In [1]:

```
import pandas as pd
import numpy as np
import matplotlib as pl
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("/home/anaconda_user/Downloads/Dataset/Social_Network_Ads.csv")
df
```

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
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1
20	15649487	Male	45	22000	1
21	15736760	Female	47	49000	1
22	15714658	Male	48	41000	1
23	15599081	Female	45	22000	1
24	15705113	Male	46	23000	1
25	15631159	Male	47	20000	1
26	15792818	Male	49	28000	1
27	15633531	Female	47	30000	1
28	15744529	Male	29	43000	0
29	15669656	Male	31	18000	0
...	...	...	...	...	...
370	15611430	Female	60	46000	1
371	15774744	Male	60	83000	1
372	15629885	Female	39	73000	0
373	15708791	Male	59	130000	1
374	15793890	Female	37	80000	0
375	15646091	Female	46	32000	1
376	15596984	Female	46	74000	0
377	15800215	Female	42	53000	0
378	15577806	Male	41	87000	1
379	15749381	Female	58	23000	1
380	15683758	Male	42	64000	0
381	15670615	Male	48	33000	1

382	15715622	Female	44	139000	1
383	15707634	Male	49	28000	1
384	15806901	Female	57	33000	1
385	15775335	Male	56	60000	1
386	15724150	Female	49	39000	1
387	15627220	Male	39	71000	0
388	15672330	Male	47	34000	1
389	15668521	Female	48	35000	1
390	15807837	Male	48	33000	1
391	15592570	Male	47	23000	1
392	15748589	Female	45	45000	1
393	15635893	Male	60	42000	1
394	15757632	Female	39	59000	0
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [4]:

```
df.shape
```

Out[4]:

```
(400, 5)
```

In [9]:

```
df.columns
```

Out[9]:

```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [10]:

```
df.dtypes
```

Out[10]:

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

In [11]:

```
df.describe()
```

Out[11]:

	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 [12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
User ID            400 non-null int64
Gender             400 non-null object
Age                400 non-null int64
EstimatedSalary    400 non-null int64
Purchased          400 non-null int64
dtypes: int64(4), object(1)
memory usage: 15.7+ KB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

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

In [13]:

```
X = df.iloc[:,[2,3]]
X
```

Out[13]:

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000
9	35	65000
10	26	80000
11	26	52000
12	20	86000
13	32	18000
14	18	82000
15	29	80000
16	47	25000
17	45	26000
18	46	28000
19	48	29000
20	45	22000
21	47	49000
22	48	41000
23	45	22000
24	46	23000
25	47	20000
26	49	28000
27	47	30000
28	29	43000
29	31	18000
...	...	...

370	60	46000
371	60	83000
372	39	73000
373	59	130000
374	37	80000
375	46	32000
376	46	74000
377	42	53000
378	41	87000
379	58	23000
380	42	64000
381	48	33000
382	44	139000
383	49	28000
384	57	33000
385	56	60000
386	49	39000
387	39	71000
388	47	34000
389	48	35000
390	48	33000
391	47	23000
392	45	45000
393	60	42000
394	39	59000
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

400 rows × 2 columns

In [14]:

```
Y = df.iloc[:,4]
Y
```

Out[14]:

```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      1
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     1
17     1
18     1
19     1
20     1
21     1
22     1
23     1
24     1
25     1
26     1
27     1
28     0
29     0
      ..
370    1
371    1
372    0
373    1
374    0
375    1
376    0
377    0
378    1
379    1
380    0
381    1
382    1
383    1
384    1
385    1
386    1
387    0
388    1
389    1
390    1
391    1
392    1
393    1
394    0
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

In [15]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [16]:

```
X_train , X_test, Y_train , Y_test = train_test_split(X,Y,test_size = 0.25)
```

In [28]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
```

In [19]:

```
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

In [20]:

```
Classifier = LogisticRegression(random_state=0)
Classifier.fit(X_train, Y_train)
```

Out[20]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [21]:

```
Y_pred = Classifier.predict(X_test)
```

In [23]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,Y_pred)
```

In [24]:

```
tp = cm[0,[0]]
```

In [26]:

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, Y_pred)*100
accuracy
```

Out[26]:

```
86.0
```

In [27]:

```
TP = cm[0,[0]]
TP
```

Out[27]:

```
array([61])
```

In [29]:

```
TN = cm[0,[1]]
TN
```

Out[29]:

```
array([7])
```

In [30]:

```
FP = cm[1,[0]]
FP
```

Out[30]:

```
array([7])
```

In [31]:

```
FN = cm[1,[1]]
FN
```

Out[31]:

```
array([25])
```

In [34]:

```
accuracy = (TP+TN)/(TP+TN+FP+FN)
print(accuracy)
```

```
[0.68]
```

In [33]:

```
error_rate =(FP+FN)/(TP+TN+FP+FN)
print(error_rate)
```

```
[0.32]
```

In [35]:

```
precision = TP/(TP+FP)
print(precision)
```

```
[0.89705882]
```

In [36]:

```
Recall = (TP/(TP+FN))
print(Recall)
```

```
[0.70930233]
```

In [38]:

```
specificity = (TN/(TN+FP))
print(specificity)
```

```
[0.5]
```