Importing Libraries

```
#to load the dataset
import pandas as pd
#to help with arrays
import numpy as np
```

```
from google.colab import files
uploaded= files.upload()
```

Choose Files DigitalAd_dataset.csv

- **DigitalAd_dataset.csv**(text/csv) - 4893 bytes, last modified: 5/28/2022 - 100% done

```
Saving DigitalAd_dataset.csv to DigitalAd_dataset.csv
```

Loading Dataset

```
dataset=pd.read_csv("DigitalAd_dataset.csv")
```

Summarise Dataset

```
#gives the rows n columns
print(dataset.shape)
#gives the first five rows
print(dataset.head(5))
```

```
(400, 3)
   Age  Salary  Status
0   18   82000       0
1   29   80000       0
2   47   25000       1
3   45   26000       1
4   46   28000       1
```

Segregating into Dependent n Independent variables

```
# X independent variables
X=dataset.iloc[:,:-1].values
#Y dependent variable
Y=dataset.iloc[:,-1].values
#indexing for iloc works like rows,cols
```

Train n Test

```
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)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
#.fit transfrom cz calculates mean nnvaraiance for each n every feature
X_test=sc.transform(X_test)
# transform is transforming all features using respectivemean n varaiance
# WE WANT TEST DATA TO BE COMPLETELY NEW
```

Training

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state=0)
model.fit(X_train,Y_train)
```

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

Prediction

```
# taking inputs
age = int(input("Customer Age:"))
salary = int(input("Customer Salary:"))
newCust = [[age,salary]]
# result
result = model.predict(sc.transform(newCust))
print(result)
if result == 1:
  print("Customer will buy")
else:
  print("Customer won't buy")
```

```
     Customer Age:19
     Customer Salary:20000
     [0]
     Customer won't buy
```

Prediction for Test Data

```
# crosschecking
Y_pred = model.predict(X_test)
print(np.concatenate((Y_pred.reshape(len(Y_pred),1), Y_test.reshape(len(Y_test),1)),1))
```

```
      [0 1]
      [0 0]
      [0 1]
      [0 0]
      [0 1]
      [0 0]
      [0 0]
```

```
 [1 1]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 1]
 [0 1]
 [1 1]
 [0 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [1 1]]
```

Evaluation

```
# Confusion Matrix
# Acuuracy = TruePositive+TrueNegative/ TruePositive+TrueNegative+Falsenegative+FalsePosit
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(Y_test, Y_pred)
cm = confusion_matrix(Y_test, Y_pred)
score = accuracy_score(Y_test, Y_pred)*100
# Printing
print("Confusion Matrix:")
print(cm)
print("Accuracy of the model:")
print(score)
```

```
     Confusion Matrix:
     [[61  0]
      [20 19]]
     Accuracy of the model:
     80.0
```

✓ 0s completed at 11:06 PM