Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2022-2023)
EC344 - Machine Learning and AI

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

**Name:** Anshal Padole **Roll No:** 2019120043 **Date:**3-11-2022

**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 the test data set.

**System Requirements:** Linux OS with Python and libraries or R or 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:

$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

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

- 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.

**Naive Bayes algorithm:**

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.

**Performance parameters of the model :**

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

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

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

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

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

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

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

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

**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.

**<u>Dataset:</u>**

| anaemia | diabetes | high_blood_pressure | sex | smoking | DEATH_EVENT |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |

**Dataset link:**
https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data?select=heart_failure_clinical_records_dataset.csv

**Number of Instances:** 20
**Number of Attributes:** 5
**Attribute Information:**

1. Anemia: Decrease of red blood cells or hemoglobin. 1 indicates decrease in RBCs and 0 indicates normal RBCs.
2. Diabetes: 1 indicates patient has diabetes and 0 indicates patient does not have diabetes.
3. High blood pressure: 1 indicates patient has high blood pressure and 0 indicates patient does not have high blood pressure.
4. Sex: 0 indicates female and 1 indicates male.
5. Smoking: 0 indicates non-smoker and 1 indicates smoker.
6. Death event: 1 indicates is dead and 0 indicates is alive.

**Code:**

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization purposes
import seaborn as sns # for statistical data visualization
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('C:/Users/Lenovo/Downloads/data2.csv')
df.head()

X = df.drop(['DEATH_EVENT'], axis=1)
y = df['DEATH_EVENT']

# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

cols = X_train.columns

from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns=[cols])
X_test = pd.DataFrame(X_test, columns=[cols])

from sklearn.naive_bayes import GaussianNB
# instantiate the model
gnb = GaussianNB()
# fit the model
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
```

```
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
# print the scores on training and test set

print('Training set score: {:.4f}'.format(gnb.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(gnb.score(X_test, y_test)))

# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])

# visualize confusion matrix with seaborn heatmap
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                         index=['Predict Positive:1', 'Predict Negative:0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
recall= TP/ float( TP+FN)
print('Recall: {0:0.4f}'.format(recall))
```

**Output**:

Model accuracy score: 0.8333

Training set score: 0.9286

Test set score: 0.8333

Confusion matrix

[[1 0]
[1 4]]

True Positives(TP) = 1

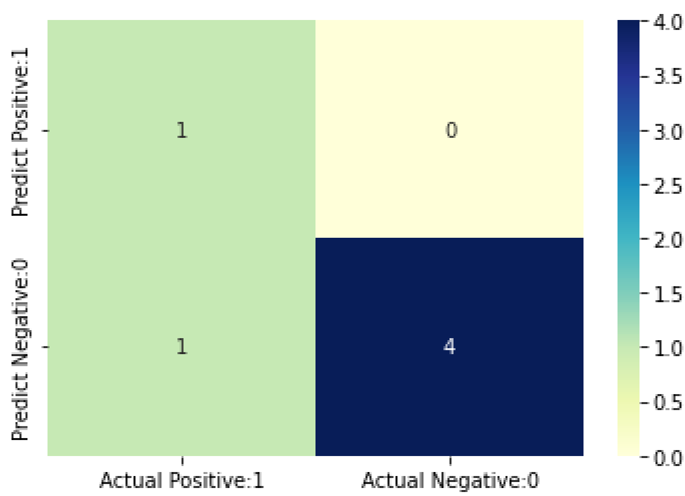
True Negatives(TN) = 4

False Positives(FP) = 0

False Negatives(FN) = 1

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.50 | 1.00 | 0.67 | 1 |
| 1 | 1.00 | 0.80 | 0.89 | 5 |
| accuracy | | | 0.83 | 6 |
| macro avg | 0.75 | 0.90 | 0.78 | 6 |
| weighted avg | 0.92 | 0.83 | 0.85 | 6 |

Recall: 0.5000



To improve the performance of a Naive Bayes Classifier

- Use probabilities for feature selection
- Remove zero observations problem by using feature smoothening
- Remove redundant features

**Conclusion:**

- We learned how the Naive Bayes classifier uses posterior probability and feature smoothing to classify an example into a class.
- Using Sci-Kit we feature engineered the dataset creating a feature vector and count vector to determine the frequency of each word in the documents.
- We build the Naive Bayes model using the Multinomial classifier and generated the performance report of the classifier for calculating accuracy, precision, recall and creating the confusion matrix.