About the Project – Twitter Sentiment Analysis

This project focused on analyzing sentiments expressed in user comments from Twitter related to various game titles. The objective was to classify comments into positive or negative sentiments using machine learning. The process began with data cleaning, which involved removing null values, links, mentions, hashtags, and stopwords to ensure high-quality input. Only positive and negative sentiment labels were retained for focused analysis. Text data was transformed into numerical form using a TF-IDF Vectorizer, selecting the top 5000 words to represent features. An 80/20 train-test split was applied, and a Logistic Regression model (with a 2000 iteration cap) was trained on this data. The model achieved an impressive 87% accuracy with balanced precision, recall, and F1-scores for both sentiment classes. Visualizations like bar plots, pie charts, and confusion matrices supported insights. The project demonstrates effective use of NLP techniques for sentiment classification and paves the way for real-time opinion monitoring.

- MD AAQIF HUSSAIN

Project Overview and challenges

- This Twitter Sentiment Analysis project aims to identify whether users express positive or negative opinions about video games on social media. Using a dataset of user comments and associated sentiments, we applied natural language processing (NLP) techniques to clean, transform, and analyze the textual data effectively.

- The model was trained using a TF-IDF vectorizer and Logistic Regression, achieving an accuracy of 86.71%. Visualization tools like bar graphs, pie charts, and confusion matrices were used to interpret results. This project demonstrates how machine learning can be applied to social media data for real-time sentiment monitoring and brand feedback analysis.

- During the project, I faced challenges such as identifying the correct text and sentiment columns, cleaning and preprocessing inconsistent data, and handling model warnings like convergence issues in Logistic Regression. Visualizing the confusion matrix also required fixing label mismatches. These obstacles improved my understanding of data handling, model tuning, and visualization techniques in machine learning.

Tools and Technologies Used

- In this project, several powerful tools and libraries were used to perform sentiment analysis on Twitter data. The entire project was developed and executed in **Google Colab**, a cloud-based Python environment that provides a convenient platform for writing, testing, and sharing code. **Python** served as the core programming language due to its versatility and rich ecosystem of data science libraries. For data manipulation and preprocessing, **Pandas** and **NumPy** were employed to efficiently handle tabular data and numerical operations.

- To process and analyze textual data, **NLTK (Natural Language Toolkit)** was used, especially for removing stopwords and cleaning the user comments. **Scikit-learn** played a central role in machine learning tasks, including data splitting, feature extraction using **TF-IDF Vectorizer**, training the **Logistic Regression** model, and evaluating performance metrics like accuracy, precision, and recall. For data visualization, **Matplotlib** and **Seaborn** were used to create plots such as sentiment distributions and confusion matrices. Finally, the **Pickle** library was used to save the trained model and vectorizer for future use, allowing the sentiment analysis system to be easily reused or deployed.

Importing important libraries

```
[3]  # Importing the necessary libraries
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     import nltk
     import re
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import classification_report, confusion_matrix
     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
     from nltk.corpus import stopwords
     import pickle
```

```
[4]  # Download NLTK stopwords
     nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Data Preparation and Cleaning

```
# uploading file manually from file
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving twitter_training.csv to twitter_training.csv
```

```
[7] #reading the csv file and adding in dataframe
    df = pd.read_csv('twitter_training.csv')
```

```
[8] #checking head of dataframe
    df.head()
```

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Data Preparation and Cleaning

Sentiment Filtering

Kept only Positive and Negative labels

```
[10] #considering two column for the further analysis
     df = df[['User_Comment', 'Sentiment']]

     df['Sentiment'] = df['Sentiment'].astype(str).str.strip().str.title()

     df = df[df['Sentiment'].isin(['Positive', 'Negative'])]

     df = df.reset_index(drop=True)
```

```
df.head(5)
```

	User_Comment	Sentiment	cleaned_comment
0	I am coming to the borders and I will kill you...	Positive	coming borders kill
1	im getting on borderlands and i will kill you ...	Positive	im getting borderlands kill
2	im coming on borderlands and i will murder you...	Positive	im coming borderlands murder
3	im getting on borderlands 2 and i will murder ...	Positive	im getting borderlands 2 murder
4	im getting into borderlands and i can murder y...	Positive	im getting borderlands murder

Null Values

Removed rows with missing User_Comment

User_Comment

```
#checking null value from the dataset
df.isnull().sum()
```

	0
User_Comment	361
Sentiment	0

dtype: int64

```
[13] #removing null value from the dataset
     df = df.dropna()
```

```
[14] # Rechecking null value
     df.isnull().sum()
```

	0
User_Comment	0
Sentiment	0

dtype: int64

Data Preparation and Cleaning

Cleaning

```python
#cleaning the dataset

stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = re.sub(r"http\\S+|www\\S+|https\\S+", '', text)    # Remove links
    text = re.sub(r'@\\w+', '', text)                         # Remove @mentions
    text = re.sub(r'#', '', text)                             # Remove # symbol
    text = re.sub(r'[^A-Za-z0-9 ]+', '', text)                # Remove special characters
    text = text.lower()                                       # Lowercase everything
    text = " ".join(word for word in text.split() if word not in stop_words)  # Remove stopwords
    return text

# Apply cleaning
df['cleaned_comment'] = df['User_Comment'].astype(str).apply(clean_text)

# Show cleaned data
df.head()
```

Cleaning Steps

Remove links, mentions, hashtags,

hashtags, special chars

Stopwords Removal

Filtered common English stopwords using NLTK

	User_Comment	Sentiment	cleaned_comment
0	I am coming to the borders and I will kill you...	Positive	coming borders kill
1	im getting on borderlands and i will kill you ...	Positive	im getting borderlands kill
2	im coming on borderlands and i will murder you...	Positive	im coming borderlands murder
3	im getting on borderlands 2 and i will murder ...	Positive	im getting borderlands 2 murder
4	im getting into borderlands and i can murder y...	Positive	im getting borderlands murder

Training and Testing Data

Train-Test Split

80% training, 20% testing data

```
#Spliting the Data for Train/Test
X = df['cleaned_comment']  # Features (inputs)
y = df['Sentiment']        # Labels (outputs)

# Split: 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

TF-IDF Vectorizer

Converts text to numerical features

Max Features

Limited to 5000 most important words

Training Data

Model trained on TF-IDF features

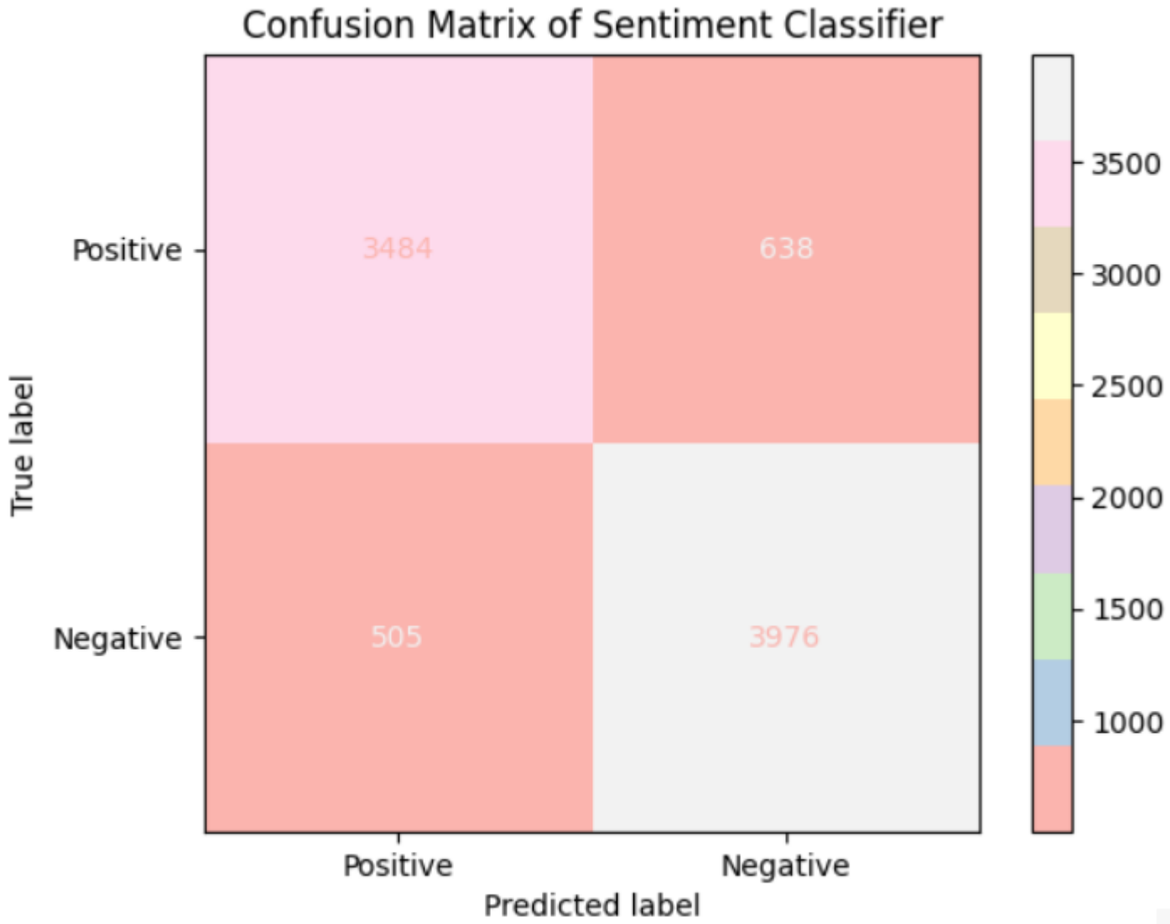
```
[17] #Converting Text to Numbers (TF-IDF Vectorizer)
     tfidf = TfidfVectorizer(max_features=5000)  # Limit to 5000 words

     X_train_tfidf = tfidf.fit_transform(X_train)
     X_test_tfidf = tfidf.transform(X_test)
```

Algorithm

Logistic Regression with 2000 max iterations

```
[18] #Training the Model (Logistic Regression)
     model = LogisticRegression(max_iter=2000)   # Increased iterations
     model.fit(X_train_tfidf, y_train)
```

```
LogisticRegression(max_iter=2000)
```



Confusion Matrix Analysis

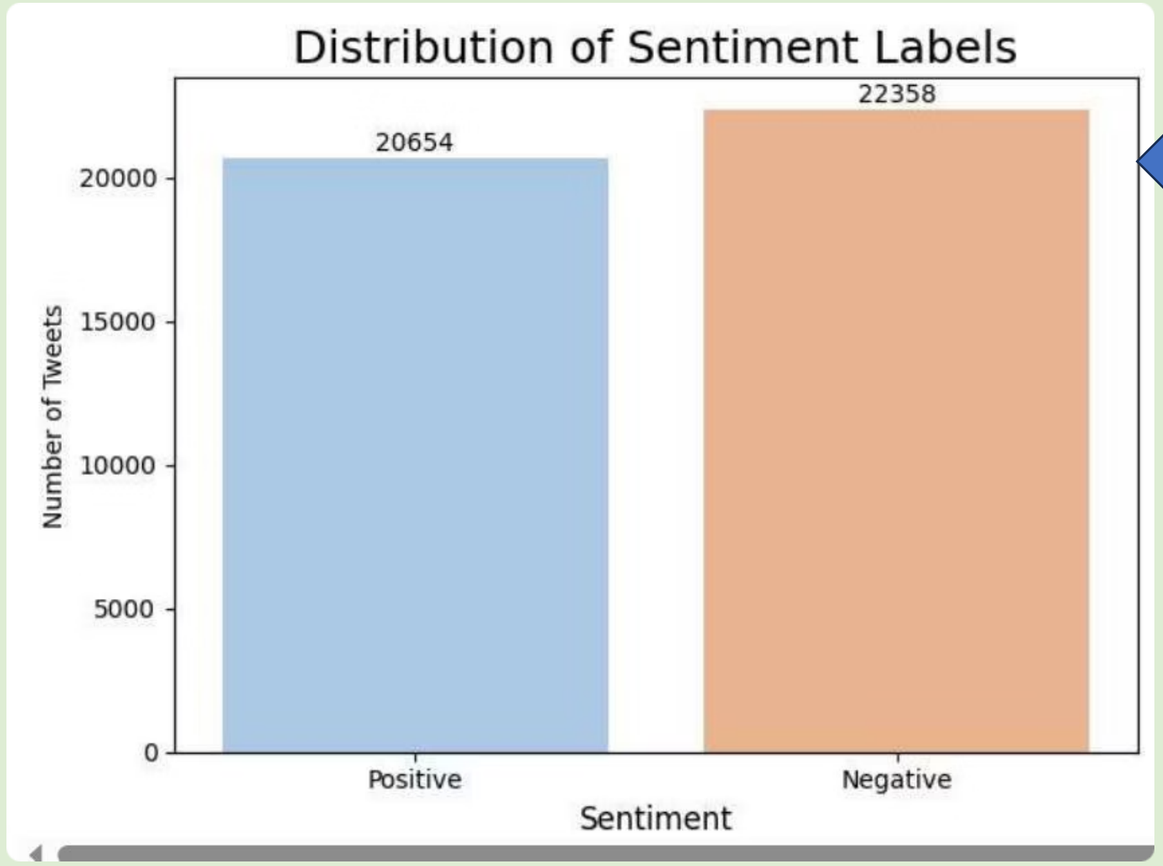
Shows true vs predicted sentiment counts for Positive and and Negative classes.

0s

```
# Predict on Test Data
y_pred = model.predict(X_test_tfidf)

# Confusion Matrix
labels = ['Positive', 'Negative']  # You can adjust based on your dataset
cm = confusion_matrix(y_test, y_pred, labels=labels)

# Plot
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot(cmap='Pastel1')
plt.title('Confusion Matrix of Sentiment Classifier')
plt.show()
```

Sentiment Distribution

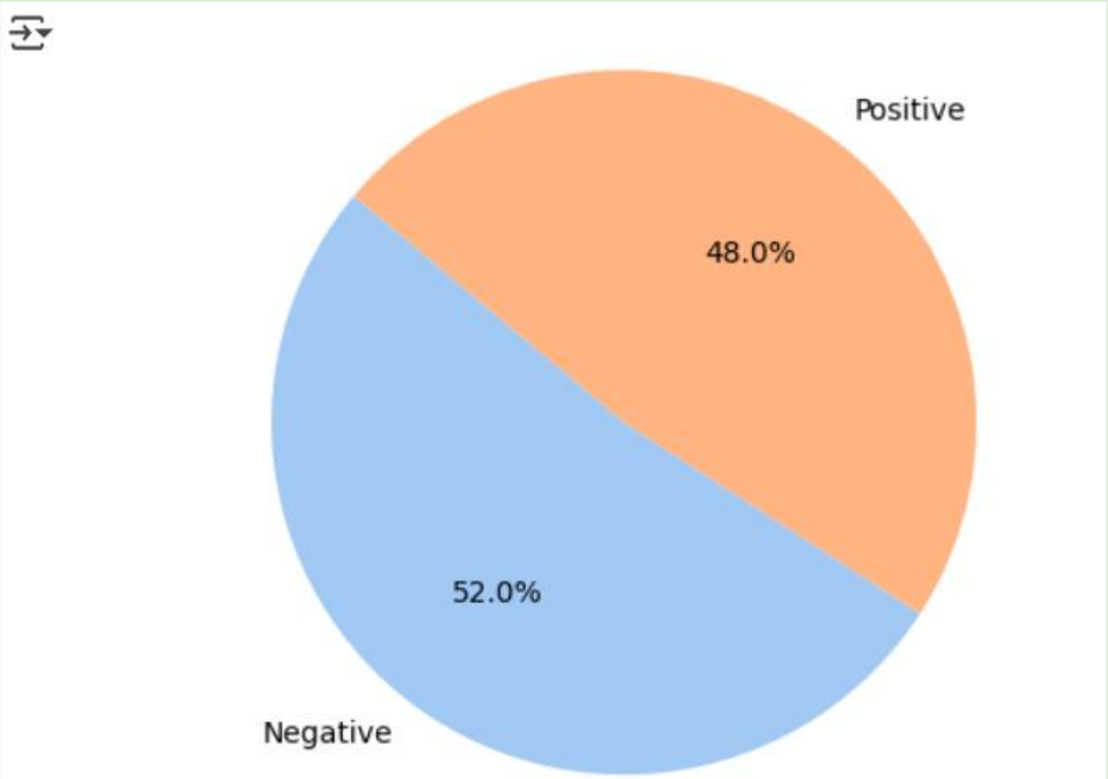
Count plot showing number of Positive and Negative tweets in tweets in dataset.




```
[26] ##Visualizing the analysis
     # 1 Create the plot
     ax = sns.countplot(x='Sentiment', hue='Sentiment', data=df, palette='pastel', legend=False)

     # Add value labels on top of bars
     for container in ax.containers:
         ax.bar_label(container, fmt='%d', label_type='edge', padding=1)

     # Customize the plot
     plt.title('Distribution of Sentiment Labels', fontsize=18)
     plt.xlabel('Sentiment', fontsize=12)
     plt.ylabel('Number of Tweets', fontsize=10)
     plt.xticks(rotation=0)  # Rotate x-labels if needed
     plt.tight_layout()  # Prevent label cutoff
     plt.show()
```

Sentiment Proportions

Pie chart visualizing percentage share of Positive vs Negative tweets.




```
# 2. Pie Chart
sentiment_counts = df['Sentiment'].value_counts()
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette('pastel'))
plt.axis('equal')
plt.show()
```

Evaluating the Model

F1-Score

Balanced performance across sentiments

sentiments

Accuracy

87% overall accuracy

```
[29] from sklearn.metrics import accuracy_score
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 86.71%
```

Precision & Recall

Both classes around 86-87%

```
##Evaluating the Model
# Classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
              precision    recall  f1-score   support

    Negative       0.86      0.89      0.87      4481
    Positive       0.87      0.85      0.86      4122

    accuracy                           0.87      8603
   macro avg       0.87      0.87      0.87      8603
weighted avg       0.87      0.87      0.87      8603

[[3976  505]
 [ 638 3484]]
```

Project summary

The Twitter Sentiment Analysis project aimed to classify user comments on video games as either Positive or Negative. The dataset was cleaned by removing neutral and irrelevant sentiments, eliminating stopwords, and standardizing text formats. TF-IDF vectorization was used to convert text into numerical features suitable for model training. A Logistic Regression model was applied, and after tuning parameters, it achieved an accuracy of 86.71%. Various visualizations, including sentiment distribution graphs and a confusion matrix, were generated to assess model performance. The project demonstrated the practical use of natural language processing and machine learning techniques for analyzing real-time social media feedback.

Thank You.

- MD AAQIF HUSSAIN