Data Analytics II (Logistic Regression)

In [14]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [15]:

```
dataset = pd.read_csv('./Social_Network_Ads.csv')
dataset.head()
```

Out[15]:

	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

In [16]:

```
dataset.shape
```

Out[16]:

```
(400, 5)
```

In [17]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [18]:

```
dataset.iloc[:, 2:5].describe()
```

Out[18]:

	Age	EstimatedSalary	Purchased
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

In [19]:

```
X = dataset[['Age', 'EstimatedSalary']]
Y = dataset['Purchased']
```

In [25]:

```
X.head()
```

Out[25]:

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

In [20]:

```
from sklearn.model_selection import train_test_split
```

In [21]:

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

In [22]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
```

In [23]:

```
stdScaler = StandardScaler()
X_train = stdScaler.fit_transform(X_train)
X_test = stdScaler.transform(X_test)
```

In [29]:

```
LRmodel = LogisticRegression()
LRmodel.fit(X_train, Y_train)
```

Out[29]:

```
LogisticRegression()
```

In [30]:

```
preds = LRmodel.predict(X_test)
print(confusion_matrix(Y_test, preds))
print(classification_report(Y_test, preds))
```

```
[[63  4]
 [ 9 24]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91        67
           1       0.86      0.73      0.79        33

    accuracy                           0.87       100
   macro avg       0.87      0.83      0.85       100
weighted avg       0.87      0.87      0.87       100
```

In [31]:

```
def changeTrainTestPlit(testsize: float):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)
    stdScaler = StandardScaler()
    X_train = stdScaler.fit_transform(X_train)
    X_test = stdScaler.transform(X_test)

    LRmodel = LogisticRegression()
    LRmodel.fit(X_train, Y_train)

    preds = LRmodel.predict(X_test)
    print(confusion_matrix(Y_test, preds))
    print(classification_report(Y_test, preds))
```

In [32]:

```
changeTrainTestPlit(.15)
```

```
[[60  5]
 [10 25]]
              precision    recall  f1-score   support

           0       0.86      0.92      0.89        65
           1       0.83      0.71      0.77        35

    accuracy                           0.85       100
   macro avg       0.85      0.82      0.83       100
weighted avg       0.85      0.85      0.85       100
```

In [33]:

```
changeTrainTestPlit(.20)
```

```
[[64  3]
 [ 5 28]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.94        67
           1       0.90      0.85      0.88        33

    accuracy                           0.92       100
   macro avg       0.92      0.90      0.91       100
weighted avg       0.92      0.92      0.92       100
```

In [34]:

```
changeTrainTestPlit(.30)
```

```
[[65  6]
 [ 6 23]]
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        71
           1       0.79      0.79      0.79        29

    accuracy                           0.88       100
   macro avg       0.85      0.85      0.85       100
weighted avg       0.88      0.88      0.88       100
```

In [ ]:

```

```