Sentiment Analysis of Product Reviews

1. **Create a Virtual Environment**

This keeps your project clean and dependencies isolated:

python3 -m venv venv

source venv/bin/activate # macOS/Linux

1. **Install the packages**

pip install pandas numpy matplotlib seaborn nltk scikit-learn wordcloud textblob

1. **Load the Dataset**

Use the Amazon Fine Food Reviews dataset on Kaggle.  
Download it and place Reviews.csv in your working directory.

python

CopyEdit

import pandas as pd

df = pd.read\_csv("Reviews.csv")

df.head()

**Clean and Preprocess the Data**

python

CopyEdit

# Drop missing or unnecessary columns

df = df[['Text', 'Score']].dropna()

# Map scores to sentiment labels

def sentiment(score):

if score <= 2:

return 'negative'

elif score == 3:

return 'neutral'

else:

return 'positive'

df['Sentiment'] = df['Score'].apply(sentiment)

# Optional: Drop neutral for binary classification

df = df[df['Sentiment'] != 'neutral']

# sentiment\_analysis.py

# -------------------------------

# STEP 1: Import Libraries

# -------------------------------

import pandas as pd

import numpy as np

import re

import nltk

import matplotlib.pyplot as plt

import seaborn as sns

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from wordcloud import WordCloud

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report, accuracy\_score

# -------------------------------

# STEP 2: Load Dataset

# -------------------------------

df = pd.read\_csv("Reviews.csv") # Make sure Reviews.csv is in the same folder

df = df[['Text', 'Score']].dropna()

# -------------------------------

# STEP 3: Label Sentiment

# -------------------------------

def sentiment(score):

if score <= 2:

return 'negative'

elif score == 3:

return 'neutral'

else:

return 'positive'

df['Sentiment'] = df['Score'].apply(sentiment)

df = df[df['Sentiment'] != 'neutral'] # Optional: Drop neutral for binary classification

# -------------------------------

# STEP 4: Preprocess Text

# -------------------------------

nltk.download('stopwords')

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

stemmer = PorterStemmer()

def preprocess(text):

text = re.sub(r'[^a-zA-Z]', ' ', text) # Remove non-alphabetical characters

text = text.lower().split()

text = [stemmer.stem(word) for word in text if word not in stop\_words]

return ' '.join(text)

df['Cleaned\_Text'] = df['Text'].apply(preprocess)

# -------------------------------

# STEP 5: Visualize Data (EDA)

# -------------------------------

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

plt.title("Sentiment Distribution")

plt.show()

positive\_words = ' '.join(df[df['Sentiment']=='positive']['Cleaned\_Text'])

negative\_words = ' '.join(df[df['Sentiment']=='negative']['Cleaned\_Text'])

# Word Clouds

WordCloud(width=800, height=400, background\_color='white').generate(positive\_words).to\_image().show()

WordCloud(width=800, height=400, background\_color='black').generate(negative\_words).to\_image().show()

# -------------------------------

# STEP 6: Text Vectorization

# -------------------------------

vectorizer = TfidfVectorizer(max\_features=5000)

X = vectorizer.fit\_transform(df['Cleaned\_Text'])

y = df['Sentiment']

# -------------------------------

# STEP 7: Train/Test Split

# -------------------------------

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

# -------------------------------

# STEP 8: Model Training & Evaluation

# -------------------------------

model = MultinomialNB()

model.fit(X\_train, y\_train)

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

print("\nClassification Report:")

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

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

# -------------------------------

# STEP 9: Predict Single Review

# -------------------------------

def predict\_sentiment(text):

cleaned = preprocess(text)

vect = vectorizer.transform([cleaned])

return model.predict(vect)[0]

# Example

sample\_review = "This product is absolutely amazing. I loved it!"

print("\nSample Review Sentiment:", predict\_sentiment(sample\_review))

Top of Form

Bottom of Form