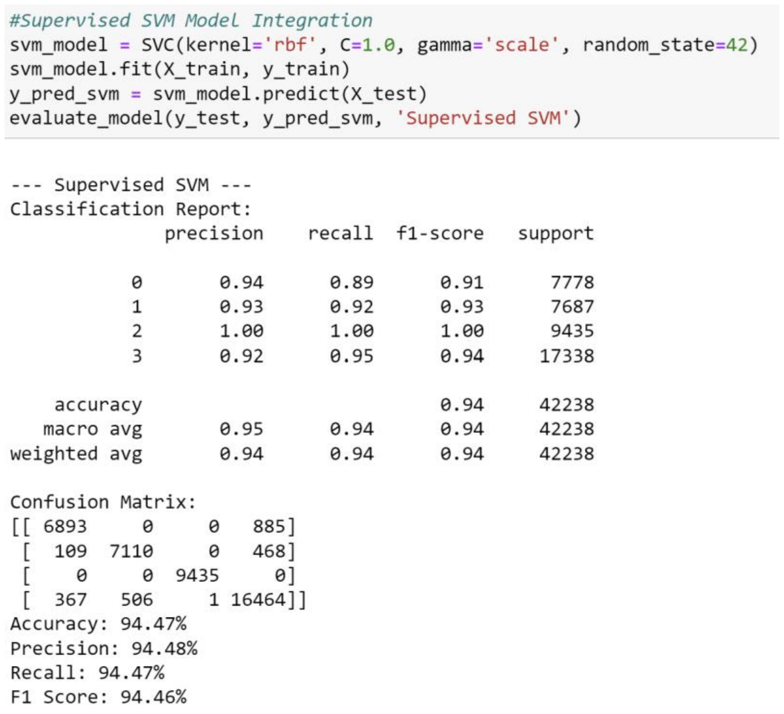
Figure 15 – Random Forest Implementation



```python
# One-Class SVM Model Integration
one_class_svm = OneClassSVM(kernel='rbf', nu=0.01, gamma='scale')
one_class_svm.fit(X_train)
y_pred_ocsvm = one_class_svm.predict(X_test)
y_pred_ocsvm = np.where(y_pred_ocsvm == 1, 0, 1)  # Adjusting labels for evaluation

# Evaluate the model
evaluate_model(y_test, y_pred_ocsvm, 'One-Class SVM')
```

```
--- One-Class SVM ---
Classification Report:
              precision    recall  f1-score   support

           0       0.21      0.99      0.35      7778
           1       0.87      0.68      0.76      7687
           2       0.00      0.00      0.00      9435
           3       0.00      0.00      0.00     17338

    accuracy                           0.31     42238
   macro avg       0.27      0.42      0.28     42238
weighted avg       0.20      0.31      0.20     42238

Confusion Matrix:
[[ 7689    89     0     0]
 [ 2456  5231     0     0]
 [ 9423    12     0     0]
 [16634   704     0     0]]
Accuracy: 30.59%
Precision: 19.68%
Recall: 30.59%
F1 Score: 20.31%
```

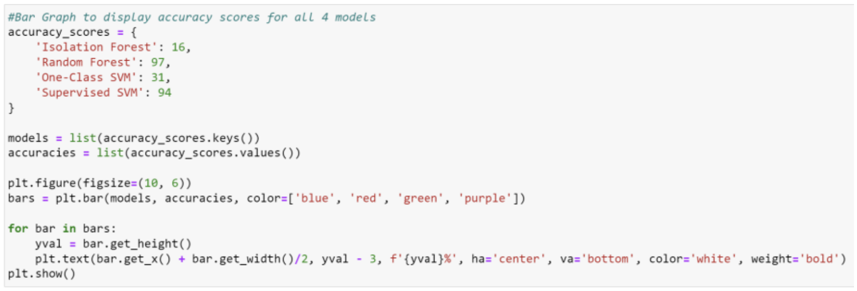
Figure 16 – One-Class SVM model Implementation



```python
#Supervised SVM Model Integration
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
evaluate_model(y_test, y_pred_svm, 'Supervised SVM')
```

```
--- Supervised SVM ---
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.91      7778
           1       0.93      0.92      0.93      7687
           2       1.00      1.00      1.00      9435
           3       0.92      0.95      0.94     17338

    accuracy                           0.94     42238
   macro avg       0.95      0.94      0.94     42238
weighted avg       0.94      0.94      0.94     42238

Confusion Matrix:
[[ 6893     0     0   885]
 [  109  7110     0   468]
 [    0     0  9435     0]
 [  367   506     1 16464]]
Accuracy: 94.47%
Precision: 94.48%
Recall: 94.47%
F1 Score: 94.46%
```

Figure 17 – Supervised SVM model implementation



```
#Bar Graph to display accuracy scores for all 4 models
accuracy_scores = {
    'Isolation Forest': 16,
    'Random Forest': 97,
    'One-Class SVM': 31,
    'Supervised SVM': 94
}

models = list(accuracy_scores.keys())
accuracies = list(accuracy_scores.values())

plt.figure(figsize=(10, 6))
bars = plt.bar(models, accuracies, color=['blue', 'red', 'green', 'purple'])

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval - 3, f'{yval}%', ha='center', va='bottom', color='white', weight='bold')
plt.show()
```

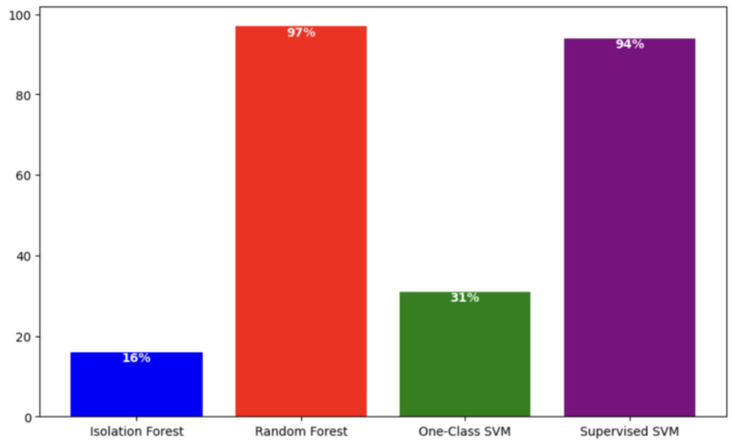
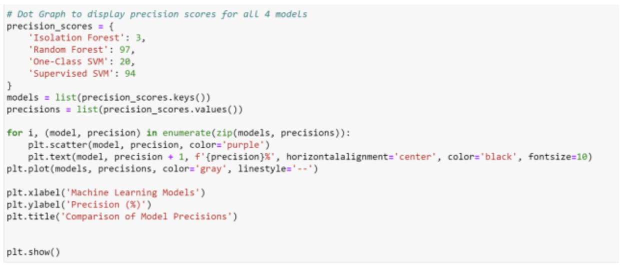


Figure 18 – Bar Graph for accuracy scores



```
# Dot Graph to display precision scores for all 4 models
precision_scores = {
    'Isolation Forest': 3,
    'Random Forest': 97,
    'One-Class SVM': 20,
    'Supervised SVM': 94
}
models = list(precision_scores.keys())
precisions = list(precision_scores.values())

for i, (model, precision) in enumerate(zip(models, precisions)):
    plt.scatter(model, precision, color='purple')
    plt.text(model, precision + 1, f'{precision}%', horizontalalignment='center', color='black', fontsize=10)
plt.plot(models, precisions, color='gray', linestyle='--')

plt.xlabel('Machine Learning Models')
plt.ylabel('Precision (%)')
plt.title('Comparison of Model Precisions')


plt.show()
```

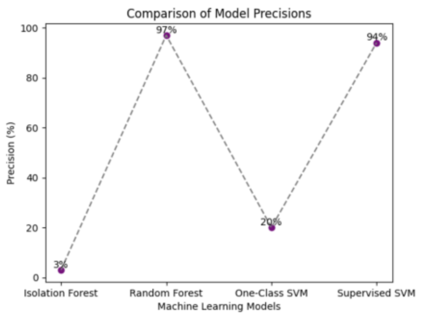
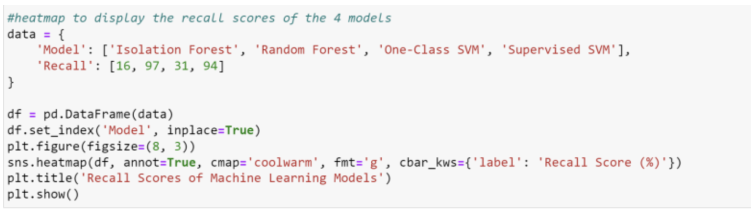


Figure 19 – Dot Graph for precision scores



```
#heatmap to display the recall scores of the 4 models
data = {
    'Model': ['Isolation Forest', 'Random Forest', 'One-Class SVM', 'Supervised SVM'],
    'Recall': [16, 97, 31, 94]
}

df = pd.DataFrame(data)
df.set_index('Model', inplace=True)
plt.figure(figsize=(8, 3))
sns.heatmap(df, annot=True, cmap='coolwarm', fmt='g', cbar_kws={'label': 'Recall Score (%)'})
plt.title('Recall Scores of Machine Learning Models')
plt.show()
```

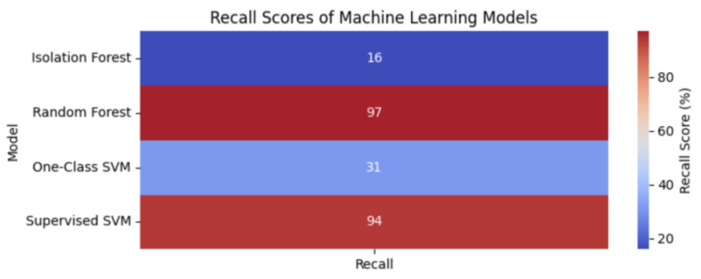
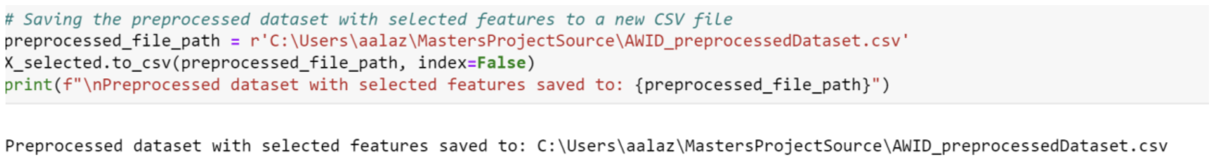


Figure 20 – Heatmap for recall scores



```
# Saving the preprocessed dataset with selected features to a new CSV file
preprocessed_file_path = r'C:\Users\aalaz\MastersProjectSource\AWID_preprocessedDataset.csv'
X_selected.to_csv(preprocessed_file_path, index=False)
print(f"\nPreprocessed dataset with selected features saved to: {preprocessed_file_path}")
```

```
Preprocessed dataset with selected features saved to: C:\Users\aalaz\MastersProjectSource\AWID_preprocessedDataset.csv
```

Figure 21 – saving the pre-processed data

References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing
You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

Anaconda. (n.d.). Download Now. [online] Available at: https://www.anaconda.com/download/success.