CLOUD APPLICATION DEVELOPMENT
GROUP 3

PROJECT 1: Big Data Analysis with IBM Cloud Databases

INTRODUCTION:

This project exemplifies the convergence of cloud technology and big data analysis, offering a versatile and efficient solution for businesses in the digital age. By automating sentiment evaluation and employing the vast computational resources of the cloud, it enables organizations to gain a deeper understanding of their customers' sentiments and opinions at scale. This valuable insight not only informs strategic decisions but also fosters improved customer engagement and satisfaction. As businesses increasingly rely on data-driven insights, the integration of cloud infrastructure enhances the project's capabilities by ensuring real-time analysis, accessibility, and adaptability to evolving data requirements, ultimately contributing to the success of cloud-based applications in today's dynamic business landscape. With the explosive growth of Twitter and other social platforms, the volume of user-generated content has reached unprecedented levels, making it a goldmine of information for businesses, researchers, and decision-makers. The primary aim of this project is to harness the power of big data technologies and natural language processing to automatically categorize tweets as positive, negative, or neutral, providing a comprehensive view of public opinion on various topics, products, or events.

DATASET:

The dataset contains enormous number of tweets such as feedbacks, product reviews and many users created contents. It is classified with six attributes and contains more than one million tweets. This big data dataset was in a format of ***.csv*** data format. This dataset comprises of positive, negative or neutral specific emotions. The attributes contain metadata such as, date of the text, user name and other information. The main objective of this dataset is to bring out valuable business insights by performing sentimental analysis to this dataset.

DATA PREPARATION AND EXPLORATION:

1.Loading Dataset:

- It uses the ***pd.read_csv()*** function from the Pandas library to read a CSV file. The file is located at drive.
- The ***df.head()*** function is used to display the first few rows of the dataset to get an initial look at the data.

Loading Dataset

```
df = pd.read_csv('/content/drive/MyDrive/IBM/Dataset/training.1600000.processed.noemoticon.csv',delimiter=',', encoding='ISO-8859-1')
df.head()
```

	polarity of tweet	id of the tweet	date of the tweet	query	user	text of the tweet
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

2.DATA SHAPE:

- The code uses ***df.shape*** to print the dimensions of the dataset, which includes the number of rows and columns.

```
[1] df.shape

    (1048572, 6)
```

3. Data Cleaning :

- ***df.info()*** provides information about the dataset, including the number of non-null entries and data types of each column.
- ***df.isnull().sum()*** is used to check for null values in the dataset. The result shows that there are no null values.

4. Column Dropping:

- The code removes unnecessary columns from the dataset using ***df.drop()***. Columns such as 'id of the ***tweet***, ***date of the tweet***, ***query***, and ***user*** are dropped, leaving only ***sentiment*** and ***text*** columns

5. Column Renaming:

- The code renames the remaining columns to ***sentiment*** and ***text*** for simplicity and clarity.

```
[ ]  #Simplifying the names of columns
     df.columns = ['sentiment', 'text']
```

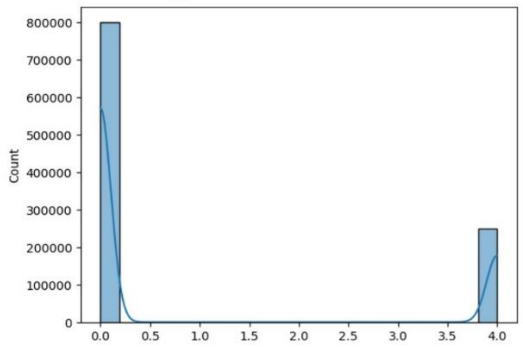
6. Sentiment Value Counts:

- The code uses ***df['sentiment'].value_counts()*** to display the counts of different sentiment labels in the sentiment column. This provides an initial understanding of the sentiment distribution in the dataset.

- These steps are essential for setting up the dataset and understanding its basic characteristics before proceeding with sentiment analysis.

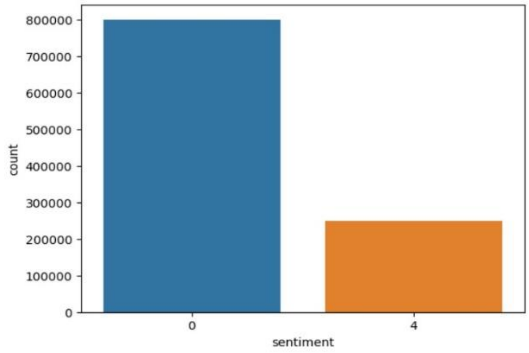
```
#visuals
sns.histplot(df['sentiment'], kde=True)
```

```
<Axes: xlabel='sentiment', ylabel='Count'>
```




```
sns.countplot(x=df['sentiment'])
```

```
<Axes: xlabel='sentiment', ylabel='count'>
```



TEXT PREPROCESSING:

We need to Import Libraries and Resources such as the Natural Language Toolkit (NLTK) library and downloads the list of English stopwords. **NLTK** is a popular library for natural language processing tasks. we need to define ***stuff_to_be_removed***, which is a list of English stopwords and punctuation symbols. These will be removed from the text data during preprocessing. Tokenization and Stemming initializes a Lancaster Stemmer, which is a stemmer used to reduce words to their root form. The text data in the DataFrame ***df['text']*** is collected in the corpus list. then for text cleaning we need to perform a series of text cleaning and preprocessing steps on the text data for each entry in the corpus Non-alphabetic characters are replaced with spaces using ***re.sub('[^a-zA-Z]', ' ', df['text'][i])***. Then the text is converted to lowercase. A new DataFrame named data cleaned is created.

```
[ ] import nltk
    nltk.download('stopwords')
    stuff_to_be_removed = list(stopwords.words('english'))+list(punctuation)
    stemmer = LancasterStemmer()
    corpus = df['text'].tolist()
    print(len(corpus))
    print(corpus[0])
```

```
1048572
is upset that he can't update his Facebook by texting it... and might cry as a result  School today also. Blah!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[ ] data_cleaned = pd.DataFrame()
    data_cleaned["text"] = final_corpus_joined
    data_cleaned["sentiment"] = df["sentiment"].values
```

```
[ ] data_cleaned
```

	text	sentiment
0	upset updat facebook text might cri result sch...	0
1	kenichan dive mani time ball manag save rest g...	0
2	whole bodi feel itchi like fire	0
3	nationwideclass behav mad see	0
4	kwesidei whole crew	0
...	...	...
1048567	grandma make dinenr mum	4
1048568	mid morn snack time bowl chees noodl yum	4
1048569	shadela say like termini movi come like word	4
1048570	destinyhop im great thaank wbuu	4
1048571	cant wait til date weekend	4

1048572 rows × 2 columns

```
[ ] #This df is created for wordcloud only
    data_eda = pd.DataFrame()
    data_eda['text'] = final_corpus
    data_eda['sentiment'] = df['sentiment'].values
    data_eda.head()
```

	text	sentiment
0	[upset, updat, facebook, text, might, cri, res...	0
1	[kenichan, dive, mani, time, ball, manag, save...	0
2	[whole, bodi, feel, itchi, like, fire]	0
3	[nationwideclass, behav, mad, see]	0
4	[kwesidei, whole, crew]	0

EXPLORATORY DATA ANALYSIS (EDA):

Exploratory Data Analysis (EDA) is a critical step in the data analysis process which involves a series of techniques and practices to examine, summarize, and visualize data sets. EDA aims to uncover patterns, trends, anomalies, and relationships within the data, providing a foundation for more in-depth analysis. During EDA, analysts create descriptive statistics, histograms, and other visualizations to gain insights into the data's structure and characteristics. This process helps identify data quality issues, informs feature selection for modeling, and guides the formulation ofhypotheses for further analysis. EDA is an essential tool for data scientists and analysts to understand their data and make informed decisions based on its findings,this is a crucial step in data analysis as it helps in understanding the structure, patterns, and relationships within the dataset by visualizing and summarizing the data, EDA can reveal any outliers, missing values, or potential problems with the data quality. It also helps in identifying the appropriate statistical techniques and models for further analysis. Conducting EDA to understand the data's characteristics,this involves statistical summaries, data visualization, and identifying patterns or trends.

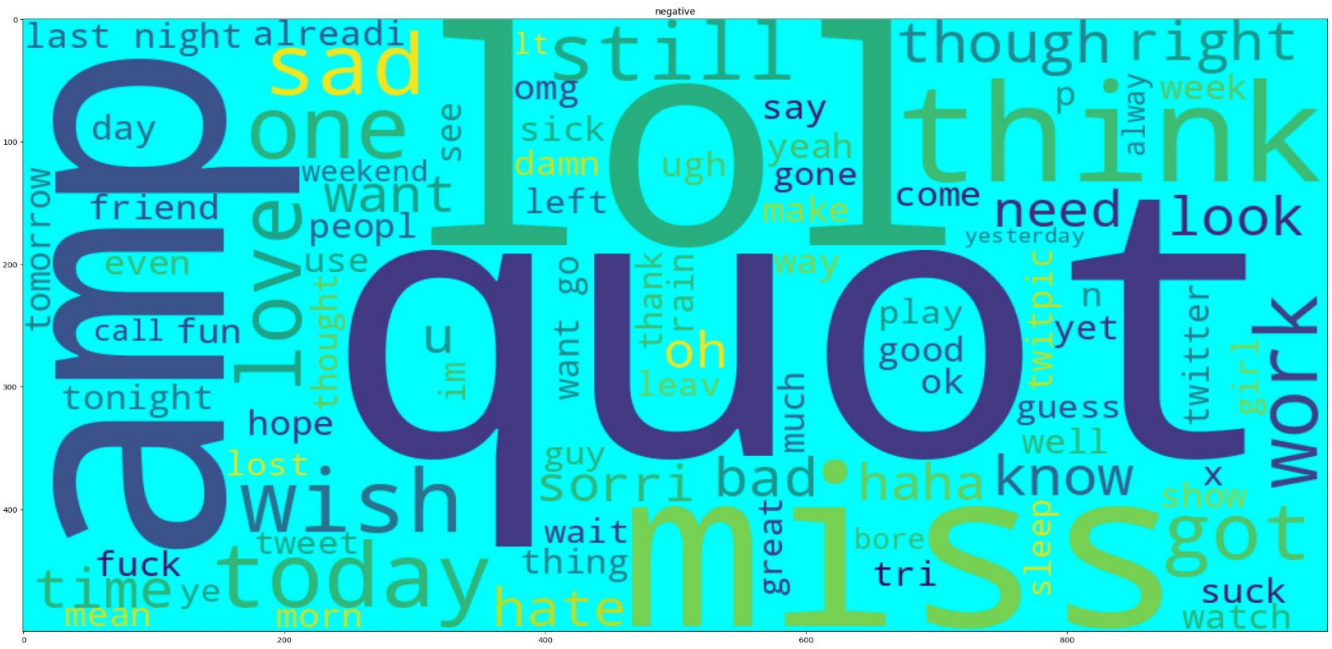
For Positives

```
[ ]  from wordcloud import WordCloud
     WordCloud()
     wordcloud = WordCloud(width=1000,
                           height=500,
                           background_color='magenta',
                           max_words = 90).generate(positive_all)

     plt.figure(figsize=(30,20))
     plt.imshow(wordcloud)
     plt.title("Positive")
     plt.show()
```


```
from wordcloud import WordCloud
WordCloud()
wordcloud = WordCloud(width=1000,
                      height=500,
                      background_color='cyan',
                      max_words = 90).generate(negative_all)

plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.title("negative")
plt.show()
```



MODEL BUILDING:

1.Data Preparation:

- **X** is assigned the "text" column of the data cleaned DataFrame, which contains the preprocessed text data.
- **Y** is assigned the "sentiment" column of the same DataFrame, which contains the sentiment labels.
- Tures of the text data stored in **X**.

2.Data Splitting:

- The data is split into training and testing sets using train_test_split from scikit-learn. The data is divided into X_train (features for training), X_test (features for testing), y_train (labels for training), and y_test (labels for testing).
- The test size is set to 20%, and a random state is specified for reproducibility.

```
[ ]  X = data_cleaned['text']
     y = data_cleaned['sentiment']
```

```
[ ]  from sklearn.feature_extraction.text import TfidfVectorizer
     tfid = TfidfVectorizer()
     Xt = tfid.fit_transform(X)
```

```
[ ]  from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(Xt, y, test_size=0.2, random_state= 42)
```

3.Logistic Regression Model:

- The code imports the logistic regression classifier from scikit-learn using from sklearn.linear_model import LogisticRegression.
- A logistic regression model is instantiated as model.
- The model is trained on the training data using model.fit(X_train, y_train).

```
[ ]  from sklearn.linear_model import LogisticRegression
     model = LogisticRegression()
     model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

4.Model Prediction:

The trained logistic regression model is used to make predictions on the testing data with y_pred = model.predict(X_test)

```
[ ]  y_pred = model.predict(X_test)
```

```
#Showing Model Summary
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
print('Classification Report:\n', classification_report(y_test, y_pred))
```

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

           0       0.85      0.95      0.90    160130
           4       0.73      0.46      0.57     49585

    accuracy                           0.83    209715
   macro avg       0.79      0.70      0.73    209715
weighted avg       0.82      0.83      0.82    209715
```

MODEL EVALUTION:

1.Displaying Model Summary:

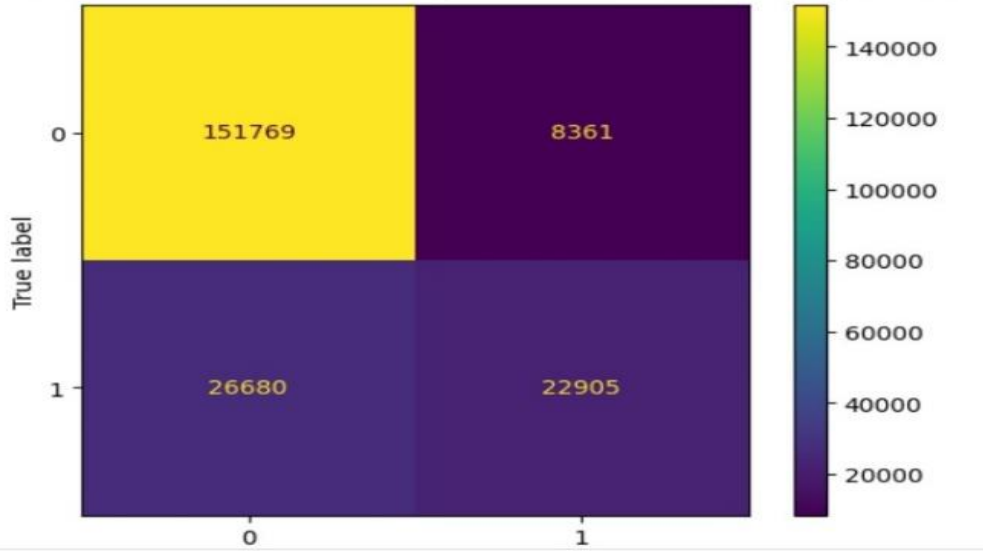
- The code imports necessary functions from scikit-learn for model evaluation: classification_report, confusion_matrix, and ConfusionMatrixDisplay.
- It prints a classification report using classification_report(y_test, y_pred). The classification report provides metrics such as precision, recall, F1-score, and support for each class (sentiment label).

2.Visualizing Model Results:

- The code uses ConfusionMatrixDisplay to create a visualization of the confusion matrix. The confusion matrix is a table used to evaluate the performance of a classification model.
- It plots the confusion matrix using t1.plot()

```
[ ]  #Visualizing Model Results
     t1 = ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred))
     t1.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x79b26ede8040>
```



BUISNESS INSIGHTS:

1. Customer Perception: Understand how customers perceive your brand, product, or service based on their tweets.
2. Brand Monitoring: Track brand mentions and sentiment to assess the impact of marketing campaigns or product launches.
3. Customer Feedback: Identify customer pain points and areas of improvement through analysis of their tweets.
4. Trend Analysis: Monitor sentiment trends over time to gauge the impact of current events or market dynamics on customer sentiment.
5. Competitor Analysis: Compare sentiment around your brand with that of competitors to gain a competitive edge.
6. Customer Service Optimization: Identify and address customer concerns and issues in real-time for improved customer service.
7. Product Development: Use feedback from tweets to enhance existing products or develop new offerings aligned with customer preferences and sentiments.

Applying sentiment analysis to a tweet dataset can thus offer valuable insights that can be leveraged to improve brand perception, customer satisfaction, and overall business performance.

CONCLUSION:

Overall, the code provides a solid foundation for sentiment analysis, from data preprocessing to model building and evaluation. The process can be further refined and expanded, and additional machine learning models and techniques can be explored to improve sentiment classification accuracy. This code serves as a valuable starting point for sentiment analysis tasks on textual data and it can be adapted and extended to suit the specific needs of different applications and industries.

THE END