LAB # 04

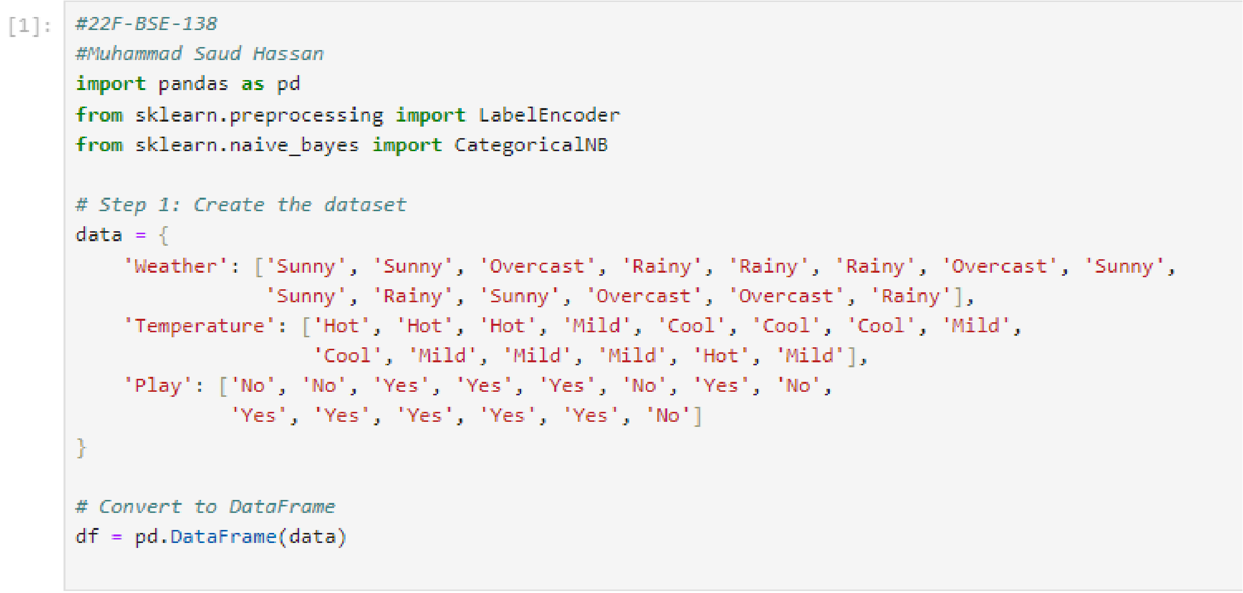
SUPERVISED LEARNING (NAÏVE BAYES ALGORITHM)

OBJECTIVE

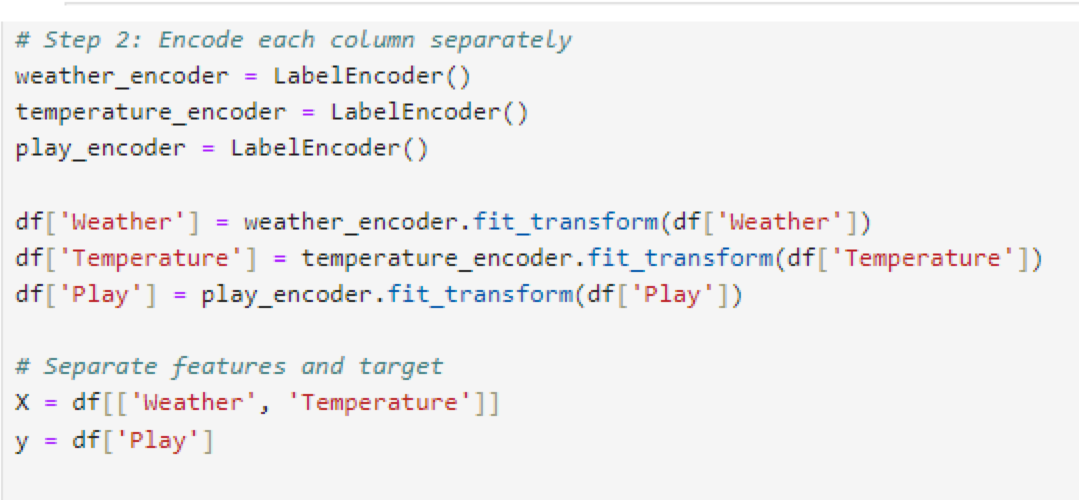
Implementing supervised learning, Naïve Bayes algorithm for training, testing and classification

Lab Tasks:

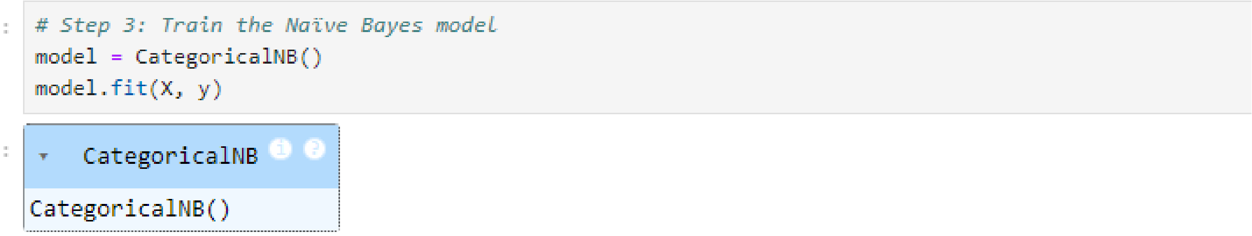
1. Implement Naïve Bayes Algorithm on the given dataset in Fig 1 to predict whether the players can play or not when the weather is overcast and the temperature is mild.



```
[1]: #22F-BSE-138
     #Muhammad Saud Hassan
     import pandas as pd
     from sklearn.preprocessing import LabelEncoder
     from sklearn.naive_bayes import CategoricalNB

     # Step 1: Create the dataset
     data = {
         'Weather': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny',
                     'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy'],
         'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild',
                         'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],
         'Play': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',
                  'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
     }

     # Convert to DataFrame
     df = pd.DataFrame(data)
```

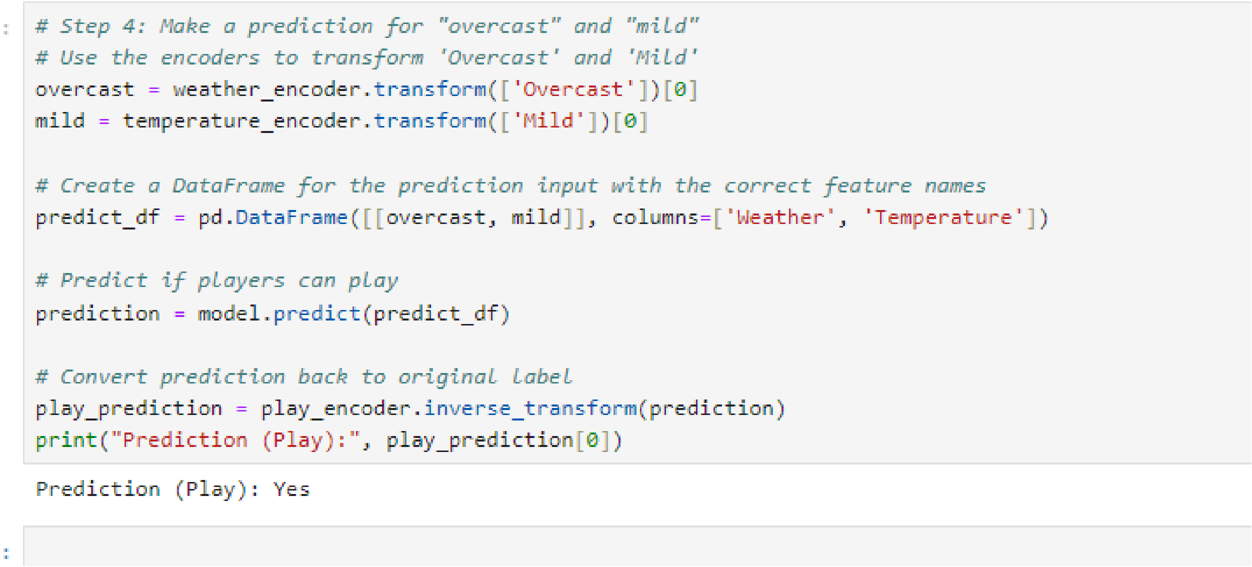


```
# Step 2: Encode each column separately
weather_encoder = LabelEncoder()
temperature_encoder = LabelEncoder()
play_encoder = LabelEncoder()

df['Weather'] = weather_encoder.fit_transform(df['Weather'])
df['Temperature'] = temperature_encoder.fit_transform(df['Temperature'])
df['Play'] = play_encoder.fit_transform(df['Play'])

# Separate features and target
X = df[['Weather', 'Temperature']]
y = df['Play']
```



```python
# Step 3: Train the Naïve Bayes model
model = CategoricalNB()
model.fit(X, y)
```

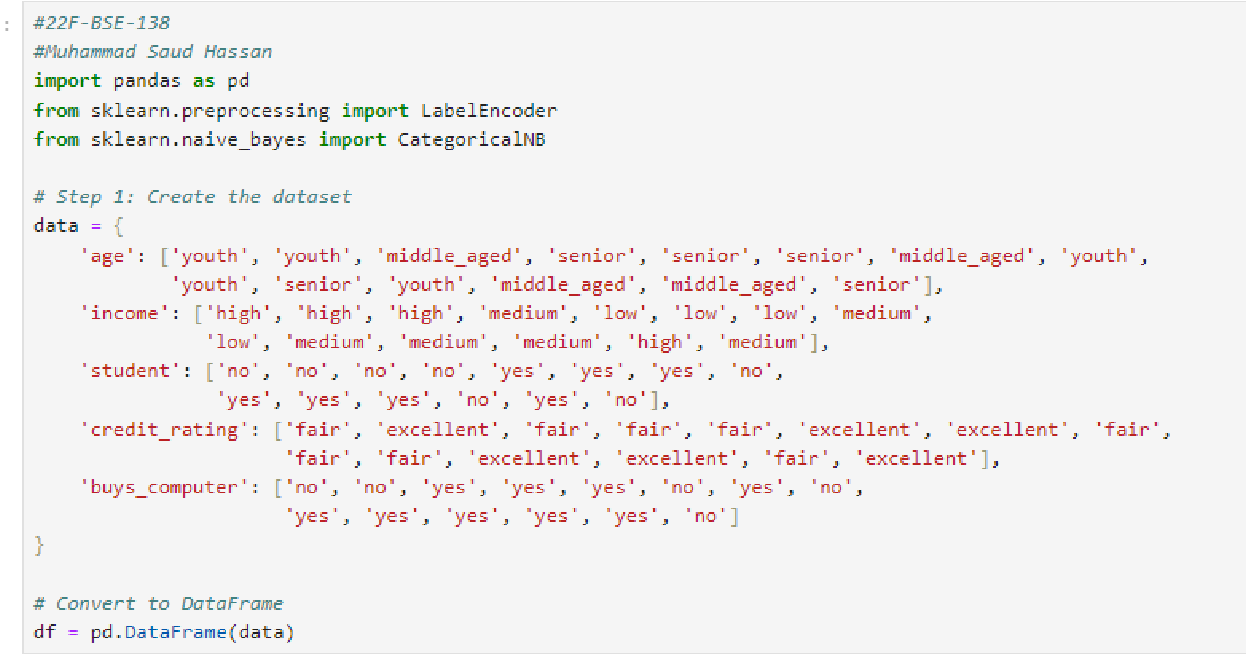
```
▾ CategoricalNB
CategoricalNB()
```



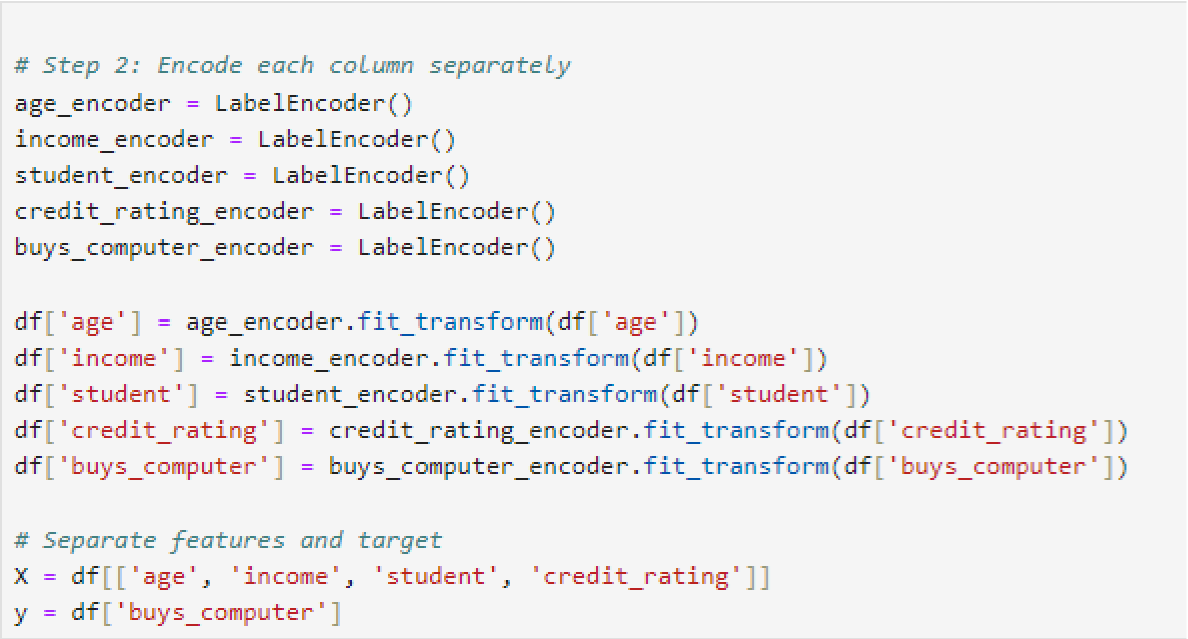
```python
# Step 4: Make a prediction for "overcast" and "mild"
# Use the encoders to transform 'Overcast' and 'Mild'
overcast = weather_encoder.transform(['Overcast'])[0]
mild = temperature_encoder.transform(['Mild'])[0]

# Create a DataFrame for the prediction input with the correct feature names
predict_df = pd.DataFrame([[overcast, mild]], columns=['Weather', 'Temperature'])

# Predict if players can play
prediction = model.predict(predict_df)

# Convert prediction back to original label
play_prediction = play_encoder.inverse_transform(prediction)
print("Prediction (Play):", play_prediction[0])
```

```
Prediction (Play): Yes
```

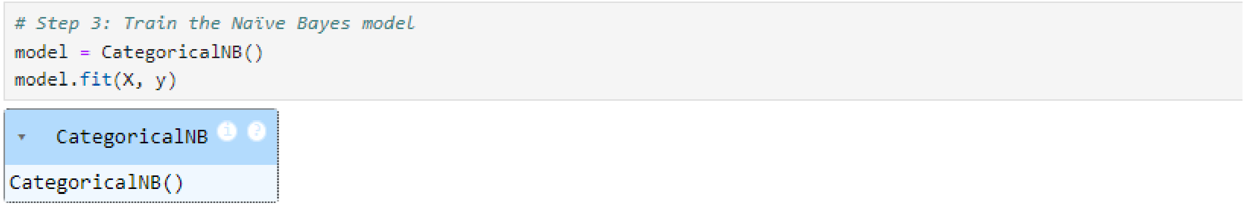
2. Consider the given dataset. Implement Naïve Bayes Algorithm to classify youth/medium/yes/fair**.**



```python
#22F-BSE-138
#Muhammad Saud Hassan
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import CategoricalNB

# Step 1: Create the dataset
data = {
    'age': ['youth', 'youth', 'middle_aged', 'senior', 'senior', 'senior', 'middle_aged', 'youth',
            'youth', 'senior', 'youth', 'middle_aged', 'middle_aged', 'senior'],
    'income': ['high', 'high', 'high', 'medium', 'low', 'low', 'low', 'medium',
               'low', 'medium', 'medium', 'medium', 'high', 'medium'],
    'student': ['no', 'no', 'no', 'no', 'yes', 'yes', 'yes', 'no',
                'yes', 'yes', 'yes', 'no', 'yes', 'no'],
    'credit_rating': ['fair', 'excellent', 'fair', 'fair', 'fair', 'excellent', 'excellent', 'fair',
                      'fair', 'fair', 'excellent', 'excellent', 'fair', 'excellent'],
    'buys_computer': ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no',
                      'yes', 'yes', 'yes', 'yes', 'yes', 'no']
}

# Convert to DataFrame
df = pd.DataFrame(data)
```

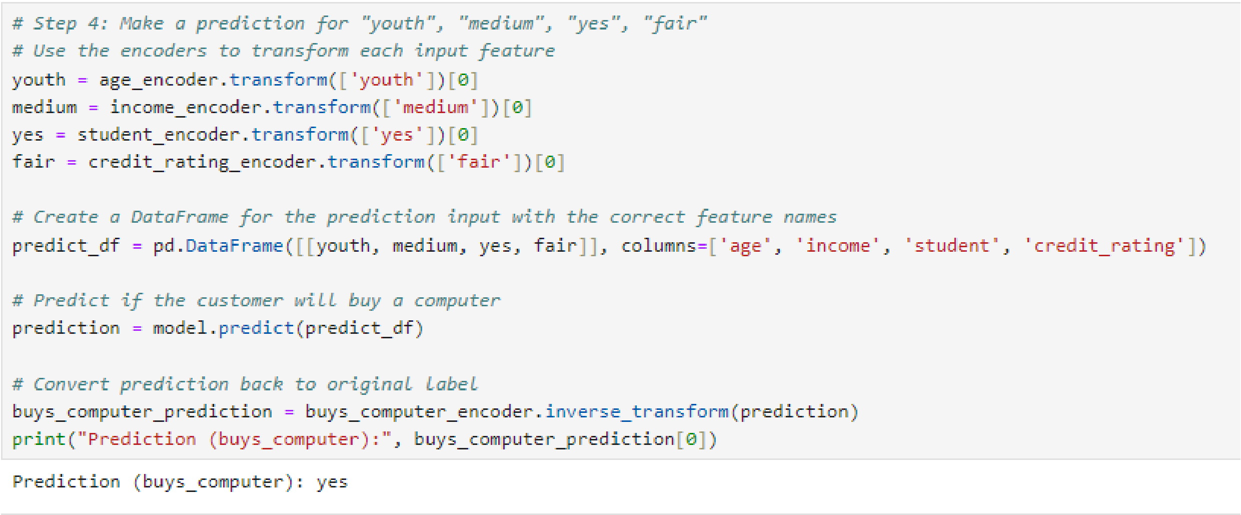


```
# Step 2: Encode each column separately
age_encoder = LabelEncoder()
income_encoder = LabelEncoder()
student_encoder = LabelEncoder()
credit_rating_encoder = LabelEncoder()
buys_computer_encoder = LabelEncoder()

df['age'] = age_encoder.fit_transform(df['age'])
df['income'] = income_encoder.fit_transform(df['income'])
df['student'] = student_encoder.fit_transform(df['student'])
df['credit_rating'] = credit_rating_encoder.fit_transform(df['credit_rating'])
df['buys_computer'] = buys_computer_encoder.fit_transform(df['buys_computer'])

# Separate features and target
X = df[['age', 'income', 'student', 'credit_rating']]
y = df['buys_computer']
```



```
# Step 3: Train the Naïve Bayes model
model = CategoricalNB()
model.fit(X, y)
```

```
▾ CategoricalNB
CategoricalNB()
```

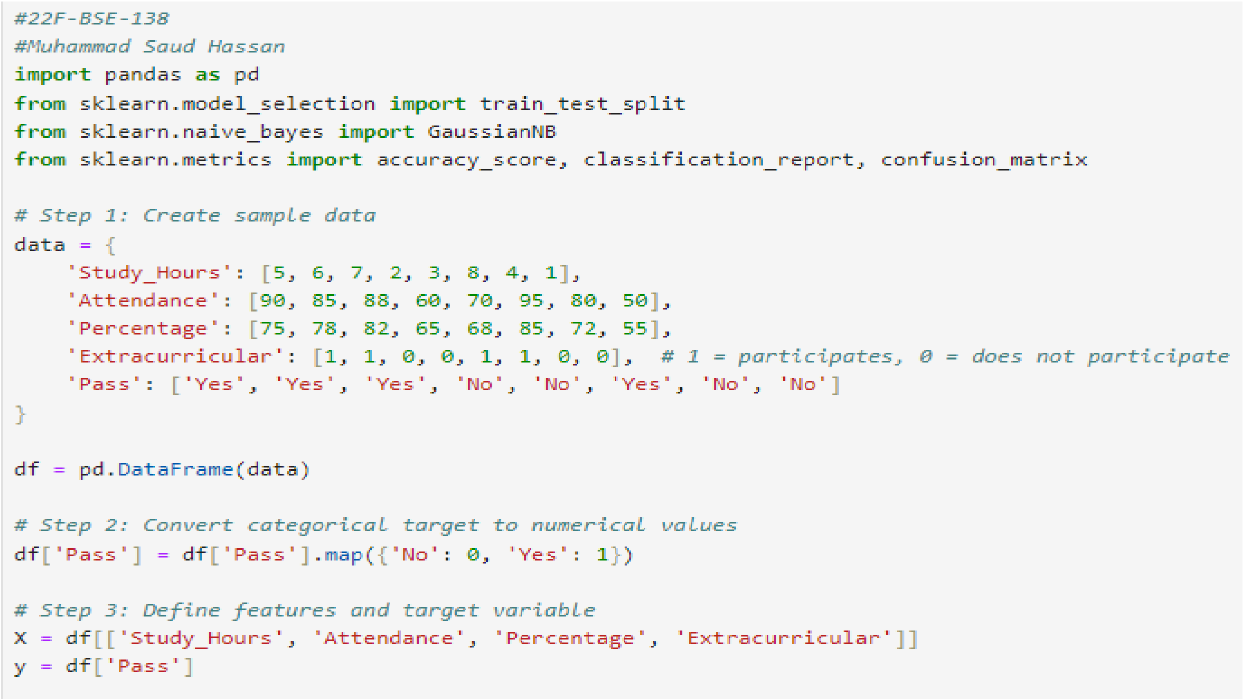


```
# Step 4: Make a prediction for "youth", "medium", "yes", "fair"
# Use the encoders to transform each input feature
youth = age_encoder.transform(['youth'])[0]
medium = income_encoder.transform(['medium'])[0]
yes = student_encoder.transform(['yes'])[0]
fair = credit_rating_encoder.transform(['fair'])[0]

# Create a DataFrame for the prediction input with the correct feature names
predict_df = pd.DataFrame([[youth, medium, yes, fair]], columns=['age', 'income', 'student', 'credit_rating'])

# Predict if the customer will buy a computer
prediction = model.predict(predict_df)

# Convert prediction back to original label
buys_computer_prediction = buys_computer_encoder.inverse_transform(prediction)
print("Prediction (buys_computer):", buys_computer_prediction[0])
```

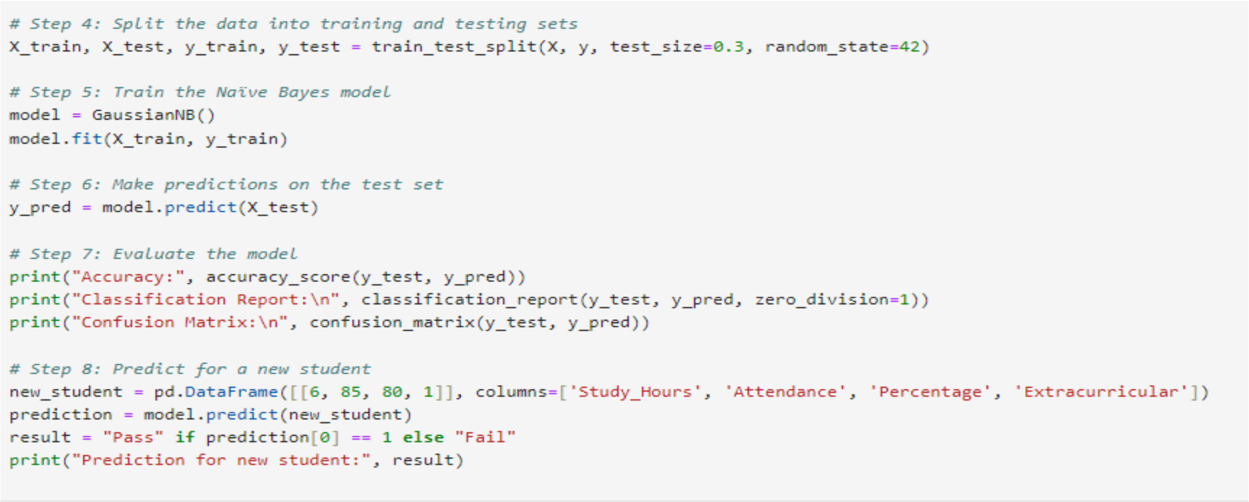
```
Prediction (buys_computer): yes
```

Home Tasks:

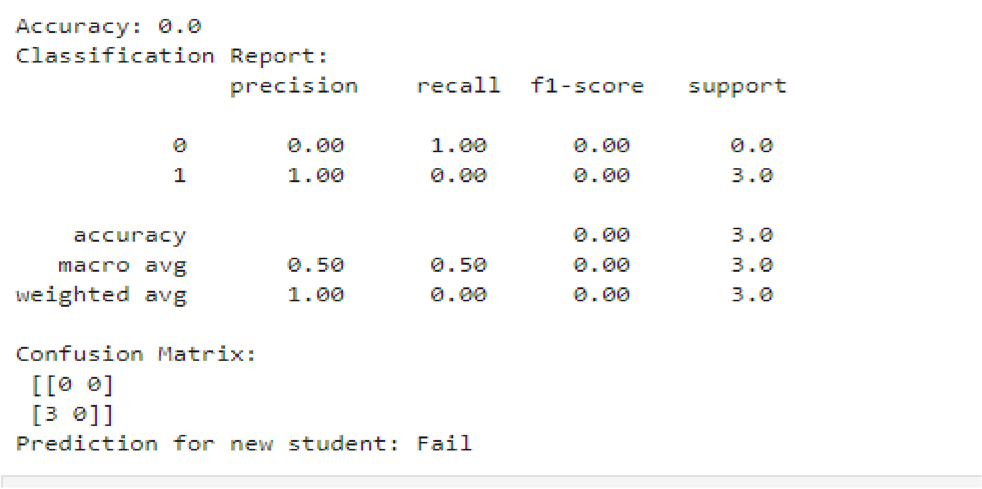
Using Naïve Bayes Algorithm, predict whether a student will pass or fail based on their study hours, attendance, percentage and participation in extra circular activities.



```python
#22F-BSE-138
#Muhammad Saud Hassan
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, confusion_matrix

# Step 1: Create sample data
data = {
    'Study_Hours': [5, 6, 7, 2, 3, 8, 4, 1],
    'Attendance': [90, 85, 88, 60, 70, 95, 80, 50],
    'Percentage': [75, 78, 82, 65, 68, 85, 72, 55],
    'Extracurricular': [1, 1, 0, 0, 1, 1, 0, 0],  # 1 = participates, 0 = does not participate
    'Pass': ['Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'No']
}

df = pd.DataFrame(data)

# Step 2: Convert categorical target to numerical values
df['Pass'] = df['Pass'].map({'No': 0, 'Yes': 1})

# Step 3: Define features and target variable
X = df[['Study_Hours', 'Attendance', 'Percentage', 'Extracurricular']]
y = df['Pass']
```



```python
# Step 4: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 5: Train the Naïve Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Step 6: Make predictions on the test set
y_pred = model.predict(X_test)

# Step 7: Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred, zero_division=1))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Step 8: Predict for a new student
new_student = pd.DataFrame([[6, 85, 80, 1]], columns=['Study_Hours', 'Attendance', 'Percentage', 'Extracurricular'])
prediction = model.predict(new_student)
result = "Pass" if prediction[0] == 1 else "Fail"
print("Prediction for new student:", result)
```



```
Accuracy: 0.0
Classification Report:
               precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       1.00      0.00      0.00       3.0

    accuracy                           0.00       3.0
   macro avg       0.50      0.50      0.00       3.0
weighted avg       1.00      0.00      0.00       3.0

Confusion Matrix:
 [[0 0]
 [3 0]]
Prediction for new student: Fail
```