Naive Bayes

Import libraries and load dataset

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Social_Network_Ads.csv')

df.head()
```

Out[2]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

observations :

* There are no missing values.

Separating independent and dependent data

In [4]:
```
# independent features
x = df.drop(columns = ['Purchased'], axis = 1)

# dependent features
y = df['Purchased']
```

Train Test Split

In [5]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)
```

Standard Scaling

In [6]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

Model Training

In [7]:
```
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

Out[7]:
```
GaussianNB()
```

Prediction

In [8]:
```
y_pred = classifier.predict(x_test)
```

Evaluation

In [9]:
```
from sklearn.metrics import confusion_matrix, accuracy_score

print(confusion_matrix(y_test, y_pred), '\n')
print(accuracy_score(y_test, y_pred))
```

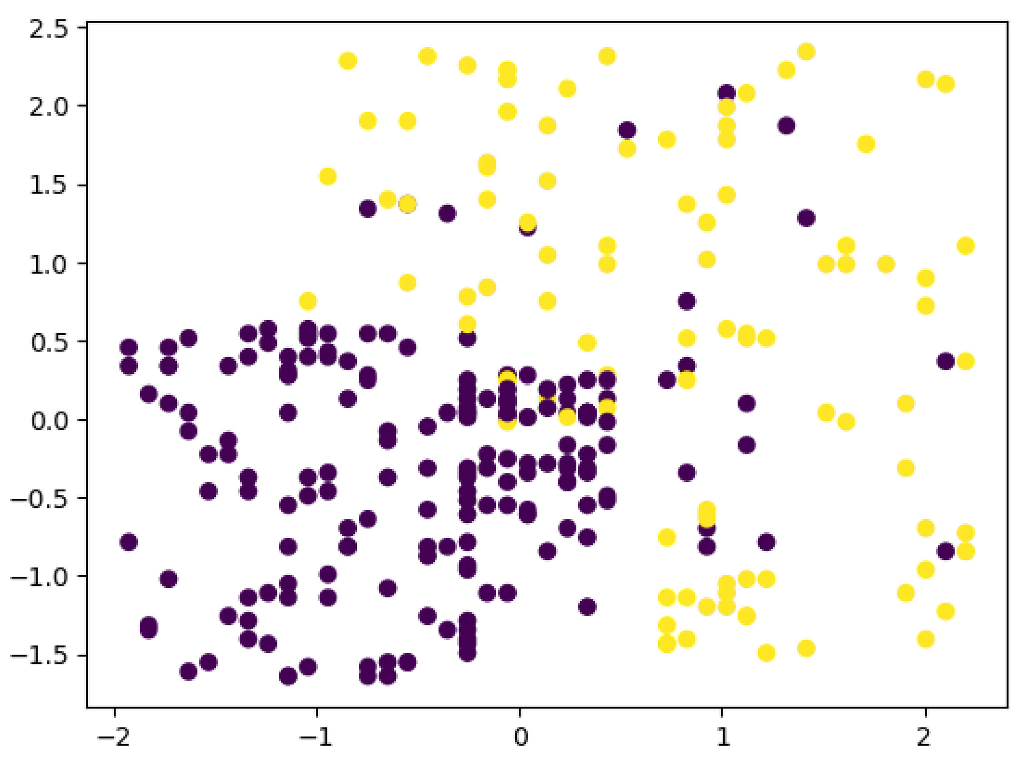
```
[[72  1]
 [ 8 39]]

0.925
```

1. Visualising training result

In [10]:
```
plt.scatter(x = x_train[:, 0], y = x_train[:, 1], c = y_train)
```

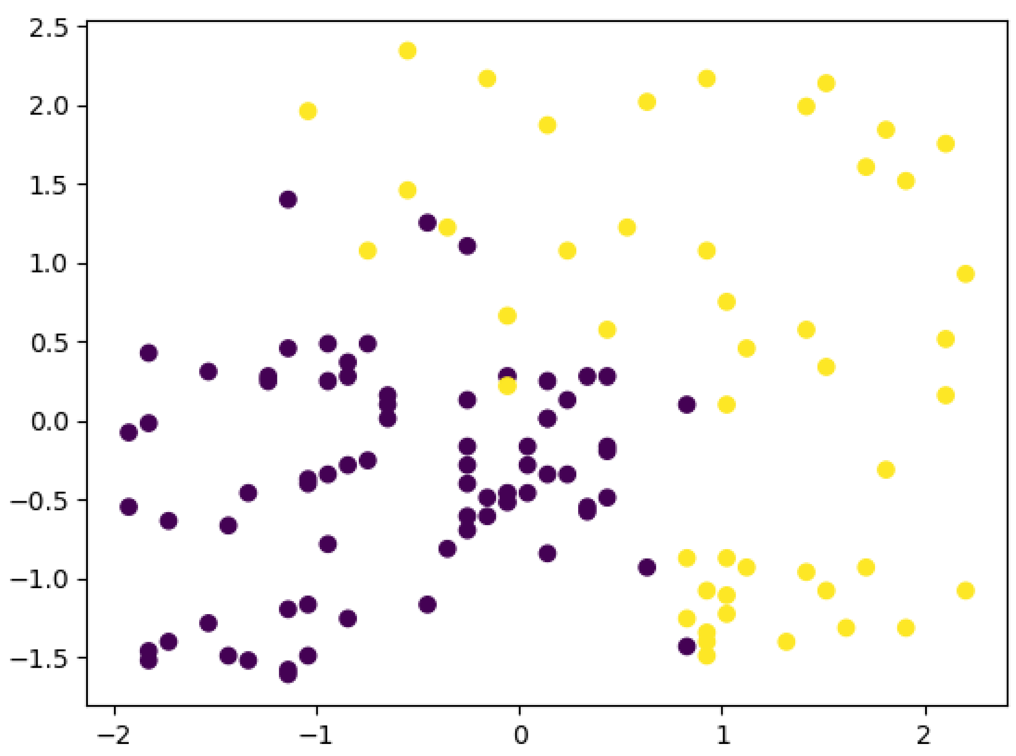
Out[10]:
```
<matplotlib.collections.PathCollection at 0x287e2b768e0>
```



2. Visualising test result

In [11]:
```
plt.scatter(x = x_test[:, 0], y = x_test[:, 1], c = y_test)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x287e2cdd940>
```



Naive Baye's Algorithm

Load dataset

In [1]:
```
from sklearn.datasets import load_iris
```

In [2]:
```
from sklearn.model_selection import train_test_split
```

In [4]:
```
x, y = load_iris(return_X_y = True)
```

In [5]:
```
print(x)
print()
print(y)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
 [5.9 3.  4.2 1.5]
 [6.  2.2 4.  1. ]
 [6.1 2.9 4.7 1.4]
 [5.6 2.9 3.6 1.3]
 [6.7 3.1 4.4 1.4]
 [5.6 3.  4.5 1.5]
 [5.8 2.7 4.1 1. ]
 [6.2 2.2 4.5 1.5]
 [5.6 2.5 3.9 1.1]
 [5.9 3.2 4.8 1.8]
 [6.1 2.8 4.  1.3]
 [6.3 2.5 4.9 1.5]
 [6.1 2.8 4.7 1.2]
 [6.4 2.9 4.3 1.3]
 [6.6 3.  4.4 1.4]
 [6.8 2.8 4.8 1.4]
 [6.7 3.  5.  1.7]
 [6.  2.9 4.5 1.5]
 [5.7 2.6 3.5 1. ]
 [5.5 2.4 3.8 1.1]
 [5.5 2.4 3.7 1. ]
 [5.8 2.7 3.9 1.2]
 [6.  2.7 5.1 1.6]
 [5.4 3.  4.5 1.5]
 [6.  3.4 4.5 1.6]
 [6.7 3.1 4.7 1.5]
 [6.3 2.3 4.4 1.3]
 [5.6 3.  4.1 1.3]
 [5.5 2.5 4.  1.3]
 [5.5 2.6 4.4 1.2]
 [6.1 3.  4.6 1.4]
 [5.8 2.6 4.  1.2]
 [5.  2.3 3.3 1. ]
 [5.6 2.7 4.2 1.3]
 [5.7 3.  4.2 1.2]
 [5.7 2.9 4.2 1.3]
 [6.2 2.9 4.3 1.3]
 [5.1 2.5 3.  1.1]
 [5.7 2.8 4.1 1.3]
 [6.3 3.3 6.  2.5]
 [5.8 2.7 5.1 1.9]
 [7.1 3.  5.9 2.1]
 [6.3 2.9 5.6 1.8]
 [6.5 3.  5.8 2.2]
 [7.6 3.  6.6 2.1]
 [4.9 2.5 4.5 1.7]
 [7.3 2.9 6.3 1.8]
 [6.7 2.5 5.8 1.8]
 [7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
 [6.8 3.  5.5 2.1]
 [5.7 2.5 5.  2. ]
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
 [6.5 3.  5.5 1.8]
 [7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
 [6.  2.2 5.  1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2. ]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Observation:

- Multiclass Classification problem as there are 3 categories in output feature.
- As input feature are continuous data we have to use gaussian naive bayes.

---

1. Train Test Split

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, random_state = 0)
```

---

2. Model Building

In [7]:
```
from sklearn.naive_bayes import GaussianNB
```

In [8]:
```
gnb = GaussianNB()
```

In [9]:
```
gnb.fit(x_train, y_train)
```

Out[9]:
```
GaussianNB()
```

---

3. Prediction

In [10]:
```
y_pred = gnb.predict(x_test)

y_pred
```

Out[10]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

---

4. Performance Metrics

In [12]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

print(accuracy_score(y_test, y_pred))
print()
print(confusion_matrix(y_test, y_pred))
print()
print(classification_report(y_test, y_pred))
```

```
1.0

[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Internal assignment

In [17]:
```
import seaborn as sns

df = sns.load_dataset('tips')
```

In [18]:
```
df
```

Out[18]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [40]:
```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()

encoded = encoder.fit_transform(df[['sex','day','time']]).toarray()
encoded
```

Out[40]:
```
array([[1., 0., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       ...,
       [0., 1., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [1., 0., 0., ..., 1., 1., 0.]])
```

In [41]:
```
import pandas as pd
df_encoded = pd.DataFrame(encoded, columns = encoder.get_feature_names_out())
```

In [45]:
```
final_df = pd.concat([df[['total_bill','tip','size']],df_encoded,df[['smoker']]],axis=1)

con = {'Yes' : 1 , 'No' :0}
final_df['smoker'] = final_df['smoker'].map(con)

final_df
```

Out[45]:

	total_bill	tip	size	sex_Female	sex_Male	day_Fri	day_Sat	day_Sun	day_Thur	time_Dinner	time_Lunch	smoker
0	16.99	1.01	2	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0
1	10.34	1.66	3	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0
2	21.01	3.50	3	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0
3	23.68	3.31	2	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0
4	24.59	3.61	4	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
239	29.03	5.92	3	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0
240	27.18	2.00	2	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1
241	22.67	2.00	2	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1
242	17.82	1.75	2	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0
243	18.78	3.00	2	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0

244 rows × 12 columns

In [50]:
```
x = final_df.iloc[:,:-1]
y = final_df.iloc[:,-1]

x , y
```

Out[50]:
```
(     total_bill   tip  size  sex_Female  sex_Male  day_Fri  day_Sat  day_Sun  \
 0         16.99  1.01     2         1.0       0.0      0.0      0.0      1.0
 1         10.34  1.66     3         0.0       1.0      0.0      0.0      1.0
 2         21.01  3.50     3         0.0       1.0      0.0      0.0      1.0
 3         23.68  3.31     2         0.0       1.0      0.0      0.0      1.0
 4         24.59  3.61     4         1.0       0.0      0.0      0.0      1.0
 ..          ...   ...   ...         ...       ...      ...      ...      ...
 239       29.03  5.92     3         0.0       1.0      0.0      1.0      0.0
 240       27.18  2.00     2         1.0       0.0      0.0      1.0      0.0
 241       22.67  2.00     2         0.0       1.0      0.0      1.0      0.0
 242       17.82  1.75     2         0.0       1.0      0.0      1.0      0.0
 243       18.78  3.00     2         1.0       0.0      0.0      0.0      0.0

      day_Thur  time_Dinner  time_Lunch
 0         0.0          1.0         0.0
 1         0.0          1.0         0.0
 2         0.0          1.0         0.0
 3         0.0          1.0         0.0
 4         0.0          1.0         0.0
 ..        ...          ...         ...
 239       0.0          1.0         0.0
 240       0.0          1.0         0.0
 241       0.0          1.0         0.0
 242       0.0          1.0         0.0
 243       1.0          1.0         0.0

 [244 rows x 11 columns],
 0      0
 1      0
 2      0
 3      0
 4      0
       ..
 239    0
 240    1
 241    1
 242    0
 243    0
 Name: smoker, Length: 244, dtype: category
 Categories (2, int64): [1, 0])
```

In [49]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state = 0)
```

In [51]:
```
gnb = GaussianNB()
gnb.fit(x_train,y_train)
y_pred = gnb.predict(x_test)
```

In [52]:
```
print(accuracy_score(y_test, y_pred))
print()
print(confusion_matrix(y_test, y_pred))
print()
print(classification_report(y_test, y_pred))
```

```
0.6351351351351351

[[30  9]
 [18 17]]

              precision    recall  f1-score   support

           0       0.62      0.77      0.69        39
           1       0.65      0.49      0.56        35

    accuracy                           0.64        74
   macro avg       0.64      0.63      0.62        74
weighted avg       0.64      0.64      0.63        74
```
