National College of Ireland

Configuration Manual

MSc Research Project
MSc Artificial Intelligence

Sayali Thakur
Student ID: x23139901

School of Computing
National College of Ireland

Supervisor: SHERESH ZAHOOR

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sayali Thakur
Student ID:	x23139901
Programme:	MSc Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	SHERESH ZAHOOR
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	910
Page Count:	11

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:	Sayali Machhindra Thakur
Date:	12th December 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

Sayali Thakur
x23139901

1 Introduction

The guide below outlines and elaborates procedures that should be followed in emulating the project setting in terms of tools, application, and settings. They describe the process of data pre-processing, feature extraction, modeling and evaluating and enables an accurate replication of the experiment.

2 Project File Details

The Google Collab program is used in this project for data preparation and exploration,modelling and evaluation. The Jupyter file is of scrapped youtube comments.

- Dissertation.ipynb: preprocessing and EDA done in this file and data is saved in CSV file.
- Thesis.ipynb: preprocessing and Modelling and evaluation are done in this file.
- Youtube_comment_scraper: this file contains youtube comment scraper code

3 Software Used

- Microsoft excel: Used for dataset storing in CSV.
- Google Collab: Used for Exploration and processing.
- Jupyter Notebook: For the modelling and evaluation.

4 System Specification

A system specification is essentially a documented outline of a system, it displays technical features and specifications that a certain system is to include. It can also provide details about the system's components, operations, layout and many more. The system configuration used in running this is illustrated in Figure 3 2 below. Schematic representation: The Figure 1 and Figure 2 below illustrates the Google Collab specification. The most basic setup would only need 8 GB of RAM, and that should be sufficient to produce the desired results but would need slightly longer time than the current methods being used.

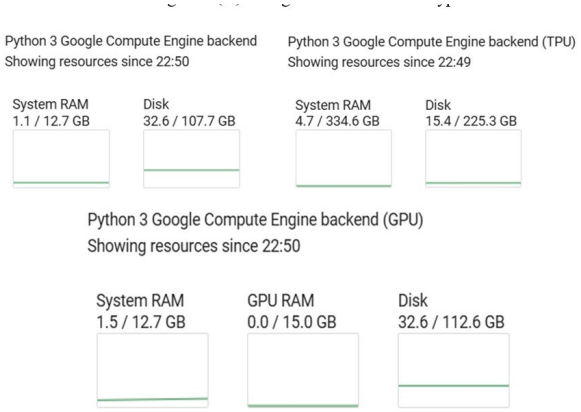

Figure 1: Google Colab Specification

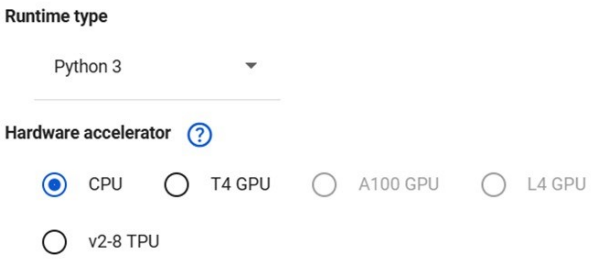

Figure 2: Google Colab Runtime Specification

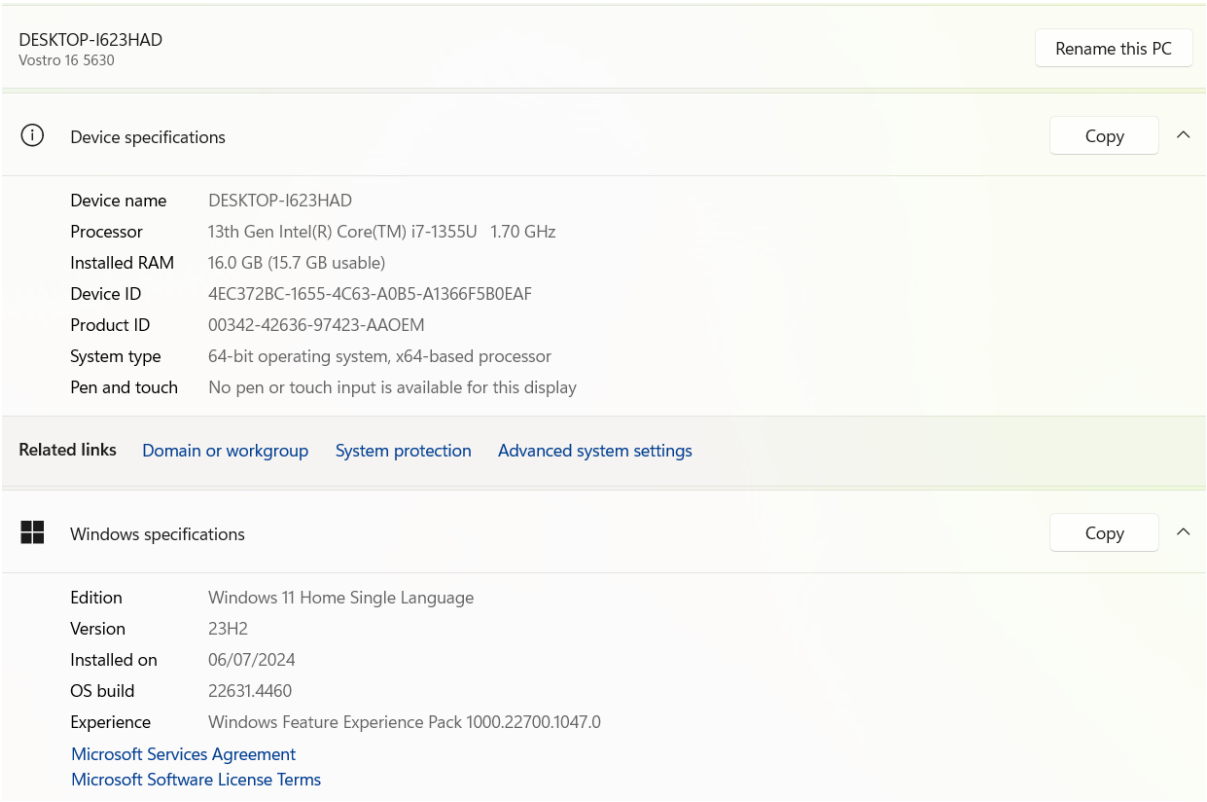

Figure 3: Machine Specification

5 Download and Install

Based on the operating system, Python must first be installed; the installation of the most recent version is advised Python 3.10.2 for Windows 11 was downloaded and installed as the most recent version of the file. A development environment is needed after Python has been installed to write, run, and view the output of code.

It is simple to use Google Collab Using a Google account, sign in and add a new file to the disk. Users have free access to computational tools like GPUs and TPUs through Google Colab, which can be utilized to execut computationally intensive tasks. jupyter notebook used for scraping the youtube comments. Jupyter Notebook is bundled with the Python distribution, Anaconda 2 as shown in Figure 4,for which a suitable installation can be downloaded depending on the operating system. The Anaconda's dashboard also features pre-installed packages like the Jupyter notebook. The first step in developing Python code is to launch the Jupyter Notebook and create a new Python file oepn .ipynb

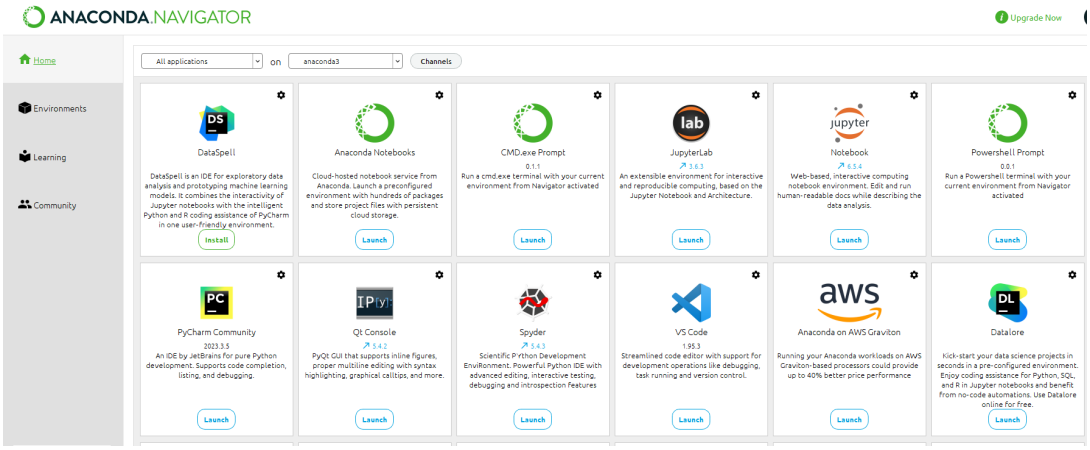

Figure 4: Anaconda Navigator Specification

6 Project Developement

After going through all these steps you can open the Jupyter notebook or Google Collab and then click the new button located at the top of the file open and load scripted file from the file reference If you scroll down to the code section, what we offered, you will have an option to run all of them at once or each cell individually.

6.1 Importing Files

Upload 'Thesis' folder on google drive. then run following command shown in Figure 8.

6.2 Python Libraries

The packages used in the project are displayed in Figure 8 below. If necessary " !pip install package-name". also emoji package needs to install first before importing eg. "!pip install emoji"

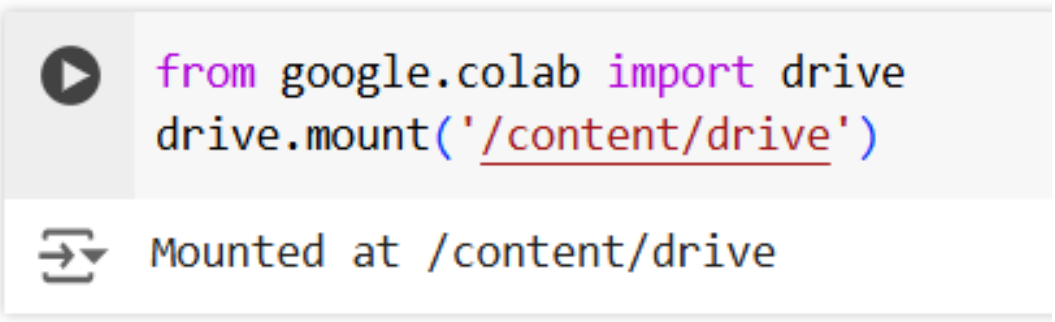

Figure 5: Procedure to mount on google drive

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
from nltk.corpus import stopwords
from nltk.tokenize import word_tokeni
from nltk.stem import WordNetLemmatizer
import emoji
# from transformers import BertTokenizer, TFBertModel
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input, Concatenate, Attention
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

Loading...

```
[ ] import os
    import csv
    import pandas as pd
    from googleapiclient.discovery import build
```

Figure 6: Packages Used in a Project

7 Dataset

The dataset used for this research consists of two main parts. Dataset 1 as shown in Figure **??** contains 19,708 rows of youtube comments. It includes columns such as username, comment, time-posted, and likes. Dataset 2 as shown in Figure **??** The text data used in this paper is obtained from the WANLP 2021 Shared Task on Sarcasm and Sentiment Detection in Arabic by Abu Farha et al. (2021). It contains 15,545 tweets which were translated into English for analysis, with columns for the tweet text (tweet), sarcasm label (sarcasm), sentiment (sentiment), and dialect (dialect). The sarcasm column indicates whether the tweet is sarcastic (1) or not (0), and dialect specifies the language variant used.

```
Dataset1 loaded successfully.
 Number of rows: 19708              username                                                comment  \
0          @trevornoah  Subscribe if you haven't already! http://bit.l...
1     @keithlenton5313  One of my favourite comedians, Trevor. You're ...
2  @praveenmalhotra159  Thank the local traitors who opened the door o...
3         @Light_spot_                                That talent 👏 👌 😍 🙌
4           @Vic-vf5tm                                That was hilarious.

            time_posted  likes
0  2021-02-26T21:22:32Z   7999
1  2024-11-14T07:57:27Z      0
2  2024-11-13T02:30:12Z      0
3  2024-11-12T22:26:49Z      0
4  2024-11-12T20:08:55Z      1
```

Figure 7: Packages Used in a Project

```
Dataset2 loaded successfully.
 Number of rows: 15545                                                    tweet  sarcasm sentiment  \
0  Dr. #Mahmoud_Al-Alaili: I see that Lieutenant ...        0       NEU
1  "With Federer, Aga, and the big boys 😍 https:/...        0       NEU
2  "Those who advocate the principle of mixing be...        1       NEG
3  "@ihe_94 @ya78m @amooo5 @badiajnikhar @Oukasaf...        1       NEG
4  "Say East Aleppo and do not say East Aleppo.....         0       NEU

  dialect
0     msa
1     msa
2     msa
3    gulf
4     msa
```

Figure 8: Packages Used in a Project

8 Preprocessing

The configuration manual offers some insight of the process undertaken to prepare the data for integration into analysis. That is why most of these steps are considered fundamental to data quality and model's performance as shown in Figure 9 and Figure 18.

- Tokenization: Used Hugging Face Tokenizer.
- Emoji extraction: Used the emoji library, One-Hot Encoding for Emoji Features.
- Feature extraction: Used Text Frequency-Inverse Document Frequency (TF-IDF), Word Embeddings, Emoji Embeddings.

```python
import pandas as pd

# Load the dataset
file_path = '/content/drive/MyDrive/Colab Notebooks/Thesis/new_tweetercomments_dataset.csv'
try:
    df = pd.read_csv(file_path)
except Exception as e:
    df = pd.read_csv(file_path, encoding='latin1')  # Fallback encoding


print("Initial Dataset Info:")
print(df.info())
print("First few rows:")
print(df.head())

# Step 1: Inspect unique values in the 'sarcasm' column
print("\nUnique values in 'sarcasm':")
print(df['sarcasm'].unique())

# Step 2: Drop rows with missing or invalid 'sarcasm' values
valid_sarcasm_values = ['TRUE', 'FALSE']
df_cleaned = df[df['sarcasm'].isin(valid_sarcasm_values)].copy()

# Step 3: Replace 'TRUE'/'FALSE' with 1/0 and ensure the column type is integer
df_cleaned['sarcasm'] = df_cleaned['sarcasm'].replace({'FALSE': 0, 'TRUE': 1}).astype(int)

# Step 4: Basic cleaning for the 'tweet' column
df_cleaned['tweet'] = df_cleaned['tweet'].str.strip()  # Remove leading/trailing spaces
df_cleaned = df_cleaned[df_cleaned['tweet'] != '']  # Remove empty strings

# Step 5: Clean remaining columns
df_cleaned = df_cleaned.dropna(subset=['sentiment', 'dialect'])

# Step 6: Save the cleaned dataset to a new CSV file
cleaned_file_path = '/content/drive/MyDrive/Colab Notebooks/Thesis/tweetercomments_cleaned_dataset.csv'
df_cleaned.to_csv(cleaned_file_path, index=False)

# Step 7: Display final dataset information and first few rows
print("\nCleaned Dataset Info:")
print(df_cleaned.info())
print("\nFirst few rows of cleaned dataset:")
print(df_cleaned.head())

print(f"\nCleaned dataset saved successfully to '{cleaned_file_path}'!")
```

Figure 9: Data Preprocessing Steps

```python
# Preprocess Text Column (Cleaned tweets should already be preprocessed)
def preprocess_text(text):
    return text.strip().lower()

print("\n---- Preprocessing Tweets ----")
data['cleaned_tweet'] = data['tweet'].apply(preprocess_text)
print("Cleaned tweets example:")
print(data['cleaned_tweet'].head())

# Extract emojis from tweets
def extract_emojis(text):
    return ''.join(c for c in text if c in emoji.EMOJI_DATA)

print("\n---- Extracting Emojis ----")
data['emojis'] = data['tweet'].apply(extract_emojis)
data['emojis'] = data['emojis'].replace('', '[NO_EMOJI]')
print("Extracted emojis example:")
print(data[['tweet', 'emojis']].head())

# One-Hot Encoding for Emoji Features
print("\n---- Emoji One-Hot Encoding ----")
emoji_encoder = OneHotEncoder(sparse_output=False)
emoji_features = emoji_encoder.fit_transform(data['emojis'].values.reshape(-1, 1))
print("Shape of One-Hot Encoded Emoji Features:", emoji_features.shape)
```

Figure 10: Data Preprocessing Next Steps

9 Data Modelling

Before modeling, the important predictors are obtained from the recursive feature elimination phase and then fed into the x-data and y-data functions with the target variable. If any of the two sets has more observations, smote will be used to balance the data before the test. train is split and fed into the models. This is shown in the figures below:

In this study, five machine learning algorithms and one is the deep learning algorithm have been employed. The best parameter will be selected when these models are tuned with the various parameters; the best parameter will be selected by the grid-search CV. Some packages need to import before running models. following are the images of the models implemented.

```
# Tokenization with BERT
print("\n---- Tokenizing Text with BERT ----")
max_seq_length = 128
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
text_inputs = tokenizer(
    list(data['cleaned_tweet']),
    truncation=True,
    padding=True,
    max_length=max_seq_length,
    return_tensors='tf'
)
print("BERT Tokenization Complete:")
print(f"Input IDs Shape: {text_inputs['input_ids'].shape}")

# Train-Test Split
print("\n---- Splitting Data into Train and Test Sets ----")
X_text = text_inputs['input_ids']
X_emoji = emoji_features
y = data['sarcasm']

X_train_text, X_test_text, X_train_emoji, X_test_emoji, y_train, y_test = train_test_split(
    X_text.numpy(), X_emoji, y, test_size=0.2, random_state=42)
```

Figure 11: Data Modelling Early Steps

```
from sklearn.metrics import accuracy_score

# Rule-based sarcasm detection: Example with keyword matching
def rule_based_sarcasm_detection(texts):
    sarcasm_indicators = ['yeah right', 'totally', 'sure', 'obviously','as if',
        'just great', 'brilliant', 'fantastic', 'what a surprise', 'how original']
    return [
        1 if any(indicator in text.lower() for indicator in sarcasm_indicators) else 0
        for text in texts
    ]

# Apply rule-based detection on cleaned_tweet
y_pred_rule_based = rule_based_sarcasm_detection(data['cleaned_tweet'])

# Evaluate accuracy of rule-based predictions
rule_based_accuracy = accuracy_score(data['sarcasm'], y_pred_rule_based)
print("Rule-Based Accuracy:", rule_based_accuracy)
# Calculate confusion matrix for Rule-Based Model
cm_rule_based = confusion_matrix(data['sarcasm'], y_pred_rule_based)
print("Confusion Matrix for Rule-Based Model:\n", cm_rule_based)
```

Figure 12: Rule Based Pattern matching

```
# Naïve Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train_combined, y_train)
y_pred_nb = nb_model.predict(X_test_combined)
print("Naïve Bayes Report:\n", classification_report(y_test, y_pred_nb))
```

Figure 13: Naïve Bayes

```
# Support Vector Machine (SVM)
svm_model = SVC(kernel='linear', probability=True)
svm_model.fit(X_train_combined_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_combined_scaled)
print("SVM Report:\n", classification_report(y_test, y_pred_svm))
```

Figure 14: SVM

```
# Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train_combined, y_train)  # No scaling for Random Forest
y_pred_rf = rf_model.predict(X_test_combined)
print("Random Forest Report:\n", classification_report(y_test, y_pred_rf))
```

Figure 15: Random Forest

```
# XGBoost Classifier
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train_combined_scaled, y_train)  # XGBoost requires scaled features
y_pred_xgb = xgb_model.predict(X_test_combined_scaled)
print("XGBoost Report:\n", classification_report(y_test, y_pred_xgb))
```

Figure 16: XGBoost

```
# Prepare inputs for BERT and emoji features
input_ids = bert_inputs['input_ids']
attention_mask = bert_inputs['attention_mask']
X_emoji = emoji_features
y = data['sarcasm'].values

# Convert TensorFlow tensors to NumPy arrays for compatibility with train_test_split
input_ids_np = input_ids.numpy()
attention_mask_np = attention_mask.numpy()

# Train-Test Split
X_train_ids, X_test_ids, X_train_mask, X_test_mask, X_train_emoji, X_test_emoji, y_train, y_test = train_test_split(
    input_ids_np, attention_mask_np, X_emoji, y, test_size=0.2, random_state=42)

# Define Multimodal Model
bert_model = TFBertModel.from_pretrained('bert-base-uncased')

# BERT branch
input_ids_layer = Input(shape=(128,), dtype=tf.int32, name='input_ids')
attention_mask_layer = Input(shape=(128,), dtype=tf.int32, name='attention_mask')

bert_output = Lambda(
    lambda x: bert_model(input_ids=x[0], attention_mask=x[1])[1],
    output_shape=(768,),  # Explicitly specify the output shape
    name="bert_embedding")([input_ids_layer, attention_mask_layer])

# Emoji branch
emoji_input = Input(shape=(X_train_emoji.shape[1],), name='emoji_input')
emoji_dense = Dense(128, activation='relu')(emoji_input)
```

Figure 17: Deep Learning 1

```
# Combine branches
combined = Concatenate()([bert_output, emoji_dense])
dropout = Dropout(0.3)(combined)
output = Dense(1, activation='sigmoid', name='output')(dropout)

# Define and compile the model
model = Model(inputs=[input_ids_layer, attention_mask_layer, emoji_input], outputs=output)
model.compile(optimizer=Adam(learning_rate=2e-5), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(
    [X_train_ids, X_train_mask, X_train_emoji],
    y_train,
    validation_data=([X_test_ids, X_test_mask, X_test_emoji], y_test),
    batch_size=32,
    epochs=3
)

# Evaluate the model
y_pred_dl = (model.predict([X_test_ids, X_test_mask, X_test_emoji]) > 0.5).astype(int)
print("Deep Learning Report:\n", classification_report(y_test, y_pred_dl))
```

Figure 18: Deep Learning 2

```
# hyper tuning
from sklearn.model_selection import GridSearchCV

# Define XGBoost parameter grid
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# Initialize XGBoost model
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

# Perform grid search
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, scoring='f1', cv=3, verbose=1)
grid_search.fit(X_train, y_train)

# Get best parameters and performance
print("Best Parameters:", grid_search.best_params_)
print("Best F1-Score:", grid_search.best_score_)

# Evaluate on test set
best_xgb_model = grid_search.best_estimator_
y_pred_tuned = best_xgb_model.predict(X_test)
print("Tuned XGBoost Report:\n", classification_report(y_test, y_pred_tuned))
```

Figure 19: Hyperparameter Tuning

```python
import shap

# Number of features in each section (TF-IDF, emoji, sentiment, POS count)
num_tfidf = 1000  # Number of TF-IDF features
num_emoji = 128   # Number of PCA-reduced emoji features

# Generate feature names dynamically
feature_names = (
    [f"TF-IDF_{i}" for i in range(num_tfidf)] +  # TF-IDF feature names
    [f"Emoji_{i}" for i in range(num_emoji)] +   # Emoji feature names
    ['Sentiment', 'POS Count']                  # Sentiment and POS Count
)

# Ensure the number of feature names matches the features in X_test
assert len(feature_names) == X_test.shape[1], \
    f"Number of generated feature names ({len(feature_names)}) does not match number of features in X_test ({X_test.shape[1]})"

# SHAP explanations for the XGBoost model
explainer = shap.Explainer(xgb_model, X_test)  # Attach X_test to the model for SHAP computation
shap_values = explainer(X_test)

# Print information
print(f"Number of features in X_test: {X_test.shape[1]}")
print(f"Generated feature names (sample): {feature_names[:5]} ... {feature_names[-5:]}")  # Display some feature names

# Visualize feature importance
shap.summary_plot(shap_values, X_test, feature_names=feature_names)
```

Figure 20: Feature Importance

References

Abu Farha, I., Zaghouani, W. and Magdy, W. (2021). Overview of the wanlp 2021 shared task on sarcasm and sentiment detection in arabic, *Proceedings of the Sixth Arabic Natural Language Processing Workshop*.