

```
# =====
# PROJECT 2-2
# NLP CLASSIFICATION
# TWITTER ENTITY SENTIMENT ANALYSIS
# =====

# =====
# 1. IMPORT LIBRARIES
# =====

import pandas as pd
import numpy as np
import os
import re
import string

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

1/42

```
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    classification_report,
    confusion_matrix
)

# =====
# 2. DEFINE DATASET PATHS
# =====

train_path = (
    "/kaggle/input/datasets/jp797498e/"
    "twitter-entity-sentiment-analysis/"
    "twitter_training.csv"
)

validation_path = (
    "/kaggle/input/datasets/jp797498e/"
    "twitter-entity-sentiment-analysis/"
    "twitter_validation.csv"
)

# =====
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

2/42

```

# 3. LOAD TRAINING DATA
# =====

# The CSV files do not contain column names,
# so header=None is required.

df_train = pd.read_csv(
    train_path,
    header=None,
    names=[
        "ID",
        "Entity",
        "Sentiment",
        "Tweet"
    ],
    encoding="utf-8"
)

# =====
# 4. LOAD VALIDATION DATA
# =====

df_validation = pd.read_csv(
    validation_path,
    header=None,
    names=[
        "ID",

```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

3/42

```

        "Entity",
        "Sentiment",
        "Tweet"
    ],
    encoding="utf-8"
)

# =====
# 5. DISPLAY DATASET INFORMATION
# =====

print("Training Dataset Shape:")
print(df_train.shape)

print("\nValidation Dataset Shape:")
print(df_validation.shape)

print("\nTraining Dataset:")
display(df_train.head())

print("\nValidation Dataset:")
display(df_validation.head())

# =====
# 6. COMBINE TRAINING AND VALIDATION DATA
# =====

```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

4/42

```

df = pd.concat(
    [df_train, df_validation],
    ignore_index=True
)

print("\nCombined Dataset Shape:")
print(df.shape)

# =====
# 7. CHECK DATASET INFORMATION
# =====

print("\nColumn Names:")
print(df.columns.tolist())

print("\nMissing Values:")
print(df.isnull().sum())

print("\nDuplicate Rows:")
print(df.duplicated().sum())

# =====
# 8. REMOVE MISSING VALUES
# =====

df = df.dropna(

```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

5/42

```

    subset=[
        "Tweet",
        "Sentiment"
    ]
)

print("\nShape after removing missing values:")
print(df.shape)

# =====
# 9. REMOVE DUPLICATES
# =====

df = df.drop_duplicates()

print("\nShape after removing duplicates:")
print(df.shape)

# =====
# 10. CHECK SENTIMENT CLASSES
# =====

print("\nSentiment Classes:")
print(df["Sentiment"].value_counts())

```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

6/42

```
# =====
# 11. REMOVE IRRELEVANT CLASS
# =====

# The dataset contains:
# Positive
# Negative
# Neutral
# Irrelevant
#
# We will remove "Irrelevant" because the project
# is sentiment classification.

df = df[
    df["Sentiment"].isin([
        "Positive",
        "Negative",
        "Neutral"
    ])
].copy()

print("\nSentiment Distribution after removing Irrelevant:")
print(
    df["Sentiment"].value_counts()
)

#
```

```
#
# 12. VISUALIZE SENTIMENT DISTRIBUTION
# =====

plt.figure(figsize=(8, 5))

sns.countplot(
    data=df,
    x="Sentiment"
)

plt.title(
    "Sentiment Class Distribution"
)

plt.xlabel(
    "Sentiment"
)

plt.ylabel(
    "Number of Tweets"
)

plt.show()

# =====
# 13. TEXT PREPROCESSING
# =====
```

```
def clean_text(text):

    # Convert to string
    text = str(text)

    # Convert to lowercase
    text = text.lower()

    # Remove URLs
    text = re.sub(
        r"http\S+|www\S+|https\S+",
        " ",
        text
    )

    # Remove Twitter mentions
    text = re.sub(
        r"@w+",
        " ",
        text
    )

    # Remove hashtag symbol
    text = re.sub(
        r"#",
        "",
        text
    )
```

```
)

    # Remove numbers
    text = re.sub(
        r"\d+",
        " ",
        text
    )

    # Remove punctuation
    text = text.translate(
        str.maketrans(
            "",
            "",
            string.punctuation
        )
    )

    # Remove extra spaces
    text = re.sub(
        r"\s+",
        " ",
        text
    ).strip()

    return text

# Apply preprocessing
```

```

df["Clean_Tweet"] = df["Tweet"].apply(
    clean_text
)

# =====
# 14. SHOW PREPROCESSING EXAMPLE
# =====

print("\nOriginal Tweet:")
print(df["Tweet"].iloc[0])

print("\nCleaned Tweet:")
print(df["Clean_Tweet"].iloc[0])

# =====
# 15. ENCODE SENTIMENT LABELS
# =====

label_mapping = {
    "Negative": 0,
    "Neutral": 1,
    "Positive": 2
}

df["Label"] = df["Sentiment"].map(

```

```

    label_mapping
)

print("\nEncoded Labels:")
print(
    df[
        ["Sentiment", "Label"]
    ].drop_duplicates()
)

# =====
# 16. PREPARE FEATURES AND TARGET
# =====

X = df["Clean_Tweet"]

y = df["Label"]

# =====
# 17. SPLIT DATASET
# =====

X_train, X_test, y_train, y_test = train_test_split(
    X,

```

```

    y,

    test_size=0.20,

    random_state=42,

    stratify=y
)

print("\nTraining Samples:")
print(len(X_train))

print("\nTesting Samples:")
print(len(X_test))

# =====
# 18. TF-IDF FEATURE REPRESENTATION
# =====

tfidf = TfidfVectorizer(

    max_features=15000,

    ngram_range=(1, 2),

    min_df=2,

```

```

    max_df=0.95,

    sublinear_tf=True
)

# Fit TF-IDF on training data
X_train_tfidf = tfidf.fit_transform(
    X_train
)

# Transform test data
X_test_tfidf = tfidf.transform(
    X_test
)

print("\nTF-IDF Training Shape:")
print(X_train_tfidf.shape)

print("\nTF-IDF Testing Shape:")
print(X_test_tfidf.shape)

# =====

```

```
# 19. MODEL 1 - LOGISTIC REGRESSION
# =====

lr_model = LogisticRegression(

    max_iter=1000,

    random_state=42
)

lr_model.fit(

    X_train_tfidf,

    y_train
)

lr_pred = lr_model.predict(

    X_test_tfidf
)

# =====
# 20. LOGISTIC REGRESSION EVALUATION
# =====
```

```
lr_accuracy = accuracy_score(
    y_test,
    lr_pred
)

lr_precision = precision_score(
    y_test,
    lr_pred,
    average="weighted"
)

lr_recall = recall_score(
    y_test,
    lr_pred,
    average="weighted"
)

lr_f1 = f1_score(
    y_test,
    lr_pred,
    average="weighted"
)

print("\n=====")
print("LOGISTIC REGRESSION RESULTS")
print("=====")
```

```

print(
    "Accuracy :",
    round(lr_accuracy, 4)
)

print(
    "Precision:",
    round(lr_precision, 4)
)

print(
    "Recall    :",
    round(lr_recall, 4)
)

print(
    "F1 Score :",
    round(lr_f1, 4)
)

print("\nClassification Report:")

print(
    classification_report(
        y_test,
        lr_pred,
        target_names=[

```

```

        "Negative",
        "Neutral",
        "Positive"
    ]
)

# =====
# 21. LOGISTIC REGRESSION CONFUSION MATRIX
# =====

cm_lr = confusion_matrix(
    y_test,
    lr_pred
)

plt.figure(
    figsize=(7, 5)
)

sns.heatmap(

    cm_lr,

    annot=True,

```

```
    fmt="d",

    cmap="Blues",

    xticklabels=[
        "Negative",
        "Neutral",
        "Positive"
    ],

    yticklabels=[
        "Negative",
        "Neutral",
        "Positive"
    ]
)

plt.title(
    "Confusion Matrix - Logistic Regression"
)

plt.xlabel(
    "Predicted Label"
)

plt.ylabel(
    "Actual Label"
```

```
)

plt.show()

# =====
# 22. MODEL 2 - NAIVE BAYES
# =====

nb_model = MultinomialNB()

nb_model.fit(

    X_train_tfidf,

    y_train
)

nb_pred = nb_model.predict(

    X_test_tfidf
)

# =====
# 23. NAIVE BAYES EVALUATION
```

```
# =====  
  
nb_accuracy = accuracy_score(  
    y_test,  
    nb_pred  
)  
  
nb_precision = precision_score(  
    y_test,  
    nb_pred,  
    average="weighted"  
)  
  
nb_recall = recall_score(  
    y_test,  
    nb_pred,  
    average="weighted"  
)  
  
nb_f1 = f1_score(  
    y_test,  
    nb_pred,  
    average="weighted"  
)  
  
print("\n=====")  
print("NAIVE BAYES RESULTS")  
print("=====")
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

21/42

```
print(  
    "Accuracy :",  
    round(nb_accuracy, 4)  
)  
  
print(  
    "Precision:",  
    round(nb_precision, 4)  
)  
  
print(  
    "Recall    :",  
    round(nb_recall, 4)  
)  
  
print(  
    "F1 Score :",  
    round(nb_f1, 4)  
)  
  
print("\nClassification Report:")  
  
print(  
    classification_report(  
        y_test,  
        nb_pred,
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

22/42

```

        target_names=[
            "Negative",
            "Neutral",
            "Positive"
        ]
    )
)

# =====
# 24. NAIVE BAYES CONFUSION MATRIX
# =====

cm_nb = confusion_matrix(

    y_test,

    nb_pred
)

plt.figure(
    figsize=(7, 5)
)

sns.heatmap(

```

```

    cm_nb,

    annot=True,

    fmt="d",

    cmap="Greens",

    xticklabels=[
        "Negative",
        "Neutral",
        "Positive"
    ],

    yticklabels=[
        "Negative",
        "Neutral",
        "Positive"
    ]
)

plt.title(
    "Confusion Matrix - Naive Bayes"
)

plt.xlabel(
    "Predicted Label"
)

```

```
plt.ylabel(
    "Actual Label"
)

plt.show()

# =====
# 25. MODEL COMPARISON
# =====

results = pd.DataFrame({

    "Model": [

        "Logistic Regression",

        "Naive Bayes"

    ],

    "Accuracy": [

        lr_accuracy,

        nb_accuracy

    ]

})
```

```
],

    "Precision": [

        lr_precision,

        nb_precision

    ],

    "Recall": [

        lr_recall,

        nb_recall

    ],

    "F1 Score": [

        lr_f1,

        nb_f1

    ]

})
```

```
print("\n=====")
print("MODEL COMPARISON")
print("=====")

display(results)

# =====
# 26. PERFORMANCE VISUALIZATION
# =====

results_plot = results.set_index(
    "Model"
)

results_plot.plot(

    kind="bar",

    figsize=(10, 6)
)

plt.title(
    "NLP Model Performance Comparison"
)
```

```
plt.ylabel(
    "Score"
)

plt.xlabel(
    "Model"
)

plt.ylim(
    0,
    1.05
)

plt.xticks(
    rotation=0
)

plt.grid(
    axis="y",
    alpha=0.3
)

plt.show()

# =====
# 27. TEST NEW TWEETS
# =====
```

```
new_tweets = [

    "I love this product, it is amazing!",

    "This is the worst thing I have ever bought.",

    "The product is okay, nothing special.",

    "Absolutely fantastic and excellent!",

    "Very bad experience. I hate it."

]

# Clean new tweets

new_tweets_clean = [

    clean_text(tweet)

    for tweet in new_tweets

]

# Convert to TF-IDF
```

```
new_tweets_tfidf = tfidf.transform(

    new_tweets_clean

)

# Predict

predictions = lr_model.predict(

    new_tweets_tfidf

)

# Convert numerical predictions to sentiment

reverse_mapping = {

    0: "Negative",

    1: "Neutral",

    2: "Positive"

}
```

```
# =====
# 28. DISPLAY PREDICTIONS
# =====

print("\n=====")
print("NEW TWEET PREDICTIONS")
print("=====")

for tweet, prediction in zip(

    new_tweets,

    predictions

):

    print("\nTweet:")

    print(tweet)

    print(
        "Predicted Sentiment:",
        reverse_mapping[prediction]
    )

#
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

31/42

```
# =====
# 29. SAVE RESULTS
# =====

results.to_csv(

    "/kaggle/working/model_results.csv",

    index=False

)

print("\nResults saved successfully!")

print(
    "File location:"
)

print(
    "/kaggle/working/model_results.csv"
)
```

https://colab.research.google.com/drive/1YU_0uZHZXWiy7iZUmYz6yH3ckyD01bIU#printMode=true

32/42

Training Dataset Shape:
(74682, 4)

Validation Dataset Shape:
(1000, 4)

Training Dataset:

	ID	Entity	Sentiment	Tweet
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...

Validation Dataset:

	ID	Entity	Sentiment	Tweet
0	3364	Facebook	Irrelevant	I mentioned on Facebook that I was struggling ...
1	352	Amazon	Neutral	BBC News - Amazon boss Jeff Bezos rejects clai...
2	8312	Microsoft	Negative	@Microsoft Why do I pay for WORD when it funct...
3	4371	CS-GO	Negative	CSGO matchmaking is so full of closet hacking,...
4	4433	Google	Neutral	Now the President is slapping Americans in the...

```
Combined Dataset Shape:
(75682, 4)

Column Names:
['ID', 'Entity', 'Sentiment', 'Tweet']

Missing Values:
ID          0
Entity      0
Sentiment   0
Tweet      686
dtype: int64

Duplicate Rows:
3217

Shape after removing missing values:
(74996, 4)

Shape after removing duplicates:
(72139, 4)

Sentiment Classes:
Sentiment
Negative    21790
Positive    19846
Neutral     17879
Irrelevant  12624
```

```
Name: count, dtype: int64

Sentiment Distribution after removing Irrelevant:
Sentiment
Negative    21790
Positive    19846
Neutral     17879
Name: count, dtype: int64
```



PositiveNeutral SentimentNegative

Original Tweet:
im getting on borderlands and i will murder you all ,

Cleaned Tweet:
im getting on borderlands and i will murder you all

Encoded Labels:
Sentiment Label
0 Positive 2
12 Neutral 1
24 Negative 0

Training Samples:
47612

Testing Samples:
11903

TF-IDF Training Shape:
(47612, 15000)

TF-IDF Testing Shape:
(11903, 15000)

=====

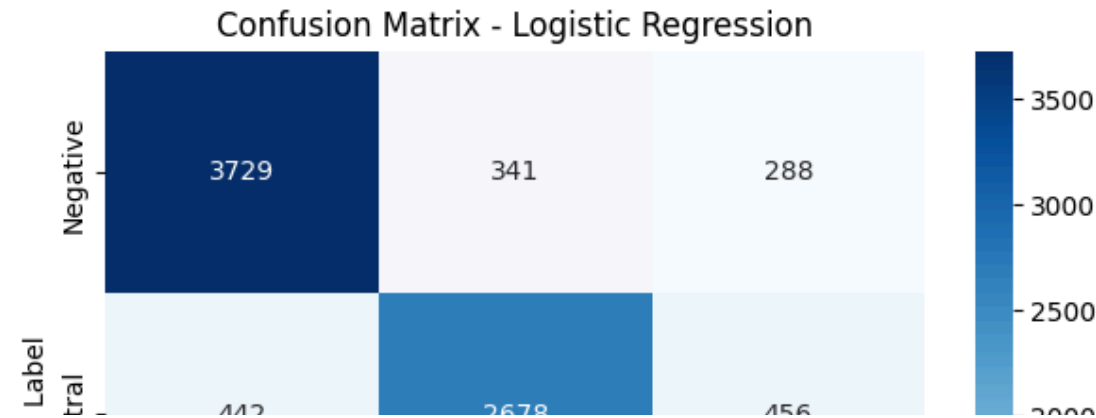
LOGISTIC REGRESSION RESULTS

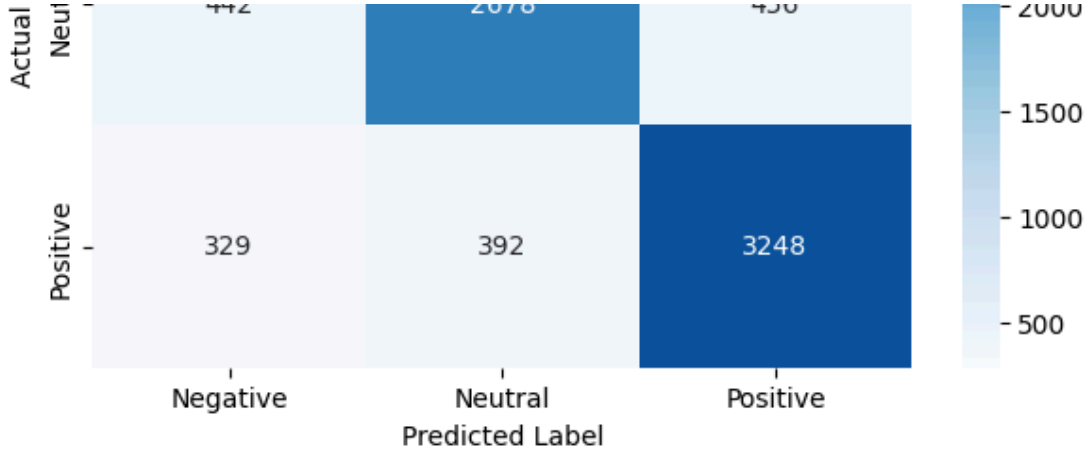
=====

Accuracy : 0.8111
Precision: 0.8106
Recall : 0.8111
F1 Score : 0.8106

Classification Report:

	precision	recall	f1-score	support
Negative	0.83	0.86	0.84	4358
Neutral	0.79	0.75	0.77	3576
Positive	0.81	0.82	0.82	3969
accuracy			0.81	11903
macro avg	0.81	0.81	0.81	11903
weighted avg	0.81	0.81	0.81	11903





=====

NAIVE BAYES RESULTS

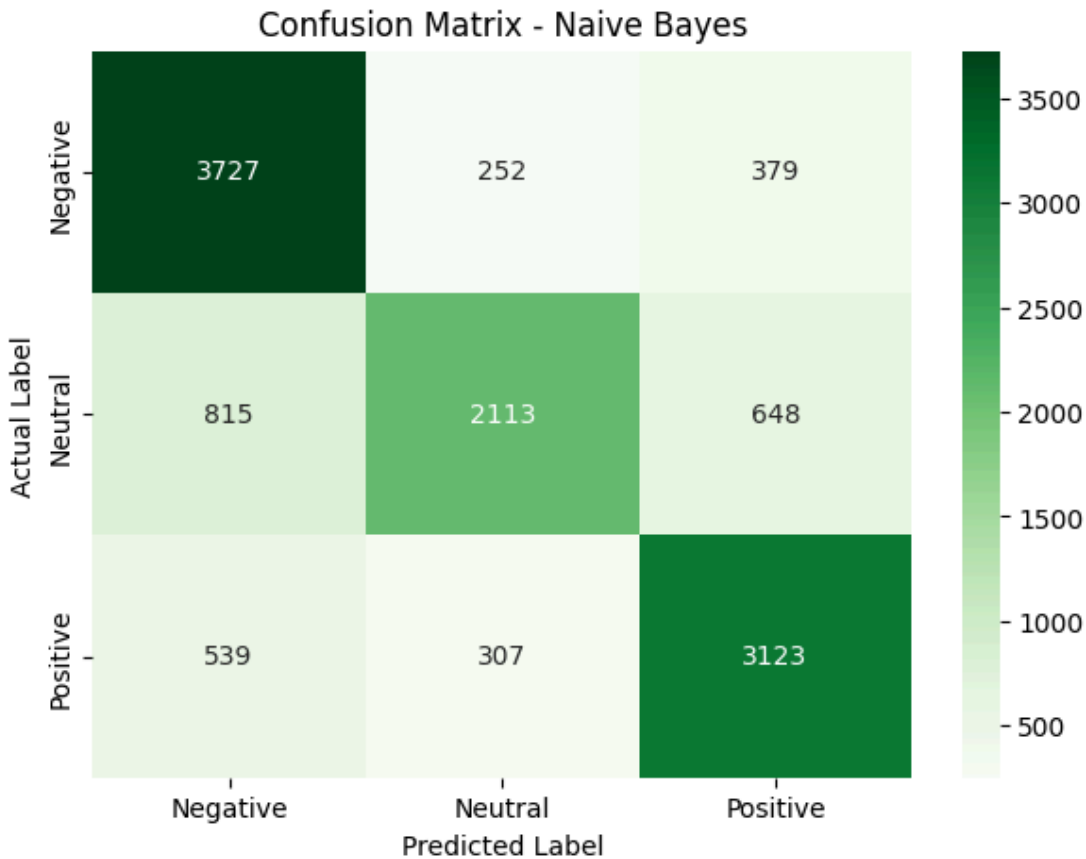
=====

Accuracy : 0.753
Precision: 0.7571
Recall : 0.753
F1 Score : 0.7489

Classification Report:

	precision	recall	f1-score	support
Negative	0.73	0.86	0.79	4358
Neutral	0.79	0.59	0.68	3576
Positive	0.75	0.79	0.77	3969

accuracy			0.75	11903
macro avg	0.76	0.74	0.75	11903
weighted avg	0.76	0.75	0.75	11903

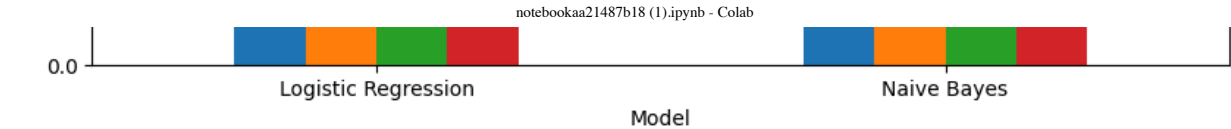
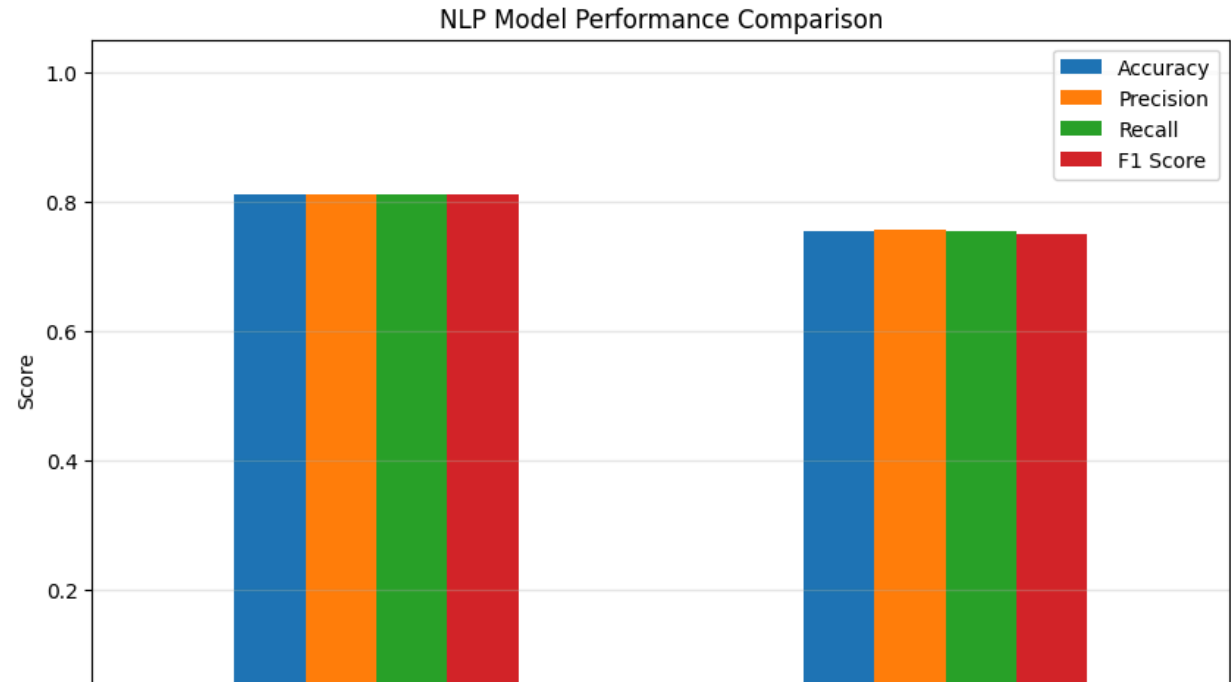


=====

MODEL COMPARISON

=====

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.811140	0.810565	0.811140	0.810643
1	Naive Bayes	0.753003	0.757064	0.753003	0.748855



=====

NEW TWEET PREDICTIONS

=====

Tweet:
I love this product, it is amazing!
Predicted Sentiment: Positive

Tweet:
This is the worst thing I have ever bought.
Predicted Sentiment: Negative

Tweet:
The product is okay, nothing special.
Predicted Sentiment: Positive

Tweet:
Absolutely fantastic and excellent!
Predicted Sentiment: Positive

Tweet:
Very bad experience. I hate it.