From Traditional to Advanced Machine Learning: A Comparative Study of Political Tweet Sentiment Analysis – Configuration Manual

MSc Research Project
MSc Data Analytics

Vishnunath Nharekkat
Student ID: x22234217

School of Computing
National College of Ireland

Supervisor: **Musfira Jilani**

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vishnunath Nharekkat
Student ID:	X22234217
Programme:	MSc Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Musfira Jilani
Submission Due Date:	12/12/2024
Project Title:	From Traditional to Advanced Machine Learning: A Comparative Study of Political Tweet Sentiment Analysis
Word Count:	930
Page Count:	7

I hereby certify that the information contained in this (my submission) is information about research I conducted for this project. All information other than my 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 another author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vishnunath Nharekkat
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's office must be placed into the assignment box located outside the office.

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

Configuration Manual

Vishnunath Nharekkat
x22234217

1. Introduction

This configuration manual provides guidelines for configurations and implementation of the sentiment analysis project on Indian Election Tweets. This paper describes the requirements, folder structure, data preparation procedures, and steps to train and test SVM and LSTM models. This document is intended to help the users manage the running process of the sentiment analysis pipeline as well as to point at the potential problems that may occur during the process.

2. Environment setup

2.1 System specification

Operating System	Windows 11 Home Edition
Installed RAM	16.00 GB
Processor	AMD Ryzeb 7 4800H with Radeon Graphics 2.90 GHz
System Type	64-Bit Operating System
Programming Language	Python Programming
Package Management	PIP
Development Environment	Jupyter Notebook

2.2 Technical specifications

The research was conducted employing a sophisticated computational tool, the Python language. The following packages were utilized:

1. Numpy
2. Pandas
3. Matplotlib
4. Sklearn
5. NLTK
6. String
7. Re
8. emoji

9. Keras
10. Tensor flow
11. Transformers

3. Project Development

3.1 Data source

The data for this research comprises the tweets collected from the Twitter platform and focus on the 2019 India General Elections, obtained from Kaggle. This is a tweet dataset with the name IndianElection19TwitterData.csv that consists of tweets containing hashtags, mentions, and keywords of the major political parties and leaders in India.

Key Details:

1. Source: Kaggle (a platform for datasets and data science projects).
2. File Format: CSV (Common-Separated Values).
3. Location: https://www.kaggle.com/datasets/yogesh239/twitter-data-about-2019-indian-general-election
4. Attributes:
    - Tweet ID (string): Unique identifier for each tweet.
    - Date and time (string): Date and time of the tweet posted.
    - Username (string): Twitter handle of the user.
    - Tweet Text (string): The full content of the tweet.
5. Total Records: 1,42,566 rows and 4 columns

	Date	User	Tweet
0	2019-05-18 23:50:47+00:00		@anjanaomkashyap I am seeing you as future #bjp spokesperson..
1	2019-05-18 23:20:00+00:00		Trinamool Congress Sitting MP Abhishek Banerjee sends a defamation notice to Senior BJP leader & Prim
2	2019-05-18 23:00:03+00:00		#LokSabhaElections2019
3	2019-05-18 22:53:54+00:00		#LokSabhaElections2019
4	2019-05-18 22:20:48+00:00		PM Modi creates a new record of being the only PM of a democratic country.
5	2019-05-18 22:16:26+00:00		My somewhat biased exit poll for India elections based on ground reports that I have been following - BJP will cr
6	2019-05-18 21:51:56+00:00		@rupasubramanya Even assuming statistical errors in the estimation , #BJP is expected to do better in the 80
7	2019-05-18 21:43:15+00:00		@abhijitmajumder Small correction. Nobody gets appointed in #BJP as PM. They work for it from the ground le
8	2019-05-18 21:36:20+00:00		We still fucking dancing ðŸ•ºðŸ¼ â™ï¸ #INC
9	2019-05-18 21:22:29+00:00		@abhijitmajumder Appointment of Successor!
10	2019-05-18 21:20:05+00:00		So Kolkata Votes in less than 5 hours.
11	2019-05-18 21:19:44+00:00		Which of the following should be top priority of Modi Government after #Loksabhaelections2019
12	2019-05-18 21:09:32+00:00		(3/3)The major difference btw 2014 and 2019 Elections is the role of media, they have become a mouth piece c
13	2019-05-18 21:00:52+00:00		After 5yrs of Modi ðŸ™
14	2019-05-18 20:41:11+00:00		@RenukaJain6
15	2019-05-18 20:35:54+00:00		(1/3)In 2014 BJP got 31.34% of votes and remaining 68.64% votes were polled to other parties, there was an ar
16	2019-05-18 20:34:01+00:00		#bjp @BJP4India @INCIndia @INCKarnataka how much you think Modi spends my tax money on fake photogra
17	2019-05-18 20:05:42+00:00		@ArvindKejriwal you actually have the audacity to think you even matter in the current political scenario... That
18	2019-05-18 19:55:37+00:00		#BreakingNews
19	2019-05-18 19:52:04+00:00		@timesfactcheck please guys report this page this page is working under @toi but in reality this page works un

Figure 1: Dataset

The username in the above image is blurred as per privacy concerns.

3.2 Data Pre-processing

- Checking the size of the dataset, data types, and is there any null values in the dataset.

```
# Size of dataset

data.shape
```

```
(142566, 1)
```

```
data.dtypes
```

```
Tweet    object
dtype: object
```

```
data.isna().sum()
```

```
Tweet    0
dtype: int64
```

Figure 2: Basic preprocessing checks

- The function for cleaning the tweets in the data includes removing URLs, mentioned usernames, hashtags, punctuations, numbers, emojis, and stop words, and converting the text into lowercase and lemmatizing it into base form.

```
# Define a function for cleaning the tweets

lemmatizer = WordNetLemmatizer()

def clean_tweet(text):
    # 1. To remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    # 2. To remove mentioned usernames
    text = re.sub(r'@\w+', '', text)
    # 3. To remove hashtags
    text = re.sub(r'#(\w+)', r'\1', text)
    # 4. To remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # 5. To remove numbers
    text = re.sub(r'\d+', '', text)
    # 6. To remove emojis
    text = emoji.demojize(text)
    # 7. Convert the text into lowercase
    text = text.lower()
    # 8. To remove stopwords
    stop_words = set(stopwords.words('english'))
    text = ' '.join(word for word in text.split() if word not in stop_words)
    # 9. Lemmatization
    text = ' '.join(lemmatizer.lemmatize(word) for word in text.split())
    return text

# Apply the cleaning function to the tweets
data['cleaned_tweet'] = data['Tweet'].apply(clean_tweet)
data
```

Figure 3: Cleaning the tweets

- Dropping the duplicates from the dataset for better training.

```
# drop duplicate values

data = data.drop_duplicates()
data
```

Figure 4: Dropping duplicates

3.3 Sentiment Analysis

- The code below shows how to perform sentiment analysis of the tweets as well as preprocessing using a tokenizer and the already-trained pipeline. First, the sentiment-analyzer pipeline from the Hugging Face library is prepared to classify the sentiment, and for tokenization the AutoTokenizer from the model "bert-base-uncased" is used. The Token_Count column is then created by adjusting the number of tokens per tweet using the tokenizer's tokenize method. Considering BERT's input size is restricted to 512 tokens for each instance, Tweets_Truncation column is included where tweets can be either truncated or padded depending on its length and then converted back into text form using the encode and decode functions of the tokenizer. Last of the features, the Sentiment column is generated as the result of the corresponding sentiment analyzer applied to the motionless first 280 characters of each tweet and containing 'POSITIVE' or 'NEGATIVE' values. This pipeline guarantees preprocessing and accurate sentiment classification besides constraining models to their capacity.

```
sentiment_analyzer = pipeline('sentiment-analysis')
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and
o/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recom
```

```
# Tokenizer initialization for checking token length
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
```

```
# Apply the count_tokens function to the 'Tweets' column
data['Token_Count'] = data['Tweets'].apply(lambda x: len(tokenizer.tokenize(x)))
```

```
Token indices sequence length is longer than the specified maximum sequence length for t
uence through the model will result in indexing errors
```

```
data['Tweets_Truncated'] = data['Tweets'].apply(
    lambda x: tokenizer.decode(tokenizer.encode(x, max_length=512, truncation=True)))
```

```
data['Sentiment'] = data['Tweets_Truncated'].apply(
    lambda x: sentiment_analyzer(x, truncation=True, padding=True)[0]['label'])
```

Figure 5: Sentiment Labeling using BERT

3.4 Exploratory Data Analysis

- Bar chart and pie chart diagram for understanding about the data.

```
# 3. Visualization: Bar chart for sentiment analysis
import matplotlib.pyplot as plt

# Visualization: Bar chart for sentiment analysis

ax = support_summary[['POSITIVE', 'NEGATIVE']].plot(kind='bar')

# Adding values on top of the bars with adjusted position
for p in ax.patches:
    height = p.get_height()
    # Position the value slightly above the bar (or below if height is very small)
    ax.annotate(f'{int(height)}',
                (p.get_x() + p.get_width() / 2,
                 height + 1),  # Add 1 to position it above the bar
                ha='center', va='bottom',
                fontsize=10)

plt.title('Sentiment Analysis of Political Support')
plt.xlabel('Political Party')
plt.ylabel('Number of Tweets')
plt.xticks(rotation=0)
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()
```

Figure 6: Bar chart diagram code

```python
# Filter the data for positive sentiments
positive_data = data[data['Sentiment'] == 'POSITIVE']

# Count the number of positive tweets for each political party, considering only BJP and INC
partwise_support = positive_data[positive_data['Political_party'].isin(['BJP', 'INC'])]['Political_party'].value_counts()

# Colors: Orange for BJP, Sky Blue for INC
colors = ['orange', '#87CEEB']  # Sky Blue

# Exploding the INC slice slightly
explode = (0, 0.1)  # Only "explode" the 2nd slice (INC)

# Create the pie chart
plt.figure(figsize=(10, 8))
plt.pie(partwise_support,
        labels=partwise_support.index,
        autopct='%1.1f%%',
        startangle=140,
        colors=colors,
        explode=explode,
        shadow=True,
        textprops=dict(color="black", fontsize=12))

# Adjust the title position
plt.title('Partwise Support (Positive Sentiment) for BJP and INC', fontsize=16, pad=20)  # Added padding

plt.axis('equal')  # Equal aspect ratio ensures that pie chart is circular.
plt.tight_layout()  # Adjusts the layout
plt.show()
```

Figure 7: Pie chart diagram code

3.5 Model Development

- Code for implementation of SVM and LSTM.

```python
]: tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2))

]: X_tfidf = tfidf_vectorizer.fit_transform(X)

]: # Apply SMOTE to balance the data
   smote = SMOTE(random_state=42)
   X_balanced, y_balanced = smote.fit_resample(X_tfidf, y)

]: # Split the data into 80% training and 30% testing sets
   X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, test_size=0.3, random_state=42)

]: # Train the SVM model
   svm_model = LinearSVC()
   svm_model.fit(X_train, y_train)

   # Make predictions
   y_pred = svm_model.predict(X_test)
```

Figure 8: SVM Model development

```python
: y_labels = label_encoder.fit_transform(y)

: X_train_lstm, X_test_lstm, y_train_lstm, y_test_lstm = train_test_split(X, y_labels, test_size=0.3, random_state=42)

: tokenizer = Tokenizer(num_words=10000)
  tokenizer.fit_on_texts(X_train_lstm)

: X_train_seq = tokenizer.texts_to_sequences(X_train_lstm)
  X_test_seq = tokenizer.texts_to_sequences(X_test_lstm)

: max_length = 100
  X_train_pad = pad_sequences(X_train_seq, maxlen=max_length)
  X_test_pad = pad_sequences(X_test_seq, maxlen=max_length)

: # Apply SMOTE for LSTM
  X_train_pad_balanced, y_train_lstm_balanced = smote.fit_resample(X_train_pad, y_train_lstm)

: lstm_model = Sequential([
      Embedding(input_dim=10000, output_dim=128, input_length=max_length),
      LSTM(64, return_sequences=True),
      Dropout(0.4),
      LSTM(32, return_sequences=False),
      Dropout(0.4),
      Dense(1, activation='sigmoid')
  ])

: lstm_model.compile(loss='binary_crossentropy', optimizer=AdamW(learning_rate=0.001), metrics=['accuracy'])
```

```
# Train LSTM model
early_stopping = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)
checkpoint_path = "best_model.keras"
model_checkpoint = ModelCheckpoint(filepath=checkpoint_path, monitor='val_loss', save_best_only=True, mode='min')
```

```
history = lstm_model.fit(
    X_train_pad_balanced, y_train_lstm_balanced,
    validation_split=0.2,
    epochs=5,
    batch_size=128,
    callbacks=[early_stopping, model_checkpoint]
)
```

```
Epoch 1/5
609/609 ━━━━━━━━━━━━━━━━━━━━ 83s 131ms/step - accuracy: 0.8080 - loss: 0.4424 - val_accuracy: 0.8643 - val_loss: 0.3151
Epoch 2/5
609/609 ━━━━━━━━━━━━━━━━━━━━ 85s 139ms/step - accuracy: 0.8877 - loss: 0.2716 - val_accuracy: 0.8734 - val_loss: 0.3026
Epoch 3/5
609/609 ━━━━━━━━━━━━━━━━━━━━ 88s 144ms/step - accuracy: 0.9135 - loss: 0.2101 - val_accuracy: 0.8709 - val_loss: 0.3214
Epoch 4/5
609/609 ━━━━━━━━━━━━━━━━━━━━ 85s 140ms/step - accuracy: 0.9336 - loss: 0.1653 - val_accuracy: 0.8670 - val_loss: 0.3533
Epoch 5/5
609/609 ━━━━━━━━━━━━━━━━━━━━ 87s 142ms/step - accuracy: 0.9478 - loss: 0.1331 - val_accuracy: 0.8670 - val_loss: 0.4097
```

```
# Evaluate LSTM model
test_loss, test_accuracy = lstm_model.evaluate(X_test_pad, y_test_lstm)
print(f'Test Accuracy: {test_accuracy * 100}')
```

Figure 9: LSTM Model development

3.6 Results

- Code for finding results of SVM and LSTM for comparison and we plotted the ROC curve and accuracy plot for a better understanding of the model's performance.

```
# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 85.91503737780334
```

```
print("\n",classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.88      0.95      0.91     21350
    POSITIVE       0.77      0.57      0.65      6474

    accuracy                           0.86     27824
   macro avg       0.82      0.76      0.78     27824
weighted avg       0.85      0.86      0.85     27824
```

```
# Confusion Matrix for SVM
svm_cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(svm_cm, display_labels=['Negative', 'Positive'])
disp.plot(cmap='Blues', values_format='d')
plt.title('SVM Confusion Matrix')
plt.show()
```

Figure 10: SVM Model Results

```
# SVM ROC Curve
fpr_svm, tpr_svm, thresholds_svm = roc_curve(y_test_binary, svm_model.decision_function(X_test))
roc_auc_svm = auc(fpr_svm, tpr_svm)

# Plotting SVM ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_svm, tpr_svm, color='blue', lw=2, label=f'SVM (AUC = {roc_auc_svm:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for SVM Model')
plt.legend(loc='lower right')
plt.show()
```

Figure 11: SVM ROC Curve

```
# Evaluate LSTM model
test_loss, test_accuracy = lstm_model.evaluate(X_test_pad, y_test_lstm)
print(f'Test Accuracy: {test_accuracy * 100}')
```

```
1305/1305 ━━━━━━━━━━━━━━━━━━━━ 26s 20ms/step - accuracy: 0.8705 - loss: 0.3028
Test Accuracy: 86.80963516235352
```

```
y_pred_lstm = (lstm_model.predict(X_test_pad) > 0.5).astype("int32")
```

```
1305/1305 ━━━━━━━━━━━━━━━━━━━━ 25s 19ms/step
```

```
print(classification_report(y_test_lstm, y_pred_lstm))
```

```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     31999
           1       0.76      0.64      0.69      9736

    accuracy                           0.87     41735
   macro avg       0.83      0.79      0.80     41735
weighted avg       0.86      0.87      0.86     41735
```

```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test_lstm, y_pred_lstm)

disp = ConfusionMatrixDisplay(conf_matrix, display_labels=['Negative', 'Positive'])
disp.plot(cmap='Blues', values_format='d')
plt.title('LSTM Confusion Matrix')
plt.show()
```

Figure 12: LSTM Model Results

```
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
```

```
# Plot Accuracy Curves
plt.figure(figsize=(12, 6))
plt.plot(train_accuracy, label='Training Accuracy', color='blue')
plt.plot(val_accuracy, label='Validation Accuracy', color='orange')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()
```

Figure 13: LSTM Accuracy plot

```
# ROC Curve for LSTM
fpr_lstm, tpr_lstm, thresholds_lstm = roc_curve(y_test_lstm, y_pred_lstm_prob)
roc_auc_lstm = auc(fpr_lstm, tpr_lstm)

# Plot ROC curve for LSTM
plt.figure(figsize=(8, 6))
plt.plot(fpr_lstm, tpr_lstm, color='green', lw=2, label=f'LSTM (AUC = {roc_auc_lstm:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for LSTM Model')
plt.legend(loc='lower right')
plt.show()
```

Figure 14: LSTM ROC Curve