ASSIGNMENT 5

Import the required libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Load the dataset

In [ ]:
```
df= pd.read_csv('Social_Network_Ads.csv')
```

In [ ]:
```
df.shape
```

Out[5]:
```
(400, 5)
```

In [ ]:
```
df.head()
```

Out[6]:

	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

Drop the column User ID

In [ ]:
```
df.drop(['User ID'],axis=1,inplace=True)
```

In [ ]:
```
df.head()
```

Out[8]:

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

In [ ]:
```
df.Purchased.value_counts()
```

Out[9]:
```
0    257
1    143
Name: Purchased, dtype: int64
```

In [ ]:
```
df.Gender.value_counts()
```

Out[10]:
```
Female    204
Male      196
Name: Gender, dtype: int64
```

In [ ]:
```
df.dtypes
```

Out[11]:
```
Gender             object
Age                 int64
EstimatedSalary     int64
Purchased           int64
dtype: object
```

Data Preprocessing

In [ ]:
```
df.isnull().sum()
```

Out[12]:
```
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

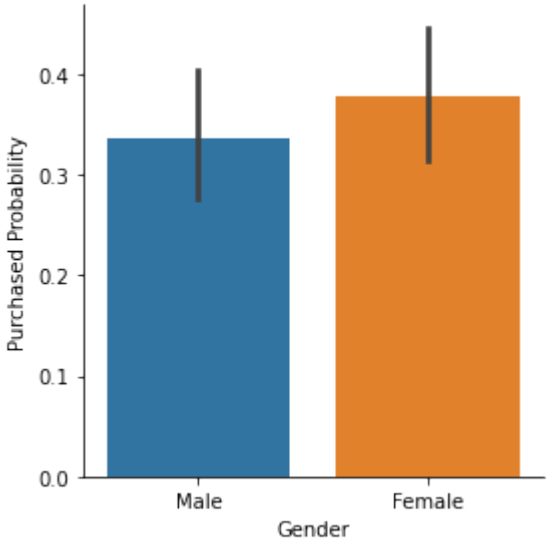
In [ ]:
```
df.describe()
```

Out[13]:

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

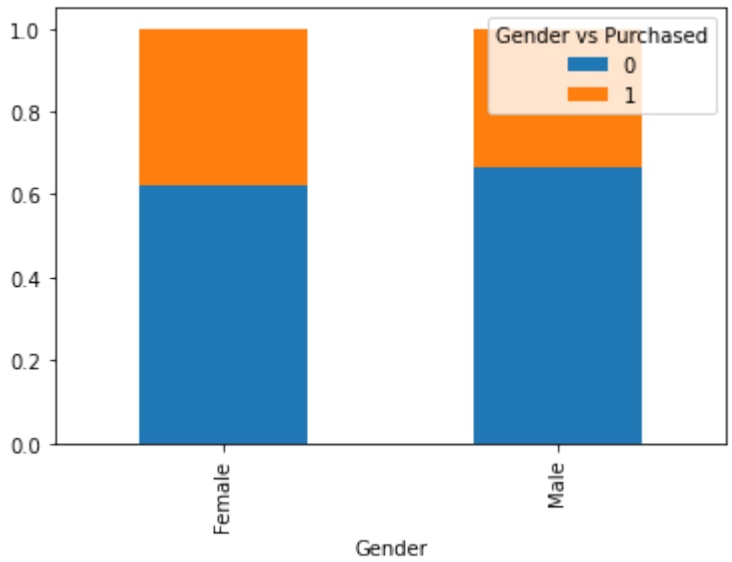
In [ ]:
```
g = sns.catplot(x = "Gender",y = "Purchased",data = df,kind = "bar",height =
g.set_ylabels("Purchased Probability")
plt.show
```

Out[14]:
```
<function matplotlib.pyplot.show>
```



In [ ]:
```
M2 = pd.crosstab(df.Gender, df.Purchased, normalize='index')
print(M2)
M2.plot.bar(figsize=(6,4),stacked=True)
plt.legend(title='Gender vs Purchased', loc='upper right')
plt.show()
```

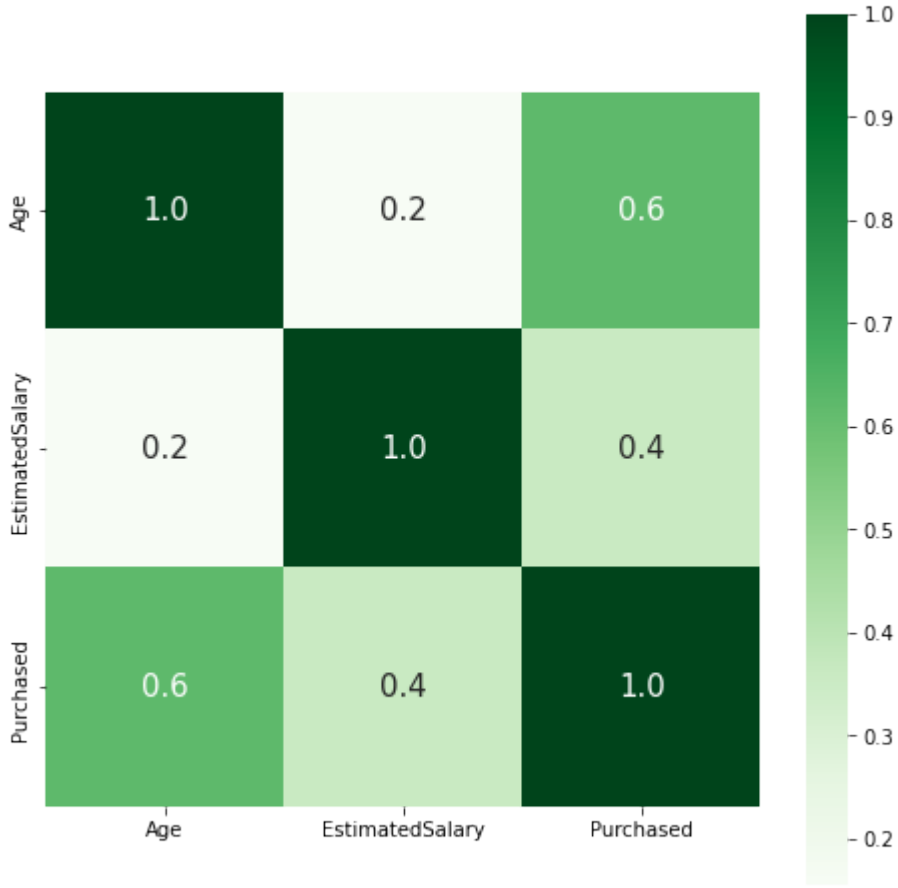
```
Purchased         0         1
Gender
Female     0.622549  0.377451
Male       0.663265  0.336735
```



In [ ]:
```
corr = df.corr()
print(corr.shape)
plt.figure(figsize=(8,8))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws=
```

```
(3, 3)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4def4b650>
```



Type *Markdown* and LaTeX: α^2

Get Input Variable into X and Outcome variable into Y

In [ ]:
```
X=df.drop(['Gender','Purchased'],axis=1)
Y= df['Purchased']
X.head()
```

Out[17]:

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

Split the data into Train set and Test set

In [ ]:
```
from sklearn.model_selection import train_test_split
# Shuffle and split the data into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rar

# Success
print("Training and testing split was successful.")
```

```
Training and testing split was successful.
```

Build the model

In [ ]:
```
from sklearn.linear_model import LogisticRegression
basemodel= LogisticRegression()
basemodel.fit(X_train,y_train)
print("Training accuracy:", basemodel.score(X_train,y_train)*100)
```

```
Training accuracy: 64.0625
```

Make predictions on test data

In [ ]:
```
y_predict= basemodel.predict(X_test)
print("Testing accuracy:", basemodel.score(X_test,y_test)*100)
```

```
Testing accuracy: 65.0
```

Normalize the data using Min Max Normalization or any other technique

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

In [ ]:
```
X=df[['Age','EstimatedSalary']]
X_scaled= scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0
print("Training and testing split was successful.")
```

```
Training and testing split was successful.
```

In [ ]:
```
model= LogisticRegression()
model.fit(X_train,y_train)
y_predict= model.predict(X_test)
print("Training accuracy:", model.score(X_train,y_train)*100)
print("Testing accuracy:", model.score(X_test,y_test)*100)
```

```
Training accuracy: 80.9375
Testing accuracy: 87.5
```

Measure the performance of the model

1. Measure the accuracy
2. Measure the perormance using Precision, Recall, Fscore, Support etc.

In [ ]:
```
from sklearn.metrics import accuracy_score
Acc=accuracy_score(y_test,y_predict)
print(Acc)
```

```
0.875
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_predict)
print(cm)
```

```
[[51  1]
 [ 9 19]]
```

In [ ]:
```
from sklearn.metrics import precision_recall_fscore_support
prf= precision_recall_fscore_support(y_test,y_predict)
print('precision:',prf[0])
print('Recall:',prf[1])
print('fscore:',prf[2])
print('support:',prf[3])
```

```
precision: [0.85 0.95]
Recall: [0.98076923 0.67857143]
fscore: [0.91071429 0.79166667]
support: [52 28]
```

In [ ]:
```
from sklearn.metrics import classification_report
cr= classification_report(y_test,y_predict)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.85      0.98      0.91        52
           1       0.95      0.68      0.79        28

    accuracy                           0.88        80
   macro avg       0.90      0.83      0.85        80
weighted avg       0.89      0.88      0.87        80
```