```
#EXPERIMENT-7
#Write a program to implement the naïve Bayesian classifier for the given dataset and compute the accuracy of the classifier.
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Load the 'adult.csv' dataset
data = pd.read_csv('adult.csv')
```

Loading...

```
# Data preprocessing
# Encode categorical variables using LabelEncoder
label_encoder = LabelEncoder()
categorical_columns = data.select_dtypes(include=['object']).columns
for col in categorical_columns:
    data[col] = label_encoder.fit_transform(data[col])

# Define features (X) and the target variable (y)
X = data.drop('income', axis=1)  # 'income' is the target column
y = data['income']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Naïve Bayes classifier (Gaussian Naïve Bayes for continuous features)
nb_classifier = GaussianNB()

# Train the classifier on the training data
nb_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = nb_classifier.predict(X_test)

# Compute the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# You can also print other classification metrics if needed
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.80
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.95      0.88      7479
           1       0.65      0.31      0.42      2290

    accuracy                           0.80      9769
   macro avg       0.73      0.63      0.65      9769
weighted avg       0.78      0.80      0.77      9769
```

```
#EXPERIMENT-8
#Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions.
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a k-Nearest Neighbors classifier
k = 3  # You can adjust the value of k
knn_classifier = KNeighborsClassifier(n_neighbors=k)

# Train the classifier on the training data
```

```
knn_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn_classifier.predict(X_test)

# Compute the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Print both correct and wrong predictions
correct_predictions = []
wrong_predictions = []
```

Loading...

```
for i in range(len(X_test)):
    if y_pred[i] == y_test[i]:
        correct_predictions.append((X_test[i], y_test[i], y_pred[i]))
    else:
        wrong_predictions.append((X_test[i], y_test[i], y_pred[i]))

print("\nCorrect Predictions:")
for x, true_label, pred_label in correct_predictions:
    print(f"True: {true_label}, Predicted: {pred_label}, Input: {x}")

print("\nWrong Predictions:")
for x, true_label, pred_label in wrong_predictions:
    print(f"True: {true_label}, Predicted: {pred_label}, Input: {x}")

# You can also print other classification metrics if needed
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.00

Correct Predictions:
True: 1, Predicted: 1, Input: [6.1 2.8 4.7 1.2]
True: 0, Predicted: 0, Input: [5.7 3.8 1.7 0.3]
True: 2, Predicted: 2, Input: [7.7 2.6 6.9 2.3]
True: 1, Predicted: 1, Input: [6.  2.9 4.5 1.5]
True: 1, Predicted: 1, Input: [6.8 2.8 4.8 1.4]
True: 0, Predicted: 0, Input: [5.4 3.4 1.5 0.4]
True: 1, Predicted: 1, Input: [5.6 2.9 3.6 1.3]
True: 2, Predicted: 2, Input: [6.9 3.1 5.1 2.3]
True: 1, Predicted: 1, Input: [6.2 2.2 4.5 1.5]
True: 1, Predicted: 1, Input: [5.8 2.7 3.9 1.2]
True: 2, Predicted: 2, Input: [6.5 3.2 5.1 2. ]
True: 0, Predicted: 0, Input: [4.8 3.  1.4 0.1]
True: 0, Predicted: 0, Input: [5.5 3.5 1.3 0.2]
True: 0, Predicted: 0, Input: [4.9 3.1 1.5 0.1]
True: 0, Predicted: 0, Input: [5.1 3.8 1.5 0.3]
True: 1, Predicted: 1, Input: [6.3 3.3 4.7 1.6]
True: 2, Predicted: 2, Input: [6.5 3.  5.8 2.2]
True: 1, Predicted: 1, Input: [5.6 2.5 3.9 1.1]
True: 1, Predicted: 1, Input: [5.7 2.8 4.5 1.3]
True: 2, Predicted: 2, Input: [6.4 2.8 5.6 2.2]
True: 0, Predicted: 0, Input: [4.7 3.2 1.6 0.2]
True: 2, Predicted: 2, Input: [6.1 3.  4.9 1.8]
True: 0, Predicted: 0, Input: [5.  3.4 1.6 0.4]
True: 2, Predicted: 2, Input: [6.4 2.8 5.6 2.1]
True: 2, Predicted: 2, Input: [7.9 3.8 6.4 2. ]
True: 2, Predicted: 2, Input: [6.7 3.  5.2 2.3]
True: 2, Predicted: 2, Input: [6.7 2.5 5.8 1.8]
True: 2, Predicted: 2, Input: [6.8 3.2 5.9 2.3]
True: 0, Predicted: 0, Input: [4.8 3.  1.4 0.3]
True: 0, Predicted: 0, Input: [4.8 3.1 1.6 0.2]

Wrong Predictions:

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Loading...