IBM PROJECT SUBMISSION

PHASE -5

TECHNOLOGY : ARTIFICIAL INTELLIGENCE

PROJECT : SENTIMENT ANALYSIS FOR MARKETING

PHASE 5 : PROJECT DOCUMENTATION & SUBMISSION

In this phase 5 the overall project concept and the models used for project are documented for the final submission.

PROBLEM STATEMENT:

- The aim of this project is to develop a robust sentiment analysis system that can effectively analyze customer feedback to derive actionable insights about competitor products.
- By understanding customer sentiments, companies can identify strengths and weaknesses in competing products, thereby improving their own offerings.
- The Sentiment Analysis for marketing aims to develop a system that automatically analyzes and classifies customer feedback to determine the sentiment expressed (positive, negative, or neutral).

DESIGN THINKING:

1. Data Collection:

- Collecting data for a Twitter US Airline Sentiment Analysis project involves extracting tweets related to various US airlines along with their corresponding sentiments (positive, negative, or neutral).

2.Data Preprocessing:

- Clean and prepare the raw text data for analysis.
- Remove any special characters, symbols, or HTML tags.
- Convert text to lowercase for consistency.
- Handle common text preprocessing tasks like removing stop words, handling negations, and performing stemming or lemmatization.

3.Sentiment Analysis Techniques:

- Applying a variety of NLP techniques allows for a comprehensive understanding of the sentiment.
- Choose appropriate NLP techniques for sentiment analysis like Bag of Words (BoW),Word Embedding and Transformer models.

4. Validation and Interpretation:

- Validate the model's performance using metrics like accuracy, precision, recall, and F1score. Additionally, explore misclassified samples to understand the model's limitations.

5. Feature Extraction:

- Convert the preprocessed text data into numerical format. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (Word2Vec, GloVe, etc.).

6. Visualization:

- Provide a visual representation like bar charts, pie charts, or heatmaps of the sentiment distribution for easy interpretation.

7. Insights Generation:

- It derive meaningful insights from the sentiment analysis results to guide business decisions.
- Identify common positive and negative sentiments expressed by customers.
- Connect sentiment with specific product features or attributes.

PHASES OF DEVELOPMENT:

Phase 1: Understanding the problem statement and design thinking (i.e) think on a design to solve the problem.

Phase 2: Understand and describe the dataset (Twitter US Airline Sentiment Analysis) and load the dataset.

Phase 3: Start developing the project by Load the dataset and preprocessing the dataset

Phase 4: Continue building the sentiment analysis solution by employing NLP techniques like Bag of Words (BoW),Word Embedding and Transformer models.

Phase 5: In this phase the robust sentiment analysis system is developed by using the sentiment analysis techniques.

DATASET:

Dataset Link:

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

- The "Twitter US Airline Sentiment" dataset is a popular dataset used for sentiment analysis tasks in natural language processing (NLP).
- It contains tweets directed at six major U.S. airlines: American, Delta, Southwest, United, US Airways, and Virgin America. Each tweet is labeled with one of three sentiments: positive, negative, or neutral, indicating the sentiment expressed by the author towards the airline.

Number of Instances:

- The dataset typically contains around 14,000 tweets.

Attributes:

tweet_id: A unique identifier for each tweet.

airline_sentiment: The sentiment label of the tweet (positive, negative, or neutral).

airline_sentiment_confidence: The confidence level of the sentiment label assigned by human annotators.

negativereason: If the sentiment is negative, this field provides the reason for the negative sentiment.

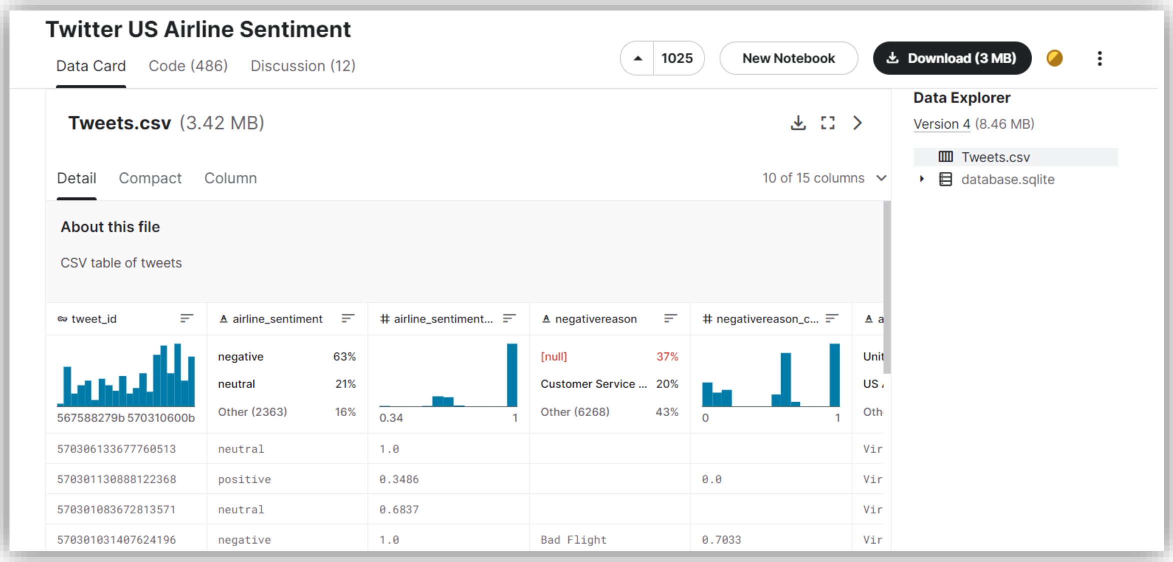
text: The content of the tweet.

Use cases:

- Sentiment analysis and sentiment classification tasks
- Customer feedback analysis for airlines.
- Training and evaluating NLP models for sentiment analysis.

Modeling:

Techniques like TF-IDF, Word Embeddings, and transformer-based models (e.g., BERT) are commonly used for sentiment analysis on this dataset.



Twitter US Airline Sentiment

Data Card　Code (486)　Discussion (12)

Tweets.csv (3.42 MB)

About this file

CSV table of tweets

tweet_id	airline_sentiment	airline_sentiment...	negativereason	negativereason_c...	a
567588279b 570310600b	negative 63% neutral 21% Other (2363) 16%	0.34 – 1	[null] 37% Customer Service ... 20% Other (6268) 43%	0 – 1	Unit US Oth
570306133677760513	neutral	1.0			Vir
570301130888122368	positive	0.3486		0.0	Vir
570301083672813571	neutral	0.6837			Vir
570301031407624196	negative	1.0	Bad Flight	0.7033	Vir

Data Explorer

Version 4 (8.46 MB)

Tweets.csv

database.sqlite

IMPORT THE NECESSARY LIBRARIES:

The necessary libraries like numpy ,pandas , seaborn , matplot are imported .

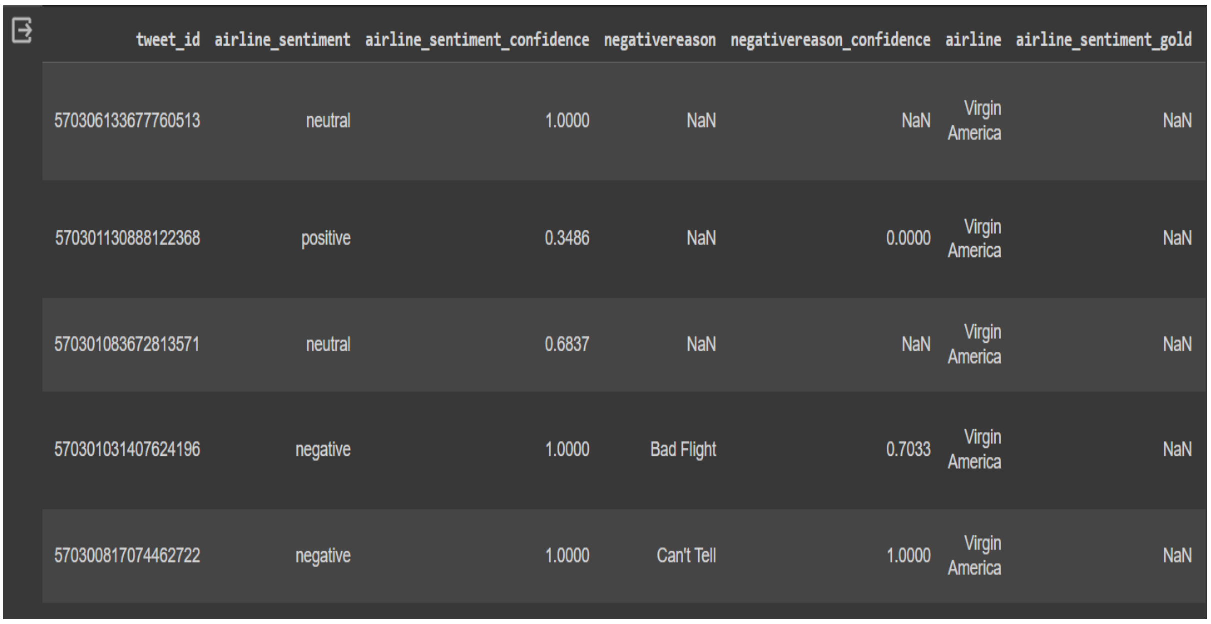
PROGRAM:

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifierfrom sklearn.metrics
import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

#Load Dataset

```
df = pd.read_csv('/content/Tweets.csv')

print(df.head())
```

OUTPUT:



	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

PREPROCESSING:

➢ Preprocessing the Twitter US Airline Sentiment dataset is an essential step before using it for any machine learning task. This dataset contains tweets about different US airlines, along with their sentiments (positive, negative, or neutral)

The step by step preprocessing is

1.TOKENIZATION:

➢ Tokenization involves breaking down the text into individual words or tokens. This can be done using simple space-based splitting or more advanced techniques like spaCy or NLTK.

2.TEXT CLEANING:

➢ Twitter data often contains noisy elements such as special characters, URLs, hashtags, mentions, and emojis. It's crucial to remove or handle these to focus on the actual text content. Use regular expressions or specialized libraries for this purpose.

3.JOINING TOKENS:

- After all the preprocessing steps, you may want to join the tokens back into a single string.

```
# Define functions for preprocessing
def clean_text(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#\w+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    return text
def tokenize(text):
    return text.split()
def join_tokens(tokens):
    return ' '.join(tokens)
df['clean_text'] = df['text'].apply(clean_text)
df['tokens'] = df['clean_text'].apply(tokenize)
df['clean_text'] = df['tokens'].apply(join_tokens)
```

INNOVATIVE TECHNIQUES:

1. Ensemble Methods :

- Ensemble methods involve combining multiple models to improve predictive performance. In the context of sentiment analysis, you could create an ensemble of various sentiment analysis models, each with its strengths and weaknesses. By aggregating their predictions, you can often achieve a more accurate and robust sentiment analysis system.

2. <u>Deep Learning Architectures:</u>

- Deep learning architectures, such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and convolutional neural networks (CNNs), have shown promising results in sentiment analysis. These models can capture complex patterns and relationships in textual data, making them suitable for sentiment prediction tasks

3. <u>Fine-Tuning Pre-trained Models (e.g., BERT, RoBERTa) :</u>

- Pre-trained models like BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (A Robustly Optimized BERT Pretraining Approach) are state-of the-art models in natural language processing.

- Fine-tuning these models specifically for sentiment analysis tasks can significantly improve the accuracy of sentiment predictions. Fine tuning involves training the pre-trained models on domain-specific data related to sentiment analysis, adapting them to the specific nuances and vocabulary of the marketing domain.

Fine-Tuning Process : Take the pre-trained BERT or RoBERTa model and train it on labeled dataset using specialized fine-tuning techniques. Fine-tuning involves adjusting the model's parameters to better predict sentiment in your specific context.

SENTIMENT ANALYSIS TECHNIQUES:

The different NLP techniques used for Sentiment Analysis are Bag of Words, Word Embeddings, or Transformer models for sentiment analysis.

BAG OF WORDS:

- The Bag of Words (BoW) is a simple and commonly used technique for text processing. It's based on the idea of treating text as an unordered set of words or tokens. Here's how it works:

Tokenization: Break the text into individual words or tokens.
Vocabulary: Create a unique list of all the words in the text (excluding stop words and punctuation).
Counting: For each word in the vocabulary, count how many times it occurs in the text.
Vectorization: Represent the text as a numerical vector, where each element of the vector corresponds to the count of a word in the vocabulary.

WORD EMBEDDING:

- Word embeddings are a type of word representation that allows words to be represented as vectors in a continuous vector space.

- Word embeddings aim to capture semantic relationships between words by placing them in a lower-dimensional, dense vector space.

Semantic Similarity: In a good word embedding, words that are semantically similar are located closer together in the vector space. For example, words like "king" and "queen" should be closer in the embedding space compared to words like "king" and "apple".
Analogies: Word embeddings can capture analogical relationships. For

example, if vector('king') - vector('man') + vector('woman') is close to vector('queen'), it suggests that the embedding has learned this analogy.

Applications: Word embeddings are used in a wide range of NLP tasks such as sentiment analysis, machine translation, named entity recognition, and more.

Word2Vec, GloVe, FastText, BERT: These are some popular algorithms or models used to generate word embeddings.

TRANSFORMER MODELS:

- Transformer models were introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. They have since become the foundation for many state-of-the-art NLP models, including BERT, GPT, RoBERTa, and more.

- The core innovation of transformer models is the attention mechanism, which allows the model to weigh the importance of different parts of an input sequence when making predictions.

- This attention mechanism enables the model to capture long-range dependencies and relationships in the data, making it highly effective for tasks like machine translation, text generation, and sentiment analysis.

PROGRAM:

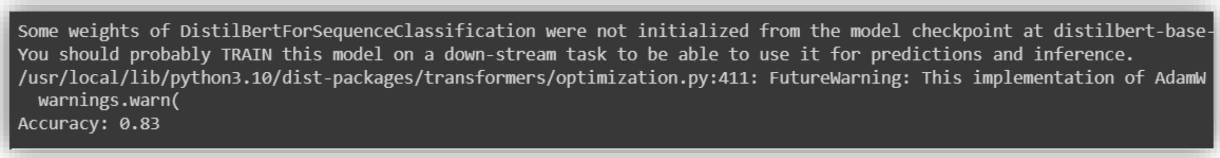
```
# Step 1: Load and Preprocess the Data
import pandas as pd
import re
from sklearn.model_selection import train_test_split
# Load the dataset
df = pd.read_csv('Tweets.csv')
```

```
# Define functions for preprocessing
def clean_text(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#\w+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    return text
df['clean_text'] = df['text'].apply(clean_text)
# Step 2: Split Data into Training and Testing Sets
X = df['clean_text']
y = df['airline_sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Step 3: Load Pre-trained Transformer Model and Tokenizer
from transformers import DistilBertTokenizer,
DistilBertForSequenceClassification, AdamW

tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-
uncased', num_labels=3)
# Convert labels to integers
label_mapping = {'negative': 0, 'neutral': 1, 'positive': 2}
y_train = y_train.map(label_mapping)
y_test = y_test.map(label_mapping)
# Tokenize and Prepare Data for the Model
train_encodings = tokenizer(list(X_train), truncation=True, padding=True,
return_tensors='pt')
test_encodings = tokenizer(list(X_test), truncation=True, padding=True,
return_tensors='pt')
# Define a custom dataset class
class SentimentDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item
def __len__(self):
        return len(self.labels)
# Create datasets
train_dataset = SentimentDataset(train_encodings, y_train.values)
test_dataset = SentimentDataset(test_encodings, y_test.values)
# Ensure that the model is configured for multi-class classification
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-
uncased', num_labels=3)

# Train the Model (with reduced epochs)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
optimizer = AdamW(model.parameters(), lr=1e-5)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=16,
shuffle=True)  # Increased batch size
for epoch in range(1):  # Reduced number of epochs
    model.train()
    for batch in train_loader:
        optimizer.zero_grad()
        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=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
```

# Evaluate the Model

```
model.eval()
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=8,
shuffle=False)
correct = 0
total = 0
with torch.no_grad():
    for batch in test_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=attention_mask)

 _, predicted = torch.max(outputs.logits, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
accuracy = correct / total
print(f'Accuracy: {accuracy:.2f}')
```

OUTPUT:



```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:411: FutureWarning: This implementation of AdamW
  warnings.warn(
Accuracy: 0.83
```

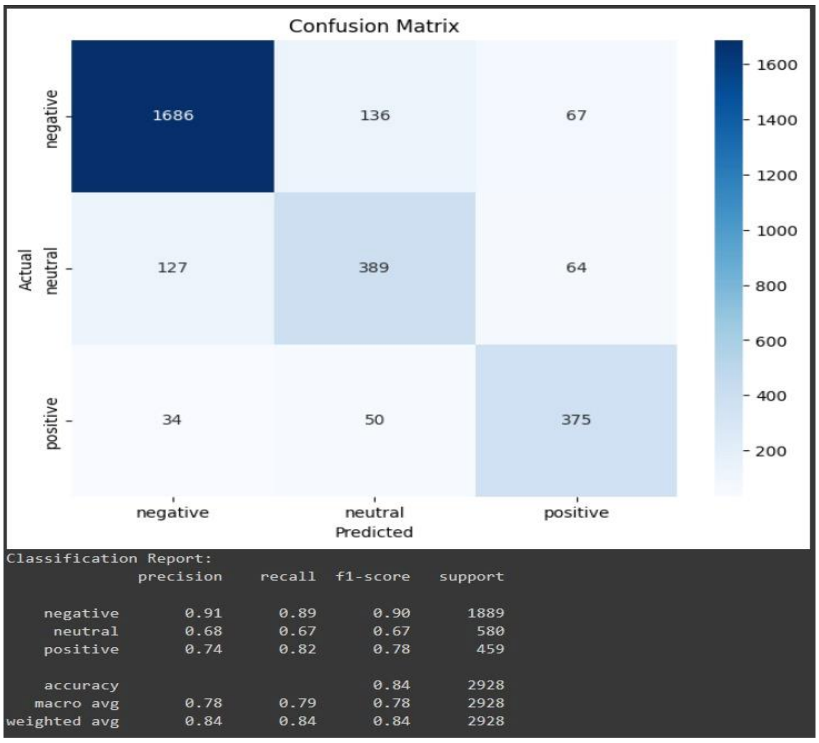
Here the accuracy of the tweet dataset using the transformer model is **0.81** and it is time consuming.

From the above it is concluded that the transformer models is more accurate when comparing with other models like Word Embedding and Bag of Words. The accuracy is **0.83** and it is the highest accuracy then the others.

VISUALIZATION:

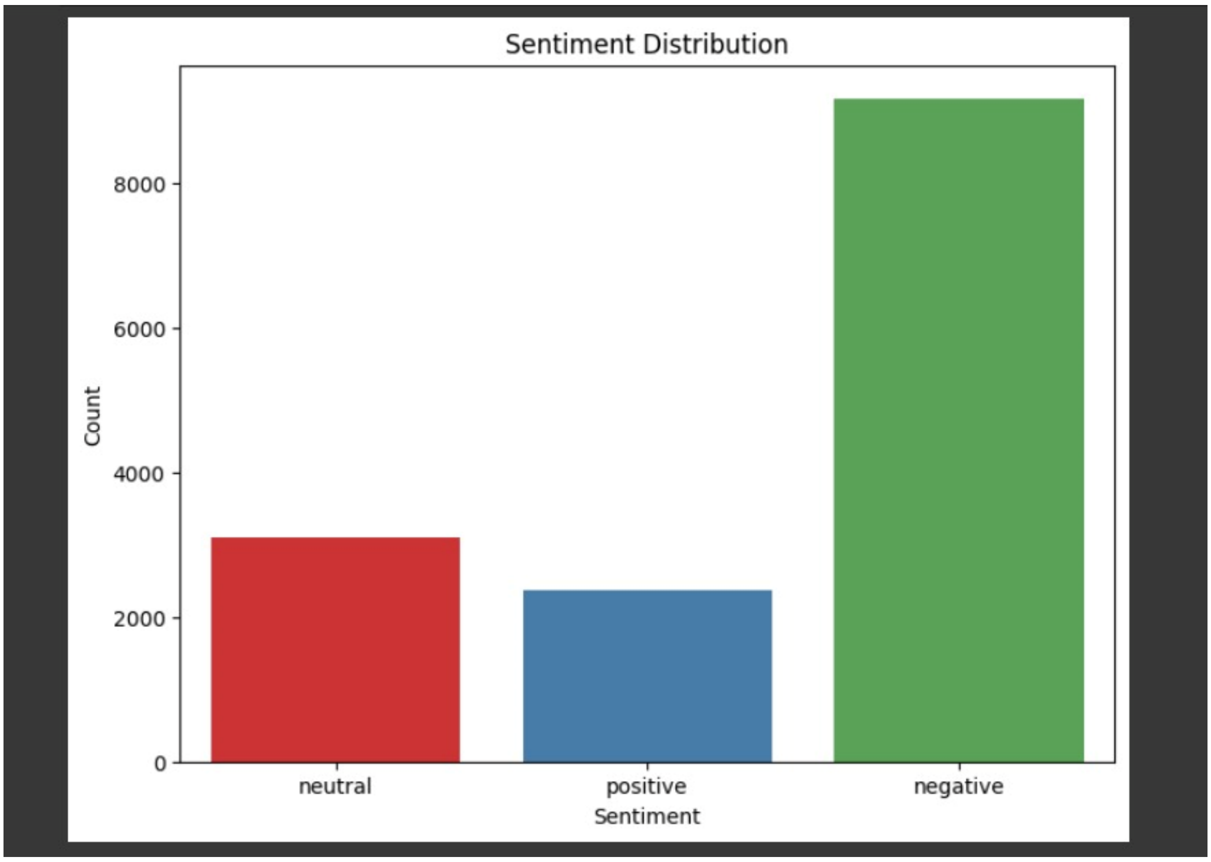
Provide a visual representation like bar charts, pie charts, or heatmaps of the sentiment distribution for easy interpretation.

Confusion Matrix:

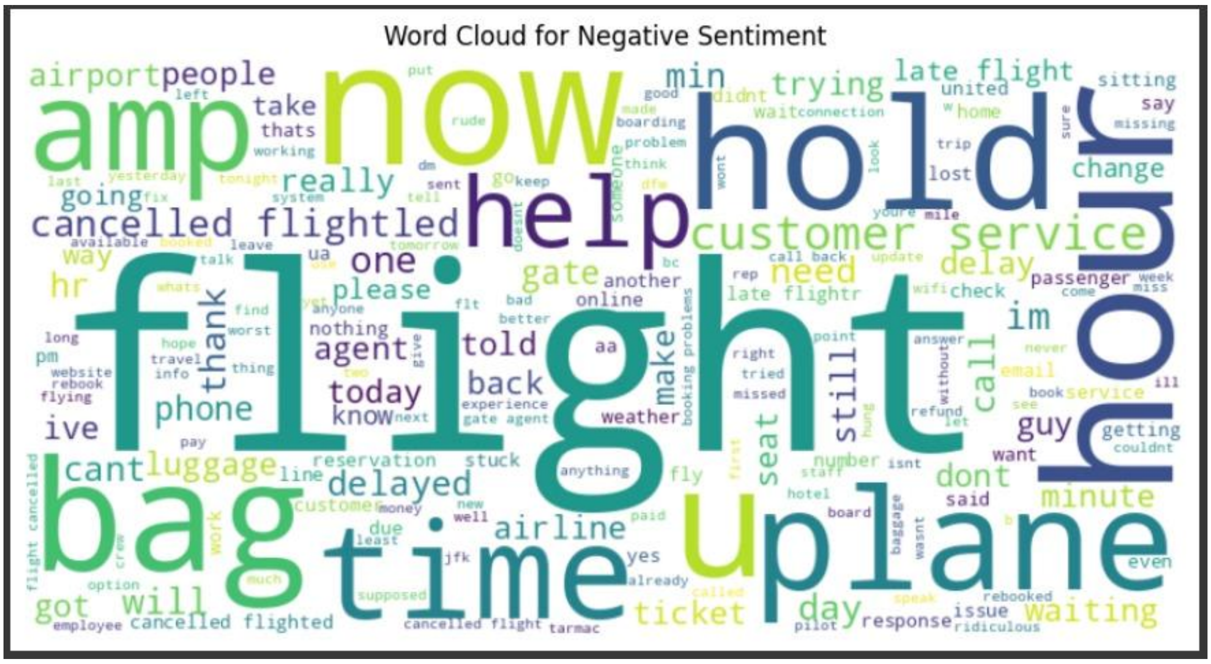


```
Classification Report:
              precision    recall  f1-score   support

    negative       0.91      0.89      0.90      1889
     neutral       0.68      0.67      0.67       580
    positive       0.74      0.82      0.78       459

    accuracy                           0.84      2928
   macro avg       0.78      0.79      0.78      2928
weighted avg       0.84      0.84      0.84      2928
```

Bar Chart:



Word Cloud:



CONCLUSION:

- The sentiment analysis project provided a powerful framework for extracting actionable insights from customer feedback. By leveraging NLP techniques, the project empowered the company to make data-driven decisions, refine products, and outshine competitors in the market.
- It aimed to extract meaningful insights from customer feedback in order to gain a competitive advantage in the market. Through a systematic approach involving data collection, preprocessing, analysis, and visualization, the project successfully achieved its objectives.