Configuration Manual

MSc Research Project
Cloud Computing

Himavanth Raavi
Student ID: x23101083

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Himavanth Raavi
Student ID:	x23101083
Programme:	Cloud Computing
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	03/01/2025
Project Title:	Configuration Manual
Word Count:	419
Page Count:	5

Signature:	Himavanth
Date:	29th January 2025

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.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Himavanth Raavi
x23101083

1 Introduction

All the requirements that are necessary for this research has been included in this configuration manual. The software and hardware requirements as well as the code required for data importing, preprocessing, model building, and evaluation has also been included.

Section 2 discuss about the information about the environment used. The data collection and loading are described in section 3. The next section explains about the data preprocessing steps. Section 5 describes about the splitting of the data, model building and the evaluation.

2 Environment

2.1 Hardware Requirement

Detailed information about the hardware and software requirements as been shown in the table below.

Operating System	Windows 11
RAM	8 GB
Hard Disc	470 GB

Table 1: System Specifications

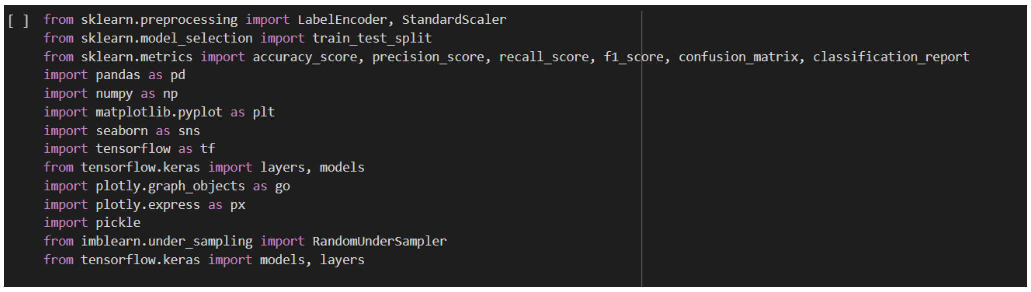
2.2 SoftwareRequirement

Programming Tools	Google Colab
Web Browser	Google Chrome
Other Required Software	Overleaf, Microsoft Word

Table 2: Software Details

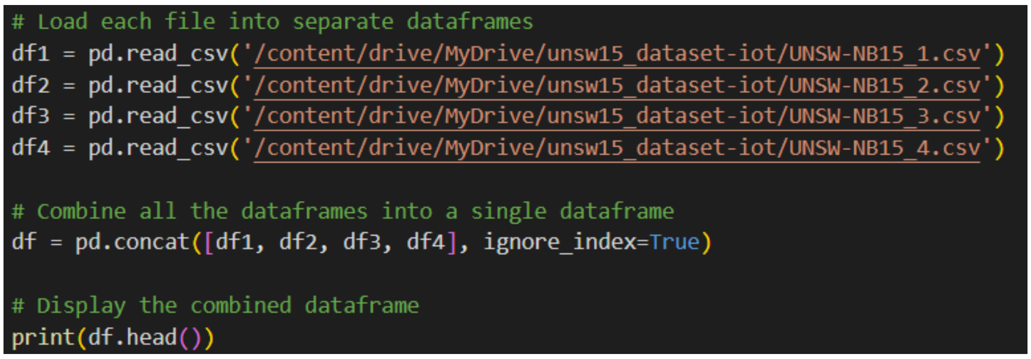
3 Data collection and loading

This section explains the code for data manipulation and importing important libraries required for data loading, cleaning and building model. Data were collected from network intrusion detection systems (UNSW-NB15 network data set).



```
[ ]  from sklearn.preprocessing import LabelEncoder, StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import tensorflow as tf
     from tensorflow.keras import layers, models
     import plotly.graph_objects as go
     import plotly.express as px
     import pickle
     from imblearn.under_sampling import RandomUnderSampler
     from tensorflow.keras import models, layers
```

Figure 1: Importing libraries

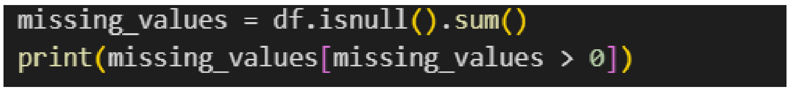


```
# Load each file into separate dataframes
df1 = pd.read_csv('/content/drive/MyDrive/unsw15_dataset-iot/UNSW-NB15_1.csv')
df2 = pd.read_csv('/content/drive/MyDrive/unsw15_dataset-iot/UNSW-NB15_2.csv')
df3 = pd.read_csv('/content/drive/MyDrive/unsw15_dataset-iot/UNSW-NB15_3.csv')
df4 = pd.read_csv('/content/drive/MyDrive/unsw15_dataset-iot/UNSW-NB15_4.csv')

# Combine all the dataframes into a single dataframe
df = pd.concat([df1, df2, df3, df4], ignore_index=True)

# Display the combined dataframe
print(df.head())
```

Figure 2: Data loading

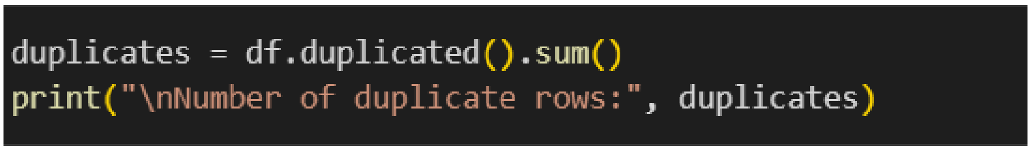
4 Data Preprocessing

In this section the data preprocessing steps and the code used to plot the charts, removing null and duplicate values.



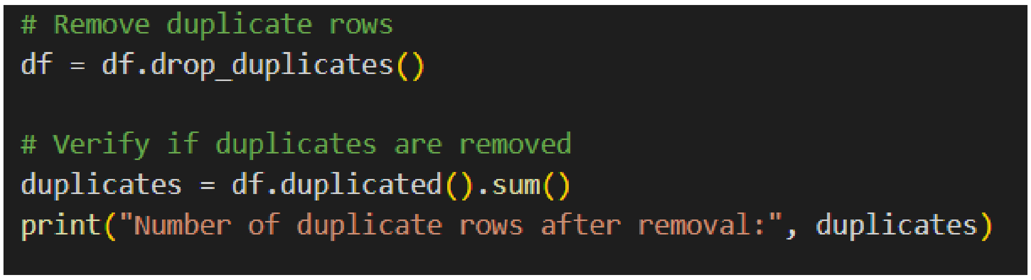
```
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])
```

Figure 3: Null Values



```
duplicates = df.duplicated().sum()
print("\nNumber of duplicate rows:", duplicates)
```

Figure 4: Duplicate Values

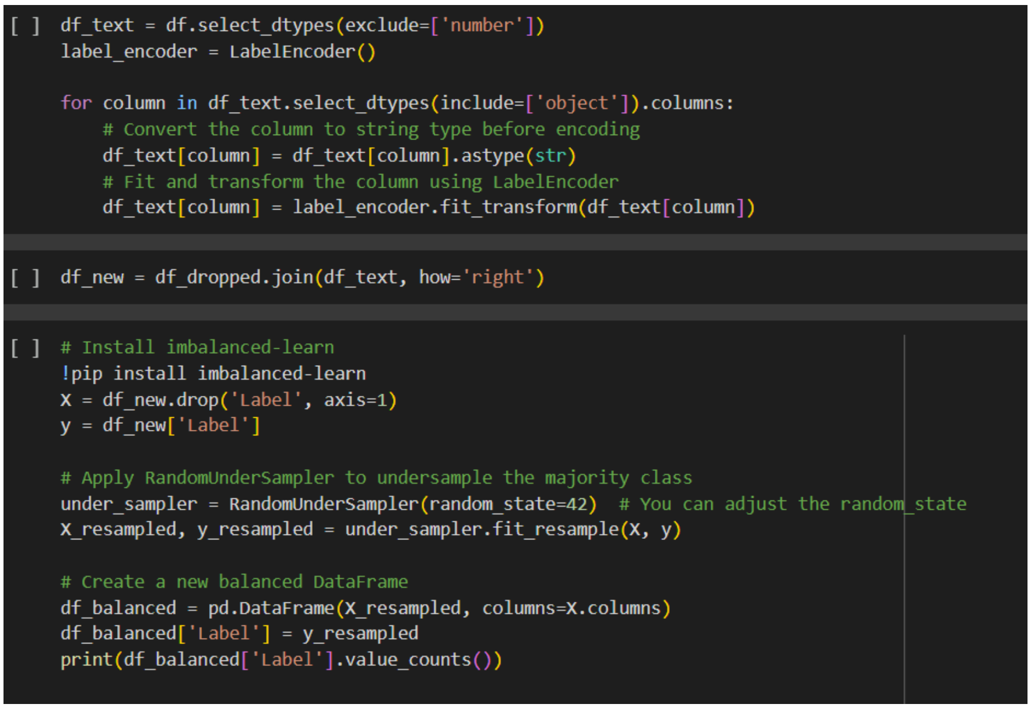


```
# Remove duplicate rows
df = df.drop_duplicates()

# Verify if duplicates are removed
duplicates = df.duplicated().sum()
print("Number of duplicate rows after removal:", duplicates)
```

Figure 5: Removing Duplicate Values

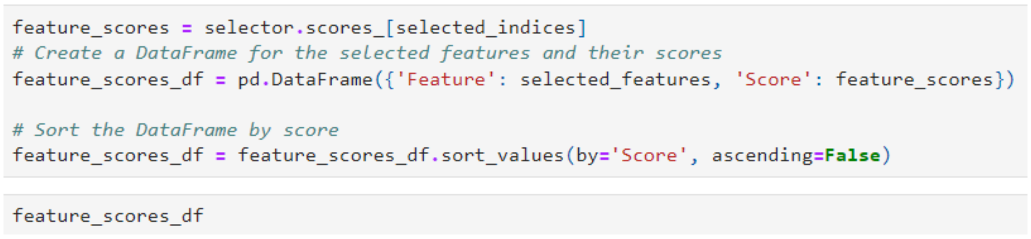
4.1 Feature Engineering and Data Balancing

Dropping highly correlated features, and encode categorical data. Apply under sampling techniques to address class imbalance and save the balanced dataset.



```
[ ] df_text = df.select_dtypes(exclude=['number'])
    label_encoder = LabelEncoder()

    for column in df_text.select_dtypes(include=['object']).columns:
        # Convert the column to string type before encoding
        df_text[column] = df_text[column].astype(str)
        # Fit and transform the column using LabelEncoder
        df_text[column] = label_encoder.fit_transform(df_text[column])

[ ] df_new = df_dropped.join(df_text, how='right')

[ ] # Install imbalanced-learn
    !pip install imbalanced-learn
    X = df_new.drop('Label', axis=1)
    y = df_new['Label']

    # Apply RandomUnderSampler to undersample the majority class
    under_sampler = RandomUnderSampler(random_state=42)  # You can adjust the random_state
    X_resampled, y_resampled = under_sampler.fit_resample(X, y)

    # Create a new balanced DataFrame
    df_balanced = pd.DataFrame(X_resampled, columns=X.columns)
    df_balanced['Label'] = y_resampled
    print(df_balanced['Label'].value_counts())
```

Figure 6: Under Sampling



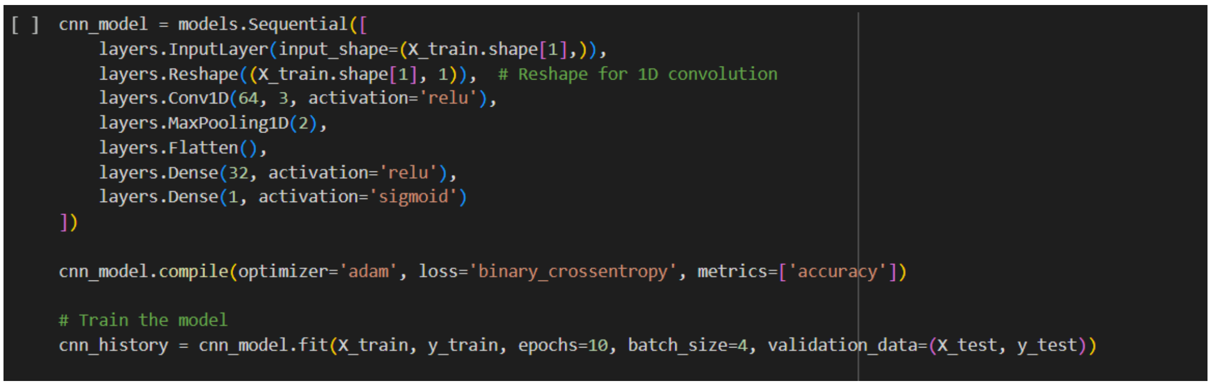
```
feature_scores = selector.scores_[selected_indices]
# Create a DataFrame for the selected features and their scores
feature_scores_df = pd.DataFrame({'Feature': selected_features, 'Score': feature_scores})

# Sort the DataFrame by score
feature_scores_df = feature_scores_df.sort_values(by='Score', ascending=False)

feature_scores_df
```

Figure 7: Feature selection

The feature_scores_df will show the features and score for each feature contributing to the target variable.

4.2 Model Training

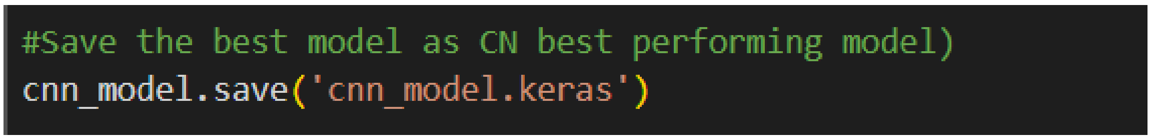
Train CNN, RNN, and Autoencoder models with the prepared dataset and evaluate their performances



```
[ ] cnn_model = models.Sequential([
        layers.InputLayer(input_shape=(X_train.shape[1],)),
        layers.Reshape((X_train.shape[1], 1)),  # Reshape for 1D convolution
        layers.Conv1D(64, 3, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Flatten(),
        layers.Dense(32, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    # Train the model
    cnn_history = cnn_model.fit(X_train, y_train, epochs=10, batch_size=4, validation_data=(X_test, y_test))
```

Figure 8: Model Training

Model Evaluation and Comparison Compare the models based on accuracy, precision, recall, and F1-score to identify the best-performing model. In this case the best model as CNN best performing mode



```
#Save the best model as CN best performing model)
cnn_model.save('cnn_model.keras')
```

Figure 9: Saving the CNN Model

5 Set Up and Connect to AWS EC2 Instance for Deploying the Web Application

Launch an EC2 Instance

- Log in to AWS Management Console.
- Navigate to EC2 → Launch Instance.
- Choose **Ubuntu 20.04 LTS** AMI.
- Select **t2.micro** or higher, based on app requirements.
- Enable **Auto-assign Public IP** under network settings.
- Create a security group to allow **ports 22 (SSH)** and **5000 (Flask default port)**.

Download Key Pair

- Create and download a key pair.
- Save the key securely for connecting via SSH.

Connect to Instance Use SSH to connect: *ssh -i "your-key.pem" ubuntu@Public-IP*

Install the dependencies and application

- Update the instance and install dependencies

  sudo apt update sudo apt upgrade -y sudo apt install python3-pip -y

Deploy Web Application

- Transfer application files to the instance:
  `scp -i "your-key.pem" <local-app-folder> ubuntu@<Public-IP>:/home/ubuntu/app`
- Navigate to the application folder and run the Flask app:
  cd /home/ubuntu/app python3 app.py

Update Security Group to Allow Flask Port Add a rule in your EC2 security group to allow inbound traffic on port **5000**.

Access the Application

- Use your EC2 **Public IP** with Flask's default port: `http://<Public-IP>:5000`
- The web app is now accessible, providing real-time IoT packet classification.