Practical No. 9 (b)

Title: Bayesian approach in data mining

Aim: Classification of data using Bayesian approach.

Software required: GaussianNB classifier from the sklearn.naive_bayes module in python.

Theory:

Classification using the Bayesian approach involves using Bayesian statistics to predict the probability of a given data instance belonging to a particular class. The Bayesian classification technique is based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions related to the event.

Bayes' Theorem:

Bayes Theorem Formula

If A and B are two events, then the **formula for the Bayes theorem** is given by:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad where \quad P(B) \neq 0$$

Where P(A|B) is the probability of condition when event A is occurring while event B has already occurred.

Bayes Theorem Derivation

Bayes Theorem can be derived for events and random variables separately using the definition of conditional probability and density.

From the definition of conditional probability, Bayes theorem can be derived for events as given below:

P(A|B) = P(A ∩ B)/ P(B), where P(B) ≠ 0

P(B|A) = P(B ∩ A)/ P(A), where P(A) ≠ 0

Here, the joint probability P(A ∩ B) of both events A and B being true such that,

P(B ∩ A) = P(A ∩ B)

P(A ∩ B) = P(A | B) P(B) = P(B | A) P(A)

P(A|B) = [P(B|A) P(A)]/ P(B), where P(B) ≠ 0

Similarly, from the definition of conditional density, Bayes theorem can be derived for two continuous random variables namely X and Y as given below:

$$f_{X|Y=y}(x) = \frac{f_{X,Y(x,y)}}{f_Y(y)}$$
$$f_{Y|X=x}(y) = \frac{f_{X,Y(x,y)}}{f_X(x)}$$

Therefore,

$$f_{X|Y=y}(x) = \frac{f_{Y|X=x}(y) f_X(x)}{f_Y(y)}$$

Example 1:

A bag I contains 4 white and 6 black balls while another Bag II contains 4 white and 3 black balls. One ball is drawn at random from one of the bags, and it is found to be black. Find the probability that it was drawn from Bag I.

Solution:

Let E_1 be the event of choosing bag I, E_2 the event of choosing bag II, and A be the event of drawing a black ball.

Then,

$$P(E_1) = P(E_2) = \frac{1}{2}$$

Also, $P(A|E_1)$ = P(drawing a black ball from Bag I) = 6/10 = 3/5

$P(A|E_2)$ = P(drawing a black ball from Bag II) = 3/7

By using Bayes' theorem, the probability of drawing a black ball from bag I out of two bags,

$$P(E_1|A) = \frac{P(E_1)P(A|E_1)}{P(E_1)P(A|E_1)+P(E_2)P(A|E_2)}$$

$$= \frac{\frac{1}{2} \times \frac{3}{5}}{\frac{1}{2} \times \frac{3}{5} + \frac{1}{2} \times \frac{3}{7}}$$

$$= \frac{7}{12}$$

Example 2:

A man is known to speak the truth 2 out of 3 times. He throws a die and reports that the number obtained is a four. Find the probability that the number obtained is actually a four.

Solution:

Let A be the event that the man reports that number four is obtained.

Let E_1 be the event that four is obtained and E_2 be its complementary event.

Then, $P(E_1)$ = Probability that four occurs = 1/6.

$P(E_2)$ = Probability that four does not occur = 1- $P(E_1)$ = 1 – (1/6) = 5/6.

Also, $P(A|E_1)$= Probability that man reports four and it is actually a four = 2/3

$P(A|E_2)$ = Probability that man reports four and it is not a four = 1/3.

By using Bayes' theorem, probability that number obtained is actually a four, $P(E_1|A)$

$$= \frac{P(E_1)P(A|E_1)}{P(E_1)P(A|E_1) + P(E_2)P(A|E_2)} = \frac{\frac{1}{6} \times \frac{2}{3}}{\frac{1}{6} \times \frac{2}{3} + \frac{5}{6} \times \frac{1}{3}}$$

$$= \frac{2}{7}$$

Types of Bayesian Classifiers:

1. **Naive Bayes Classifier**: The Naive Bayes classifier assumes that the features (predictor variables) are conditionally independent given the class label. Despite this simplifying assumption, Naive Bayes often performs well in practice and is computationally efficient.
2. **Gaussian Naive Bayes**: This variant of Naive Bayes assumes that continuous features follow a Gaussian (normal) distribution.
3. **Multinomial Naive Bayes**: Suitable for discrete data and assumes that features follow a multinomial distribution.
4. **Bernoulli Naive Bayes**: Designed for binary/boolean features and assumes that features are binary-valued.

Applications:

Bayesian classification is widely used in various domains, including:

- **Email Spam Filtering**: Classifying emails as spam or non-spam based on features such as the presence of certain keywords, sender information, etc.
- **Medical Diagnosis**: Predicting the likelihood of a patient having a particular disease based on symptoms and medical test results.
- **Document Classification**: Categorizing documents into predefined categories such as sports, politics, finance, etc.
- **Sentiment Analysis**: Analyzing text data to determine the sentiment (positive, negative, neutral) based on word frequencies and patterns.

Program/Code/queries:

Implementing the Bayesian approach in data mining involves using Bayes' theorem to classify or predict outcomes based on observed data. Below is a step-by-step guide to implementing a basic Bayesian classifier using Python and the Naive Bayes algorithm as an example.

Step 1: Import Libraries

First, import the required libraries. For this example, we'll use the GaussianNB classifier from the sklearn.naive_bayes module.

import numpy as np from sklearn.model_selection import train_test_split from sklearn.naive_bayes import GaussianNB from sklearn.metrics import accuracy_score, classification_report

Step 2: Prepare the Dataset

Create a sample dataset for binary classification. For simplicity, consider a dataset with two features (e.g., age and income) and a binary target variable (e.g., whether a person buys a product or not).

Sample dataset: age, income, buys_product data = np.array([[25, 50000, 0], [30, 70000, 0], [35, 90000, 1], [20, 30000, 0], [40, 100000, 1]]) X = data[:, :-1] # Features: age, income y = data[:, -1] # Target variable: buys_product

Step 3: Split the Dataset

Split the dataset into training and test sets to evaluate the classifier's performance.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Step 4: Train the Naive Bayes Classifier

Instantiate the Gaussian Naive Bayes classifier and fit it to the training data.

clf = GaussianNB() clf.fit(X_train, y_train)

Step 5: Make Predictions

Use the trained classifier to make predictions on the test data.

y_pred = clf.predict(X_test)

Step 6: Evaluate the Classifier

Evaluate the performance of the classifier by calculating accuracy and generating a classification report.

accuracy = accuracy_score(y_test, y_pred) print(f'Accuracy: {accuracy * 100:.2f}%') print('Classification Report:') print(classification_report(y_test, y_pred))

Output/Snapshots:

Result/Conclusion:

Above basic example demonstrates how to implement a Bayesian classifier using the Naive Bayes algorithm in Python. In practice, we can apply Bayesian approaches to more complex datasets, including text classification, medical diagnosis, and other applications where probabilistic modeling is beneficial. By leveraging Bayesian techniques, we can make informed predictions and gain insights from data by considering prior knowledge and observed evidence to make probabilistic inferences.