**AIM :-**

```
import numpy as np
import pandas as pd
```

```
from google.colab import files
uploaded=files.upload()
df=pd.read_csv('Social_Network_Ads.csv')
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Social_Network_Ads.csv to Social_Network_Ads.csv

| | 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 |

```
df.shape
```

```
(400, 5)
```

```
df.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
```

```
X=df[['Age','EstimatedSalary']]
y=df['Purchased']
#data is separated x-independent y-dependent
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30)
X_train.shape
```

```
(280, 2)
```

```
X_train.isna().sum()
```

| | 0 |
|---|---|
| **Age** | 0 |
| **EstimatedSalary** | 0 |

**dtype:** int64

```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
```

```
imputer.fit(X_train[['Age',"EstimatedSalary"]]) #.fit method- finds the mean
```

```
▾ SimpleImputer
SimpleImputer()
```

```
X_train[['Age','EstimatedSalary']]=imputer.transform(X_train[['Age','EstimatedSalary']])
```

```
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
scalar.fit(X_train)
X_train_scaled=scalar.transform(X_train)
X_test=scalar.transform(X_test)
```

```
X_train_scaled
```

```
array([[0.73809524, 0.37037037],
       [0.95238095, 0.59259259],
       [0.66666667, 0.0962963 ],
       [0.45238095, 0.48148148],
       [0.35714286, 0.0962963 ],
       [0.26190476, 0.20740741],
       [0.21428571, 0.11851852],
       [0.76190476, 0.03703704],
       [0.26190476, 0.48148148],
       [0.23809524, 0.12592593],
       [0.85714286, 0.68888889],
       [0.66666667, 0.12592593],
       [0.4047619 , 0.60740741],
       [0.26190476, 0.20740741],
       [0.04761905, 0.15555556],
       [0.4047619 , 0.32592593],
       [0.45238095, 0.9037037 ],
       [0.54761905, 0.42222222],
       [0.66666667, 0.05185185],
       [0.21428571, 0.28888889],
       [0.52380952, 0.34074074],
       [0.54761905, 0.42222222],
       [0.26190476, 0.5037037 ],
       [0.52380952, 0.41481481],
       [0.80952381, 1.        ],
       [0.23809524, 0.2962963 ],
       [0.19047619, 0.48148148],
       [0.4047619 , 0.54074074],
       [0.9047619 , 0.33333333],
       [0.52380952, 0.68148148],
       [0.14285714, 0.08888889],
       [0.45238095, 0.42222222],
       [0.4047619 , 0.33333333],
       [0.95238095, 0.17037037],
       [0.21428571, 0.55555556],
       [0.73809524, 0.54814815],
       [0.73809524, 0.52592593],
       [0.57142857, 0.28888889],
       [0.04761905, 0.52592593],
       [0.02380952, 0.08148148],
       [0.97619048, 0.1037037 ],
       [0.5       , 0.88148148],
       [0.54761905, 0.22222222],
       [0.54761905, 0.48148148],
       [0.4047619 , 0.44444444],
       [0.45238095, 0.28148148],
       [0.        , 0.4962963 ],
       [0.02380952, 0.04444444],
       [0.54761905, 0.42222222],
       [0.23809524, 0.21481481],
       [0.97619048, 0.85185185],
       [0.30952381, 0.76296296],
       [0.28571429, 0.34814815],
       [0.28571429, 0.88888889],
       [0.26190476, 0.5037037 ],
       [0.76190476, 0.54074074],
       [0.73809524, 0.0962963 ],
       [0.23809524, 0.32592593],
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train_scaled,y_train)
```

▾ LogisticRegression

```
LogisticRegression()
```

```
model.score(X_train_scaled,y_train)
```

```
0.8285714285714286
```

```
model.score(X_test,y_test)
```

```
0.85
```

```
Start coding or generate with AI.
```