```
# 5 Apply logical regression Model techniques to predict the
# data on any dataset.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=pd.read_csv('/content/drive/MyDrive/KRAI/User_data.csv')
dataset
```

| | 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 |
| **...** | ... | ... | ... | ... | ... |
| **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

```
x=dataset.iloc[:,[2,3]].values

y=dataset.iloc[:,4].values
print(x)
print(y)
```

```
 [    59  29000]
 [    58  47000]
 [    46  88000]
 [    38  71000]
 [    54  26000]
 [    60  46000]
 [    60  83000]
 [    39  73000]
 [    59 130000]
 [    37  80000]
 [    46  32000]
 [    46  74000]
 [    42  53000]
 [    41  87000]
 [    58  23000]
 [    42  64000]
 [    48  33000]
 [    44 139000]
 [    49  28000]
 [    57  33000]
 [    56  60000]
 [    49  39000]
 [    39  71000]
 [    47  34000]
 [    48  35000]
 [    48  33000]
 [    47  23000]
 [    45  45000]
 [    60  42000]
 [    39  59000]
 [    46  41000]
 [    51  23000]
 [    50  20000]
 [    36  33000]
 [    49  36000]]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
```

```
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
X_train=sc_x.fit_transform(X_train)
X_test=sc_x.transform(X_test)
print(X_train[0:10,:])
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]
```

```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(X_train,y_train)
```

```
▾       LogisticRegression
LogisticRegression(random_state=0)
```

```
y_pred=classifier.predict(X_test)
```

```
# import the metrics class
from sklearn.metrics import confusion_matrix

cnf=confusion_matrix(y_test, y_pred)
cnf
```

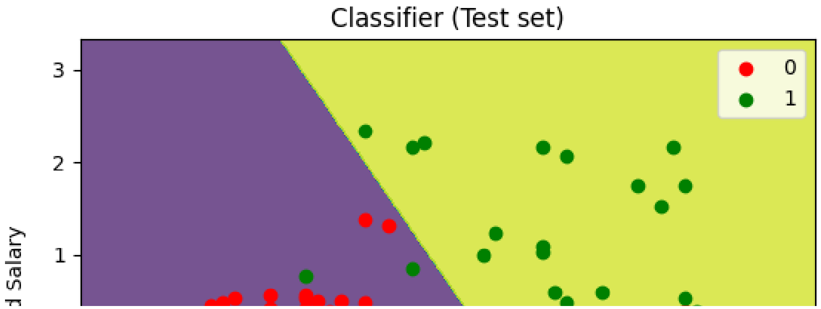
```
array([[65,  3],
       [ 8, 24]])
```

```
from sklearn.metrics import accuracy_score
print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.89
```

```
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                            stop = X_set[:, 0].max() + 1, step = 0.01),
                    np.arange(start = X_set[:, 1].min() - 1,
                            stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(
            np.array([X1.ravel(), X2.ravel()]).T).reshape(
            X1.shape), alpha = 0.75, cmsp = ListedColormap(('green','red')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == i, 1],
                c = ListedColormap(('red','green'))(i), label = j)

plt.title('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-19-1f549b236efd>:9: UserWarning: The following kwargs were not used by contour: 'cmsp'
  plt.contourf(X1, X2, classifier.predict(
<ipython-input-19-1f549b236efd>:17: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence i
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == i, 1],
```


Classifier (Test set)
d Salary
● 0
● 1