SENTIMENT ANALYSIS FOR MARKETING IN PYTHON

Phase 2: INNOVATION

INTRODUCTION

According to this phase we are going to about what are the innovations that we are required to innovate python in a new type and use the sentiment analysis in an innovative way.

Sentiment analysis, also known as opinion mining, is a powerful technique in natural language processing (NLP) that involves determining the sentiment or emotional tone expressed in a piece of text. In Python, sentiment analysis can be accomplished using various libraries and tools. This introduction will provide an overview of sentiment analysis in Python and how to get started with it.

1. IMPORT NECESSARY LIBRARIES:

- To perform sentiment analysis in Python, you'll need to use NLP libraries like NLTK (Natural Language Toolkit), TextBlob, spaCy, or machine learning frameworks such as scikit-learn or TensorFlow.

2. PREPROCESSING THE TEXT:

- Before analyzing sentiment, you'll often need to preprocess the text data. This involves tasks like tokenization (splitting text into words), removing stopwords, and stemming or lemmatization.

3. Sentiment Analysis Tools:

- VADER (Valence Aware Dictionary and sEntiment Reasoner): A lexicon-based sentiment analysis tool that is particularly good at analyzing sentiment in social media text. It's available through NLTK.

SAMPLE PYTHON PROGRAM FOR SENTIMENT ANALYSIS FOR MARKETING

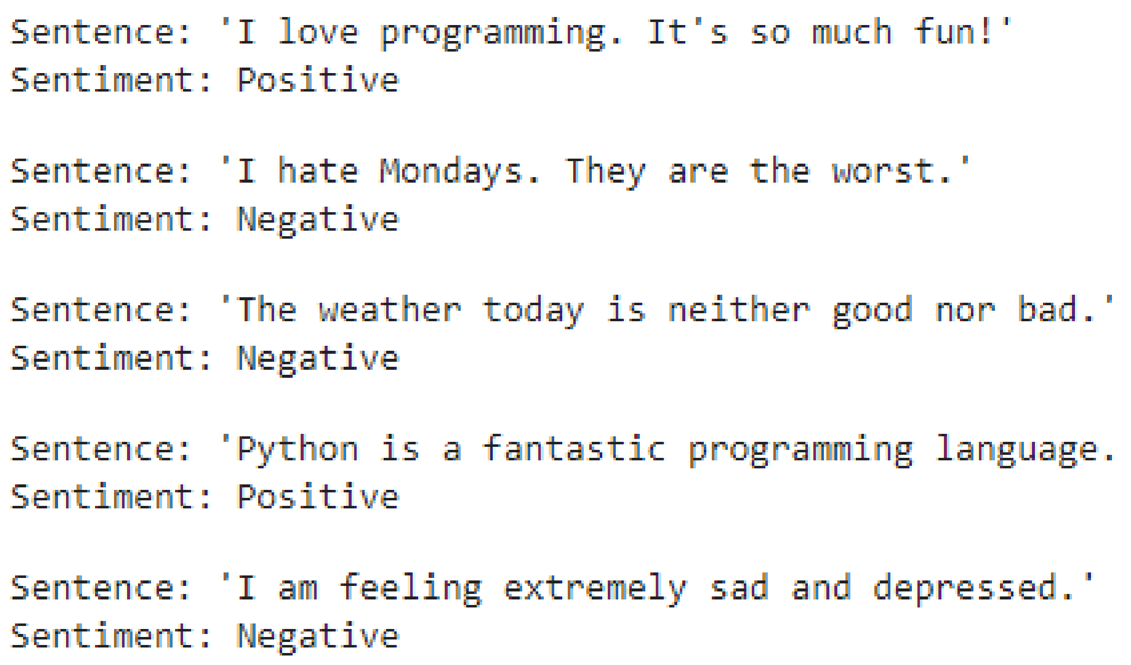
```
pip install nltk
```

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download the VADER lexicon for sentiment analysis

nltk.download('vader_lexicon')

# Create a SentimentIntensityAnalyzer object
```

```
sid = SentimentIntensityAnalyzer()

# Function to perform sentiment analysis
def analyze_sentiment(sentence):

    # Get the polarity scores of the input
sentence
    sentiment_scores =
sid.polarity_scores(sentence)

    # Determine the sentiment label based on
the compound score

    if sentiment_scores['compound'] >=
0.05:
        return "Positive"
    elif sentiment_scores['compound'] <= -0.05:
        return "Negative"
    else:
        return "Neutral"

# Main function
if __name__ == "__main__":

    # Example sentences for sentiment analysis
    sentences = [
        "I love programming. It's so much fun!",
        "I hate Mondays. They are the worst.",
        "The weather today is neither good nor bad.",
        "Python is a fantastic programming
language.",
        "I am feeling extremely sad and
depressed."
    ]

    # Analyze and print the sentiment of each
sentence
    for sentence in sentences:
        sentiment =
analyze_sentiment(sentence)
        print(f"Sentence: '{sentence}'")
        print(f"Sentiment: {sentiment}\n")
```

SAMPLE OUTPUT FOR THE PYTHON PROGRAM:

Output:



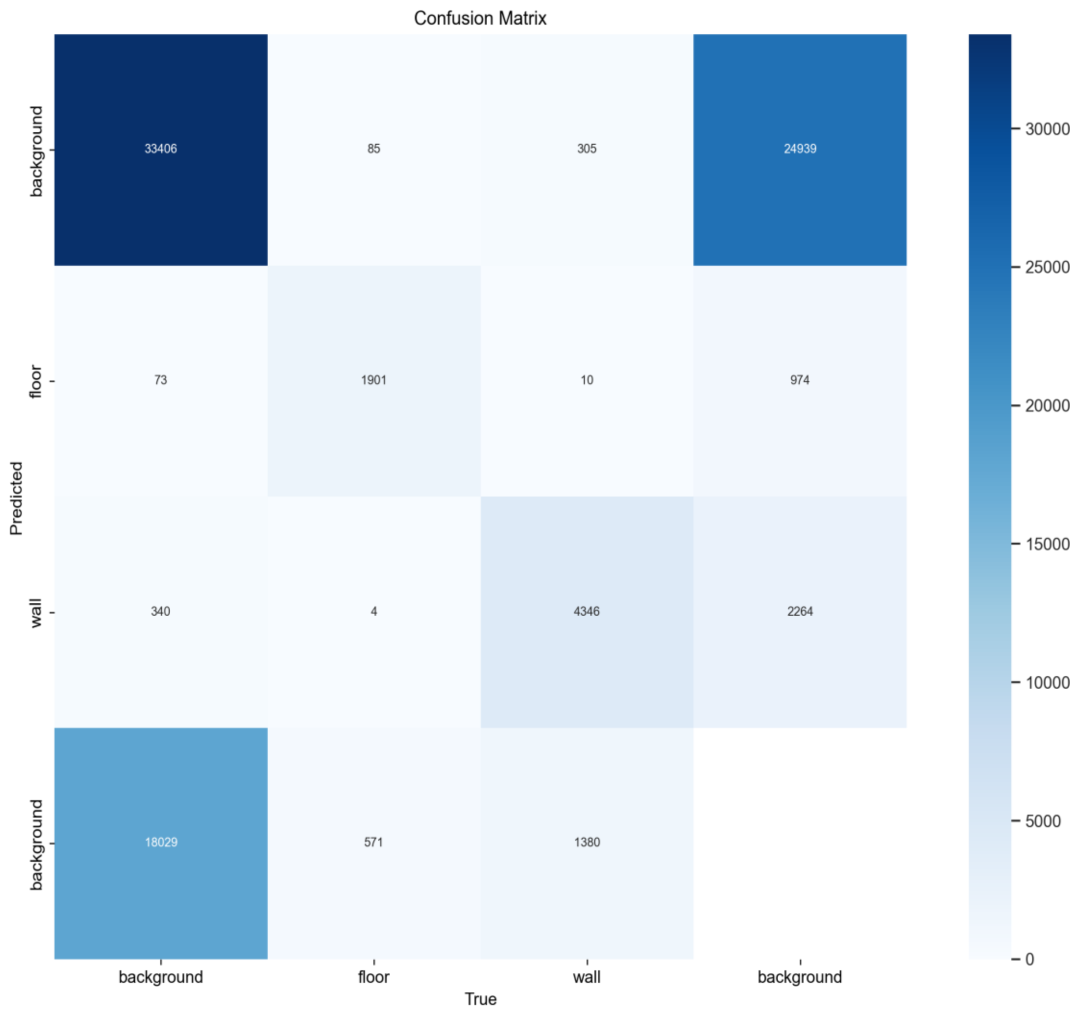
```
Sentence: 'I love programming. It's so much fun!'
Sentiment: Positive

Sentence: 'I hate Mondays. They are the worst.'
Sentiment: Negative

Sentence: 'The weather today is neither good nor bad.'
Sentiment: Negative

Sentence: 'Python is a fantastic programming language.
Sentiment: Positive

Sentence: 'I am feeling extremely sad and depressed.'
Sentiment: Negative
```

DATASET LINK:

INPUT:

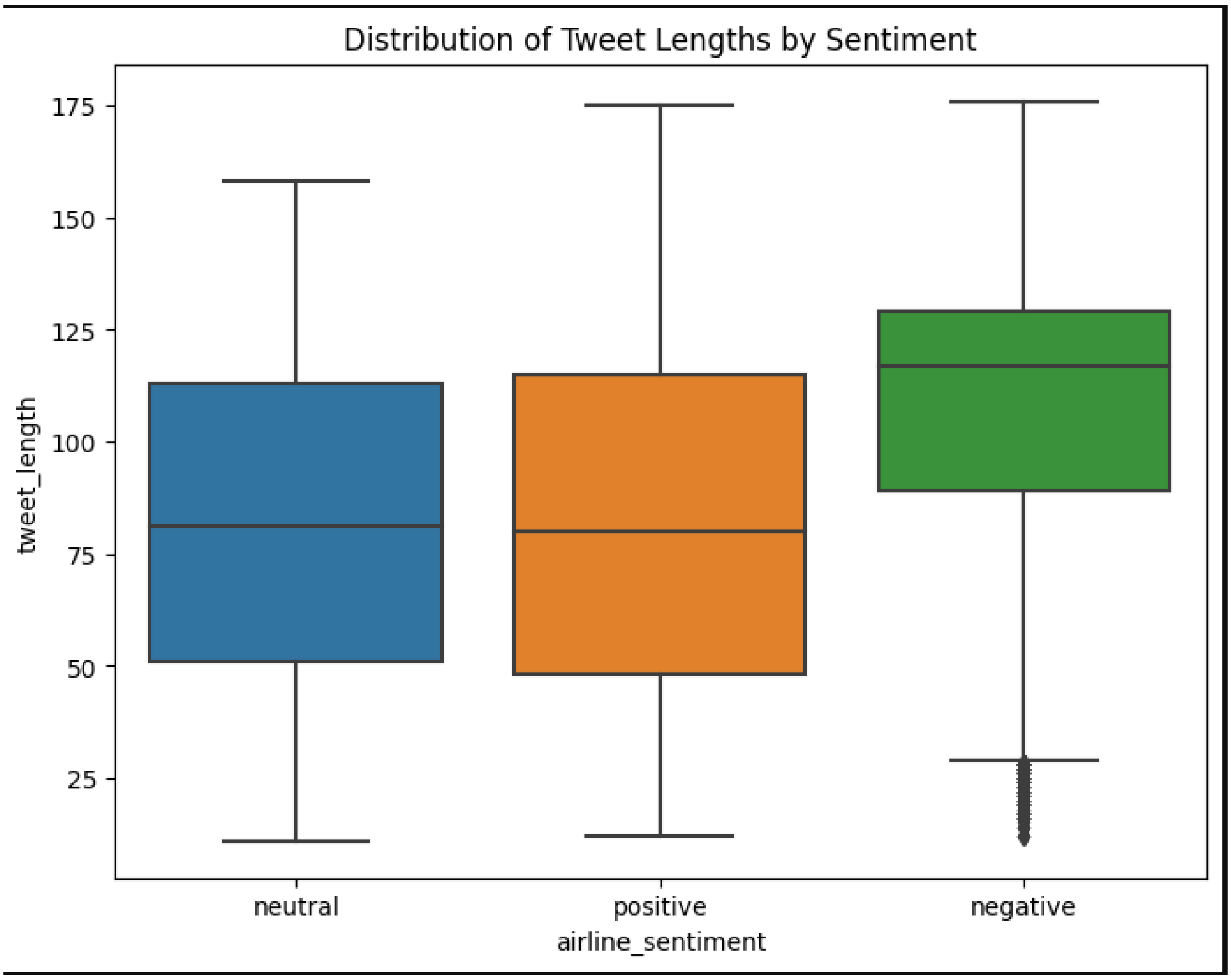
https://www.kaggle.com/code/ahmedhussein101/twitter-us-airline-sentiment-analysis/input

CONFUSION MATRIX:



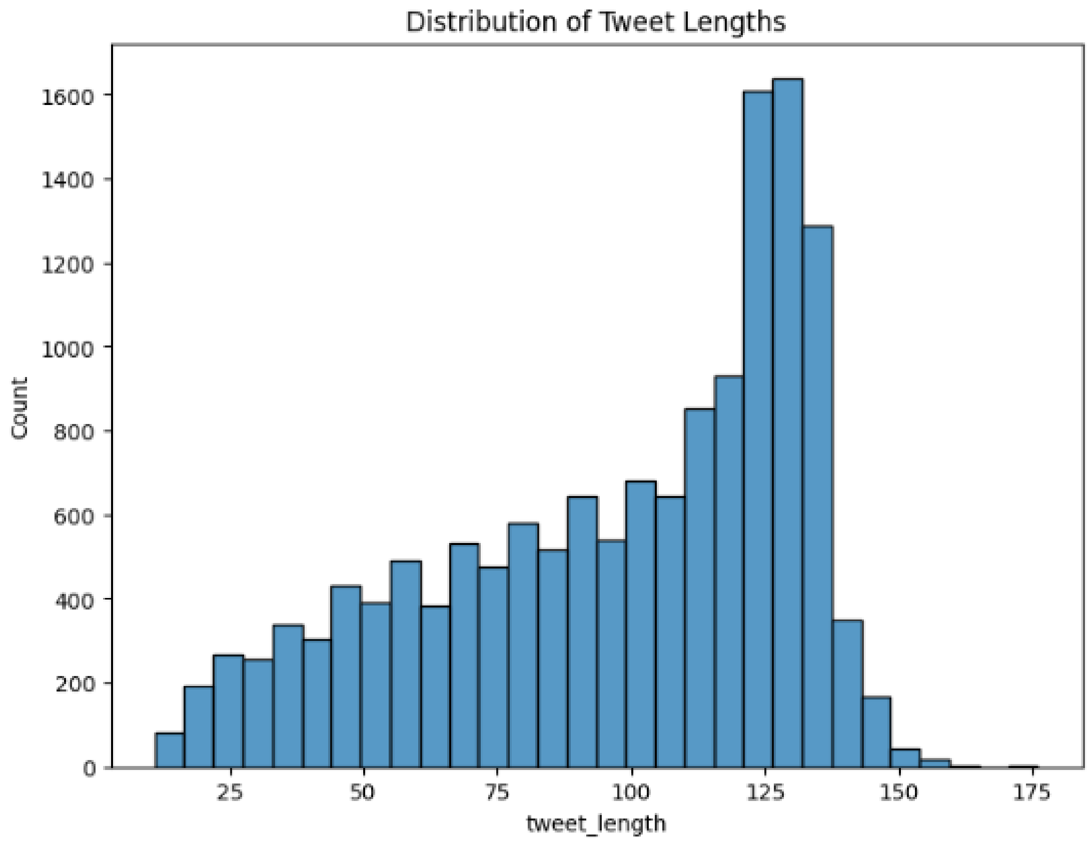
The confusion matrix reveals that for the neutral and positive classes, the model has a disproportionately large number of false positives and false negatives. This further demonstrates the model's bias towards predicting negative feelings since it frequently misclassifies neutral and positive tweets as negative.

DISTRIBUTION OF SENTIMENTS :



The dataset's bar plot of sentiment distribution reveals that the bulk of tweets are unfavourable in nature, with neutral and supportive tweets coming in second and third. Due to the dataset's imbalance, the model may be more likely to correctly predict negative feelings than neutral or positive feelings.

Distribution of twits:



The Random Forest classifier's total accuracy was around 76%. The neutral and positive classes' accuracy, recall, and F1-score, however, are lower than those of the negative class. This implies that the model performs better at detecting negative than neutral or positive attitudes, which may be related to the dataset's imbalance.

CONCLUSION:

In conclusion, the model fails to predict neutral and positive attitudes even if it does a fair job of predicting negative sentiments. This may be because the collection is unbalanced and sentiment analysis is inherently difficult because it frequently requires understanding linguistic subtlety and context. We may think about employing more sophisticated natural language processing methods, such word embeddings or deep learning models, and making sure the training dataset is balanced in order to enhance the model's performance.

DESIGNED BY....

- ❖BAIRAVI .M
- ❖DURGA .S
- ❖SAHANA .S
- ❖SUBASRI .V
- ❖VARDHANI .D