**Data Analytics - II : Logistic Regression**

**Problem Statement**

**1. Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset.  
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.**

In [30]:

*#imports*

**import** numpy **as** np

**import** pandas **as** pd

**import** seaborn **as** sns

**import** warnings

**import** matplotlib.pyplot **as** plt

warnings**.**filterwarnings("ignore")

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** confusion\_matrix,classification\_report

In [9]:

data **=** pd**.**read\_csv("Social\_Network\_Ads.csv")

In [10]:

data**.**sample(5)

Out[10]:

|  | **User ID** | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| --- | --- | --- | --- | --- | --- |
| **233** | 15614187 | Male | 49 | 86000 | 1 |
| **147** | 15749130 | Female | 41 | 30000 | 0 |
| **134** | 15800061 | Female | 28 | 55000 | 0 |
| **72** | 15595228 | Female | 20 | 23000 | 0 |
| **73** | 15782530 | Female | 33 | 113000 | 0 |

In [11]:

data**.**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 [12]:

data**.**isna()**.**sum()

Out[12]:

User ID 0

Gender 0

Age 0

EstimatedSalary 0

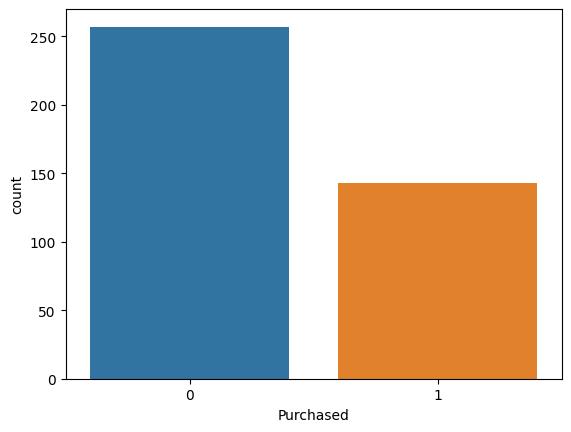
Purchased 0

dtype: int64

In [14]:

*# Target label : 'Purchased'*

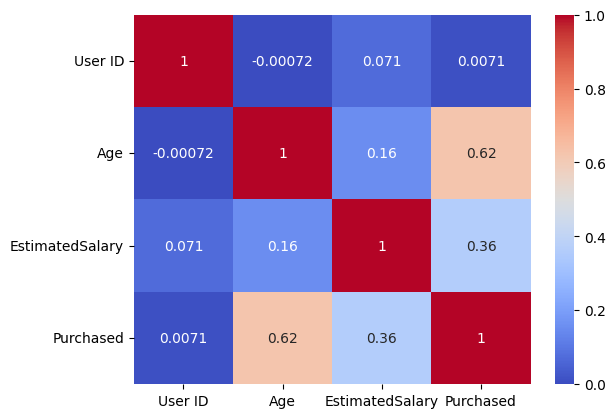
sns**.**countplot(data **=** data, x **=** 'Purchased');



In [16]:

*# Finding useful features*

sns**.**heatmap(data**.**corr(), annot **=** **True**, cmap**=** 'coolwarm' );



In [17]:

features **=** data[['Age', 'EstimatedSalary']]

label **=** data['Purchased']

In [18]:

scaler **=** StandardScaler()

features **=** scaler**.**fit\_transform(features)

In [20]:

x **=** features

y **=** label

In [21]:

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.2, random\_state**=**42)

**Model**

In [23]:

model **=** LogisticRegression()

model**.**fit(x\_train, y\_train)

Out[23]:

LogisticRegression()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

**Prediction**

In [26]:

y\_pred **=** model**.**predict(x\_test)

In [27]:

y\_pred

Out[27]:

array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,

0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,

1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0], dtype=int64)

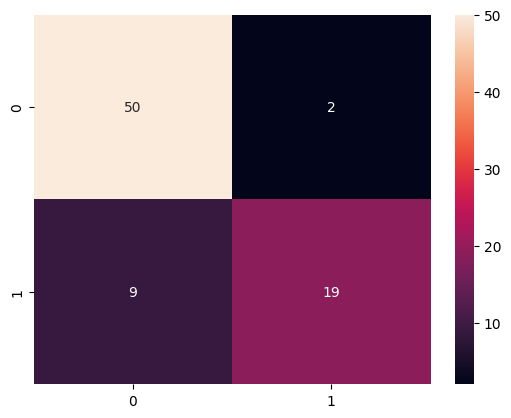
**Evaluation**

In [29]:

sns**.**heatmap(confusion\_matrix(y\_test, y\_pred), annot**=** **True**)

Out[29]:

<AxesSubplot:>



In [31]:

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.85 0.96 0.90 52

1 0.90 0.68 0.78 28

accuracy 0.86 80

macro avg 0.88 0.82 0.84 80

weighted avg 0.87 0.86 0.86 80