LAB # 04

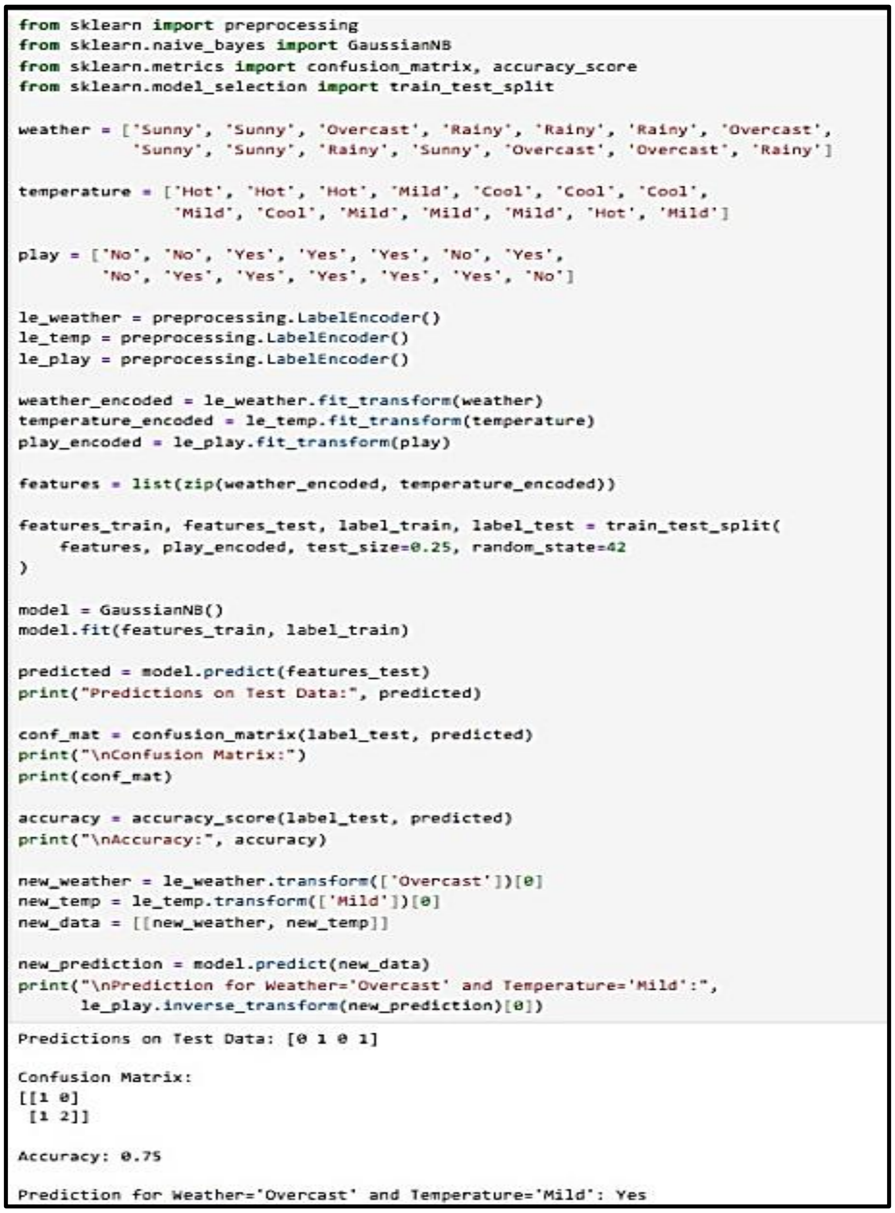
SUPERVISED LEARNING (NAÏVE BAYES ALGORITHM)

OBJECTIVE

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

Lab Task:

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



```
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

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']

le_weather = preprocessing.LabelEncoder()
le_temp = preprocessing.LabelEncoder()
le_play = preprocessing.LabelEncoder()

weather_encoded = le_weather.fit_transform(weather)
temperature_encoded = le_temp.fit_transform(temperature)
play_encoded = le_play.fit_transform(play)

features = list(zip(weather_encoded, temperature_encoded))

features_train, features_test, label_train, label_test = train_test_split(
    features, play_encoded, test_size=0.25, random_state=42
)

model = GaussianNB()
model.fit(features_train, label_train)

predicted = model.predict(features_test)
print("Predictions on Test Data:", predicted)

conf_mat = confusion_matrix(label_test, predicted)
print("\nConfusion Matrix:")
print(conf_mat)

accuracy = accuracy_score(label_test, predicted)
print("\nAccuracy:", accuracy)

new_weather = le_weather.transform(['Overcast'])[0]
new_temp = le_temp.transform(['Mild'])[0]
new_data = [[new_weather, new_temp]]

new_prediction = model.predict(new_data)
print("\nPrediction for Weather='Overcast' and Temperature='Mild':",
      le_play.inverse_transform(new_prediction)[0])
```

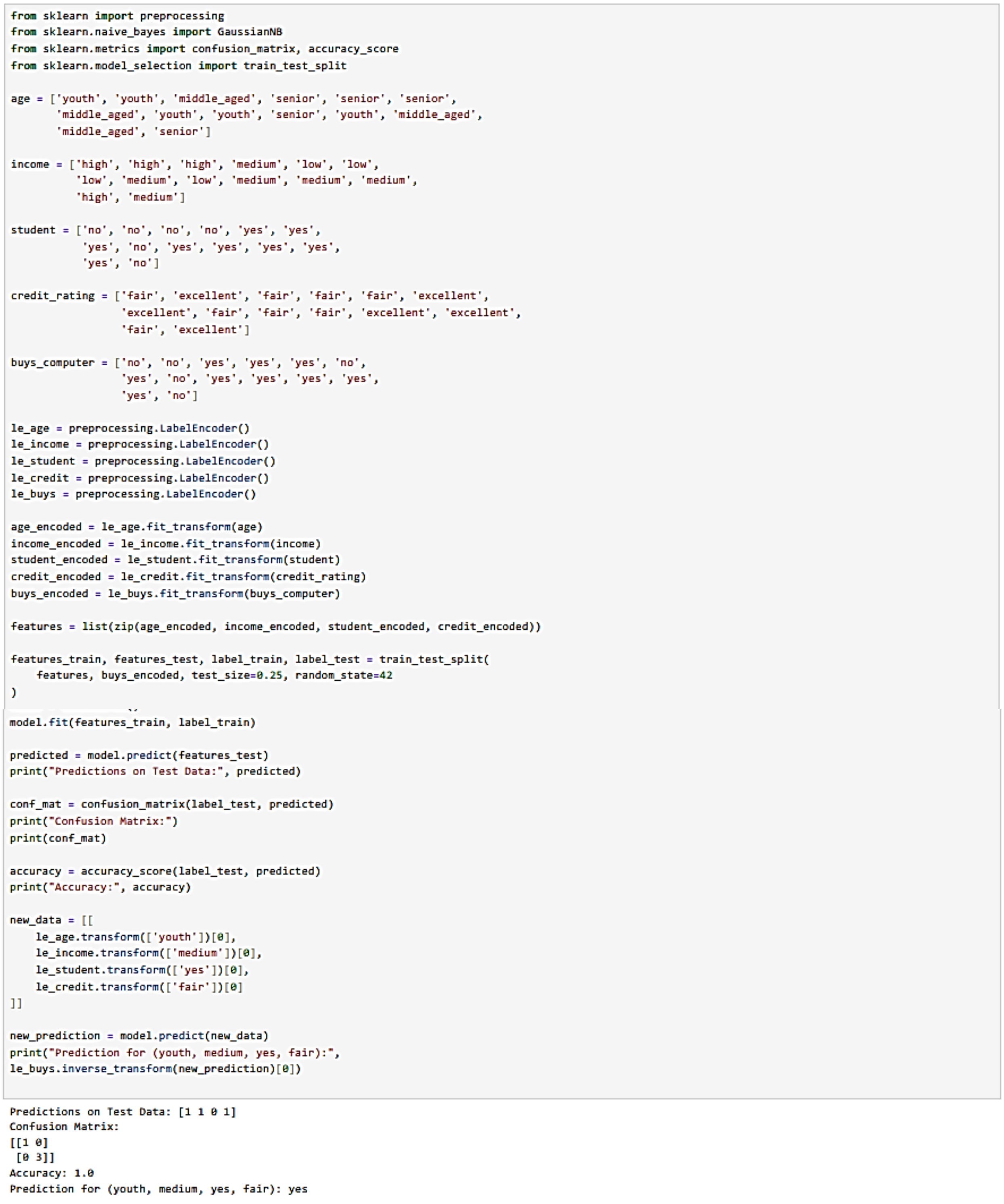
```
Predictions on Test Data: [0 1 0 1]

Confusion Matrix:
[[1 0]
 [1 2]]

Accuracy: 0.75

Prediction for Weather='Overcast' and Temperature='Mild': Yes
```

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



```python
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

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', 'yes',
           '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']

le_age = preprocessing.LabelEncoder()
le_income = preprocessing.LabelEncoder()
le_student = preprocessing.LabelEncoder()
le_credit = preprocessing.LabelEncoder()
le_buys = preprocessing.LabelEncoder()

age_encoded = le_age.fit_transform(age)
income_encoded = le_income.fit_transform(income)
student_encoded = le_student.fit_transform(student)
credit_encoded = le_credit.fit_transform(credit_rating)
buys_encoded = le_buys.fit_transform(buys_computer)

features = list(zip(age_encoded, income_encoded, student_encoded, credit_encoded))

features_train, features_test, label_train, label_test = train_test_split(
    features, buys_encoded, test_size=0.25, random_state=42
)
model.fit(features_train, label_train)

predicted = model.predict(features_test)
print("Predictions on Test Data:", predicted)

conf_mat = confusion_matrix(label_test, predicted)
print("Confusion Matrix:")
print(conf_mat)

accuracy = accuracy_score(label_test, predicted)
print("Accuracy:", accuracy)

new_data = [[
    le_age.transform(['youth'])[0],
    le_income.transform(['medium'])[0],
    le_student.transform(['yes'])[0],
    le_credit.transform(['fair'])[0]
]]

new_prediction = model.predict(new_data)
print("Prediction for (youth, medium, yes, fair):",
le_buys.inverse_transform(new_prediction)[0])
```

```
Predictions on Test Data: [1 1 0 1]
Confusion Matrix:
[[1 0]
 [0 3]]
Accuracy: 1.0
Prediction for (youth, medium, yes, fair): yes
```