SENTIMENT ANALYSIS FOR MARKETING

PHASE 2

NAME	SARAVANAN S
TEAM ID	proj_212173_team_2

Project : Sentient Analysis For Marketing

Introduction:

Sentiment analysis using BERT and RoBERTa models is a powerful approach to extract sentiment information from text data. These models, based on transformer architecture, have achieved state-of-the-art performance on various NLP tasks, including sentiment analysis. In this introduction, I'll walk you through the steps to perform sentiment analysis using the Hugging Face Transformers library, which provides pre-trained BERT and RoBERTa models.

Data Collection and Preprocessing:

- Importing the dataset : Obtain a comprehensive dataset containing relevant features such as tweet count, tweet timezone, tweet id, etc.,
- Data pre-processing : Clean the data by handling missing values, outliers and categorical variables. Standardize or normalize numerical features

Exploratory Data Analysis(EDA):

- Visualize and analysis the dataset to gain insights into the relationship between variables.
- Identify correlations and patterns that can inform features selected and engineering

ADVANCED TECHNIQUES:

- BERT or RoBERTa for Text Embeddings:

  First, you can use BERT or RoBERTa to generate text embeddings (vectors) for your text data. These embeddings capture the semantic information of the text, which you can then use as input to a regression model.
- Random Forest Regressor:

Random Forest is an ensemble learning method that can handle both regression and classification tasks effectively. It's known for its ability to capture complex relationships in the data.

- Gradient Boosting Regressor (e.g., XGBoost, LightGBM, or CatBoost): Gradient boosting algorithms often provide excellent predictive performance by combining the predictions of multiple weak learners. Each of these libraries (XGBoost, LightGBM, and CatBoost) has its advantages and can be fine-tuned for optimal results.

DATA SOURCE:

A good data source for Sentimental analysis for marketing using nlp should be Accurate, Complete, Covering the reviews of customers from all possible ways like Social Media, Direct review and trends of products.

Dataset Link: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

PROGRAM:

SENTIMENT ANALYSIS FOR MARKETING

IMPORTING DEPENDENCIES:

```
import pandas as pd

import numpy as np

import torch

import tokenize

import seaborn as sns

import matplotlib.pyplot as plt

import nltk

import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments

from skearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

import xgboost as xg
```

Loading Data:

```
dataset=pd.read_csv('Tweets.csv')

dataset.info()

print(dataset.shape)

print(dataset['airline_sentiment'].value_counts())
```

```
Out[1]:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   tweet_id                      14640 non-null  int64  
 1   airline_sentiment             14640 non-null  object 
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object 
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object 
 6   airline_sentiment_gold        40 non-null     object 
 7   name                          14640 non-null  object 
 8   negativereason_gold           32 non-null     object 
 9   retweet_count                 14640 non-null  int64  
 10  text                          14640 non-null  object 
 11  tweet_coord                   1019 non-null   object 
 12  tweet_created                 14640 non-null  object 
 13  tweet_location                9907 non-null   object 
 14  user_timezone                 9820 non-null   object 
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
(14640, 15)
negative    9178
neutral     3099
positive    2363
```

```
Name: airline_sentiment, dtype: int64
```

Pre-Process the Data:

```
def preprocess_text(text):
    # Remove punctuations and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)

    # Single character removal
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)

    # Removing multiple spaces
    text = re.sub(r'\s+', ' ', text)

    # Converting to Lowercase
    text = text.lower()

    # Lemmatization
    #text = text.split()
    #lemmatizer = WordNetLemmatizer()
    #text = [lemmatizer.lemmatize(word) for word in text if not word in
set(stopwords.words('english'))]
    #text = ' '.join(text)

    return text

# Apply the preprocessing to the 'text' column
df['text'] = df['text'].apply(preprocess_text)

# Display the first 5 rows of the dataframe after preprocessing
df.head()
```

output:

```
S.no   airline_sentiment      text
-----  ---------------------  ------------------------------------------------------------
0      neutral                virginamerica what dhepburn said
1      positive               virginamerica plus you ve added commercials t...
2      neutral                virginamerica didn today must mean need to ta...
```

```
3       negative                virginamerica it really aggressive to blast o...
4       negative                virginamerica and it a really big bad thing a...
```

DATA CLEANING:

```
data = data[['airline_sentiment', 'text']]

data['airline_sentiment'] = data['airline_sentiment'].map({'positive': 2, 'neutral': 1, 'negative': 0})
```

SPLIT THE DATA INTO TRAINING AND TESTING SETS:

```
X = data['text']

y = data['airline_sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

REGRESSION MODELS:

LOGISTIC REGRESSION:

```
model=LogisticRegression(max_iter=10000)

model.fit(train_vec,train_labels)
```

Output : LogisticRegression(max_iter=10000)

RANDOM FORESTING:

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

rf_classifier.fit(X_train_tfidf, y_train)

rf_predictions = rf_classifier.predict(X_test_tfidf)
```

Output:

```
Classification Report for Random Forest:
```

```
              precision    recall  f1-score   support

    negative       0.79      0.93      0.85      1889
     neutral       0.58      0.36      0.44       580
    positive       0.73      0.56      0.64       459

    accuracy                           0.76      2928
   macro avg       0.70      0.62      0.64      2928
weighted avg       0.74      0.76      0.74      2928
```

r_train_accuracy, r_test_accuracy, r_train_auc, r_test_auc= check_scores(RandomForestClassifier(random_state=0).fit(x_train, y_train), x_train,x_test,y_train,y_test)

Output:

```
Train confusion matrix is:
[[6829   26]
 [   5 1795]]

Test confusion matrix is:
[[2215  108]
 [ 238  325]]

              precision    recall  f1-score   support

           0       0.90      0.95      0.93      2323
           1       0.75      0.58      0.65       563

    accuracy                           0.88      2886
   macro avg       0.83      0.77      0.79      2886
weighted avg       0.87      0.88      0.87      2886


Train accuracy score:  0.996418255343732
Test accuracy score:  0.8801108801108801

Train ROC-AUC score:  0.9982442661479861
Test ROC-AUC score:  0.8956867344777572

Are under Precision-Recall curve: 0.6526104417670683
Area under ROC-AUC: 0.7441899264879837
```

GRADIANT BOOSTING CLASSIFICATION:

gb_classifier = GradientBoostingClassifier(n_estimators=100, random_state=42)

gb_classifier.fit(X_train_tfidf, y_train)

```
gb_predictions = gb_classifier.predict(X_test_tfidf)
```

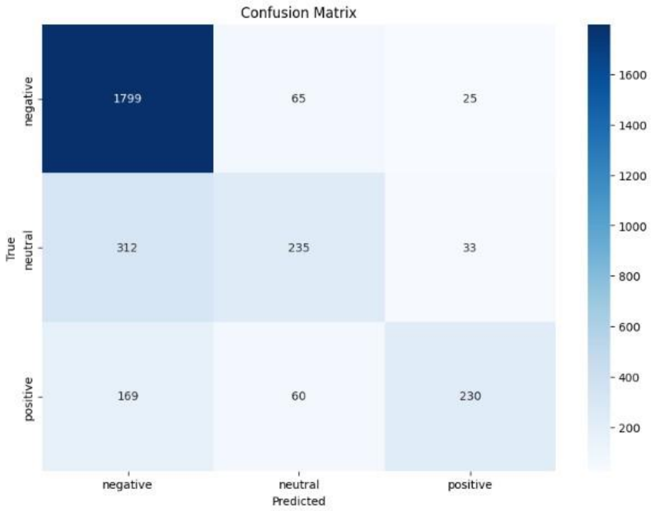
Output:

```
Classification Report for Gradient Boosting:

              precision    recall  f1-score   support

    negative       0.76      0.96      0.85      1889
     neutral       0.67      0.24      0.35       580
    positive       0.74      0.54      0.63       459

    accuracy                           0.75      2928
   macro avg       0.72      0.58      0.61      2928
weighted avg       0.74      0.75      0.71      2928
```

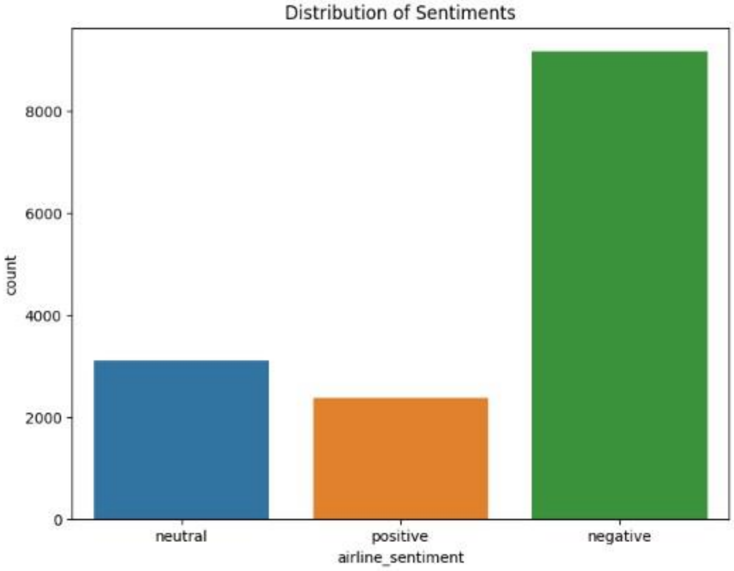
PLOTING THE REGRESSION MODELS:

CONFUSION MATRIX

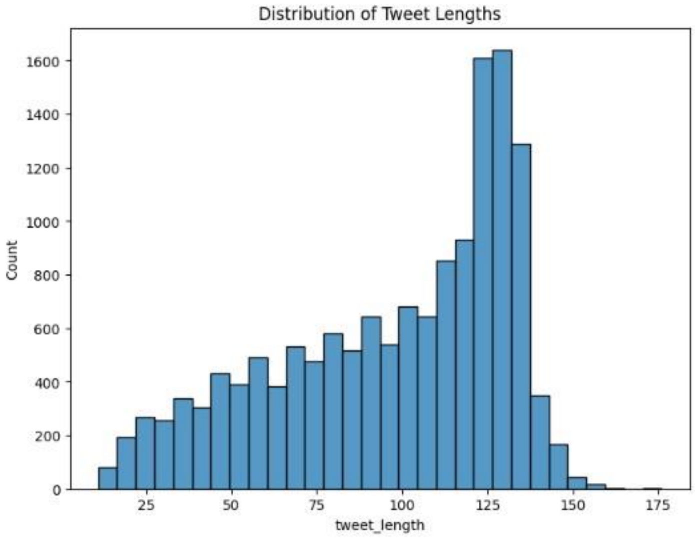
```
def plot_confusion_matrix(y_test, y_pred):
    cm = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cm, index = [i for i in ['negative', 'neutral', 'positive']],
                  columns = [i for i in ['negative', 'neutral', 'positive']])
    plt.figure(figsize = (10,7))
    sns.heatmap(df_cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()
plot_confusion_matrix(y_test, y_pred)
```



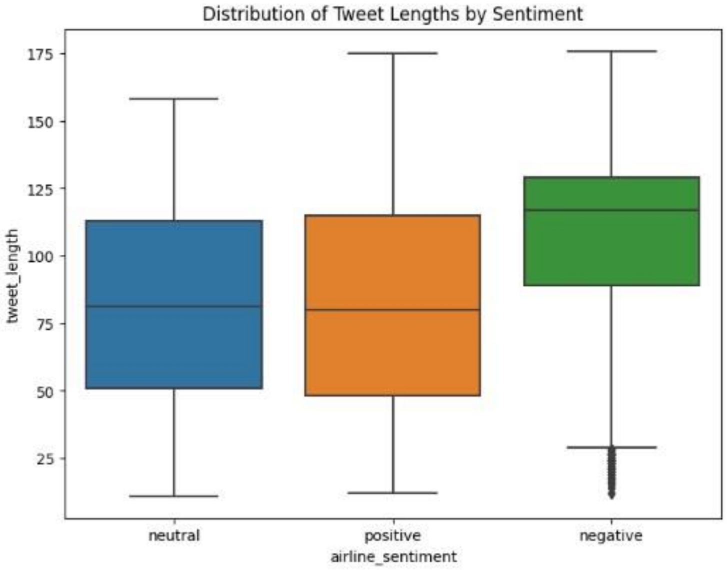
```
# Creating  column 'tweet_length'
df['tweet_length'] = df['text'].apply(len)
# distribution of sentiments
plt.figure(figsize=(8,6))
sns.countplot(x='airline_sentiment', data=df)
plt.title('Distribution of Sentiments')
plt.show()
```



```
# Histogram of tweet lengths
plt.figure(figsize=(8,6))
sns.histplot(df['tweet_length'], bins=30)
plt.title('Distribution of Tweet Lengths')
plt.show()
```



```
# Boxplot of tweet lengths
plt.figure(figsize=(8,6))
sns.boxplot(x='airline_sentiment', y='tweet_length', data=df)
plt.title('Distribution of Tweet Lengths by Sentiment')
plt.show()
```



CONCLUSION:

- In the phase 2 conclusion, I summarize the key findings and insights from the advanced techniques. We will reiterate the impact of these techniques on the improving the accuracy and robustness of Sentiment analysis.