Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

BE-ETRX B	**Sub- AIML Lab**
Name: Shubham Sawant	**UID :2019110050**

Name of the Experiment: **Implement the Naïve-Bayes classifier**

Objective: Implement the naïve Bayesian Classifier model to classify a set of documents that you have assumed. Calculate the accuracy, precision, and recall for your data set.

Outcomes:

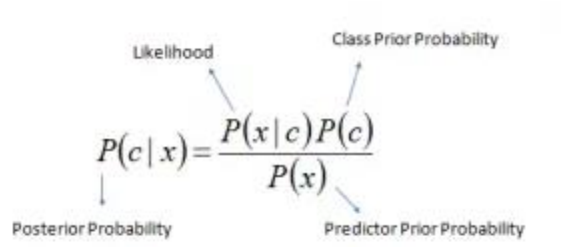
1. Find the conditional probabilities of attributes of the train data using Bayes theorem and follow the steps of the algorithm.
2. Apply the Naïve-Bayes algorithm to classify the given documents.
3. Apply Parameter smoothing for non-occurring values of attributes while calculation.
4. Find accuracy, precision, recall of the model for test data set.

System Requirements: Windows with MATLAB

Theory:

Naive Bayes algorithm is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c\mid x)=\frac{P(x\mid c)P(c)}{P(x)}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

<u>Naive Bayes algorithm:</u>

Step 1: Convert the data set into a frequency table

Step 2: Create a likelihood table by finding the probabilities

Step 3: Calculate the posterior probability of each feature with respect to the class.

Step 4: If for a certain feature the probability evaluates to zero use feature smoothening for correction.

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

Step 5: Classify the example into the class for which the probability is highest.

<u>Performance parameters of the model :</u>

Accuracy: It is the ratio of number of correct predictions to the total number of input samples.

$$\boldsymbol{Accuracy} = \frac{No.of\ correct\ prediction}{No.of\ total\ predictions\ made}$$

Precision: Precision is defined as the fraction of the examples which are actually positive among all the examples which we predicted positive.

$$\boldsymbol{Precision} = \frac{No.of\ correct\ prediction}{No.of\ total\ returned\ predictions}$$

Recall: We define recall as, among all the examples that actually positive, what fraction did we detect as positive?

$$\boldsymbol{Recall} = \frac{No.of\ correct\ prediction}{No.of\ actual\ correct\ values}$$

F1-score: F1 Score is the Harmonic Mean between precision and recall.

$$\boldsymbol{Precision} = \frac{2\ x\ Precision\ x\ Recall}{Precision\ +\ Recall}$$

Confusion Matrix: Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model.

There are 4 important terms :

- True Positives: The cases in which we predicted YES and the actual output was also YES.
- True Negatives: The cases in which we predicted NO and the actual output was NO.
- False Positives: The cases in which we predicted YES and the actual output was NO.
- False Negatives: The cases in which we predicted NO and the actual output was YES.

Data Set Link: https://archive.ics.uci.edu/ml/machine-learning-databases/00193/CTG.xls

Dataset Description:

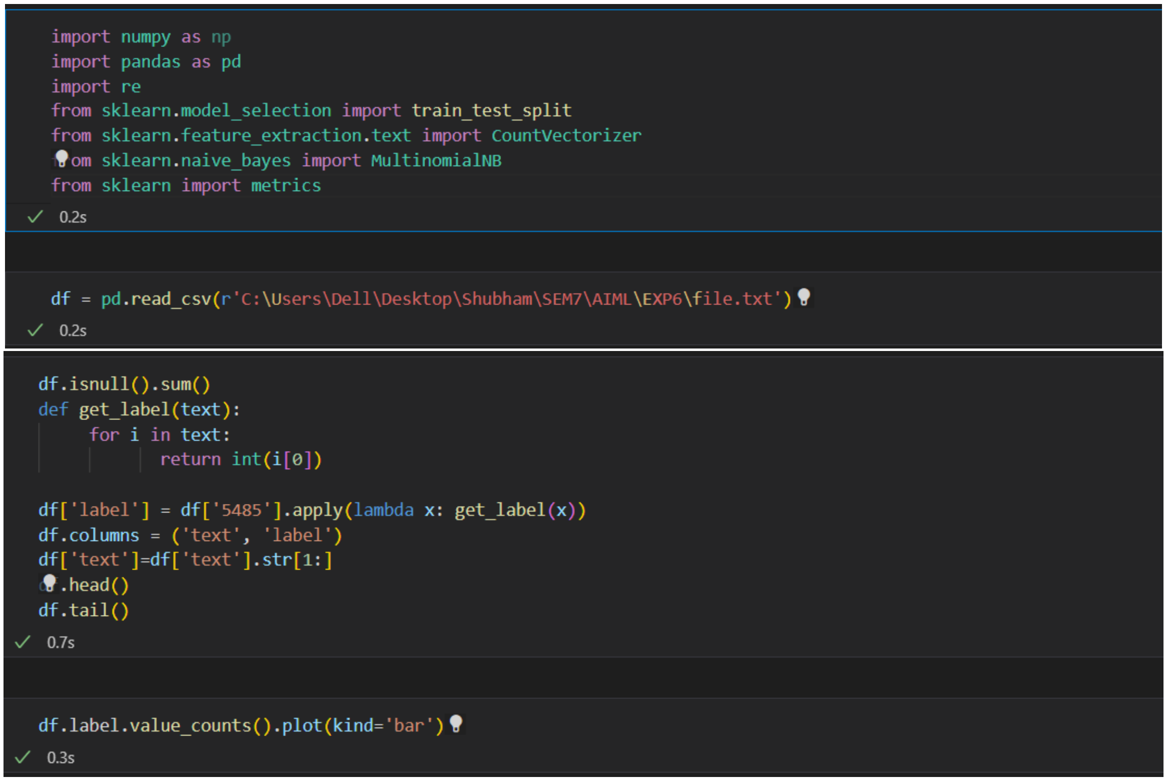
Number of Instances: 5485

Number of Attributes: 2

Attribute Information:
The dataset contains only one column of text.
The column has lables in it with in the range 1 to 8.

Code:

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**



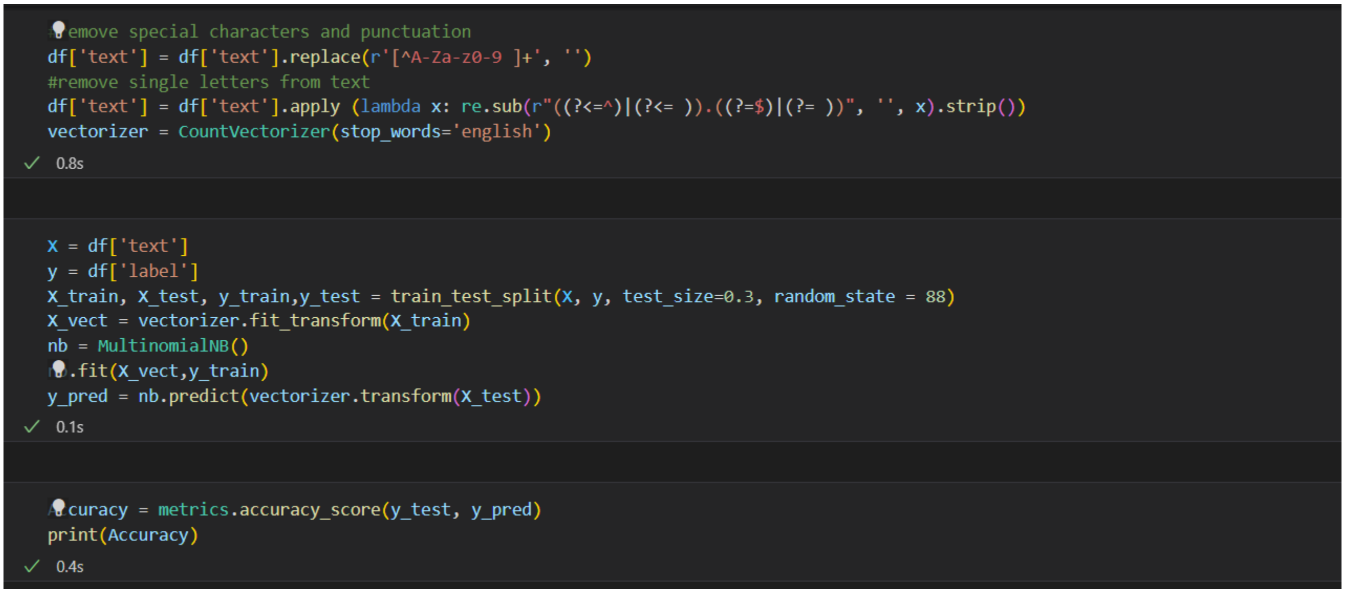
```
import numpy as np
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```
✓ 0.2s

```
df = pd.read_csv(r'C:\Users\Dell\Desktop\Shubham\SEM7\AIML\EXP6\file.txt')
```
✓ 0.2s

```
df.isnull().sum()
def get_label(text):
    for i in text:
        return int(i[0])

df['label'] = df['5485'].apply(lambda x: get_label(x))
df.columns = ('text', 'label')
df['text']=df['text'].str[1:]
df.head()
df.tail()
```
✓ 0.7s

```
df.label.value_counts().plot(kind='bar')
```
✓ 0.3s

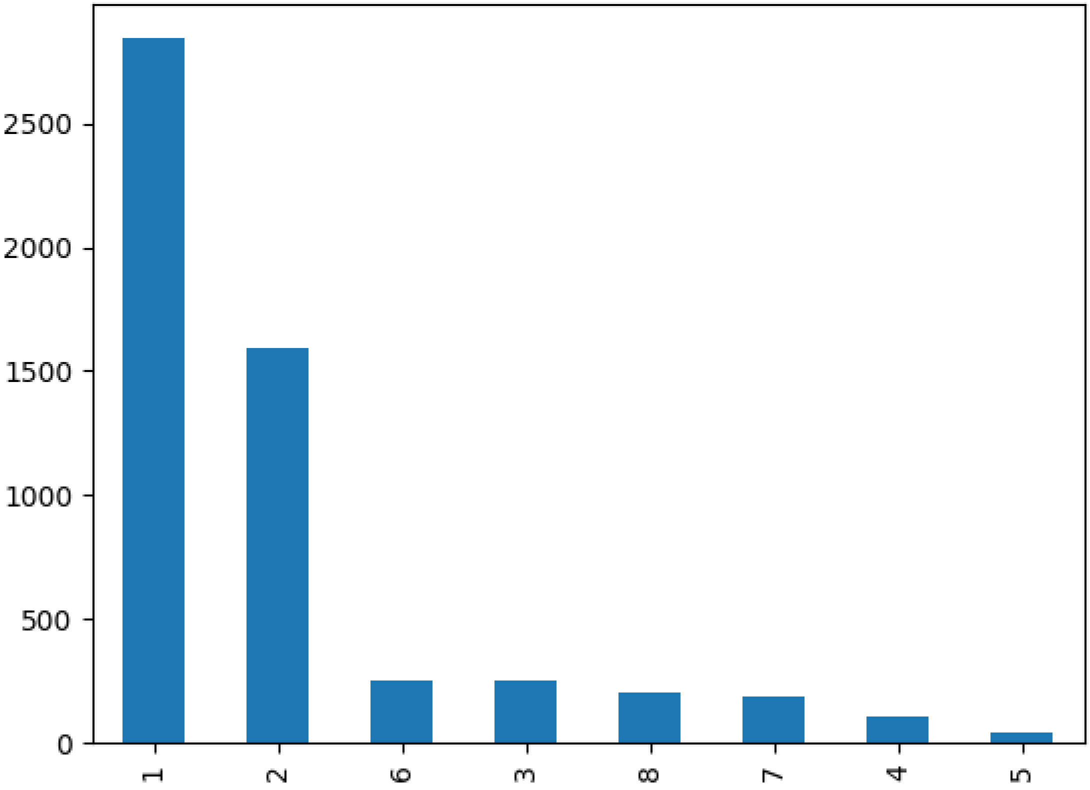


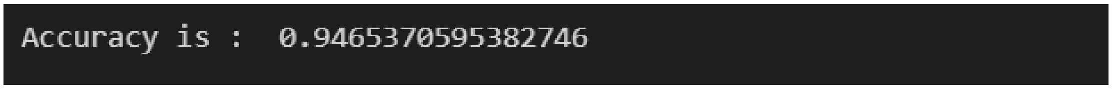
```
#remove special characters and punctuation
df['text'] = df['text'].replace(r'[^A-Za-z0-9 ]+', '')
#remove single letters from text
df['text'] = df['text'].apply (lambda x: re.sub(r"((?<=^)|(?<= )).((?=$)|(?= ))", '', x).strip())
vectorizer = CountVectorizer(stop_words='english')
```
✓ 0.8s

```
X = df['text']
y = df['label']
X_train, X_test, y_train,y_test = train_test_split(X, y, test_size=0.3, random_state = 88)
X_vect = vectorizer.fit_transform(X_train)
nb = MultinomialNB()
nb.fit(X_vect,y_train)
y_pred = nb.predict(vectorizer.transform(X_test))
```
✓ 0.1s

```
Accuracy = metrics.accuracy_score(y_test, y_pred)
print(Accuracy)
```
✓ 0.4s

Interpretation of output:





```
Accuracy is :  0.9465370595382746
```

Interpretation:

- There are 1 input features each with different units having values in different ranges. Hence, we need to standardize and normalize the data for better performance.
- The number of positive labels is more than negative labels. The data has slight imbalance.
- Due to this imbalance, the model might fail to correctly classify negative examples. But accuracy is high and training and test set scores are almost equal. Hence there is no overfitting.
- The class follows a gaussian distribution.

Conclusion:

- Naïve Bayes is fast and easy to implement but its biggest disadvantage is that the requirement of predictors to be independent.
- Naïve Bayes performs very well on high dimensional data like text or documents or even images.
- This is a probability-based classifier which uses Bayes theorem for classification.