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)

Academic SEM: VII **Year: 2022-23**

Experiment: Implement the Naïve-Bayes classifier

Name: Darshan Somani **Date: 15, Nov 2022**

Objective:To explore the multi-layer perceptron algorithm using back-propagation

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: 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:

Likelihood Class Prior Probability

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

Posterior Probability 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.

<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.

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

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 :

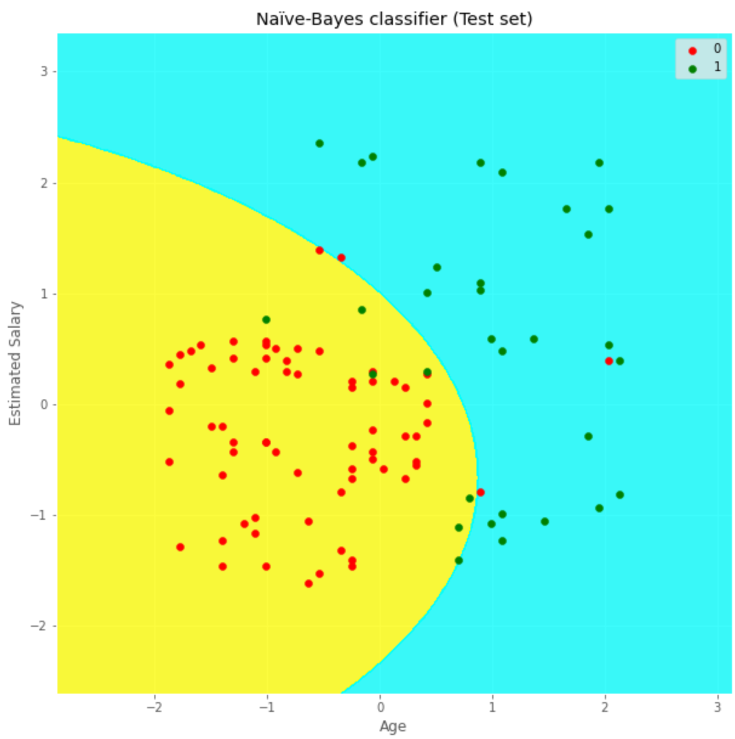
- True Positives: The cases in which we predicted YES and the actual output was also YES.
- True Negatives: The cases in which we predicted NO and the actual output was NO.
- False Positives: The cases in which we predicted YES and the actual output was NO.
- False Negatives: The cases in which we predicted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

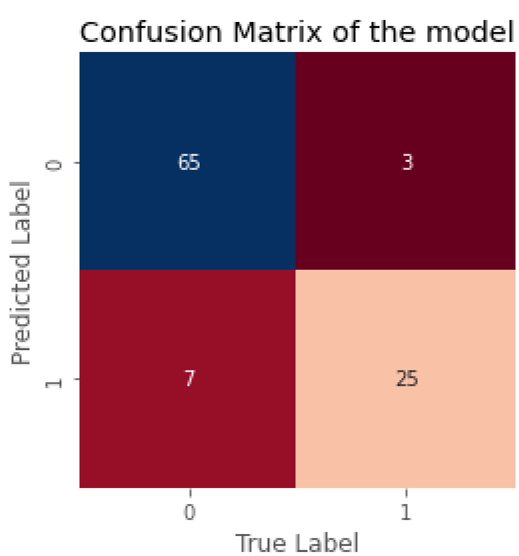
Code

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Importing the dataset
dataset = pd.read_csv('/content/drive/MyDrive/data/SPAM text message 20170820 -
Data.csv')
X = data.loc[:,['Age','EstimatedSalary']]
y = data.iloc[:,-1]
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
 random_state = 0)
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,
0].max() + 1, step = 0.01),
                        np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,
1].max() + 1, step = 0.01))

plt.figure(figsize=(10,10))
plt.contourf(X1,          X2,          classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('yellow', 'cyan')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naïve-Bayes classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, square=True, annot=True, cmap='RdBu', cbar=False,
xticklabels=['0', '1'], yticklabels=['0', '1'])
plt.title("Confusion Matrix of the model")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')
total = cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1]
accuracy = (cm[0][0]+cm[1][1])/total
error_rate = (cm[0][1]+cm[1][0])/total
precision = cm[1][1]/(cm[0][1]+cm[1][1])
recall = cm[1][1]/(cm[1][0]+cm[1][1])
```

```
f1_score = 2*((precision*recall)/(precision+recall))
print('Accuracy of the model is %.2f%%' %(accuracy*100))
print('Error rate of the model is %.2f%%' %(error_rate*100))
print('Precision of the model is %.2f%%' %(precision*100))
print('Recall of the model is %.3f%%' %(recall*100))
print('F1 Score of the model is %.2f%%' %(f1_score*100))
```

Output:





```
Accuracy of the model is 90.00%
Error rate of the model is 10.00%
Precision of the model is 89.29%
Recall of the model is 78.125%
F1 Score of the model is 83.33%
```

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 built 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.