Configuration Manual

MSc Research Project
Data Analytics

Yogeshwar Bodicherla Rambob
Student ID: x22176322

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Yogeshwar Bodicherla Rambob
Student ID:	x22176322
Programme:	Data Analytics
Year:	2023 - 2024
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	31st January 2024
Project Title:	Perfecting Intrusion Detection System using Machine Learning Algorithm
Word Count:	298
Page Count:	5

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:	Yogeshwar Bodicherla Rambob
Date:	31st January 2024

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

Yogeshwar Bodicherla Rambob
x22176322

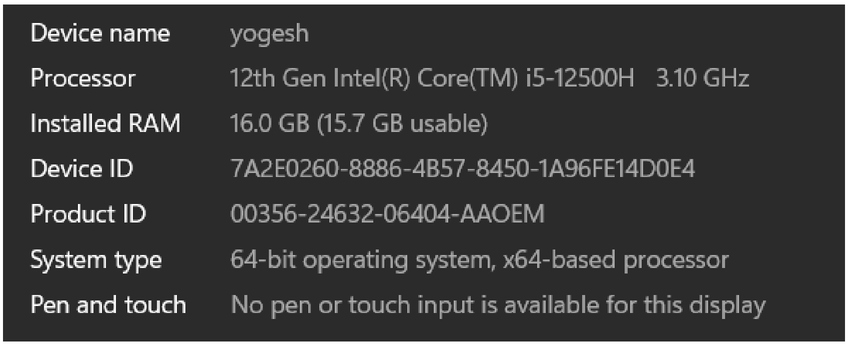
1 Introduction

This Configuration Manual details every prerequisite to replicate the research and its results in a specific environment. The hardware and software prerequisites, as well as a code fragment, are for exploratory data analysis and import. The report is organized as follows: Section 2, Provides details regarding System configuration.

2 System Configuration

A description of the system configuration employed for project execution can be found in this section of the configuration manual.

2.1 System Hardware Requirement



Device name	yogesh
Processor	12th Gen Intel(R) Core(TM) i5-12500H 3.10 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	7A2E0260-8886-4B57-8450-1A96FE14D0E4
Product ID	00356-24632-06404-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: System Hardware Configuration

2.2 System Software Requirement

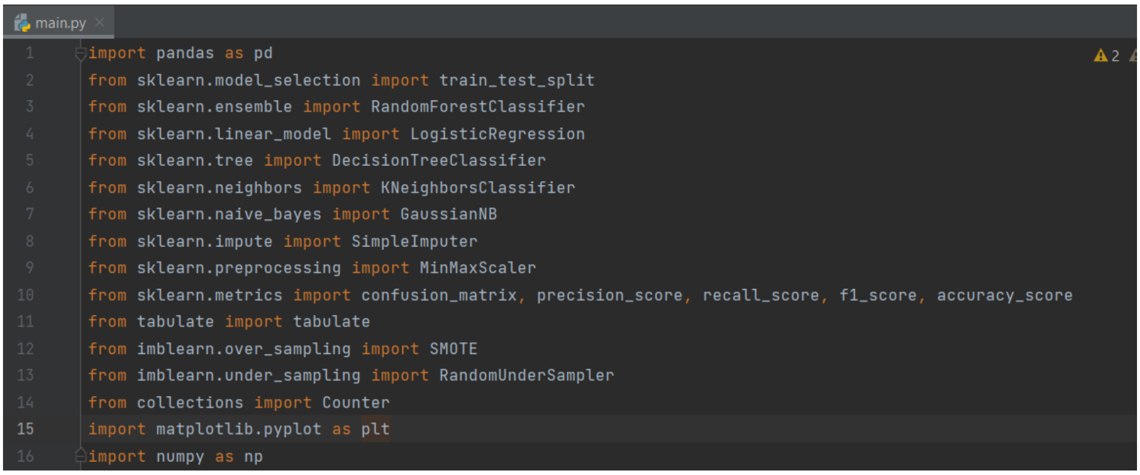
PyCharm 2022.3.3 (Professional Edition) Build PY-223.8836.43, built on March 10, 2023 Licensed to yogeshwar Bodicherla Rambob Subscription is active until January 22, 2024. For educational use only. Runtime version: 17.0.6+1-b653.34 amd64 VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o. Windows 11 10.0 GC: G1 Young Generation, G1 Old Generation Memory: 2012M Cores: 16 Registry: ide.images.show.chessboard=true

3 Data Collection

The data source used is the CICIDS 2019 Data Set, which is taken from Kaggle. The link is as follows: https://www.kaggle.com/datasets/tarundhamor/cicids-2019-dataset

4 The Exploration of Data

A comprehensive inventory of all the Python libraries necessary for the execution of the entire project is depicted in Figure 2.



```
main.py
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, accuracy_score
from tabulate import tabulate
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np
```

Figure 2: Python Libraries used for execution

```
# Load data into a Pandas dataframe
#MSSQL
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\MSSQL1.csv',
                 dtype={'column_with_mixed_types': str})
# UDP
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\UDP1.csv',
                 dtype={'column_with_mixed_types': str})
# DNS
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\DNS1.csv',
                 dtype={'column_with_mixed_types': str})
# DrDoS1
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\LDAP\\DrDoS1.csv',
                 dtype={'column_with_mixed_types': str})
# NetBIOS
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\NetBIOS\\DrDOsNetBIOS1_3.csv',
                 dtype={'column_with_mixed_types': str})
# SNMP
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\DrDoS_SNMP_data_1_3_per.csv\\DrDoS_SNMP_data_1_3_per.csv',
                 dtype={'column_with_mixed_types': str})
# NTP
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\DrDoS_NTP_data_data_5_per.csv\\DrDoS_NTP_data_data_5_per.csv',
                 dtype={'column_with_mixed_types': str})
# SSDP
df = pd.read_csv('C:\\Users\\donyo\\OneDrive\\Documents\\Datamining\\datasets\\DrDoS_SSDP_data_2_per.csv\\DrDoS_SSDP_data_2_per.csv',
                 dtype={'column_with_mixed_types': str})
```

Figure 3: Loading of data set for different models

```python
# Function to evaluate and compare models
def evaluate_models(X_train, X_test, y_train, y_test, models, objective):
    results = []
    for model_name, model in models.items():
        # For models requiring feature scaling
        scaler = MinMaxScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        # Train and evaluate the model
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        # Evaluate model performance
        conf_mat = confusion_matrix(y_test, y_pred)
        tp, tn, fp, fn = conf_mat[1, 1], conf_mat[0, 0], conf_mat[0, 1], conf_mat[1, 0]
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred)
        accuracy = accuracy_score(y_test, y_pred)
        # Append results to the list
        results.append([model_name, tp, tn, fp, fn, precision, recall, f1, accuracy])
    # Display the results using tabulate
    headers = ["Model", "True Positive", "True Negative", "False Positive", "False Negative", "Precision", "Recall", "F1 Score", "Accuracy"]
    print(tabulate(results, headers=headers, tablefmt="pretty"))
    # Find the best algorithm based on the defined objective
    best_algorithm = max(results, key=lambda x: x[headers.index(objective.capitalize()) + 1])
    print(f"\nBest Algorithm based on {objective}: {best_algorithm[0]}")

    return results, best_algorithm[0], headers  # Include headers in the returned values
```

Figure 4: Evaluate and comparing model

```python
# Sample a fraction of the dataset
df_sampled = df.sample(frac=0.1, random_state=42)

# Drop columns with mixed types and non-numeric values
df = df_sampled.select_dtypes(exclude='object')
```

Figure 5: Data Sampling and Preprocessing

```python
# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean')

# Clip extreme values before imputation
df_clipped = np.clip(df, -1e10, 1e10)

# Impute missing values with the mean
try:
    df_imputed = pd.DataFrame(imputer.fit_transform(df_clipped), columns=df.columns)
except ValueError as e:
    print(f"Error during imputation: {e}")
    print("Check for infinity or too large values in each column:")
    print(df_clipped.columns[(~np.isfinite(df_clipped)).any()])

    raise  # Reraise the exception to terminate the script
```

Figure 6: Data Imputation

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df_imputed.iloc[:, :-1], df_imputed.iloc[:, -1], test_size=0.2,
                                                    random_state=42)
```

Figure 7: Data Splitting

```
# Check the class distribution before applying sampling
print("Class distribution before sampling:", Counter(y_train))

# Apply SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Apply RandomUnderSampler
rus = RandomUnderSampler(random_state=42)
X_train_resampled, y_train_resampled = rus.fit_resample(X_train_resampled, y_train_resampled)

# Check the class distribution after sampling
print("Class distribution after sampling:", Counter(y_train_resampled))
```

Figure 8: Class Imbalance Handling

```
# Define models
models = {
    'Random Forest': RandomForestClassifier(n_estimators=100),
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'KNN': KNeighborsClassifier(),
    'GaussianNB': GaussianNB(),
}
```

Figure 9: Model Definitions

```
# Evaluate and compare models
results, best_algorithm, headers = evaluate_models(X_train_resampled, X_test, y_train_resampled, y_test, models, 'Recall')
```

Figure 10: Model Evaluation and Comparison

```
# Function to plot metrics
def plot_metrics(results, headers):
    metrics = ['Precision', 'Recall', 'F1 Score', 'Accuracy']
    num_metrics = len(metrics)

    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
    fig.suptitle('Model Evaluation Metrics', fontsize=16)

    for i in range(num_metrics):
        metric_values = [result[headers.index(metrics[i])] for result in results]
        row, col = np.divmod(i, 2)
        ax = axes[row, col]

        ax.bar(models.keys(), metric_values, color=['blue', 'orange', 'green', 'red', 'purple'])
        ax.set_title(metrics[i])
        ax.set_ylabel(metrics[i])
        ax.set_xlabel('Model')

    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

# Plot all metrics
plot_metrics(results, headers)
```

Figure 11: Plotting Metrics

References

Fosic, I., Zagar, D., & Grgic, K. (2022). Network traffic verification based on a public dataset for IDS Systems and Machine Learning Classification algorithms. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 1-4). doi: 10.23919/mipro55190.2022.9803674

Chahal, J. K., et al. (2021). KAS-ids: A machine learning based Intrusion Detection System. In *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)* (pp. 353-358). doi: 10.1109/ispcc53510.2021.9609402