جامعة أم القرى
UMM AL-QURA UNIVERSITY

DATA ANALYSIS

COURSE PRESENTER
(DR. Omaimah)

Sentiment140 dataset with 1.6 million tweets

STUDENT	ID
SHAHAD AMER	444005703
AREEJ TALEB	444002403

DEPARTMENT OF (INFORMATION SCIENCE-DATA SCIENCE)
COLLEGE OF COMPUTER AND INFORMATION SYSTEMS
UMM AL-QURA UNIVERSITY

2024

INTRODUCTION

The **Sentiment140 dataset** is a widely used resource for sentiment analysis, comprising 1.6 million tweets. It was created to facilitate research and development in natural language processing (NLP) and machine learning, particularly in sentiment classification.

The tweets were collected using Twitter's API and pre-processed to remove irrelevant information, ensuring a clean and rich dataset for analysis.

OBJECTIVE

perform sentiment analysis on the Sentiment140 dataset to classify the sentiment expressed in tweets as positive, negative, or neutral.

DATA EXPLORATION

In this step we are focusing on understanding the structure of the data by:

Displaying Data Information

df.columns: Prints the names of all columns in the DataFrame.
df.describe().T: Provides descriptive statistics (like count, mean, standard deviation, etc.) for numerical columns, transposed for easier reading.
df.info(): Displays a concise summary of the DataFrame, including the number of non-null entries and data types for each column.
df.head(): Shows the first five rows of the DataFrame, giving a quick look at the data.

Renaming Columns

renames the columns to more meaningful names. This improves readability and makes it easier to understand the data:

- The first column is renamed to "target".
- The second column is renamed to "id".
- The date column is renamed to "date."
- A flag column is renamed to "flag".
- A user column is renamed to "user".

The sixth column (the exact name isn’t specified) is renamed to "text" for the textual data

Checking for Missing Values

There are no missing values

Selecting Columns for Analysis

Select 'target', 'user', 'flag', 'id', 'date', 'text' columns this ensures that the upcoming operations focus only on these specific columns

Counting Unique Values in Each Column

It uses value_counts() to display the frequency of unique values in that column, which helps in understanding the distribution of data.

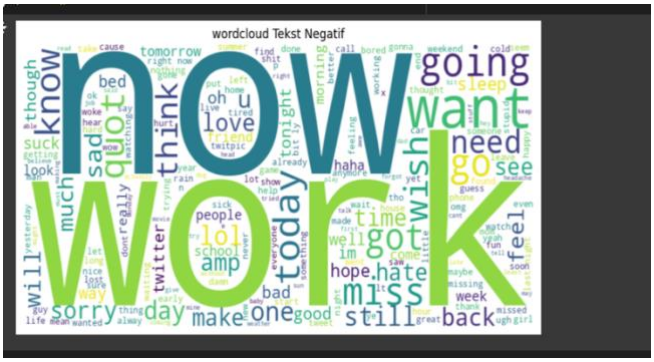
VISUALIZATION

```python
import matplotlib.pyplot as plt
from wordcloud import WordCloud
# النصوص الإيجابية
positive_text = ' '.join(df[df['target'] == 4]['text'])  # تأكد من استخدام القيمة الصحيحة للإيجابية

# إنشاء سحابة الكلمات للنصوص الإيجابية
positive_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(positive_text)

# عرض سحابة الكلمات للنصوص الإيجابية
plt.figure(figsize=(10, 5))
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('wordcloudTekst Positif')  # عنوان السحابة
plt.show()
```

```python
import matplotlib.pyplot as plt
from wordcloud import WordCloud
# النصوص السلبية
negative_text = ' '.join(df[df['target'] == 0]['text'])  # تأكد من استخدام القيمة الصحيحة للسلبية

# إنشاء سحابة الكلمات للنصوص السلبية
negative_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(negative_text)

# عرض سحابة الكلمات للنصوص السلبية
plt.figure(figsize=(10, 5))
plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('wordcloud Tekst Negatif')  # عنوان السحابة
plt.show()
```

The overall function of this code is to create a visual representation of the most common words found in a collection of positive texts from a Data Frame. The words that appear most frequently are displayed in larger font sizes, making it easy to identify key themes or sentiments in the text data.

Also used the same code for the negative words, but change the target to 0 (negative).



we can see the difference between the negative word cloud and the positive word cloud, the words like love, thank, and ha-ha…are the most frequent in positive texts and the words like now, work, and sorry are the most frequent in negative texts.

TEXT PREPROCESSING

Cleaning the Text Column

- Check for 'text' Column
- Remove URLs
- Remove Usernames and Hashtags
- Remove Special Characters
- Convert to Lowercase
- Error handling

Cleaning the Date Column

- Check for 'date' Column
- Convert to Datetime

Cleaning the Target Column

- Check for 'target' Column
- Convert to Categorical

Cleaning the Flag Column

- Check for 'flag' Column
- Strip Extra Spaces

Cleaning the ID Column

- Check for 'id' Column

removing stop words

- Stop words are commonly used words (like "and," "the," "is," etc.) that are often filtered out in natural language processing (NLP) because they carry little meaningful information.

Define Processing Function: defined a function called tokenize_and_lemmatize that:

- Tokenizes the input text into individual words.
- Applies lemmatization to each token to convert it to its base form

Display Cleaned Data: After all the cleaning steps, the first few rows of the cleaned Data Frame are displayed using print, allowing for a quick review of the results.

BUILD NAÏVE BAYES MODEL

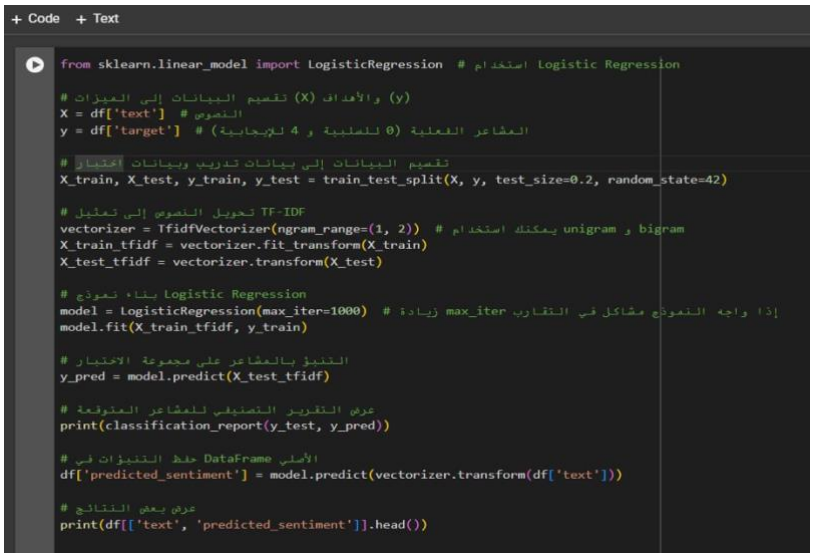
```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# تقسيم البيانات إلى الميزات (X) والأهداف (y)
X = df['text']  # النصوص
y = df['target']  # المشاعر الفعلية (0 للسلبية و 4 للإيجابية)

# تقسيم البيانات إلى بيانات تدريب وبيانات اختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# تحويل النصوص إلى تمثيل TF-IDF
vectorizer = TfidfVectorizer(ngram_range=(1, 2))
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# التنبؤ بالمشاعر على مجموعة الاختبار
y_pred = model.predict(X_test_tfidf)

# عرض التقرير التصنيفي للمشاعر المتوقعة
print(classification_report(y_test, y_pred))

# حفظ التنبؤات في DataFrame الأصلي
df['predicted_sentiment'] = model.predict(vectorizer.transform(df['text']))

# عرض بعض النتائج
print(df[['text', 'predicted_sentiment']].head())
```

In This step, splits a text dataset into training and testing sets, transforms the text into a numerical format using TF-IDF, trains a **Multinomial Naive Bayes model** for sentiment classification, makes predictions, evaluates the model's performance, and stores the predictions in the original DataFrame. Finally, it displays some of the results.

```
              precision    recall  f1-score   support

           0       0.77      0.80      0.78    159494
           4       0.79      0.76      0.78    160506

    accuracy                           0.78    320000
   macro avg       0.78      0.78      0.78    320000
weighted avg       0.78      0.78      0.78    320000

                                                text  predicted_sentiment
0  upset update facebook texting might cry result...                    0
1  dived many times ball managed save 50 rest go ...                    0
2                  whole body feels itchy like fire                     0
3                                behaving mad see                       0
4                                      whole crew                       4
```

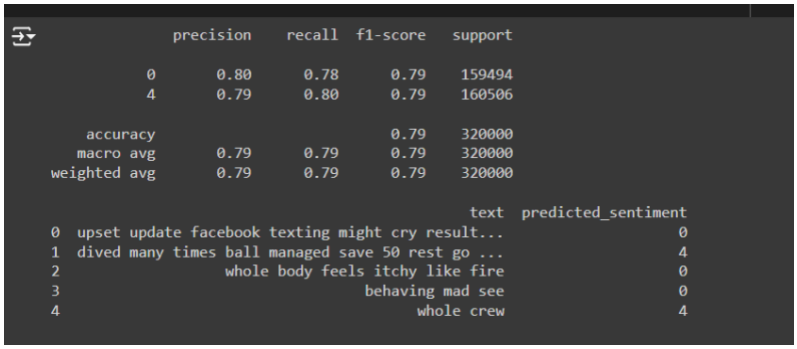
The overall accuracy of the model is **0.78**, indicating it correctly classified **78%** of the total instances across both classes.

BUILD LOGISTIC REGRESSION MODEL



```
+ Code + Text

from sklearn.linear_model import LogisticRegression  # استخدام Logistic Regression

# تقسيم البيانات إلى الميزات (X) والأهداف (y)
X = df['text']  # النصوص
y = df['target']  # المشاعر الفعلية (0 للسلبية و 4 للإيجابية)

# تقسيم البيانات إلى بيانات تدريب وبيانات اختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# تحويل النصوص إلى تمثيل TF-IDF
vectorizer = TfidfVectorizer(ngram_range=(1, 2))  # يمكنك استخدام unigram و bigram
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# بناء نموذج Logistic Regression
model = LogisticRegression(max_iter=1000)  # زيادة max_iter إذا واجه النموذج مشاكل في التقارب
model.fit(X_train_tfidf, y_train)

# التنبؤ بالمشاعر على مجموعة الاختبار
y_pred = model.predict(X_test_tfidf)

# عرض التقرير التصنيفي للمشاعر المتوقعة
print(classification_report(y_test, y_pred))

# حفظ التنبؤات في DataFrame الأصلي
df['predicted_sentiment'] = model.predict(vectorizer.transform(df['text']))

# عرض بعض النتائج
print(df[['text', 'predicted_sentiment']].head())
```

This code performs sentiment analysis using Logistic Regression by preparing data, training the model, making predictions, and displaying results.



```
              precision    recall  f1-score   support

           0       0.80      0.78      0.79    159494
           4       0.79      0.80      0.79    160506

    accuracy                           0.79    320000
   macro avg       0.79      0.79      0.79    320000
weighted avg       0.79      0.79      0.79    320000

                                                text  predicted_sentiment
0  upset update facebook texting might cry result...                    0
1  dived many times ball managed save 50 rest go ...                    4
2                  whole body feels itchy like fire                     0
3                                behaving mad see                       0
4                                      whole crew                       4
```

The overall accuracy of the model is **0.79**, meaning it correctly classified **79%** of the instances across both classes.