EXPERIMENT NO. 4

Date of Performance:

Date of Submission:

AIM: Implementation of Bayesian Classification Algorithm.

Software used: Java / C/ Python

Theory:

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c).

Likelihood Class Prior Probability

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

Posterior Probability Predictor Prior Probability

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

Above, a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

- 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.
- **Step 3**: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.
- **Problem:** Players will play if weather is sunny. Is this statement correct?

  We can solve it using above discussed method of posterior probability.

- P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)
- Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64
- Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

  Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Advantages:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

PROGRAM and OUTPUT:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
[ ] from google.colab import files
    uploaded = files.upload()

    Choose Files  No file chosen        Up
    Saving Facts.csv to Facts.csv
```

```
[ ] from pandas.core.internals.construction import dataclasses_to_dicts
    import io
    df = pd.read_csv(io.BytesIO(uploaded['Facts.csv']))
    x = df.iloc[:,:7].values
    y = df['Equipments_inv']
    df.head(5)
```

	f_id	sr_no	User_id	Reg_id	Card_no	Sol_no	Vacant_rooms	Equipments required	Equipments_inv	Mediclaim Diversification
0	10	1	9206	5728	65229335	898	36	SPECIALIZED	15	6
1	20	2	8901	5496	93720416	461	33	BASIC	14	9
2	30	3	5967	3814	25854311	954	25	BASIC	23	7
3	40	4	1751	976	41381338	545	10	ADVANCE	17	7
4	50	5	7931	5967	32133322	191	46	SPECIALIZED	20	6

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

```
[ ] from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    x_train = sc.fit_transform(x_train)
    x_test = sc.transform(x_test)
```

```
[ ] from sklearn.naive_bayes import GaussianNB
    classifier = GaussianNB()
    classifier.fit(x_train, y_train)

    GaussianNB()
```

```
[ ] y_pred = classifier.predict(x_test)
    y_pred

    array([27, 20, 27, 27])
```

```
[ ] from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test,y_pred)
    from sklearn.metrics import accuracy_score
    print("Accuracy: ",accuracy_score(y_test,y_pred))
    cm

    Accuracy:  0.0
    array([[0, 0, 0, 0, 1, 0],
           [0, 0, 0, 0, 0, 0],
           [0, 1, 0, 0, 0, 0],
           [0, 0, 0, 0, 1, 0],
           [0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 1, 0]])

[ ] df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
    df
```

	Real Values	Predicted Values
7	28	27
12	21	20
14	24	27
0	15	27

CONCLUSION: Thus implemented Bayesian Classification Algorithm.

SIGN AND REMARK

DATE

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	