|  |  |
| --- | --- |
|  | **EXP.NO:8**  **DATE:13-09-2023** |
| **Aim:** Implement Naive Bayes Theorem to Classify the English Text using python.  **DESCRIPTION**:  **The Naive Bayes algorithm**  Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. The dataset is divided into two parts, namely, feature matrix and the response/target vector.The Feature matrix (X) contains all the vectors(rows) of the dataset in which each vector consists of the value of dependent features. The number of features is d i.e. X = (x1,x2,x2, xd).The Response/target vector (y) contains the value of class/group variable for each row of feature matrix.  **Program:**  print("NAIVE BAYES ENGLISH TEST CLASSIFICATION")  import numpy as np, pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.datasets import fetch\_20newsgroups  from sklearn.feature\_extraction.text import TfidfVectorizer  from sklearn.naive\_bayes import MultinomialNB  from sklearn.pipeline import make\_pipeline  from sklearn.metrics import confusion\_matrix, accuracy\_score  sns.set() # use seaborn plotting style  # Load the dataset  data = fetch\_20newsgroups()# Get the text categories  text\_categories = data.target\_names# define the training set  train\_data = fetch\_20newsgroups(subset="train", categories=text\_categories)# define the test set  test\_data = fetch\_20newsgroups(subset="test", categories=text\_categories)  print("We have {} unique classes".format(len(text\_categories)))  print("We have {} training samples".format(len(train\_data.data)))  print("We have {} test samples".format(len(test\_data.data)))  # let’s have a look as some training data let it 5th only  #print(test\_data.data[5])  # Build the model  model = make\_pipeline(TfidfVectorizer(), MultinomialNB())# Train the model using the training data  model.fit(train\_data.data, train\_data.target)# Predict the categories of the test data  predicted\_categories = model.predict(test\_data.data)  print(np.array(test\_data.target\_names)[predicted\_categories])  # plot the confusion matrix  mat = confusion\_matrix(test\_data.target, predicted\_categories)  sns.heatmap(mat.T, square = True, annot=True, fmt = "d", xticklabels=train\_data.target\_names,yticklabels=train\_data.target\_names)  plt.xlabel("true labels")  plt.ylabel("predicted label")  plt.show()  print("The accuracy is {}".format(accuracy\_score(test\_data.target,predicted\_categories)))  **Output:** | |