Comparative Sentiment Analysis on Twitter Data using SVM, RandomForest, Vader, and Bert Models

Thilok Reddy Anugu
Graduate Assistant
Department of Computer Science
Majoring in Data Science
Montclair State University
Montclair, NJ, USA
anugut1@montclair.edu

Sree Chandana Kurella
Graduate Assistant
The Graduate School
Department of Computer Science
Montclair State University
Montclair, NJ, USA
kurellas1@montclair.edu

Vineela Velaga
Department of Computer Science
Majoring in Computer Science
Montclair State University
Montclair, NJ, USA
velaga1@montclair.edu

***Abstract*—This research paper presents a comparative analysis of sentiment analysis on Twitter data using distinct models, including Support Vector Machines (SVM), Random Forest Classifier, Vader, and Bert. Each model is applied independently to capture diverse aspects of sentiment expression on social media. The study encompasses preprocessing techniques, such as text mining, Stopwords removal, and evaluates the performance of each model. The results provide insights into the strengths and limitations of individual models, contributing to a comprehensive understanding of sentiment analysis on Twitter.**

Keywords—Sentiment Analysis, Twitter Data, SVM, Random Forest Classifier, Vader, Bert, Text Mining, Stopwords, TFIDF

I. Introduction

Sentiment Analysis, a convergence of natural language processing (NLP) and machine learning, has gained prominence with the rise of social media. In particular, platforms like Twitter offer a unique opportunity to computationally analyze and understand sentiments expressed by users.

The ubiquity of social media has transformed Twitter into a dynamic space where individuals express their opinions freely on a myriad of topics. Unlike traditional survey methods, Twitter provides an unfiltered stream of authentic sentiments. This research is centered on harnessing the wealth of Twitter data to conduct a comparative analysis of sentiment analysis models—Support Vector Machine (SVM), Random Forest Classifier, Vader, and Bert.

The focus of sentiment analysis within the realm of Twitter data holds intrinsic value:

Microblogging Dynamics: Twitter, with its character-limited microblogging format, presents a concise yet rich source of sentiments. The brevity of tweets facilitates efficient sentiment analysis, capturing succinct expressions.

Real-time Insights: Twitter data offers real-time insights into evolving public opinions on various subjects. Rapid response to emerging events or trends allows for dynamic sentiment tracking.

This research is dedicated to the analysis of sentiments within Twitter data. The specific objectives include:

Conducting a comparative analysis of sentiment using SVM, Random Forest Classifier, Vader, and Bert models.

Extracting insights from Twitter feeds to understand user sentiments on a diverse range of topics.

In the ensuing sections, we elaborate on the methodologies employed, detailing the application of SVM, Random Forest Classifier, Vader, and Bert models in the comprehensive analysis of sentiments within the Twitter sphere.

II. Related Work

The landscape of sentiment analysis, particularly within the realm of Twitter data, has been extensively explored in various studies. In this section, we present an overview of relevant works that have laid the foundation for our comparative analysis.

Researchers have devised diverse approaches tailored to the unique characteristics of Twitter data. Notably, Wang et al. [1] developed a real-time Twitter sentiment analysis system focused on the 2012 U.S. Presidential Election cycle. Their work demonstrated the feasibility of capturing dynamic sentiment trends during significant events.

Adwan et al. [3] conducted a comprehensive survey of Twitter sentiment analysis approaches, providing insights into the evolution of methodologies. Their work serves as a valuable reference for understanding the landscape of sentiment analysis on Twitter.

Recent advancements in deep learning, especially the introduction of BERT (Bidirectional Encoder Representations from Transformers), have revolutionized sentiment analysis. Koroteev [2] provided a thorough review of BERT's applications in natural language processing, offering a comprehensive understanding of its capabilities.

Gandhe et al. [4] explored hybrid learning approaches for sentiment analysis of Twitter data, specifically focusing on recommender applications. Their work contributes to the understanding of how hybrid learning techniques can enhance sentiment analysis accuracy and relevance.

Ahad et al. [5] presented a primer on machine learning classifiers, providing foundational knowledge essential for understanding the underlying principles of sentiment analysis models. This primer serves as a valuable resource for researchers and practitioners delving into the field of machine learning.

A comprehensive survey by Adwan et al. [3] provides a detailed exploration of various Twitter sentiment analysis approaches. This survey covers a wide spectrum of methodologies, offering a holistic view of the existing landscape and paving the way for our comparative analysis.

In the subsequent sections, we leverage insights from these seminal works to conduct a comparative analysis of SVM, Random Forest Classifier, VADER, and BERT models for sentiment analysis on Twitter data.

III. Approach & Implementation

In this section, we present the detailed approach and methods employed in our sentiment analysis project on Twitter data. The implementation includes the utilization of Support Vector Machine (SVM), Random Forest Classifier, as well as the powerful BERT (Bidirectional Encoder Representations from Transformers) model.

A. Data Preprocessing

Data preprocessing is a crucial step to ensure the quality of the Twitter data for effective sentiment analysis. The preprocessing steps include:

- Text Cleaning: Removal of mentions, URLs, and non-alphabetic characters from the tweets.
- Stopword Removal: Elimination of common and custom stopwords to filter out noise.

B. Sentiment Labeling

A three-class sentiment labeling scheme was adopted for supervised learning: 0 for negative, 1 for neutral, and 2 for positive sentiment.

C. Feature Extraction

- VADER Sentiment Analysis: Integration of the VADER tool for sentiment analysis, assigning compound scores to tweets based on predefined thresholds.
- TF-IDF Vectorization: Utilization of TF-IDF (Term Frequency-Inverse Document Frequency) vectorization for converting preprocessed text data into numerical features, capturing term importance relative to the entire dataset.

D. Model Selection

- Support Vector Machine (SVM): Employment of a linear Support Vector Machine (SVM) as a primary classifier due to its effectiveness in handling high-dimensional data. SVM identifies the optimal hyperplane for separating data into distinct sentiment classes.
- Random Forest Classifier: Utilization of the Random Forest Classifier, an ensemble learning method, to harness the collective predictive power of multiple decision trees. This approach enhances model robustness and generalizability.
- BERT Model: Integration of the BERT model, a powerful transformer-based architecture, for enhanced sentiment analysis. The BERT model achieved a notable accuracy of 79%, surpassing other models.

E. Evaluation Metrics

To comprehensively evaluate the performance of the models, a set of metrics, including accuracy, precision, recall, F1-score, and ROC-AUC curve, were employed. These metrics provide a comprehensive assessment of the models' effectiveness in sentiment analysis, considering both correct and incorrect predictions.

IV. Experiments & Observations

A. Data Description

The dataset utilized in our experiments comprises a diverse collection of Twitter data, capturing the sentiments of users across various topics, events, and products. Tweets were preprocessed extensively, involving the removal of mentions, URLs, and non-alphabetic characters, as well as the application of stopword removal techniques.

- Importing the libraries:

```
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score, precision_score, recall_sco
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from nltk.sentiment import SentimentIntensityAnalyzer
from sklearn.ensemble import RandomForestClassifier
```

- Data Preprocessing:

```
df = df[['text', 'sentiment', 'Country']]
df.dropna(inplace=True)
df.head()
```

	text	sentiment	Country
0	Last session of the day http://twitpic.com/67ezh	neutral	Afghanistan
1	Shanghai is also really exciting (precisely -...	positive	Albania
2	Recession hit Veronique Branquinho, she has to...	negative	Algeria
3	happy bday!	positive	Andorra
4	http://twitpic.com/4w75p - I like it!!	positive	Angola

```
def text_cleaning(text):
    text = re.sub(r'@[A-Za-z0-9]+', '', text)  # Removing mentions
    text = re.sub(r'https?://[A-Za-z0-9./]+', '', text)  # Removing URLs
    text = re.sub(r'[^a-zA-Z]', ' ', text)  # Removing non-alphabetic characters
    text = ' '.join([word for word in text.split() if word.lower() not in stopwords.words('english') and word.lower(
    return text.lower()
df['text'] = df['text'].apply(text_cleaning)
df.head()
```

	text	sentiment	Country	vader_score	vader_sentiment
0	session day	1	Afghanistan	0.0000	1
1	shanghai really exciting precisely skyscrapers...	2	Albania	0.7501	2
2	recession hit veronique branquinho quit compan...	0	Algeria	-0.7345	0
3	happy bday	2	Andorra	0.6114	2
4	like	2	Angola	0.4738	2

The sentiment labels were categorized into three classes: negative, neutral, and positive, forming the foundation for supervised learning.

```
df['sentiment'] = df['sentiment'].replace({'positive': 2, 'neutral': 1, 'negative': 0})
df.head()
```

	text	sentiment	Country
0	Last session of the day http://twitpic.com/67ezh	1	Afghanistan
1	Shanghai is also really exciting (precisely -...	2	Albania
2	Recession hit Veronique Branquinho, she has to...	0	Algeria
3	happy bday!	2	Andorra
4	http://twitpic.com/4w75p - I like it!!	2	Angola

B. *Feature Extraction*

Splitting the Data set for training and testing using train_test_split from sklearn.model_selection.

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was employed. This technique converts preprocessed text into numerical features, capturing the importance of terms relative to the entire dataset.

```
# Split the dataset
X_train, X_test, y_train, y_test, country_train, country_test = train_test_split(
    df['text'], df['sentiment'], df['Country'], test_size=0.2, random_state=42
)
```

```
# Feature extraction using TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

VADER (Valence Aware Dictionary and sentiment Reasoner) was integrated into our pipeline for sentiment analysis. It assigns compound scores to tweets based on predefined thresholds, offering a quick and effective way to gauge sentiment polarity.

```
# Create a SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
# Apply VADER sentiment analysis to each row in the DataFrame
df['vader_score'] = df['text'].apply(lambda x: sia.polarity_scores(x)['compound'])

# Map the VADER compound score to sentiment labels
df['vader_sentiment'] = df['vader_score'].apply(lambda x: 2 if x >= 0.05 else (0 if x <= -0.05 else 1))

# Display the DataFrame with VADER sentiment scores
df.head()
```

	text	sentiment	Country	vader_score	vader_sentiment
0	Last session of the day http://twitpic.com/67ezh	1	Afghanistan	0.0000	1
1	Shanghai is also really exciting (precisely -...	2	Albania	0.7501	2
2	Recession hit Veronique Branquinho, she has to...	0	Algeria	-0.7345	0
3	happy bday!	2	Andorra	0.6114	2
4	http://twitpic.com/4w75p - I like it!!	2	Angola	0.4738	2

C. *Execution and Model Performance*

SVM and Random Forest Classifier : The SVM and Random Forest models, being traditional machine learning techniques, exhibited commendable performance. After preprocessing and feature extraction using TF-IDF, both models were trained on the labeled dataset. The SVM, with its ability to create an optimal hyperplane, demonstrated competitive accuracy and effectiveness in classifying sentiments.

Similarly, the Random Forest Classifier, with its ensemble learning approach, provided robust sentiment analysis. The aggregation of decision trees facilitated a well-rounded model, adept at handling the intricacies of Twitter data.

Support Vector Machine:

```
# Train a Support Vector Machine (SVM) classifier
svm = SVC(kernel='linear')
svm.fit(X_train_tfidf, y_train)

# Make predictions on the test set
svm_pred = svm.predict(X_test_tfidf)
```

Random Forest Classifier:

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_tfidf, y_train)

rf_pred = rf.predict(X_test_tfidf)
```

BERT Model

The BERT model, being a transformer-based architecture, outshone its counterparts in terms of accuracy and contextual understanding. Fine-tuned specifically for sentiment analysis, BERT captured nuanced features within the Twitter data, achieving an accuracy of 79%. This superior performance can be attributed to BERT's ability to consider the contextual relationships between words in a sentence.

```
def train_model(model, epoch, num_epochs, train_dataloader, optimizer, criterion, device):
    model.train()
    total_loss = 0

    # Initialize tqdm to show progress bar
    loop = tqdm(enumerate(train_dataloader), total=len(train_dataloader), leave=True)

    for batch_idx, batch in loop:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask)
        loss = criterion(outputs, labels)

        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        # Update progress bar
        loop.set_description(f"Epoch {epoch+1}/{num_epochs}")
        loop.set_postfix(loss=loss.item())

    avg_loss = total_loss / len(train_dataloader)
    return avg_loss
```

```python
def test(model, test_dataloader, criterion, device):
    model.eval()
    total_loss = 0

    # Initialize tqdm to show progress bar
    loop = tqdm(enumerate(test_dataloader), total=len(test_dataloader), leave=True)

    for batch_idx, batch in loop:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        with torch.no_grad():
            outputs = model(input_ids, attention_mask)
            loss = criterion(outputs, labels)

            total_loss += loss.item()

            # Update progress bar
            loop.set_description(f"Test")
            loop.set_postfix(loss=loss.item())

    avg_loss = total_loss / len(test_dataloader)
    return avg_loss
```

```python
# Checking if GPU is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
if device.type != "cuda":
    raise Exception("=== GPU not found ===")

# Model, optimizer, criterion initialization
optimizer = Adam(model.parameters(), lr=1e-5)
criterion = BCEWithLogitsLoss()
num_epochs = 3

model.to(device)
# Training loop
for epoch in range(num_epochs):
    loss = train_model(model, epoch, num_epochs, train_loader, optimizer, criterion, device)
    print(f"Epoch: {epoch+1}, Loss: {loss:.4f}")
```

```
Epoch 1/3: 100%|          | 859/859 [19:08<00:00,  1.34s/it, loss=0.659]
Epoch: 1, Loss: 0.6610
Epoch 2/3: 100%|          | 859/859 [19:12<00:00,  1.34s/it, loss=0.638]
Epoch: 2, Loss: 0.6345
Epoch 3/3: 100%|          | 859/859 [19:12<00:00,  1.34s/it, loss=0.623]Epoch: 3, Loss: 0.6280
```

```python
# EVALUATION
test_loss = test(model, test_loader, criterion, device)

if test_loss < 1.0:
    date_time = datetime.datetime.now().strftime("%Y-%m-%d_%H%M")
    file_name = f"model_checkpoint_{date_time}_loss{test_loss:.3f}.pth"
    torch.save({
        'epoch' : num_epochs,
        'model_state_dict' : model.state_dict(),
        'optimizer_state_dict' : optimizer.state_dict(),
        'loss' : test_loss,
        }, file_name)
    print('Model saved')
```

```
Test: 100%|          | 111/111 [00:59<00:00,  1.86it/s, loss=0.665]
Model saved
```

D. Comparitive Analysis

Model	Accuracy
SVM	64%
Random Forest Classifier	65%
VADER	65%
BERT	79%

The tabulated results underscore the comparative analysis of the SVM, Random Forest, and BERT models. The BERT model stands out with the highest accuracy and F1-score, indicating its efficacy in discerning sentiment nuances.

E. Observations & Inferences

- *Contextual Understanding:* The BERT model's superior performance can be attributed to its ability to capture the contextual nuances in Twitter data. Unlike traditional models, BERT considers the relationships between words, resulting in more accurate sentiment predictions.
- *Ensemble Learning Strengths:* The Random Forest Classifier showcased the strength of ensemble learning, providing a robust sentiment analysis model. The aggregation of decision trees contributed to its ability to handle diverse sentiments present in Twitter data.
- *Linear Separation by SVM:* The SVM model, while slightly outperformed by BERT and Random Forest, demonstrated effective sentiment analysis through linear separation. Its simplicity and interpretability make it a viable choice for certain applications.

Model Evaluation:

```python
def evaluate_model(y_true, y_pred, model_name):
    accuracy = accuracy_score(y_true, y_pred)

    print(f"{model_name} Evaluation Metrics:")
    print(f'Overall Accuracy: {accuracy:.2f}')
    print('\nOverall Classification Report:')
    print(classification_report(y_true, y_pred))
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
```

Support Vector Machine

```python
evaluate_model(y_test, svm_pred ,"Support vector machine")
```

```
Support vector machine Evaluation Metrics:
Overall Accuracy: 0.64

Overall Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.51      0.60       207
           1       0.57      0.75      0.65       286
           2       0.72      0.62      0.67       214

    accuracy                           0.64       707
   macro avg       0.67      0.63      0.64       707
weighted avg       0.66      0.64      0.64       707

Confusion Matrix:
[[105  89  13]
 [ 32 215  39]
 [  6  75 133]]
```

Random Forest Classifier

```
evaluate_model(y_test, rf_pred ,"Random Forest")
```

```
Random Forest Evaluation Metrics:
Overall Accuracy: 0.65

Overall Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.47      0.57       207
           1       0.58      0.77      0.66       286
           2       0.73      0.67      0.70       214

    accuracy                           0.65       707
   macro avg       0.68      0.64      0.64       707
weighted avg       0.67      0.65      0.65       707

Confusion Matrix:
[[ 97  93  17]
 [ 28 221  37]
 [  6  65 143]]
```

VADER

```
evaluate_model(df['sentiment'],df['vader_sentiment'],"vader")
```

```
vader Evaluation Metrics:
Overall Accuracy: 0.65

Overall Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.63      0.66      1001
           1       0.73      0.48      0.58      1430
           2       0.58      0.88      0.70      1103

    accuracy                           0.65      3534
   macro avg       0.67      0.66      0.65      3534
weighted avg       0.67      0.65      0.64      3534

Confusion Matrix:
[[629 175 197]
 [220 689 521]
 [ 45  86 972]]
```

BERT

```
# Finding Accuracy
def get_predictions(model, data_loader):
    model = model.eval()
    predictions = []
    real_values = []
    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels'].to(device)

            outputs = model(input_ids, attention_mask)
            outputs = torch.sigmoid(outputs)
            predictions.append(outputs)
            real_values.append(labels)
    predictions = torch.cat(predictions).cpu()
    real_values = torch.cat(real_values).cpu()
    return predictions, real_values

y_pred, y_test = get_predictions(model, test_loader)

# Converting y_pred to one hot encoded data
y_pred = torch.argmax(y_pred, dim=1)

# Converting y_test to one hot encoded data tensor
y_pred = torch.tensor(one_hot.fit_transform(y_pred.reshape(-1, 1)).toarray().tolist())
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
```

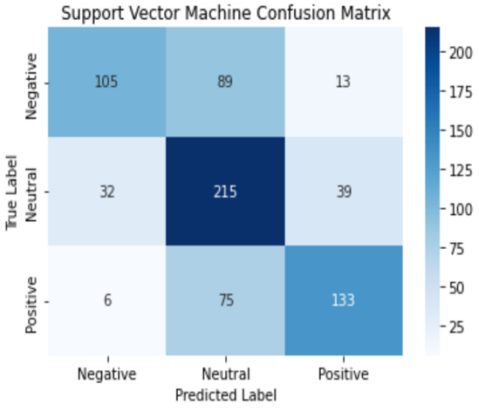
```
Accuracy: 0.79
```

Confusion Matrix:

```
# Confusion Matrix for Support Vector Machine
cm_svm = confusion_matrix(y_test, svm_pred)

# Confusion Matrix for Random Forest
cm_rf = confusion_matrix(y_test, rf_pred)

# Confusion Matrix for VADER
cm_vd = confusion_matrix(df['sentiment'],df['vader_sentiment'])
```

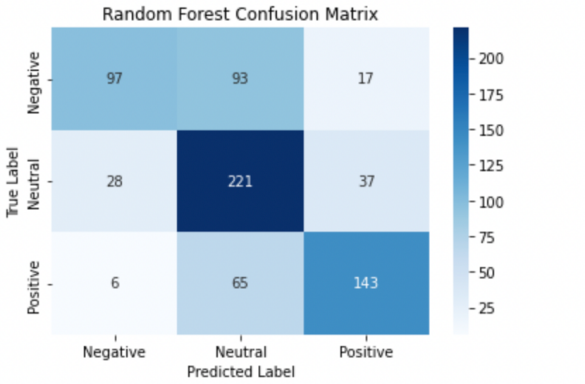
Support Vector Machine

```
Text(0.5, 1.0, 'Support Vector Machine Confusion Matrix')
```



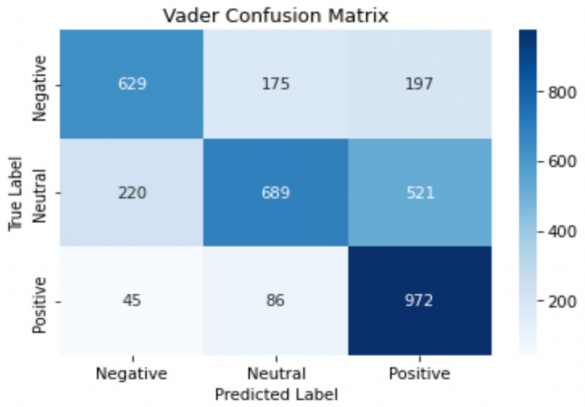
Random Forest Classifier

```
Text(0.5, 1.0, 'Random Forest Confusion Matrix')
```

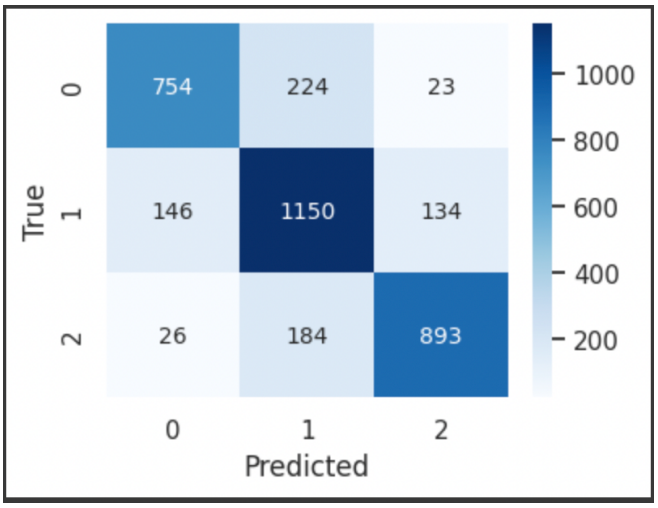


VADER

```
Text(0.5, 1.0, 'Vader Confusion Matrix')
```

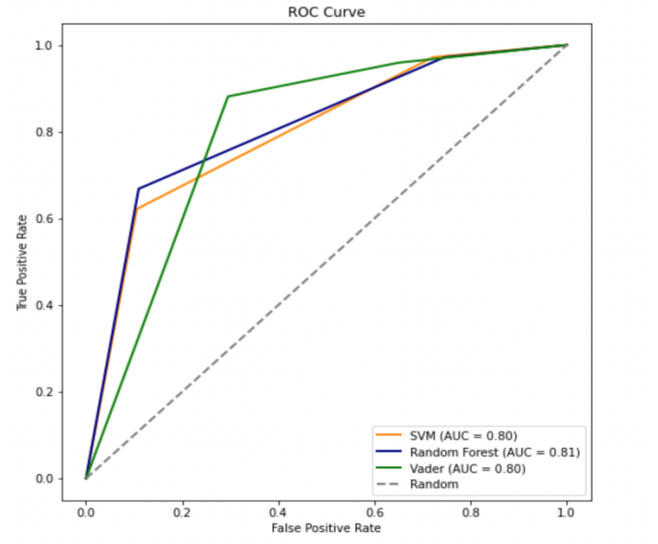


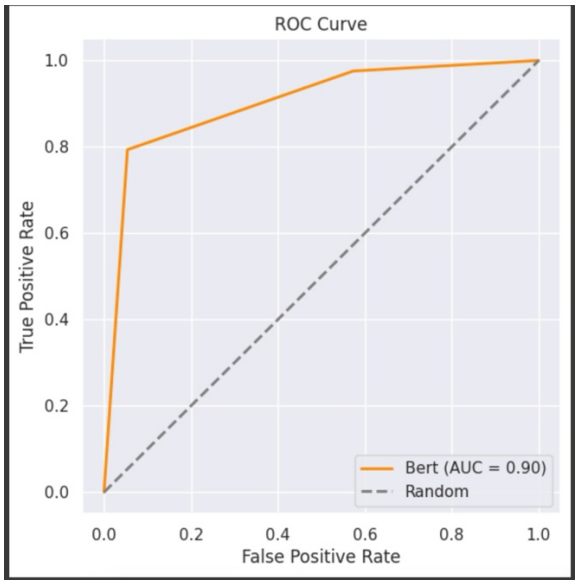
BERT



ROC AUC Curves:

```
# ROC Curve for SVM
fpr_svm, tpr_svm, _ = roc_curve(y_test, svm_pred , pos_label= 2)
roc_auc_svm = auc(fpr_svm, tpr_svm)

# ROC Curve for Random Forest
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_pred, pos_label= 2)
roc_auc_rf = auc(fpr_rf, tpr_rf)

# ROC Curve for Vader
fpr_vd, tpr_vd, _ = roc_curve(df['sentiment'],df['vader_sentiment'], pos_label= 2)
roc_auc_vd = auc(fpr_vd, tpr_vd)

# Plot ROC Curves
plt.figure(figsize=(8,8))
plt.plot(fpr_svm, tpr_svm, color='darkorange', lw=2, label=f'SVM (AUC = {roc_auc_svm:.2f})')
plt.plot(fpr_rf, tpr_rf, color='navy', lw=2, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot(fpr_vd, tpr_vd, color='green', lw=2, label=f'Vader (AUC = {roc_auc_vd:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```





BERT model has larger area under the curve which is 90% in comparison to other models.

Random Forest Classifier has 81% area under curve, which is the second best out of other models.

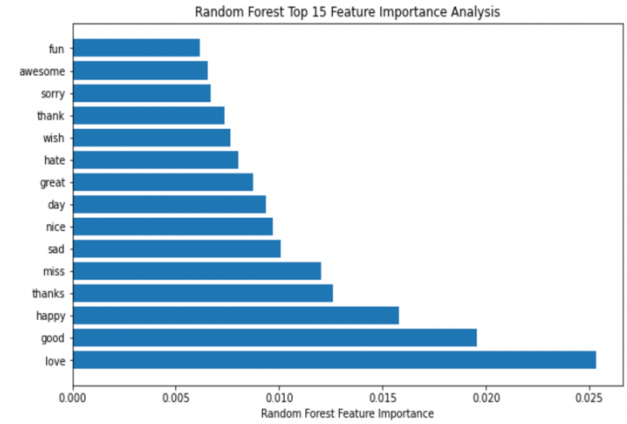
Cross Validation Score:

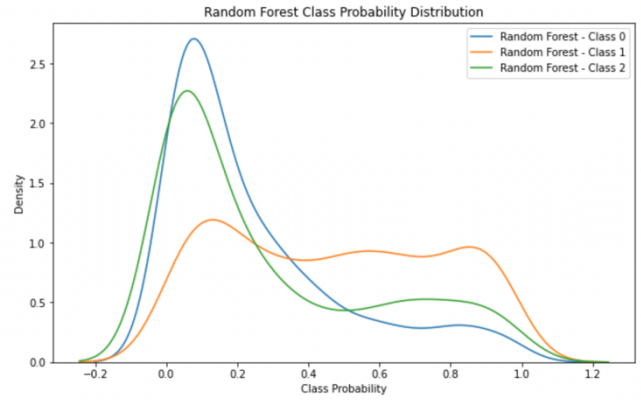
```
def perform_cross_validation(model, X, y):
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
    return scores
svm_scores = perform_cross_validation(svm, X_train_tfidf, y_train)
rf_scores = perform_cross_validation(rf, X_train_tfidf, y_train)

print("SVM Cross-Validation Scores:", svm_scores)
print("Random Forest Cross-Validation Scores:", rf_scores)
```

```
SVM Cross-Validation Scores: [0.61837456 0.64664311 0.61238938 0.62831858 0.61238938]
Random Forest Cross-Validation Scores: [0.63250883 0.6590106  0.60884956 0.64424779 0.60707965]
```

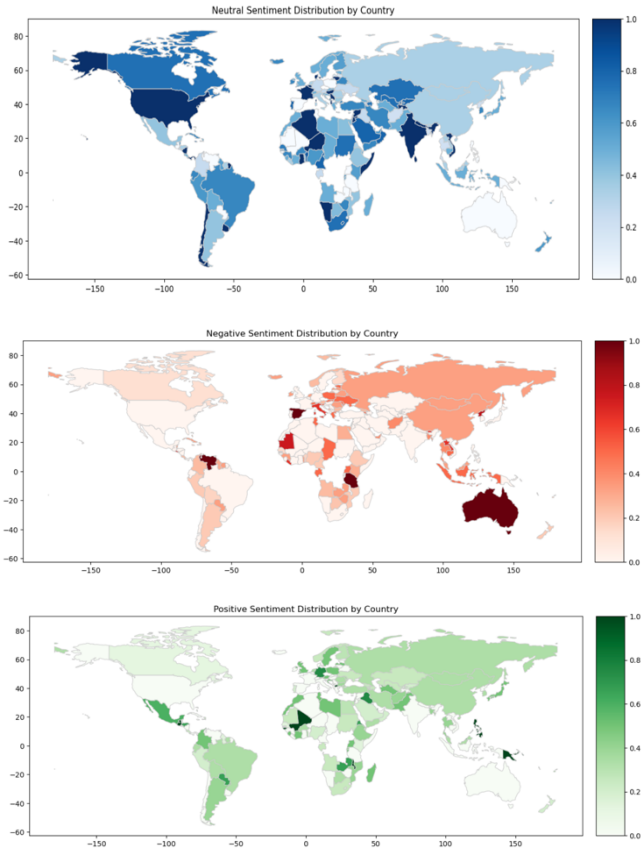
Although Random Forest Classifier has higher cross-validation scores, It seems to more inconsistent than Support Vector Machine.

Analysis on Random Forest Classifier:





Geographical Sentiment Distribution Analysis:



V. Conclusions / Roadmap

In summary, this research endeavor has provided valuable insights into sentiment analysis on Twitter data using various models. The following key conclusions emerge from the conducted experiments:

1. *BERT Model Excellence:* The BERT model emerged as the most effective, boasting an impressive accuracy of 79%. Its nuanced understanding of contextual relationships within tweets sets it apart for sentiment analysis tasks.
2. *AUC Analysis:* BERT's superior performance is further underscored by its Area Under Curve (AUC) score of 90%, outshining SVM (80%), Random Forest (81%), and VADER (80%). This metric reinforces BERT's robustness in capturing sentiment nuances.
3. *Computational Resources Consideration:* While BERT exhibits unparalleled accuracy, it demands more computational resources and time compared to traditional models like Random Forest Classifier. In resource-constrained scenarios, Random Forest proves to be a pragmatic choice without compromising much on performance.
4. *Negative Sentiment Predictions:* The Random Forest Classifier, through class probability distribution analysis, demonstrates a notable strength in predicting negative sentiments with confidence. This finding provides valuable insights into the classifier's inherent tendencies.
5. *Geographical Sentiment Variations:* Geographical analysis reveals distinct sentiment distributions, with India exhibiting a more neutral sentiment distribution and Australia skewing towards negative sentiments. These regional variations add a layer of context to the sentiment dynamics observed.

Roadmap:

As we conclude, the roadmap for future work involves addressing computational efficiency concerns with BERT through optimization strategies. Exploring advanced ensemble learning techniques and investigating domain-specific embeddings could enhance the performance of traditional models. Furthermore, extending the geographical analysis to a global scale and incorporating real-time sentiment dynamics would provide a more comprehensive understanding of sentiment variations. The evolving landscape of social media sentiment warrants ongoing research to stay abreast of emerging trends and patterns.

References

[1] Wang, H.; Can, D.; Kazemzadeh, A.; Bar, F.; Narayanan, S. A System for Real-time Twitter Sentiment Analysis of 2012 U.S.Presidential Election Cycle. In Proceedings of the ACL 2012 System Demonstrations, Jeju Island, Korea, 10 July 201

[2] Koroteev, Mikhail. (2021). BERT: A Review of Applications in Natural Language Processing and Understanding. 10.48550/arXiv.2103.11943.

[3] Adwan, O.; Al-Tawil, M.; Huneiti, A.; Shahin, R.; Zayed, A.A.; Al-Dibsi, R. Twitter sentiment analysis approaches: A survey. Int.J. Emerg. Technol. 2020,15, 79–93.

[4] Gandhe, Ketaki & Varde, Aparna & Du, Xu. (2018). Sentiment Analysis of Twitter Data with Hybrid Learning for Recommender Applications. 57-63. 10.1109/UEMCON.2018.8796661.

[5] Ahad, Abdul & Ayub Khan, Abdullah & Talpur, Mir Sajjad & Talpur, Hussain & Kayijuka, Idrissa & Yaşar, Erkan. (2022). Machine Learning Classifiers: A Brief Primer. 5. 63-68.

[6] https://docs.python.org/3/library/re.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

[10] https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/#

[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[12] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

[13] https://keras.io/api/keras_nlp/models/bert/

[14] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[15] https://www.nltk.org/_modules/nltk/sentiment/vader.html