EXPERMINT: 04

- **Aim: Implementation of Bayesian algorithm.**
- **Theory:**

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

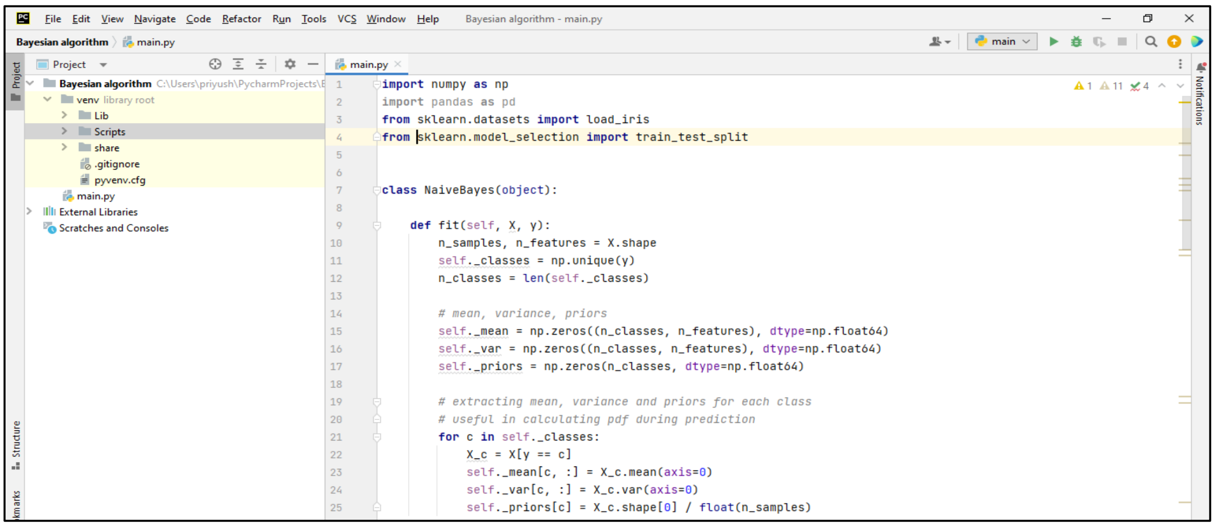
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

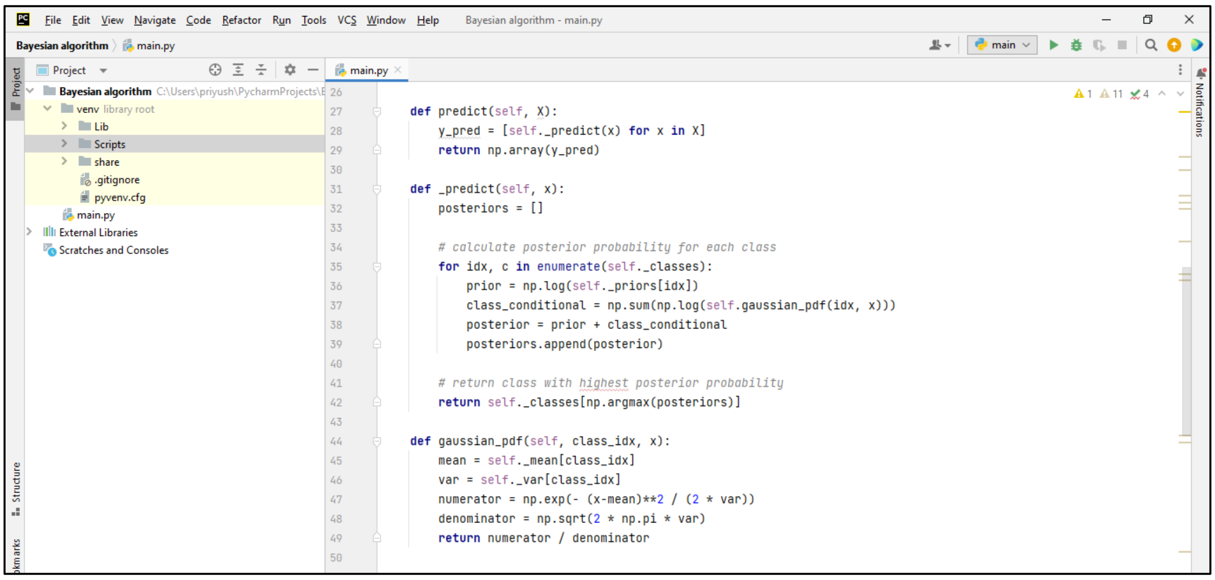
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

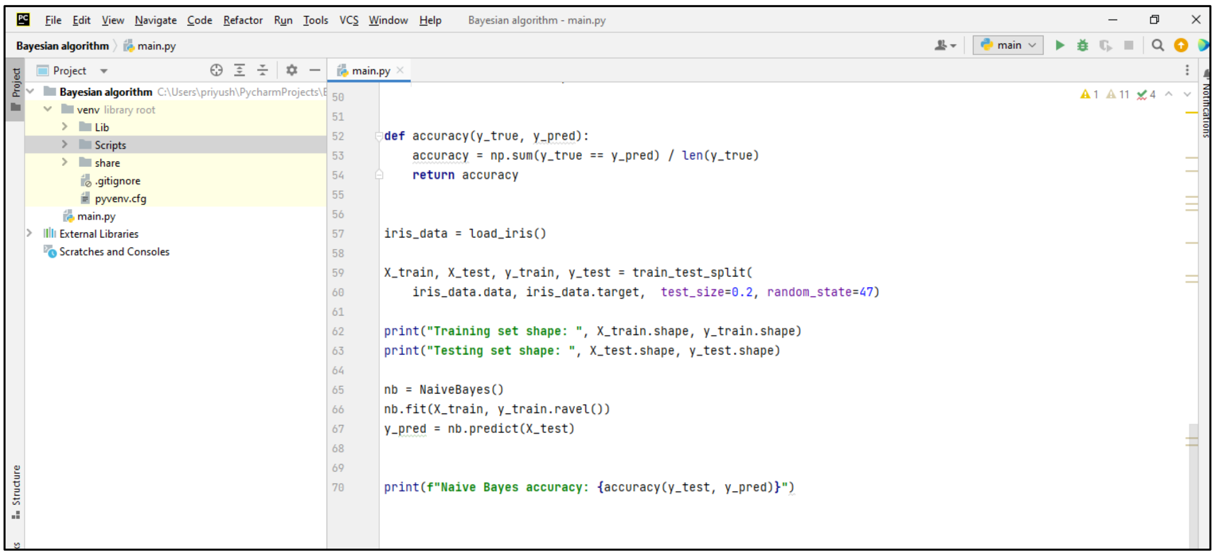
Code:


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Bayesian algorithm - main.py
Project
Bayesian algorithm C:\Users\priyush\PycharmProjects\E
venv library root
Lib
Scripts
share
.gitignore
pyvenv.cfg
main.py
External Libraries
Scratches and Consoles

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split


class NaiveBayes(object):

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self._classes = np.unique(y)
        n_classes = len(self._classes)

        # mean, variance, priors
        self._mean = np.zeros((n_classes, n_features), dtype=np.float64)
        self._var = np.zeros((n_classes, n_features), dtype=np.float64)
        self._priors = np.zeros(n_classes, dtype=np.float64)

        # extracting mean, variance and priors for each class
        # useful in calculating pdf during prediction
        for c in self._classes:
            X_c = X[y == c]
            self._mean[c, :] = X_c.mean(axis=0)
            self._var[c, :] = X_c.var(axis=0)
            self._priors[c] = X_c.shape[0] / float(n_samples)
```

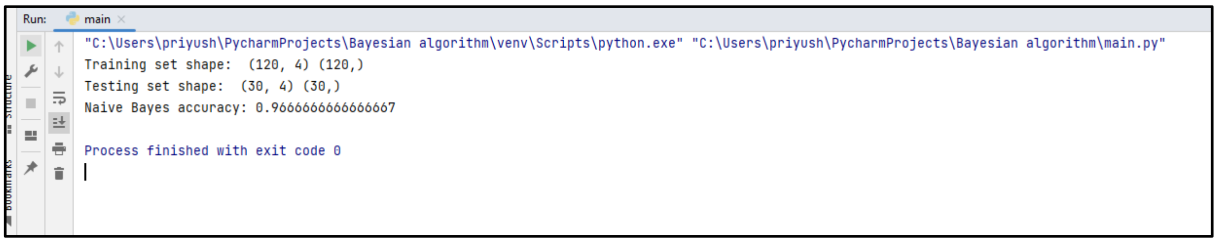


```python

    def predict(self, X):
        y_pred = [self._predict(x) for x in X]
        return np.array(y_pred)

    def _predict(self, x):
        posteriors = []

        # calculate posterior probability for each class
        for idx, c in enumerate(self._classes):
            prior = np.log(self._priors[idx])
            class_conditional = np.sum(np.log(self.gaussian_pdf(idx, x)))
            posterior = prior + class_conditional
            posteriors.append(posterior)

        # return class with highest posterior probability
        return self._classes[np.argmax(posteriors)]

    def gaussian_pdf(self, class_idx, x):
        mean = self._mean[class_idx]
        var = self._var[class_idx]
        numerator = np.exp(- (x-mean)**2 / (2 * var))
        denominator = np.sqrt(2 * np.pi * var)
        return numerator / denominator

```



```python


def accuracy(y_true, y_pred):
    accuracy = np.sum(y_true == y_pred) / len(y_true)
    return accuracy


iris_data = load_iris()

X_train, X_test, y_train, y_test = train_test_split(
    iris_data.data, iris_data.target,  test_size=0.2, random_state=47)

print("Training set shape: ", X_train.shape, y_train.shape)
print("Testing set shape: ", X_test.shape, y_test.shape)

nb = NaiveBayes()
nb.fit(X_train, y_train.ravel())
y_pred = nb.predict(X_test)


print(f"Naive Bayes accuracy: {accuracy(y_test, y_pred)}")
```

Output :



```
"C:\Users\priyush\PycharmProjects\Bayesian algorithm\venv\Scripts\python.exe" "C:\Users\priyush\PycharmProjects\Bayesian algorithm\main.py"
Training set shape:  (120, 4) (120,)
Testing set shape:  (30, 4) (30,)
Naive Bayes accuracy: 0.9666666666666667

Process finished with exit code 0
```

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for text classification problems.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for Credit Scoring.
- It is used in medical data classification.
- It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as Spam filtering and Sentiment analysis.

● Conclusion:

We implemented Naive **Bayes algorithms** are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They are fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent.