National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Venkata Sai Charan Vinnamuri
StudentID:22156461

School of Computing
National College of Ireland

Supervisor: Jitender Kumar Sharma

National College of Ireland
Project Submission Sheet School of Computing

Student Name:	Venkata Sai Charan Vinnamuri
Student ID:	22156461
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Jitender Kumar Sharma
Submission Due Date:	16/09/2024
Project Title:	Novel Approaches for Real-Time Detection of DDoS Attacks in Cloud Computing Environments Using Advanced Machine Learning Techniques
Word Count:	
Page Count:	

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:	Vinnamuri Venkata Sai Charan
Date:	16th September, 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

Venkata Sai Charan
22156461

1. Introduction

This document provides the steps to create and install the required softwares, tools and files to perform the research mentionted in research report with title “Novel Approaches for Real-Time Detection of DDoS Attacks in Cloud Computing Environments Using Advanced Machine Learning Techniques”.

2.Prerequisites

Before setting up and running the project, import all necessary libraries using below commands:

```
import pandas as pd
from sklearn.model_selection import train_test_split  from
sklearn.preprocessing import StandardScaler, MinMaxScaler  from
sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score  import time
```

3.Load and Explore the Dataset

Read the csv datafile “DDoS Dataset.csv” and load the traffic dataset into pandas dataframe which is table-like data structure with rows and columns, ideal for data manipulation and analysis. Next print few rows of the structured dataset to see the attributes and information of the data and few rows of the structured dataset.

```
file_path = 'DDoS Dataset.csv'  # Path to the dataset
dataset = pd.read_csv(file_path)  # Load the dataset

# Display basic information about the dataset to understand its structure
print("Dataset Information:")
print(dataset.info())
print("\nFirst 5 rows of the dataset:")
print(dataset.head())
```

Figure 1

The above code in the figure 1 prints the data information and first 5 rows of the dataset.

4. Run the data preprocessing cells

4.1 Handle the missing values if any:

Ensure that the dataset is free of missing values by replacing them with 0. Handling the missing values ensures the data consistency and data integrity.

```
# Replace missing values with 0 (if any)
dataset.fillna(0, inplace=True)
```

Figure 2

The DataFrame now contains no “NaN” values, which prevents potential issues during model training or analysis.

4.2 Feature selection

Drop the unnecessary features like “source_IP”, “Destination_IP” in ‘X’ using drop function as they don’t need to train the model.
Define the target variable ‘y’.

```
X = dataset.drop(columns=['Source_IP', 'Destination_IP', 'Label'])  # Features
y = dataset['Label']  # Target variable
```

Figure 3

4.3 Split the data

Using the function ‘train_test_split()’ from the ‘sklearn.model_selection’ module in Scikitlearn split data into test subsets.
First split: Split the data into training (70%), temp(30%)
Second split: validation (15%), and testing (15%) sets

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

Figure 4

4.4 Feature Scaling

Apply Min-Max scaling to normalize the feature values between 0 and 1. Create an instance of the MinMaxScaler from the sklearn.preprocessing module using scaler = MinMaxscaler() syntax. Fitting and Transforming the Training Data using fit() and transform() function. Finally transform the validation and test data.

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)  # Fit and transform the training data
X_val_scaled = scaler.transform(X_val)  # Transform the validation data
X_test_scaled = scaler.transform(X_test)  # Transform the test data
```

Figure 5

5. Train the models

5.1 Train with Random forest model

Initialize and train the Random Forest model with 100 decision trees. Below is the code to train the model:

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)

Output:

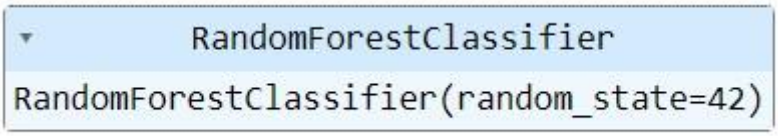

Figure 5

5.2 Train the dataset with SVM model

Initialize and train the Support Vector Machine model with a linear kernel.

Code: svm_model = SVC(kernel='linear', probability=True, random_state=42) svm_model.fit(X_train_scaled, y_train)

output:

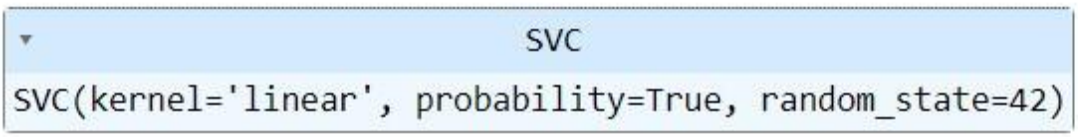

Figure 6

6. Hybrid Model - Ensemble Learning

6.1 Create a Voting Classifier

Combine the Random Forest and SVM models using a soft voting approach.

Code:

```
hybrid_model = VotingClassifier(estimators=[('rf', rf_model), ('svm', svm_model)], voting='soft')
hybrid_model.fit(X_train_scaled, y_train)  # Train the hybrid model
```

Output:

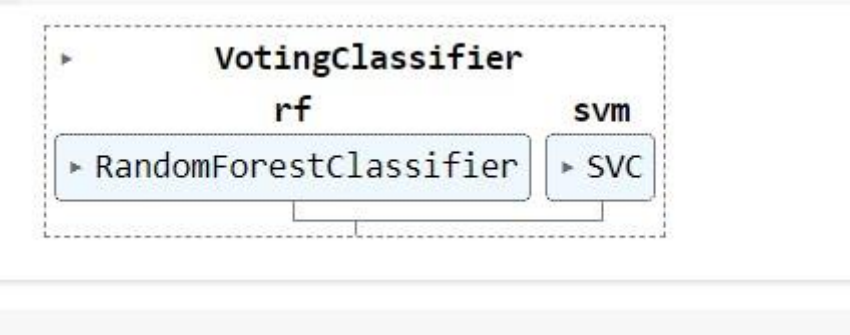

Figure 7

7. Evaluate the model

7.1 Evaluate on the Validation Set for Random Forest

You should assess your model's performance after it has been trained. Metrics like accuracy, precision, recall, F1-score, and potentially a confusion matrix are usually used for this.

```
y_val_pred_rf = rf_model.predict(X_val_scaled)

# Confusion Matrix for Random Forest
conf_matrix_rf = confusion_matrix(y_val, y_val_pred_rf)
print("\nRandom Forest - Confusion Matrix:")
print(conf_matrix_rf)

# Classification Metrics for Random Forest
accuracy_rf = accuracy_score(y_val, y_val_pred_rf)
precision_rf = precision_score(y_val, y_val_pred_rf)
recall_rf = recall_score(y_val, y_val_pred_rf)
f1_rf = f1_score(y_val, y_val_pred_rf)
roc_auc_rf = roc_auc_score(y_val, rf_model.predict_proba(X_val_scaled)[:, 1])

print(f"\nRandom Forest - Accuracy: {accuracy_rf:.2f}")
print(f"Random Forest - Precision: {precision_rf:.2f}")
print(f"Random Forest - Recall: {recall_rf:.2f}")
print(f"Random Forest - F1 Score: {f1_rf:.2f}")
print(f"Random Forest - ROC AUC Score: {roc_auc_rf:.2f}")
```

Figure 8

Fig 8 evaluates and prints metrics like confusion matrix, accuracy, precision, recall, F1-score, ROC AUC score.

7.2 Evaluate on the Validation Set for SVM

Classify the different classes and get insights of number of true positives, true negatives, false positives, and false negatives using confusion matrix

```
y_val_pred_svm = svm_model.predict(X_val_scaled)

# Confusion Matrix for SVM
conf_matrix_svm = confusion_matrix(y_val, y_val_pred_svm)
print("\nSVM - Confusion Matrix:")
print(conf_matrix_svm)

# Classification Metrics for SVM
accuracy_svm = accuracy_score(y_val, y_val_pred_svm)
precision_svm = precision_score(y_val, y_val_pred_svm)
recall_svm = recall_score(y_val, y_val_pred_svm)
f1_svm = f1_score(y_val, y_val_pred_svm)
roc_auc_svm = roc_auc_score(y_val, svm_model.predict_proba(X_val_scaled)[:, 1])

print(f"\nSVM - Accuracy: {accuracy_svm:.2f}")
print(f"SVM - Precision: {precision_svm:.2f}")
print(f"SVM - Recall: {recall_svm:.2f}")
print(f"SVM - F1 Score: {f1_svm:.2f}")
print(f"SVM - ROC AUC Score: {roc_auc_svm:.2f}")
```

Figure 9

Fig 9 prints the computed metrics with two decimal points of precision for easier interpretation.

7.3 Evaluate on the Validation Set

Predict labels for the validation set using the trained hybrid model

y_val_pred = hybrid_model.predict(X_val_scaled)

7.4 Generate the confusion matrix to evaluate the performance of the model

conf_matrix = confusion_matrix(y_val, y_val_pred)
print("\nHybrid Model Confusion Matrix:") print(conf_matrix)

Classification Metrics:

Calculate and print key performance metrics: Accuracy, Precision, Recall, F1-Score, and ROC AUC Score

```
accuracy = accuracy_score(y_val, y_val_pred)
precision = precision_score(y_val, y_val_pred)
recall = recall_score(y_val, y_val_pred)
f1 = f1_score(y_val, y_val_pred)
roc_auc = roc_auc_score(y_val, hybrid_model.predict_proba(X_val_scaled)[:, 1])

print(f"\nHybrid Model Accuracy: {accuracy:.2f}")
print(f"Hybrid Model Precision: {precision:.2f}")
print(f"Hybrid Model Recall: {recall:.2f}")
print(f"Hybrid Model F1 Score: {f1:.2f}")
print(f"Hybrid Model ROC AUC Score: {roc_auc:.2f}")
```

8. Real-Time Performance Evaluation

8.1 Latency Measurement

Measure the time taken by the model to make predictions on the test set with the below syntax.

start_time = time.time() # Start timer
y_test_pred = hybrid_model.predict(X_test_scaled) # Predict test labels end_time = time.time() # End timer

latency = end_time - start_time # Calculate latency print(f"\nLatency: {latency:.4f} seconds") # Print latency

8.2 Throughput Measurement

Calculate the throughput as the number of samples processed per second.

```
throughput = len(X_test) / latency
print(f"Throughput: {throughput:.2f} samples per second")  # Print throughput
```