**Amazon Sentiments Analysis - Code**

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

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import nltk

import sklearn

from wordcloud import WordCloud

**# Step 1: Load Data**

import pandas as pd

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

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

df = df[df['Score'] != 3 ]

df['Sentiment'] = df['Score'].apply(lambda x: 'positive' if x > 3 else 'negative')

**# Step 2: Text Cleaning**

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

import re

nltk.download('stopwords')

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

stemmer = PorterStemmer()

def clean\_text(text):

text = re.sub(r'[^a-zA-Z]', ' ', text.lower())

words = text.split()

words = [stemmer.stem(w) for w in words if w not in stop\_words]

return ' '.join(words)

df['CleanText'] = df['Text'].apply(clean\_text)

**# Step 3: WordCloud Visualization**

from wordcloud import WordCloud

import matplotlib.pyplot as plt

**# Positive reviews**

wc\_pos = WordCloud(width=800, height=400).generate(' '.join(df[df['Sentiment'] == 'positive']['CleanText']))

plt.figure(figsize=(10,5))

plt.imshow(wc\_pos, interpolation='bilinear')

plt.axis('off')

plt.title("Positive Reviews Word Cloud")

plt.show()

**Output – Positive word cloud**

A close up of words

AI-generated content may be incorrect.

**# Negative reviews**

wc\_neg = WordCloud(width=800, height=400).generate(' '.join(df[df['Sentiment'] == 'negative']['CleanText']))

plt.figure(figsize=(10,5))

plt.imshow(wc\_neg, interpolation='bilinear')

plt.axis('off')

plt.title("Negative Reviews Word Cloud")

plt.show()

**Output – Negative word cloud**

A close-up of words

AI-generated content may be incorrect.

**# Step 4: Naive Bayes Sentiment Classifier**

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

X = df['CleanText']

y = df['Sentiment']

vectorizer = CountVectorizer()

X\_vec = vectorizer.fit\_transform(X)

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

model = MultinomialNB()

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

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

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

**Output – Classification Report**

A screenshot of a graph

AI-generated content may be incorrect.

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

**# Compute the confusion matrix**

cm = confusion\_matrix(y\_test, y\_pred, labels=['negative', 'positive'])

**# Create a ConfusionMatrixDisplay object**

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['negative', 'positive'])

**# Plot it**

disp.plot(cmap=plt.cm.Reds)

plt.title("Confusion Matrix")

plt.show()

**Output – Confusion Matrix**

A diagram of a negative and negative matrix

AI-generated content may be incorrect.