# **LAB-09**

Dated: 24-4-2025

**Exercise:**

1. Perform data pre-processing before feeding to classifier. Also do Exploratory Data Analysis

2. Apply the Naïve Bayes classifier on “movie\_reviews.csv” dataset. Determine the highest accuracy achieved and identify the hyperparameter values that contribute to this optimal performance.

**PROGRAM:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style("darkgrid")

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

df2=pd.read\_csv("/content/movie\_reviews.csv")

df2.head()

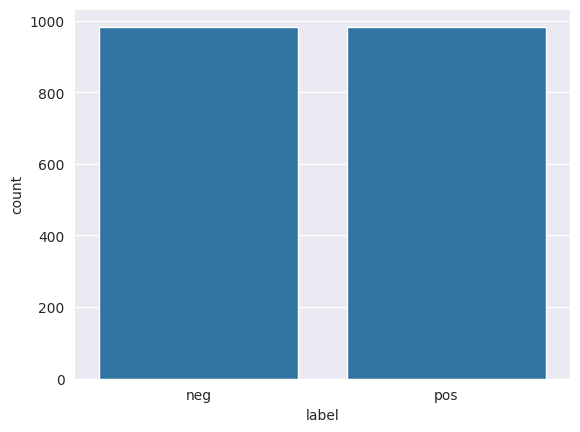
df2.info()

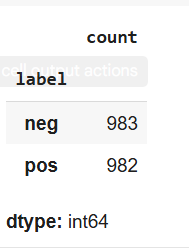
df2.isnull.sum()

df2.dropna(subset=['review'], inplace=True)

df2['label'].value\_counts()

sns.countplot(data=df2, x="label");

****



from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

cv = CountVectorizer(stop\_words='english')

matrix = cv.fit\_transform(df2[df2["label"] == 'neg']['review'])

freqs = zip(cv.get\_feature\_names\_out(), matrix.sum(axis=0).tolist()[0])

# sort from largest to smallest

print("Top 20 words used for Negative Reviews")

print(sorted(freqs, key=lambda x: -x[1])[:20])

**NAÏVE BAYES CLASSIFIER**

# Import libraries

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

from sklearn.naive\_bayes import MultinomialNB

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report,ConfusionMatrixDisplay, roc\_auc\_score, roc\_curve)

from sklearn.preprocessing import LabelBinarizer

# Feature and target

X = df2['review']

y = df2['label']

# Train-test split

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

# TF-IDF Vectorization

tfidf = TfidfVectorizer(stop\_words='english')

X\_train\_tfidf = tfidf.fit\_transform(X\_train)

X\_test\_tfidf = tfidf.transform(X\_test)

# Hyperparameter tuning using GridSearchCV

param\_grid = {

    'alpha': [0.1, 0.5, 1.0, 2.0, 5.0]

}

grid\_search = GridSearchCV(MultinomialNB(), param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)

grid\_search.fit(X\_train\_tfidf, y\_train)

# Best model

best\_nb = grid\_search.best\_estimator\_

# Predictions

y\_pred = best\_nb.predict(X\_test\_tfidf)

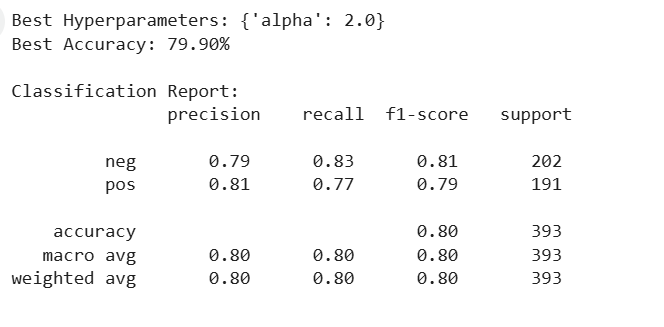
# Results

print(f"Best Hyperparameters: {grid\_search.best\_params\_}")

print(f"Best Accuracy: {accuracy\_score(y\_test, y\_pred)\*100:.2f}%")

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

OUTPUT:



3. Visualize the performance of classifier using confusion matrix, classification report,

and ROC curve.

**PROGRAM:**

# Confusion Matrix

conf\_mat = confusion\_matrix(y\_test, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=conf\_mat, display\_labels=best\_nb.classes\_)

disp.plot(cmap='Blues')

plt.title("Confusion Matrix")

plt.grid(False)

plt.show()

# ROC Curve

# Binarize the labels

lb = LabelBinarizer()

y\_test\_bin = lb.fit\_transform(y\_test)

y\_pred\_prob = best\_nb.predict\_proba(X\_test\_tfidf)[:,1]

fpr, tpr, thresholds = roc\_curve(y\_test\_bin, y\_pred\_prob)

roc\_auc = roc\_auc\_score(y\_test\_bin, y\_pred\_prob)

plt.figure(figsize=(8,6))

plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc\_auc:.2f})')

plt.plot([0,1], [0,1], 'k--')  # Diagonal

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**

