Phase 4

Date	**28-10-2023**
Team ID	**1005**
Project Name	**Sentiment Analysis for Marketing**

Table of Contents

SENTIMENT ANALYSIS FOR MARKETING

1.PROBLEM STATEMENT:

Develop a sentiment analysis model to classify tweets related to airlines into positive, neutral, or negative sentiments. The goal is to provide airlines with a tool to analyse customer feedback on social media platforms, enabling them to better understand customer sentiment and improve their services.

2.DATA SPLITTING:

In this step, we split the dataset into training and testing sets to assess the model's performance. Specifically:

- X_train: The training data consisting of tweet text for the first 11,712 entries.
- Y_train: Corresponding sentiment labels for the training data.
- X_test: The testing data containing tweet text for the remaining entries.
- Y_test: Corresponding sentiment labels for the testing data.

3.TEXT VECTORIZATION:

To make the text data suitable for machine learning models, we use the TF-IDF vectorization technique. This process involves converting the tweet text into numerical vectors. Key information:

- TfidfVectorizer: We initialize the TF-IDF vectorizer.
- train_vectors: Vectorized representation of training data.
- test_vectors: Vectorized representation of testing data.

4.MODEL BUILDING AND EVALUATION:

This section covers the creation and evaluation of two different models: Support Vector Classifier (SVC) and Multinomial Naive Bayes (MultinomialNB).

4.1. Support Vector Classifier (SVC)

We employ the SVC model to classify the sentiment of the airline tweets. The following steps are taken:

- **Model Initialization**: We initialize the SVC model.
- **Model Training:** The model is trained using the training vectors and corresponding sentiment labels.
- **Predictions:** We make predictions on the test vectors.
- **Accuracy Score:** The accuracy score is calculated to assess the model's performance.

4.2. Naive Bayes (MultinomialNB)

In addition to SVC, we employ the MultinomialNB model to classify sentiments. The steps include:

- **Model Initialization:** Initialization of the MultinomialNB model.
- **Model Training:** Training the model with the training vectors and sentiment labels.
- **Predictions:** Generating predictions on the test vectors.
- **Accuracy Score:** Calculating the accuracy score for performance evaluation.

Importing libraries

```
import nltk

from textblob import TextBlob

import spacy

pip install nltk spacy textblob
```

loading the data:

```
from google.colab import files

uploaded=files.upload()
```

```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
```

```
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

 # Scatter and density plots

```
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df[columnNames]
```

```
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

Let's check 1st file: ../input/Tweets.csv

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# Tweets.csv has 14640 rows in reality, but we are only loading/previewing the first 1000 rows
df1 = pd.read_csv('../input/Tweets.csv', delimiter=',', nrows = nRowsRead)
df1.dataframeName = 'Tweets.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')
```

output:

There are 1000 rows and 15 column

let's take a quick look at what the data looks like:

```
df1.head(5)
```

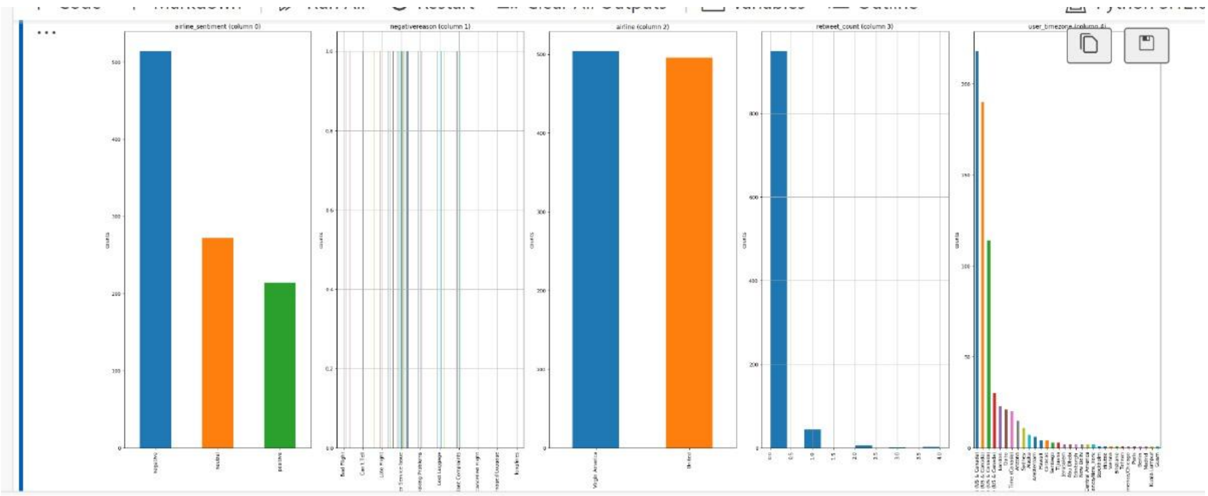
output:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retweet_count	text	tweet_coord	tweet_created	tweet_location	user_timezone
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	0	@VirginAmerica What @dhepburn said.	NaN	2015-02-24 11:35:52 -0800	NaN	Eastern Time (US & Canada)
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica plus you've added commercials t...	NaN	2015-02-24 11:15:59 -0800	NaN	Pacific Time (US & Canada)
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0	@VirginAmerica I didn't today... Must mean I n...	NaN	2015-02-24 11:15:48 -0800	Lets Play	Central Time (US & Canada)
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica it's really aggressive to blast...	NaN	2015-02-24 11:15:36 -0800	NaN	Pacific Time (US & Canada)
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica and it's a really big bad thing...	NaN	2015-02-24 11:14:45 -0800	NaN	Pacific Time (US & Canada)

#plotting

plotPerColumnDistribution(df1, 10, 5)

output:



5. RESULTS AND PERFORMANCE

The results of both models are presented in this section. Specifically:

- **Predicted Results:** The model's predictions for the test data.
- **Accuracy Scores:** Accuracy scores for both the SVC and MultinomialNB models.

6.CONCLUSION:

This documentation concludes the process of building and evaluating sentiment analysis models for airline tweets. The accuracy scores and model performance indicate the effectiveness of the models. The choice between the two models may depend on specific project requirements.