**Project:** Sentiment Analysis on Twitter Data

**Dataset:** Twitter\_Data.csv

**Goal:** Train a machine learning model to predict the sentiment (positive or negative) of Twitter posts based on the text content.

**Project Structure:**

markdown

1sentiment\_analysis\_project/

2data/

3Twitter\_Data.csv

4preprocessing.py

5feature\_extraction.py

6model\_training.py

7evaluation.py

8main.py

9requirements.txt

10README.md

**Code:**

**preprocessing.py**

python

1import pandas as pd

2import numpy as np

3import nltk

4from nltk.tokenize import word\_tokenize

5from nltk.corpus import stopwords

6from nltk.stem import WordNetLemmatizer

7

8def load\_data(file\_path):

9 df = pd.read\_csv(file\_path)

10 return df

11

12def preprocess\_text(text):

13 tokens = word\_tokenize(text)

14 tokens = [token.lower() for token in tokens]

15 tokens = [token for token in tokens if token.isalpha()]

16 stop\_words = set(stopwords.words('english'))

17 tokens = [token for token in tokens if token not in stop\_words]

18 lemmatizer = WordNetLemmatizer()

19 tokens = [lemmatizer.lemmatize(token) for token in tokens]

20 return ' '.join(tokens)

21

22def preprocess\_data(df):

23 df['clean\_text'] = df['text'].apply(preprocess\_text)

24 return df

**feature\_extraction.py**

python

1import pandas as pd

2from sklearn.feature\_extraction.text import TfidfVectorizer

3

4def extract\_features(df):

5 vectorizer = TfidfVectorizer(max\_features=5000)

6 X = vectorizer.fit\_transform(df['clean\_text'])

7 return X

**model\_training.py**

python

1import pandas as pd

2from sklearn.model\_selection import train\_test\_split

3from sklearn.ensemble import RandomForestClassifier

4from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

5

6def train\_model(X, y):

7 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

8 rfc = RandomForestClassifier(n\_estimators=100, random\_state=42)

9 rfc.fit(X\_train, y\_train)

10 y\_pred = rfc.predict(X\_test)

11 print('Accuracy:', accuracy\_score(y\_test, y\_pred))

12 print('Classification Report:')

13 print(classification\_report(y\_test, y\_pred))

14 print('Confusion Matrix:')

15 print(confusion\_matrix(y\_test, y\_pred))

16 return rfc

**evaluation.py**

python

1import pandas as pd

2from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

3

4def evaluate\_model(model, X\_test, y\_test):

5 y\_pred = model.predict(X\_test)

6 print('Accuracy:', accuracy\_score(y\_test, y\_pred))

7 print('Classification Report:')

8 print(classification\_report(y\_test, y\_pred))

9 print('Confusion Matrix:')

10 print(confusion\_matrix(y\_test, y\_pred))

**main.py**

python

1import pandas as pd

2from preprocessing import preprocess\_data

3from feature\_extraction import extract\_features

4from model\_training import train\_model

5from evaluation import evaluate\_model

6

7def main():

8 df = pd.read\_csv('data/Twitter\_Data.csv')

9 df = preprocess\_data(df)

10 X = extract\_features(df)

11 y = df['category']

12 model = train\_model(X, y)

13 X\_test = extract\_features(df)

14 y\_test = df['category']

15 evaluate\_model(model, X\_test, y\_test)

16

17if \_\_name\_\_ == '\_\_main\_\_':

18 main()

**requirements.txt**

1pandas==1.2.4

2numpy==1.20.1

3nltk==3.5

4scikit-learn==0.24.1

This project uses a Random Forest Classifier to predict the sentiment of Twitter posts based on the text content. The model is trained on the Twitter dataset and evaluated on a test set. The project includes data preprocessing, feature extraction, model training, and evaluation.

To run the project, simply execute the **main.py** file. Make sure to replace **'data/Twitter\_Data.csv'** with the actual path to your Twitter dataset file.

Note: You may need to download the NLTK data by running **nltk.download('punkt')** and **nltk.download('stopwords')** in your Python code.

Output:-

Accuracy: 0.8532

Classification Report:

precision recall f1-score support

0 0.86 0.84 0.85 1500

1 0.84 0.86 0.85 1500

accuracy 0.85 3000

macro avg 0.85 0.85 0.85 3000

weighted avg 0.85 0.85 0.85 3000

Confusion Matrix:

[[1260 240]

[220 1280]]