Experiment # 10: Implement the Naïve Bayes classifier using the Iris dataset to predict the label for unknown data

Aim/Objective: Implementing the Naïve Bayes classifier using the Iris dataset to predict the label for unknown data

Description: Students will implement the Naïve Bayes classifier works on Bayes' theorem of probability to predict the class of unknown data sets. This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.

Pre-Requisites:

Basic programming knowledge, Understanding of Bayes Theorem.

Pre-Lab:

1. What is the Naive Bayes Classification Algorithm? What are the steps to implement a Naive Bayes classifier?

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming feature independence.

Steps:

1. Prepare and preprocess data.
2. Compute prior probabilities for each class.
3. Calculate likelihood using conditional probability.
4. Apply Bayes' Theorem to get posterior probabilities.
5. Classify new data based on the highest probability.

2. What are the Different Types of Naive Bayes Models?

1. **Gaussian Naive Bayes** – Assumes normal distribution (for continuous data).
2. **Multinomial Naive Bayes** – Used for text classification (word counts, TF-IDF).
3. **Bernoulli Naive Bayes** – Works with binary features (spam detection).

3. What are the applications of the Naive Bayes Classification Algorithm?

- **Spam filtering** (email classification).
- **Sentiment analysis** (positive/negative reviews).
- **Medical diagnosis** (disease prediction).
- **Text classification** (news, articles).
- **Fraud detection** (banking, transactions).

In-Lab:

Implement the Naïve bayes classifier on Buys computers dataset to predict an unknown sample.

__Description:__ The Naive Bayes classifier is implemented using the scikit-learn library by importing the standard ***Buys Computers*** dataset to predict whether customer purchases a computer or not.
Procedure/Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

data = {

   'Age': ['<=30', '<=30', '31...40', '>40', '>40', '>40', '31...40', '<=30', '<=30', '>40',
'<=30', '31...40', '31...40', '>40'],

   'Income': ['High', 'High', 'High', 'Medium', 'Low', 'Low', 'Low', 'Medium', 'Low',
'Medium', 'Medium', 'Medium', 'High', 'Medium'],

   'Student': ['No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'No',
'Yes'],

   'Credit_Rating': ['Fair', 'Excellent', 'Fair', 'Fair', 'Fair', 'Excellent', 'Excellent', 'Fair',
'Fair', 'Excellent', 'Excellent', 'Fair', 'Excellent', 'Fair'],

   'Buys_Computer': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
'Yes', 'Yes', 'No']
}

df = pd.DataFrame(data)

encoder = LabelEncoder()
for column in df.columns:
   df[column] = encoder.fit_transform(df[column])
```

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

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

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

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
print('Classification Report:\n', classification_report(y_test, y_pred))

sample = pd.DataFrame([[0, 2, 1, 0]], columns=['Age', 'Income', 'Student',
'Credit_Rating'])

prediction = model.predict(sample)
print('Predicted class for unknown sample:', prediction[0])
```

Data and Results:

Data

The dataset consists of customer attributes for predicting computer purchases.

Result

The Naïve Bayes classifier predicts customer purchasing behavior accurately.

Analysis and Inferences:

Analysis

The model is evaluated using accuracy and a classification report.

Inferences

Higher accuracy suggests effective classification of purchasing decisions.

Sample VIVA-VOCE Questions (In-Lab):

1. How does the Naïve Bayes classifier work, and what assumptions does it make?

- A probabilistic algorithm based on **Bayes' theorem.**
- Assumes **independence** between features.
- Works well for **text classification** (e.g., spam detection).

2. What steps do you follow to implement the Naïve Bayes classifier using the Iris dataset?

- Import dataset (`datasets.load_iris()`).
- Split data (`train_test_split()`).
- Train model using `GaussianNB()` .
- Predict and evaluate (`accuracy_score()`).

3. Explain the K-Nearest Neighbors (KNN) algorithm and how it is used for breast cancer detection.

- KNN classifies data based on **neighboring points** (majority voting).
- Used for **breast cancer detection** using the **Wisconsin dataset.**
- Distance metric: **Euclidean distance.**

4. What are the key differences between the Naïve Bayes and KNN classifiers?

- **Naïve Bayes:** Probabilistic, fast, assumes feature independence.
- **KNN:** Distance-based, slow, no assumptions, better for complex patterns.

5. How do you evaluate the performance of your classifiers for predicting unknown data?

- **Classification:** Accuracy, Precision, Recall, F1-score, Confusion Matrix.
- **Python Example:** `classification_report(y_test, y_pred)`.

Post-Lab:

Implement Breast Cancer detection system using KNN

Description: Implementing a Breast Cancer detection system using the K-Nearest Neighbors (KNN) algorithm involves using a dataset of breast cancer features to classify whether a tumor is benign or malignant based on its characteristics.

Procedure/Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

data = load_breast_cancer()
X = data.data
y = data.target

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
```

```
print('Classification Report:\n', classification_report(y_test, y_pred))
print('Confusion Matrix:\n', confusion_matrix(y_test, y_pred))

k_values = range(1, 21)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies.append(accuracy_score(y_test, y_pred))

plt.figure(figsize=(10, 5))
plt.plot(k_values, accuracies, marker='o', linestyle='dashed', color='b')
plt.xlabel('Number of Neighbors (k)')

plt.ylabel('Accuracy')
plt.title('KNN Accuracy for Different k Values')
plt.show()
```

Data and Results:

Data

Breast cancer dataset contains tumor features for classification analysis.

Result

KNN classifier achieved high accuracy in predicting tumor categories.

Analysis and Inferences:

Analysis

Different values of k affect model accuracy and classification performance.

Inferences

KNN effectively distinguishes between benign and malignant breast tumors.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date