In [1]:
```
# # Output variable -> y
# y -> Whether the client has subscribed a term deposit or not
# Binomial ("yes" or "no")
```

1.Import Libraries

In [2]:
```
#Import Nessasry Libraries
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,roc_auc_score,roc_curve
```

2.Import dataset

In [3]:
```
#Read csv file
#Take only nessasary columns
data=pd.read_csv("bank_data.csv",usecols=['age','balance','duration','campaign','y'])
data.head() #TOP 5 rows
```

Out[3]:

	age	balance	duration	campaign	y
0	58	2143	261	1	0
1	44	29	151	1	0
2	33	2	76	1	0
3	47	1506	92	1	0
4	33	1	198	1	0

3.Data Undestanding

In [4]:
```
#indicate columns name
data.columns
```

Out[4]:
```
Index(['age', 'balance', 'duration', 'campaign', 'y'], dtype='object')
```

In [5]:
```
#indicate number of rows and columns
data.shape
```

Out[5]:
```
(45211, 5)
```

In [6]:
```
#Indicate any null values avilible
data.isnull().sum()
```

Out[6]:
```
age         0
balance     0
duration    0
campaign    0
y           0
dtype: int64
```

In [7]:
```
#Infomation of null entry and memomry usage
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       45211 non-null  int64
 1   balance   45211 non-null  int64
 2   duration  45211 non-null  int64
 3   campaign  45211 non-null  int64
 4   y         45211 non-null  int64
dtypes: int64(5)
memory usage: 1.7 MB
```

In [8]:
```
#No dublicate data avilible
data[data.duplicated()]
```

Out[8]:

	age	balance	duration	campaign	y
1252	43	0	187	1	0
4586	34	0	150	1	0
4819	32	0	91	1	0
9056	36	0	174	1	0
10293	37	0	137	1	0
...	...	...	...	...	...
42505	46	0	155	1	0
43344	42	0	158	1	0
43608	48	0	85	1	0
44554	50	0	120	2	0
44842	31	0	173	1	0

94 rows × 5 columns

In [9]:
```
#Drop Dublicate data it might impact accuracy
data=data.drop_duplicates(ignore_index=True)
```

In [10]:
```
#After Droping dublicate Columns
data.shape
```

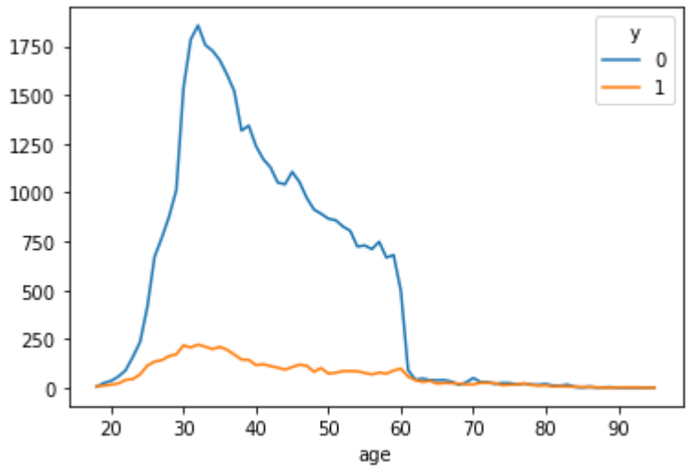
Out[10]:
```
(45117, 5)
```

In [11]:
```
#Value indicate data distribution
data['y'].value_counts()
```

Out[11]:
```
0    39828
1     5289
Name: y, dtype: int64
```

In [12]:
```
pd.crosstab(data.age,data.y).plot(kind='line')
# it indicate that age of 20-60 has more rejection of application while 60-90 almost everybody
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bfa6722760>
```



4.Data Preparation

In [13]:
```
s=data[['age','balance','duration']]
```

In [14]:
```
#It transforms the data in such a manner that it has mean as 0 and standard deviation as 1.
# It arranges the data in a standard normal distribution
scale=StandardScaler()
scaled_x=scale.fit_transform(s)
scaled_x=pd.DataFrame(scaled_x,columns=s.columns)
```

In [15]:
```
#input and ouput data
x=pd.concat([scaled_x,data['campaign']],axis=1)
y=data['y']
```

In [16]:
```
x.head()
```

Out[16]:

	age	balance	duration	campaign
0	1.606287	0.255338	0.010054	1
1	0.288348	-0.438415	-0.416782	1
2	-0.747176	-0.447275	-0.707806	1
3	0.570763	0.046293	-0.645721	1
4	-0.747176	-0.447603	-0.234407	1

5.Model Building

In [17]:
```
model=LogisticRegression(intercept_scaling=1,
                         l1_ratio=None,
                         max_iter=5000,
                         multi_class='auto',
                         n_jobs=None,
                         penalty='l2',
                         random_state=None,
                         solver='lbfgs',
                         tol=0.0001,
                         verbose=0
                        )
```

In [18]:
```
#Fit The Model
model.fit(x,y)
```

Out[18]:
```
LogisticRegression(max_iter=5000)
```

In [19]:
```
#Coeeficient of model
model.coef_
```

Out[19]:
```
array([[ 0.08170551,  0.11288593,  0.91340364, -0.14006492]])
```

6.Model Testing

In [20]:
```
#Predict output
y_pred=model.predict(x)
y_pred
```

Out[20]:
```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [21]:
```
#Data Frame visualization actual vs predict output
result=pd.DataFrame({"Actual":y,"prediction":y_pred})
result.head()
```

Out[21]:

	Actual	prediction
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

7. Accuracy Score

In [22]:
```
#Accuracy = Number of correct predictions Total number of predictions
accuracy_score(y,y_pred)
```

Out[22]:
```
0.8888223951060575
```

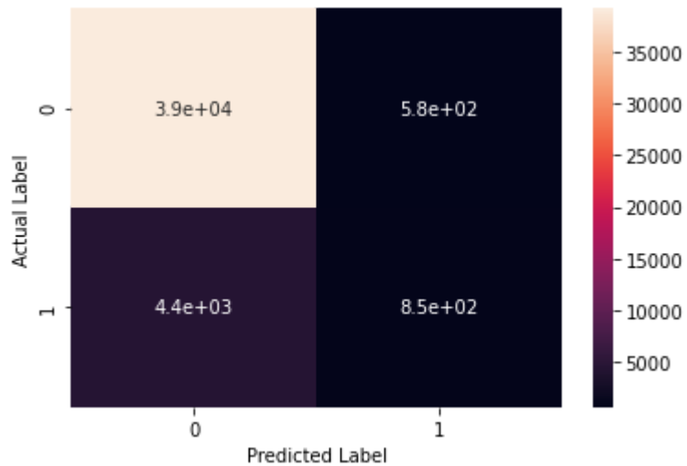
8. Confusion matrix

In [23]:
```
#table that is often used to describe the performance of a classification model
# The matrix compares the actual target values with those predicted by the machine learning model
confusion_matrix(y,y_pred)
```

Out[23]:
```
array([[39248,   580],
       [ 4436,   853]], dtype=int64)
```

In [24]:
```
#heatmap is a graphical representation of data that uses a system of color-coding to represent different values.
sns.heatmap(confusion_matrix(y,y_pred),annot=True)
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
```

Out[24]:
```
Text(0.5, 15.0, 'Predicted Label')
```



9. Classification Report

In [25]:
```
#Classification report
print(classification_report(y,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     39828
           1       0.60      0.16      0.25      5289

    accuracy                           0.89     45117
   macro avg       0.75      0.57      0.60     45117
weighted avg       0.86      0.89      0.86     45117
```

10.Model Evalution

In [26]:
```
new_data=pd.DataFrame({'age':44,'balance':29,'duration':151,'campaign':1},index=[0])
new_data
```

Out[26]:

	age	balance	duration	campaign
0	44	29	151	1

In [27]:
```
y_pred_new=model.predict(new_data)
y_pred_new  #Model Predict on our new data
```

Out[27]:
```
array([1], dtype=int64)
```