```
import pandas as pd
import numpy as np
import seaborn as sn

df=pd.read_csv("Social_Network_Ads.csv")

df.head()
```

	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.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
```

```
df.shape
```

```
(400, 5)
```

```
df.isnull().sum()
```

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

```
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000

```
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000
```

Encoding gender feature

transfer string to number

```
df['Gender'].nunique()
```

```
2
```

```
df['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
df['Gender'].replace(['Male','Female'],[1,0],inplace=True)
```

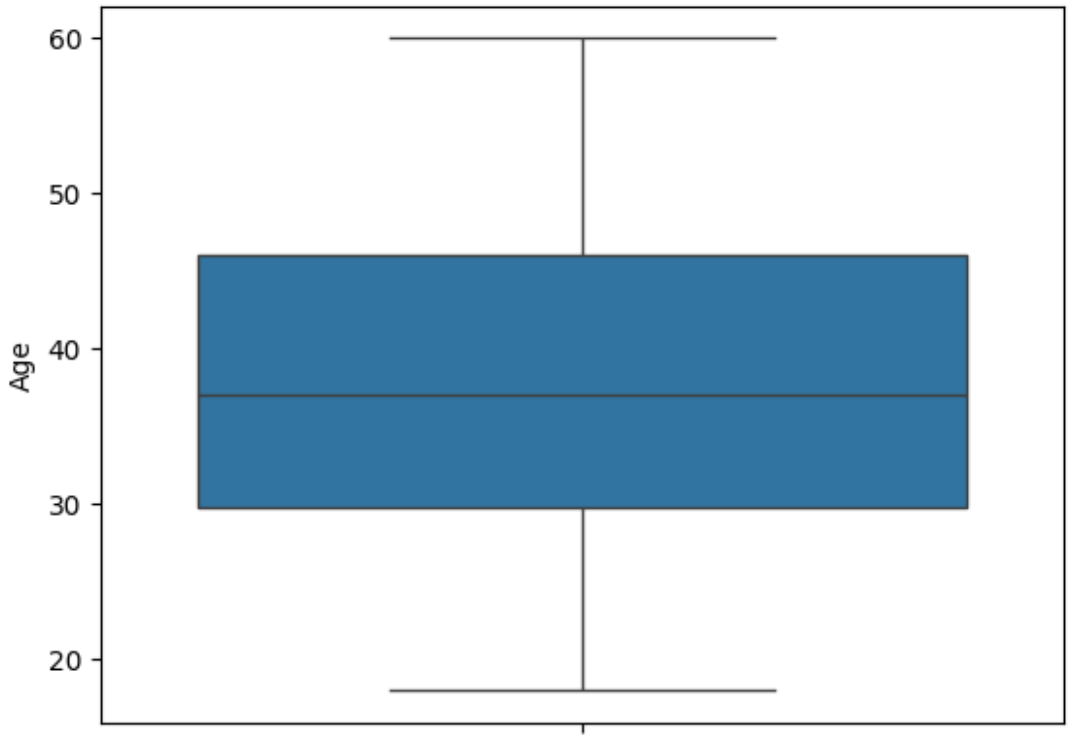
```
df.sample(5)
```

```
      User ID  Gender  Age  EstimatedSalary  Purchased
18   15704583       1   46            28000          1
119  15701962       1   41            59000          0
215  15779529       0   60           108000          1
273  15589449       1   39           106000          1
233  15614187       1   49            86000          1
```

Exploratory Data Analysis(EDA)

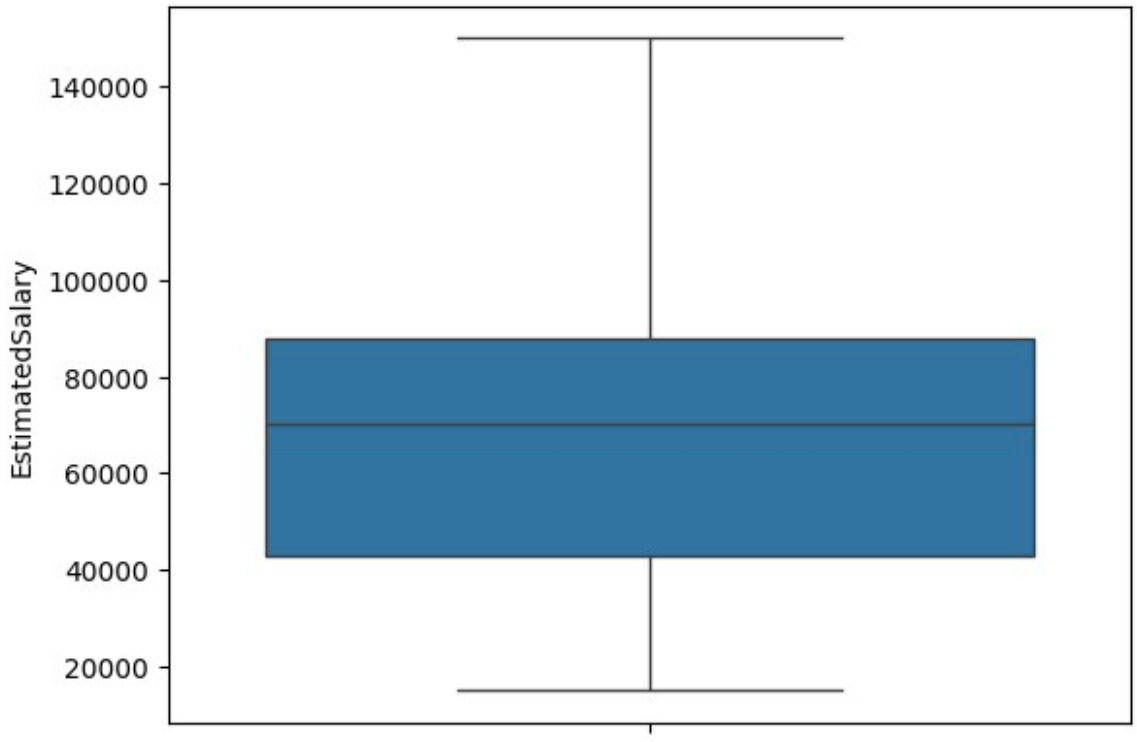
```
sn.boxplot(df['Age'])
```

```
<Axes: ylabel='Age'>
```




```
sn.boxplot(df['EstimatedSalary'])
```

```
<Axes: ylabel='EstimatedSalary'>
```



Setting value of x and y

```
x=df.iloc[:,1:4];
y=df.iloc[:,4];

x.head()
```

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

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Purchased, dtype: int64
```

Splitting dataset into training and testing dataset

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=42)

x_train.shape
```

```
(320, 3)
```

```
y_train.shape
```

```
(320,)
```

```
x_test.shape
```

```
(80, 3)
```

Scaling

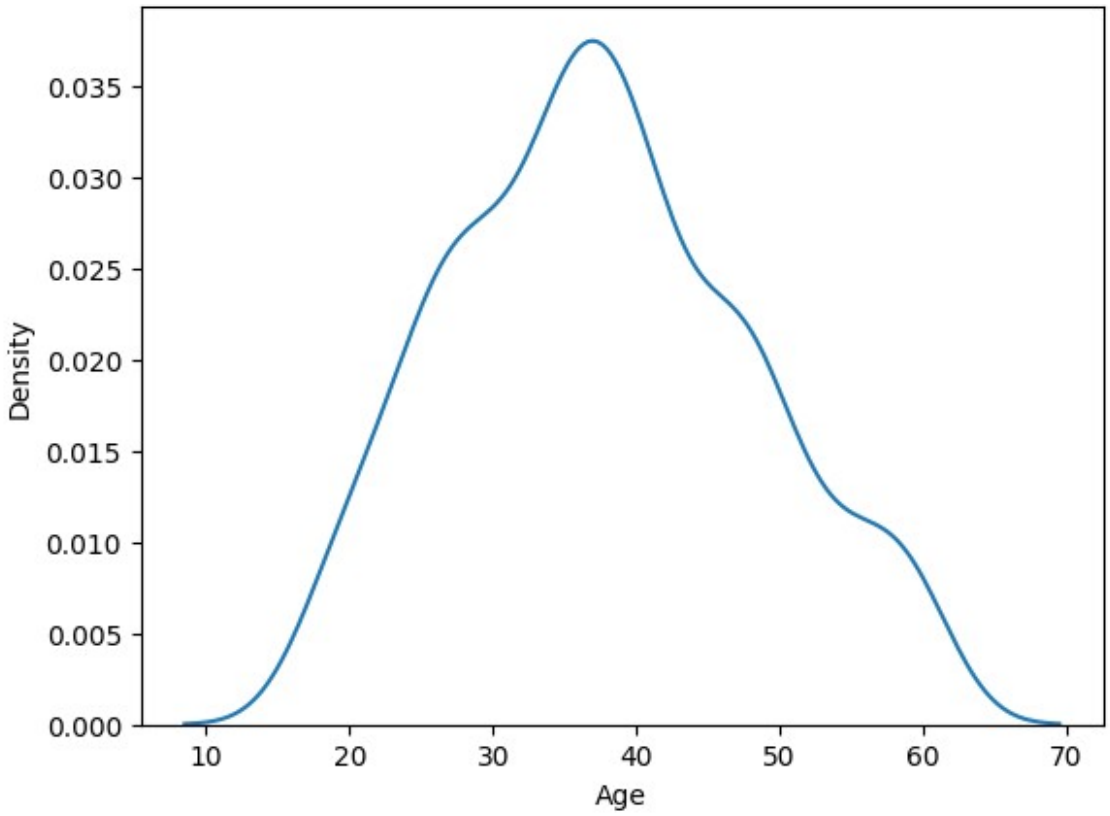
Checking Distribution of age and estimated salary

```
df['Age'].skew()
```

```
0.23133746309642822
```

```
sn.kdeplot(df['Age'])
```

```
<Axes: xlabel='Age', ylabel='Density'>
```

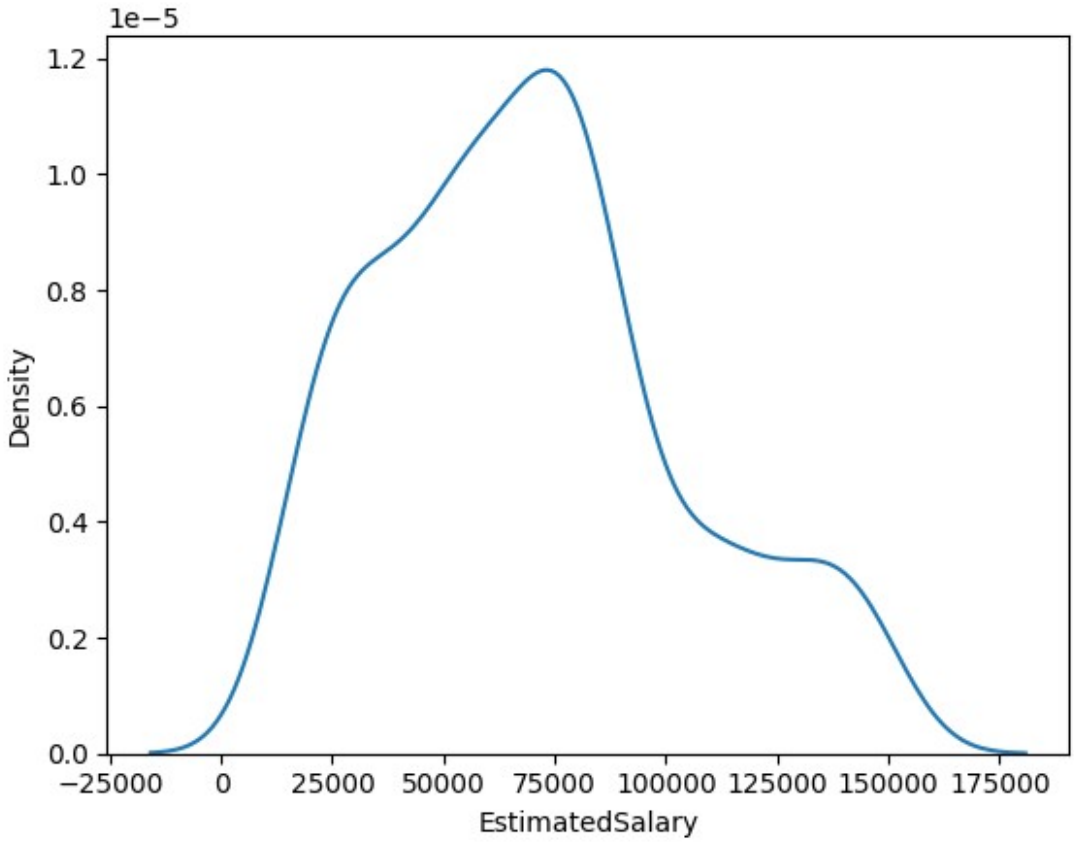



```
df['EstimatedSalary'].skew()
```

```
0.49502362888993623
```

```
sn.kdeplot(df['EstimatedSalary'])
```

```
<Axes: xlabel='EstimatedSalary', ylabel='Density'>
```



age and estimated salary are normally distributed

as its normal apply standard scaler

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)

x_test
```

```
array([[-1.        ,  0.79753468, -1.40447546],
       [-1.        ,  2.07309956,  0.51542886],
       [-1.        , -0.96863208, -0.76450736],
       [-1.        ,  0.99377543,  0.74814454],
       [-1.        , -0.87051171, -1.22993871],
       [-1.        , -0.77239133, -0.24089709],
       [-1.        ,  0.89565505,  1.06812859],
       [-1.        , -0.87051171,  0.36998156],
       [ 1.        ,  0.20881242,  0.13726589],
       [ 1.        ,  0.40505317, -0.15362871],
       [-1.        , -0.28178945, -0.15362871],
       [-1.        ,  1.4843773 , -1.05540195],
```

```
       [-1.        , -1.45923396, -0.64814952],
       [-1.        , -1.75359508, -1.37538601],
       [ 1.        , -0.77239133,  0.4863394 ],
       [ 1.        , -0.28178945,  1.09721805],
       [-1.        ,  1.38625693, -0.93904411],
       [-1.        ,  0.79753468,  0.10817643],
       [ 1.        ,  0.11069205, -0.82268628],
       [ 1.        ,  1.77873843, -0.29907601],
       [-1.        , -1.55735433, -1.25902817],
       [ 1.        , -0.87051171,  0.28271318],
       [ 1.        ,  0.89565505, -1.37538601],
       [-1.        ,  2.07309956,  0.16635535],
       [ 1.        , -1.85171546, -1.49174384],
       [ 1.        ,  1.28813655, -1.37538601],
       [ 1.        ,  0.40505317,  0.28271318],
       [ 1.        , -0.0855487 , -0.50270222],
       [-1.        ,  1.68061805,  1.59173886],
       [-1.        , -1.85171546, -1.43356492],
       [-1.        ,  0.79753468, -0.85177573],
       [ 1.        , -1.85171546, -0.00818141],
       [ 1.        , -0.18366908,  2.14443859],
       [ 1.        , -0.96863208,  0.25362372],
       [ 1.        ,  0.20881242,  1.06812859],
       [ 1.        , -0.28178945,  0.13726589],
       [ 1.        , -0.0855487 , -0.4445233 ],
       [-1.        ,  0.01257167, -0.15362871],
       [ 1.        , -1.16487283, -1.17175979],
       [-1.        , -1.94983583, -0.06636033],
       [ 1.        ,  0.99377543, -1.08449141],
       [-1.        , -1.36111358, -0.4445233 ],
       [ 1.        , -1.94983583, -0.53179168],
       [ 1.        ,  0.89565505, -1.46265438],
       [ 1.        , -1.75359508, -0.61906006],
       [-1.        ,  0.60129393,  1.99899129],
       [ 1.        , -0.87051171, -0.26998655],
       [-1.        , -0.67427095,  0.02090805],
       [ 1.        ,  0.99377543, -0.85177573],
       [-1.        , -0.37990983, -0.79359682],
       [ 1.        , -1.26299321,  0.25362372],
       [-1.        ,  1.4843773 ,  0.3408921 ],
       [-1.        ,  0.01257167, -0.4445233 ],
       [ 1.        , -1.26299321,  0.28271318],
       [-1.        , -0.0855487 ,  0.28271318],
       [-1.        , -1.06675246, -1.14267033],
       [ 1.        ,  2.17121993,  0.92268129],
       [-1.        , -1.16487283,  1.38811264],
       [ 1.        , -0.67427095,  0.10817643],
       [ 1.        , -0.67427095,  0.16635535],
       [-1.        ,  0.3069328 , -0.56088114],
```

```
       [-1.        , -0.28178945, -0.38634438],
       [-1.        ,  1.38625693,  0.57360778],
       [-1.        , -0.96863208,  0.4863394 ],
       [ 1.        , -0.96863208, -0.32816546],
       [-1.        , -1.06675246,  1.94081237],
       [-1.        ,  0.40505317,  0.57360778],
       [-1.        ,  0.89565505,  2.14443859],
       [-1.        ,  0.11069205, -0.32816546],
       [-1.        , -0.4780302 ,  1.24266535],
       [-1.        ,  1.38625693,  1.96990183],
       [ 1.        , -1.85171546,  0.42816048],
       [ 1.        , -1.06675246, -0.35725492],
       [ 1.        , -1.45923396, -1.46265438],
       [ 1.        ,  0.89565505, -1.05540195],
       [-1.        , -0.28178945, -0.5899706 ],
       [ 1.        ,  1.77873843,  1.82445454],
       [-1.        ,  1.58249768, -1.28811763],
       [-1.        , -0.28178945, -0.67723898],
       [-1.        , -0.0855487 ,  0.22453427]])
```

Build Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
y_pred=lr.predict(x_test)

df1=pd.DataFrame({"Actual":y_test, "Predicted":y_pred})

df1.head()
```

```
     Actual  Predicted
209       0          0
280       1          1
33        0          0
210       1          1
93        0          0
```

```
from sklearn.metrics import
confusion_matrix,accuracy_score,precision_score,recall_score

print(confusion_matrix(y_test,y_pred))
```

```
[[50  2]
 [ 7 21]]
```

```
print(accuracy_score(y_test,y_pred))
```

```
0.8875
```

```
(50+21)/(50+2+21+7)
```

```
0.8875
```

```
print(precision_score(y_test,y_pred))
```

```
0.9130434782608695
```

```
print(recall_score(y_test,y_pred))
```

```
0.75
```