Sale Prediction from Existing customer - Logistic Regression

Problem statement: Build a ML model which predicts whether the new custmer will buy the product or not based on its age and salary.

1. Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
```

2. Data Gathering and summarizing

In [2]:
```
df=pd.read_csv("DigitalAd_dataset.csv")
df.head()
```

Out[2]:

	Age	Salary	Status
0	18	82000	0
1	29	80000	0
2	47	25000	1
3	45	26000	1
4	46	28000	1

In [3]:
```
df.shape
```

Out[3]:
```
(400, 3)
```

In [4]:
```
df.describe()
```

Out[4]:

	Age	Salary	Status
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

3. Separating dataset into dependent(y) and independent(x) features.

In [5]:
```
x=df.drop('Status', axis=1)
y=df['Status']
print(x.head())
print('*'*20)
print(y.head())
```

```
   Age  Salary
0   18   82000
1   29   80000
2   47   25000
3   45   26000
4   46   28000
********************
0    0
1    0
2    1
3    1
4    1
Name: Status, dtype: int64
```

4. Spliting dataset into training and testing

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_
```

In [7]:
```
x_train.head()
```

Out[7]:

	Age	Salary
224	46	82000
195	46	22000
13	47	30000
200	47	43000
239	37	146000

In [8]:
```
y_train.head()
```

Out[8]:
```
224    0
195    0
13     1
200    0
239    1
Name: Status, dtype: int64
```

5. Model building

In [9]:
```
LR_model=LogisticRegression(random_state=0)
```

In [10]:
```
LR_model.fit(x_train, y_train)
```

Out[10]:
```
LogisticRegression
LogisticRegression(random_state=0)
```

6. Model evaluation

In [11]:
```
# Training accuracy
y_pred_train=LR_model.predict(x_train)
cnf_matrix=confusion_matrix(y_train, y_pred_train)
print("Confusion matrix: \n", cnf_matrix)
print('*'*20)
acc_score=accuracy_score(y_train, y_pred_train)
print("Accuracy score: ", acc_score)
```

```
Confusion matrix:
 [[206   0]
 [114   0]]
********************
Accuracy score:  0.64375
```

In [12]:
```
# Testing accuracy
y_pred=LR_model.predict(x_test)
cnf_matrix=confusion_matrix(y_test, y_pred)
print("Confusion matrix: \n", cnf_matrix)
print('*'*20)
acc_score=accuracy_score(y_test, y_pred)
print("Accuracy score: ", acc_score)
```

```
Confusion matrix:
 [[51  0]
 [29  0]]
********************
Accuracy score:  0.6375
```

Problem:

Since both the features have different scales, there is a chance that higher weightage is given to features with higher magnitude. This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biassed towards one feature.

Solution:

we scale our data to make all the features contribute equally to the result.

7. Feature Scaling

In [13]:
```
from sklearn.preprocessing import StandardScaler
std_scalar=StandardScaler()
x_std_train=std_scalar.fit_transform(x_train)
x_std_test=std_scalar.transform(x_test)
```

8. Model building after feature scaling

In [14]:
```
Model=LogisticRegression(random_state=0)
```

In [15]:
```
Model.fit(x_std_train, y_train)
```

Out[15]:
```
LogisticRegression(random_state=0)
```

9. Model evaluation after scaling

In [16]:
```
# Training accuracy
y_pred_train=Model.predict(x_std_train)
cnf_matrix=confusion_matrix(y_train, y_pred_train)
print("Confusion matrix: \n", cnf_matrix)
print('*'*20)
acc_score=accuracy_score(y_train, y_pred_train)
print("Accuracy score: ", acc_score)
```

```
Confusion matrix:
 [[189  17]
 [ 34  80]]
********************
Accuracy score:  0.840625
```

In [22]:
```
# Testing accuracy
y_pred=Model.predict(x_std_test)
cnf_matrix=confusion_matrix(y_test, y_pred)
print("Confusion matrix: \n", cnf_matrix)
print('*'*20)
acc_score=accuracy_score(y_test, y_pred)
print("Accuracy score: ", acc_score*100)
```

```
Confusion matrix:
 [[46  5]
 [ 8 21]]
********************
Accuracy score:  83.75
```

Prediction

Predicting, wheather new customer with Age & Salary will buy a product or not.

In [21]:
```
age=int(input("Enter new customer age: "))
salary=int(input("Enter Salary or new customer: "))
dict1={'Age':[age], 'Salary':[salary]}
data=pd.DataFrame(dict1)
prediction=Model.predict(std_scalar.transform(data))
if prediction==1:
    print("Customer will Buy")
else:
    print("Customer won't Buy")
```

```
Enter new customer age: 46
Enter Salary or new customer: 28000
Customer won't Buy
```