In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
sna_df = pd.read_csv('Social_Network_Ads.csv')
sna_df.head(10)
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

In [3]:

```
sna_df.isnull().sum()
```

Out[3]:

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [4]:

```
sna_df.describe()
```

Out[4]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [5]:

```
sna_df = sna_df.replace('Male',1)
```

In [6]:
```
sna_df = sna_df.replace('Female',0)
```

In [7]:
```
sna_df
```

Out[7]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	1	19	19000	0
1	15810944	1	35	20000	0
2	15668575	0	26	43000	0
3	15603246	0	27	57000	0
4	15804002	1	19	76000	0
...	...	...	...	...	...
395	15691863	0	46	41000	1
396	15706071	1	51	23000	1
397	15654296	0	50	20000	1
398	15755018	1	36	33000	0
399	15594041	0	49	36000	1

400 rows × 5 columns

In [8]:
```
sna_df.corr()
```

Out[8]:

	User ID	Gender	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.025249	-0.000721	0.071097	0.007120
Gender	-0.025249	1.000000	-0.073741	-0.060435	-0.042469
Age	-0.000721	-0.073741	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	-0.060435	0.155238	1.000000	0.362083
Purchased	0.007120	-0.042469	0.622454	0.362083	1.000000

In [9]:
```
#Spliting the dataset in independent and dependent variables
X = sna_df.loc[:, ['Age', 'EstimatedSalary','Gender']].values
y = sna_df['Purchased'].values
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_
```

In [11]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [12]:
```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
logisticregression = LogisticRegression()
logisticregression.fit(X_train, y_train)
```

Out[12]:
```
LogisticRegression()
```

In [13]:
```
y_pred = logisticregression.predict(X_test)
print(y_pred)
```

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

```
In [14]: y_compare = np.vstack((y_test,y_pred)).T
```

```
In [15]: print(y_compare)
```

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

In [16]:

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('true negatives (TN): Both, actual and predicted values are false: ', cm[0,0
print('true positives (TP): Both, actual and predicted values are true: ', cm[1,1]
print('false positives (FP): Predicted value is yes but actual is false: ', cm[0,1
print('false negative (FN): Predicted value is no but actual is true: ', cm[1,0])
```

```
[[45  3]
 [10 22]]
true negatives (TN): Both, actual and predicted values are false:  45
true positives (TP): Both, actual and predicted values are true:  22
false positives (FP): Predicted value is yes but actual is false:  3
false negative (FN): Predicted value is no but actual is true:  10
```

In [17]:

```
from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,y_pred)*100
```

In [18]:

```
score
```

Out[18]:

```
83.75
```

In [19]:

```
(cm[0,1] + cm[1,0]) / (cm[0,0]+ cm[1,1] + cm[0,1] + cm[1,0])
```

Out[19]:

```
0.1625
```

In [20]:

```
#Mean Squared error
print(np.mean((y_pred-y_test)**2))
```

```
0.1625
```

In [21]:

```
precision = cm[1,1]  / (cm[1,1] +  cm[0,1] )
```

In [22]:

```
precision
```

Out[22]:

```
0.88
```

In [23]:

```
recall = cm[1,1]  / (cm[1,1] + cm[1,0] )
```

In [24]:

```
recall
```

Out[24]:

```
0.6875
```