```
from google.colab import files
uploaded = files.upload()
```

Choose Files Twitter_sen…nt_data.csv

- **Twitter_sentiment_data.csv**(text/csv) - 20895533 bytes, last modified: 29/6/2025 - 100% done

```
Saving Twitter_sentiment_data.csv to Twitter_sentiment_data.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
```

```
nltk.download('stopwords')
```

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

```
#load and view data
df = pd.read_csv("Twitter_sentiment_data.csv")
df.head()
```

	clean_text	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0

```
#data cleaning
df.columns
```

```
Index(['clean_text', 'category'], dtype='object')
```

```
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import re

ps = PorterStemmer()
corpus = []

for i in range(0, len(df)):
    review = re.sub('[^a-zA-Z]', ' ', str(df['clean_text'][i]))
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if word not in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

```
 #convert text to numbers
 from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=3000)
X = cv.fit_transform(corpus).toarray()
y = df['category']
```

```
#split data into train and set tasks
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("Any NaNs in y_train:", y_train.isnull().sum())
```

```
X_train shape: (130384, 3000)
y_train shape: (130384,)
Any NaNs in y_train: 5
```

```
# Create a mask of non-null labels
mask = ~y_train.isnull()
```

```
# Filter both X_train and y_train based on the mask
X_train = X_train[mask]
y_train = y_train[mask]

# Flatten y (optional but safe)
y_train = y_train.values.ravel()
y_test = y_test.values.ravel()
```

```
#train a naive bayes model
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
# Clean y_test NaNs
test_mask = ~pd.isnull(y_test)
X_test = X_test[test_mask]
y_test = y_test[test_mask]

# Flatten without .values
y_test = y_test.ravel()
```

```
#make predictions and evaluate
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test)

print("📊 Accuracy:", accuracy_score(y_test, y_pred))
print("\n📋 Classification Report:\n")
print(classification_report(y_test, y_pred))
```

```
📊 Accuracy: 0.7436951586181506

📋 Classification Report:

              precision    recall  f1-score   support

        -1.0       0.66      0.66      0.66      7061
         0.0       0.77      0.74      0.75     10974
         1.0       0.76      0.79      0.78     14559

    accuracy                           0.74     32594
   macro avg       0.73      0.73      0.73     32594
weighted avg       0.74      0.74      0.74     32594
```

```
#testing sentence
sample = ["I love this product, it's amazing!"]
sample_cleaned = re.sub('[^a-zA-Z]', ' ', sample[0])
sample_cleaned = sample_cleaned.lower()
sample_cleaned = sample_cleaned.split()
sample_cleaned = [ps.stem(word) for word in sample_cleaned if word not in stopwc
sample_cleaned = ' '.join(sample_cleaned)
sample_vec = cv.transform([sample_cleaned]).toarray()

model.predict(sample_vec)
```

```
array([1.])
```

Start coding or generate with AI.