social_net_ads

February 1, 2022

https://raw.githubusercontent.com/shivang98/Social-Network-ads-Boost/master/Social_Network_Ads.csv

Raw Data : https://github.com/shivang98/Social-Network-ads-Boost/blob/master/Social_Network_Ads.csv

[ ]:
```
!pip3 install numpy
!pip3 install pandas
!pip3 install matplotlib
!pip3 install seaborn
!pip3 install sklearn
```

```
Requirement already satisfied: numpy in ./assign5_venv/lib/python3.8/site-
packages (1.22.1)
Requirement already satisfied: pandas in ./assign5_venv/lib/python3.8/site-
packages (1.4.0)
Requirement already satisfied: python-dateutil>=2.8.1 in
./assign5_venv/lib/python3.8/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
./assign5_venv/lib/python3.8/site-packages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.18.5; platform_machine != "aarch64" and
platform_machine != "arm64" and python_version < "3.10" in
./assign5_venv/lib/python3.8/site-packages (from pandas) (1.22.1)
Requirement already satisfied: six>=1.5 in ./assign5_venv/lib/python3.8/site-
packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: matplotlib in ./assign5_venv/lib/python3.8/site-
packages (3.5.1)
Requirement already satisfied: fonttools>=4.22.0 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (4.29.0)
Requirement already satisfied: numpy>=1.17 in ./assign5_venv/lib/python3.8/site-
packages (from matplotlib) (1.22.1)
Requirement already satisfied: cycler>=0.10 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (9.0.0)
Requirement already satisfied: python-dateutil>=2.7 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: packaging>=20.0 in
```

```
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib) (3.0.7)
Requirement already satisfied: six>=1.5 in ./assign5_venv/lib/python3.8/site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: seaborn in ./assign5_venv/lib/python3.8/site-
packages (0.11.2)
Requirement already satisfied: pandas>=0.23 in
./assign5_venv/lib/python3.8/site-packages (from seaborn) (1.4.0)
Requirement already satisfied: matplotlib>=2.2 in
./assign5_venv/lib/python3.8/site-packages (from seaborn) (3.5.1)
Requirement already satisfied: scipy>=1.0 in ./assign5_venv/lib/python3.8/site-
packages (from seaborn) (1.7.3)
Requirement already satisfied: numpy>=1.15 in ./assign5_venv/lib/python3.8/site-
packages (from seaborn) (1.22.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
./assign5_venv/lib/python3.8/site-packages (from pandas>=0.23->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
./assign5_venv/lib/python3.8/site-packages (from pandas>=0.23->seaborn) (2021.3)
Requirement already satisfied: pillow>=6.2.0 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(9.0.0)
Requirement already satisfied: packaging>=20.0 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(21.3)
Requirement already satisfied: pyparsing>=2.2.1 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(3.0.7)
Requirement already satisfied: fonttools>=4.22.0 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(4.29.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(1.3.2)
Requirement already satisfied: cycler>=0.10 in
./assign5_venv/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(0.11.0)
Requirement already satisfied: six>=1.5 in ./assign5_venv/lib/python3.8/site-
packages (from python-dateutil>=2.8.1->pandas>=0.23->seaborn) (1.16.0)
Requirement already satisfied: sklearn in ./assign5_venv/lib/python3.8/site-
packages (0.0)
Requirement already satisfied: scikit-learn in
./assign5_venv/lib/python3.8/site-packages (from sklearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
./assign5_venv/lib/python3.8/site-packages (from scikit-learn->sklearn) (3.0.0)
Requirement already satisfied: joblib>=0.11 in
./assign5_venv/lib/python3.8/site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in
```

```
./assign5_venv/lib/python3.8/site-packages (from scikit-learn->sklearn) (1.22.1)
Requirement already satisfied: scipy>=1.1.0 in
./assign5_venv/lib/python3.8/site-packages (from scikit-learn->sklearn) (1.7.3)
```

[ ]:
```
import numpy as NP
import pandas as PD
import seaborn as SNS
import matplotlib.pyplot as MPLOT
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import MinMaxScaler
```

[ ]:
```
DF = PD.read_csv("https://raw.githubusercontent.com/shivang98/
 ↪Social-Network-ads-Boost/master/Social_Network_Ads.csv")
print("Shape :",DF.shape)
print("Size :",DF.size)
DF.sample   (10)
```

```
Shape : (400, 5)
Size : 2000
```

[ ]:

	User ID	Gender	Age	EstimatedSalary	Purchased
355	15606472	Male	60	34000	1
34	15724858	Male	27	90000	0
2	15668575	Female	26	43000	0
173	15581654	Female	34	43000	0
273	15589449	Male	39	106000	1
127	15745232	Male	26	32000	0
281	15685536	Male	35	61000	0
20	15649487	Male	45	22000	1
376	15596984	Female	46	74000	0
317	15684861	Male	35	55000	0

[ ]:
```
DF.dtypes
```

[ ]:
```
User ID             int64
Gender             object
Age                 int64
EstimatedSalary     int64
Purchased           int64
dtype: object
```

[ ]:
```
DF.describe()
```

[ ]:

	Gender	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000	400.000000

```
mean      0.490000   37.655000    69742.500000    0.357500
std       0.500526   10.482877    34096.960282    0.479864
min       0.000000   18.000000    15000.000000    0.000000
25%       0.000000   29.750000    43000.000000    0.000000
50%       0.000000   37.000000    70000.000000    0.000000
75%       1.000000   46.000000    88000.000000    1.000000
max       1.000000   60.000000   150000.000000    1.000000
```

[ ]:
```
DF.isna().sum()
```

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

[ ]:
```
DF.isnull().sum()
```

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

0.0.1 Dropping unwanted Information

We will drop User ID Column as it is not useful in classification

[ ]:
```
DF.drop(columns=['User ID'], inplace=True)
DF.sample(10)
```

[ ]:

	Gender	Age	EstimatedSalary	Purchased
317	Male	35	55000	0
282	Male	37	70000	1
31	Female	27	137000	1
372	Female	39	73000	0
342	Female	38	65000	0
341	Male	35	75000	0
109	Female	38	80000	0
94	Female	29	83000	0
36	Female	33	28000	0
174	Female	34	72000	0

0.0.2 Converting Data to Category

Gender == Categorical Data –> So will convert into number (encoding)

[ ]:
```
DF.dtypes
```

[ ]:
```
Gender             object
Age                 int64
EstimatedSalary     int64
Purchased           int64
dtype: object
```

[ ]:
```
DF['Gender'] = DF['Gender'].astype('category')
DF.dtypes
```

[ ]:
```
Gender             category
Age                   int64
EstimatedSalary       int64
Purchased             int64
dtype: object
```

[ ]:
```
DF['Gender'] = DF['Gender'].cat.codes
DF.sample(10)
```

[ ]:

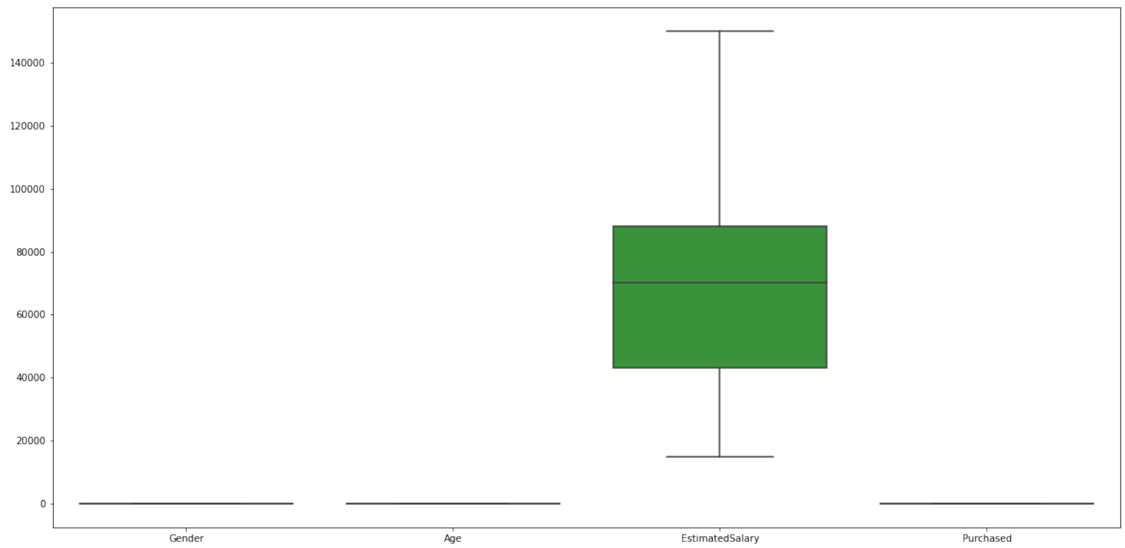
	Gender	Age	EstimatedSalary	Purchased
381	1	48	33000	1
22	1	48	41000	1
382	0	44	139000	1
36	0	33	28000	0
82	1	20	49000	0
364	1	42	104000	1
161	1	25	90000	0
335	0	36	54000	0
334	1	57	60000	1
359	1	42	54000	0

0.0.3 Removing Outliers

Now will detect and remove oultiers

[ ]:
```
fig, ax = MPLOT.subplots(figsize=(20,10))
SNS.boxplot(data=DF, ax=ax)
```

[ ]:
```
<AxesSubplot:>
```



[ ]:
```
def Remove_Outlier(DataFrame, Col_Name) :
    Upper_Threshold = DataFrame[Col_Name].mean() + 3*DataFrame[Col_Name].std()
    Lower_Threshold = DataFrame[Col_Name].mean() - 3*DataFrame[Col_Name].std()
    Count = DataFrame[(DataFrame[Col_Name] >= Upper_Threshold ) |␣
 ↪(DataFrame[Col_Name] <= Lower_Threshold)][Col_Name].count()
    print("[ * ] In", str(Col_Name), "->", "High :" ,Upper_Threshold, "| Low :
 ↪", Lower_Threshold, "| -> Outliers Detected :", Count)
    return DataFrame[((DataFrame[Col_Name] >= Lower_Threshold ) &␣
 ↪(DataFrame[Col_Name] <= Upper_Threshold))]
```

[ ]:
```
Remove_Outlier( DF, 'Gender' )
Remove_Outlier( DF, 'Age' )
Remove_Outlier( DF, 'EstimatedSalary' )
Remove_Outlier( DF, 'Purchased' )
```

```
[ * ] In Gender -> High : 1.9915781172171041 | Low : -1.0115781172171041 | ->
Outliers Detected : 0
[ * ] In Age -> High : 69.10362979192374 | Low : 6.206370208076258 | -> Outliers
Detected : 0
[ * ] In EstimatedSalary -> High : 172033.38084727435 | Low :
-32548.380847274355 | -> Outliers Detected : 0
[ * ] In Purchased -> High : 1.797091890790607 | Low : -1.0820918907906072 | ->
Outliers Detected : 0
```

[ ]:

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0

```
4          1    19             76000          0
..        …   …              …        …
395        0    46             41000          1
396        1    51             23000          1
397        0    50             20000          1
398        1    36             33000          0
399        0    49             36000          1

[400 rows x 4 columns]
```

---

1 Corelation Matrix

[ ]:
```python
fig, ax = MPLOT.subplots(figsize=(15,10))
SNS.heatmap(DF.corr(), annot = True, ax=ax)
```

[ ]:
```
<AxesSubplot:>
```

---

2 Train Model

[ ]:
```python
X = DF[['Age', 'EstimatedSalary']]
Y = DF['Purchased']

X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = train_test_split(
    X,
    Y,
    test_size=0.2,
    random_state=42)


model = LogisticRegression()
model.fit(X_TRAIN, Y_TRAIN)
output = model.predict(X_TEST)


print("Mean Absolute Error :", mean_absolute_error(Y_TEST, output))
print("Model Score (Accuracy) :", model.score(X_TEST, Y_TEST))
```

```
Mean Absolute Error : 0.35
Model Score (Accuracy) : 0.65
```

---

3 Data Normalization and Training Model

[ ]:
```python
DF.describe()
```

[ ]:

	Gender	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000	400.000000
mean	0.490000	37.655000	69742.500000	0.357500
std	0.500526	10.482877	34096.960282	0.479864
min	0.000000	18.000000	15000.000000	0.000000
25%	0.000000	29.750000	43000.000000	0.000000
50%	0.000000	37.000000	70000.000000	0.000000
75%	1.000000	46.000000	88000.000000	1.000000
max	1.000000	60.000000	150000.000000	1.000000

[ ]:
```python
X = DF[['Age', 'EstimatedSalary']]
Y = DF['Purchased']

X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

# fit scaler on training data
norm = MinMaxScaler().fit(X_TRAIN)
```

```python
# transform training data
X_TRAIN = norm.transform(X_TRAIN)

# fit scaler on training data
norm = MinMaxScaler().fit(X_TEST)

# transform training data
X_TEST = norm.transform(X_TEST)


model = LogisticRegression()
model.fit(X_TRAIN, Y_TRAIN)
output = model.predict(X_TEST)


print("Mean Absolute Error :", mean_absolute_error(Y_TEST, output))
print("Model Score (Accuracy) :", model.score(X_TEST, Y_TEST))
```

```
Mean Absolute Error : 0.125
Model Score (Accuracy) : 0.875
```