LAB # 4

SUPERVISED LEARNING (NAÏVE BAYES ALGORITHM)

Lab Tasks:-

Weather	Temperature	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

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.

Code:-

```
data = [
    ['Sunny', 'Hot', 'No'],
    ['Sunny', 'Hot', 'No'],
    ['Overcast', 'Hot', 'Yes'],
    ['Rainy', 'Mild', 'Yes'],
    ['Rainy', 'Cool', 'Yes'],
    ['Rainy', 'Cool', 'No'],
    ['Overcast', 'Cool', 'Yes'],
    ['Sunny', 'Mild', 'No'],
    ['Sunny', 'Cool', 'Yes'],
    ['Rainy', 'Mild', 'Yes'],
    ['Sunny', 'Mild', 'Yes'],
    ['Overcast', 'Mild', 'Yes'],
    ['Overcast', 'Hot', 'Yes'],
    ['Rainy', 'Mild', 'No']
]
yes_data = [row for row in data if row[2] == 'Yes']
no_data = [row for row in data if row[2] == 'No']

p_yes = len(yes_data) / len(data)
p_no = len(no_data) / len(data)

weather = 'Overcast'
temperature = 'Mild'

def likelihood(attribute_index, value, subset):
    count = sum(1 for row in subset if row[attribute_index] == value)
    return count / len(subset)

p_weather_yes = likelihood(0, weather, yes_data)
p_temp_yes = likelihood(1, temperature, yes_data)

p_weather_no = likelihood(0, weather, no_data)
p_temp_no = likelihood(1, temperature, no_data)

p_yes_given = p_weather_yes * p_temp_yes * p_yes
p_no_given = p_weather_no * p_temp_no * p_no

print("P(Yes | Overcast, Mild) =", p_yes_given)
print("P(No  | Overcast, Mild) =", p_no_given)
if p_yes_given > p_no_given:
    print("\n Prediction: Players will PLAY.")
else:
    print("\n Prediction: Players will NOT play.")
```

Output:-

```
P(Yes | Overcast, Mild) = 0.12698412698412698
P(No  | Overcast, Mild) = 0.0

 Prediction: Players will PLAY.
```

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

age	*income*	*student*	*credit_rating*	*Class: buys_computer*
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Code:-

```
data = [
    ['youth', 'high', 'no', 'fair', 'no'],
    ['youth', 'high', 'no', 'excellent', 'no'],
    ['middle_aged', 'high', 'no', 'fair', 'yes'],
    ['senior', 'medium', 'no', 'fair', 'yes'],
    ['senior', 'low', 'yes', 'fair', 'yes'],
    ['senior', 'low', 'yes', 'excellent', 'no'],
    ['middle_aged', 'low', 'yes', 'excellent', 'yes'],
    ['youth', 'medium', 'no', 'fair', 'no'],
    ['youth', 'low', 'yes', 'fair', 'yes'],
    ['senior', 'medium', 'yes', 'fair', 'yes'],
    ['youth', 'medium', 'yes', 'excellent', 'yes'],
    ['middle_aged', 'medium', 'no', 'excellent', 'yes'],
    ['middle_aged', 'high', 'yes', 'fair', 'yes'],
    ['senior', 'medium', 'no', 'excellent', 'no']
]
yes_data = [row for row in data if row[4] == 'yes']
no_data = [row for row in data if row[4] == 'no']

p_yes = len(yes_data) / len(data)
p_no = len(no_data) / len(data)
test = ['youth', 'medium', 'yes', 'fair']

def likelihood(index, value, subset):
    count = sum(1 for row in subset if row[index] == value)
    return count / len(subset)

p_age_yes = likelihood(0, test[0], yes_data)
p_income_yes = likelihood(1, test[1], yes_data)
p_student_yes = likelihood(2, test[2], yes_data)
p_credit_yes = likelihood(3, test[3], yes_data)
p_age_no = likelihood(0, test[0], no_data)
p_income_no = likelihood(1, test[1], no_data)
p_student_no = likelihood(2, test[2], no_data)
p_credit_no = likelihood(3, test[3], no_data)

p_yes_given = p_age_yes * p_income_yes * p_student_yes * p_credit_yes * p_yes
p_no_given = p_age_no * p_income_no * p_student_no * p_credit_no * p_no

print("P(Yes | X) =", p_yes_given)
print("P(No  | X) =", p_no_given)
if p_yes_given > p_no_given:
    print("\n Prediction: Buys computer = YES")
else:
    print("\n Prediction: Buys computer = NO")
```

Output:-

```
P(Yes | X) = 0.028218694885361547
P(No  | X) = 0.006857142857142858

 Prediction: Buys computer = YES
```

Home Task:-

- Take a Dataset and use both KNN and Naïve Bayes Algorithm on it.

Code:-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import CategoricalNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

data = {
    'RoomType': [
        'Deluxe','Standard','Suite','Standard','Suite','Deluxe','Deluxe','Suite','Standard','Deluxe',
        'Suite','Standard','Deluxe','Suite','Standard','Deluxe','Suite','Standard','Deluxe','Suite',
        'Deluxe','Standard','Suite','Deluxe','Suite','Standard','Deluxe','Standard','Suite','Deluxe'
    ],    'Season': [
        'Summer','Winter','Rainy','Summer','Winter','Rainy','Summer','Winter','Rainy','Summer',
        'Winter','Rainy','Summer','Winter','Rainy','Summer','Winter','Rainy','Summer','Winter',
        'Summer','Winter','Rainy','Summer','Winter','Rainy','Summer','Winter','Rainy','Summer'
    ],    'Payment': [
        'Card','Cash','Card','Cash','Card','Card','Cash','Card','Card','Cash',
        'Cash','Card','Card','Cash','Cash','Card','Card','Cash','Card','Cash',
        'Card','Card','Cash','Cash','Card','Card','Cash','Cash','Card','Card'
    ],    'Discount': [
        'Yes','No','Yes','No','Yes','Yes','No','Yes','No','Yes',
        'No','Yes','Yes','No','Yes','Yes','No','Yes','No','Yes',
        'Yes','No','Yes','No','Yes','No','Yes','No','Yes','Yes'
    ],    'Booked': [
        'Yes','No','Yes','No','Yes','Yes','No','Yes','No','Yes',
        'No','Yes','Yes','No','Yes','Yes','No','Yes','No','Yes',
        'Yes','No','Yes','No','Yes','No','Yes','No','Yes','Yes'
    ]
}df = pd.DataFrame(data)
print("Hotel Booking Dataset (30 records):\n")
print(df.head())

le = LabelEncoder()
for col in df.columns:
    df[col] = le.fit_transform(df[col])

X = df[['RoomType', 'Season', 'Payment', 'Discount']]
y = df['Booked']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

nb = CategoricalNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)

print("\nNaïve Bayes Results:")
print("Accuracy:", round(accuracy_score(y_test, y_pred_nb), 2))
print(classification_report(y_test, y_pred_nb))

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_s, y_train_s)
y_pred_knn = knn.predict(X_test_s)

print("\nKNN Results:")
print("Accuracy:", round(accuracy_score(y_test_s, y_pred_knn), 2))
print(classification_report(y_test_s, y_pred_knn))
```

Output:-

```
Hotel Booking Dataset:

    RoomType  Season Payment Discount Booked
0     Deluxe  Summer    Card      Yes    Yes
1   Standard  Winter    Cash       No     No
2      Suite   Rainy    Card      Yes    Yes
3   Standard  Summer    Cash       No     No
4      Suite  Winter    Card      Yes    Yes

Naïve Bayes Results:
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         6

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9


KNN Results:
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         6

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9
```