This project basically tells wheather or not the user purchase some good on the basis of their age,salary,gender.

Importing the libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing the dataset and let's do some analysis

In [2]:

```
df = pd.read_csv('C:/Users/dell/Downloads/Social_Network_Ads.csv')
df.head()
```

Out[2]:

	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

In [3]:

```
## Let's check for the null values

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

Out[3]:

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

In [4]:

```
## Let's check for the datatypes

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
```

In [5]:

```
## Here we need to work with two columns -- User ID and Gender
## 1. We don't see any value of user id so we need to drop it.
## 2. As machine learning model cannot take text values so we need to work with gender va
lues.

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

Out[5]:

	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
...	...	...	...	...
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

400 rows × 4 columns

In [6]:

```
## To accomplish the second task we will be using one hot endoder because we are deling w
ith Nominal categorical data

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

Out[6]:

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

In [7]:

```
Gender_value = pd.get_dummies(df.Gender)
Gender_value
```

Out[7]:

	Female	Male
0	0	1
1	0	1
2	1	0
3	1	0
4	0	1
...	...	...
395	1	0
396	0	1
397	1	0
398	0	1
399	1	0

400 rows × 2 columns

In [8]:

```
Gender_value.drop(['Male'],axis = 'columns',inplace = True)
Gender_value
```

Out[8]:

	Female
0	0
1	0
2	1
3	1
4	0
...	...
395	1
396	0
397	1
398	0
399	1

400 rows × 1 columns

In [9]:

```
df.drop(['Gender'],axis = 'columns',inplace = True)
df
```

Out[9]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

In [10]:

```
## By using one hot loading encoder we might rul into multicolliniarity problem
## so let's drop a column of Male but befor that let's merge the value

merge_df = pd.concat([df,Gender_value],axis = 'columns')
merge_df
```

Out[10]:

	Age	EstimatedSalary	Purchased	Female
0	19	19000	0	0
1	35	20000	0	0
2	26	43000	0	1
3	27	57000	0	1
4	19	76000	0	0
...	...	...	...	...
395	46	41000	1	1
396	51	23000	1	0
397	50	20000	1	1
398	36	33000	0	0
399	49	36000	1	1

400 rows × 4 columns

merge_df.shape()

Splitting the datset into Training and Testing set

In [25]:

```
y = merge_df['Purchased']
```

In [28]:

```
x = merge_df.drop('Purchased',axis =1)
```

In [29]:

```
x
```

Out[29]:

	Age	EstimatedSalary	Female
0	19	19000	0
1	35	20000	0
2	26	43000	1
3	27	57000	1
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	0
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

In [31]:

```
#from sklearn.model_selection import train_test_split

#training_data, testing_data = train_test_split(merge_df, test_size=0.2, random_state=25)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, random_state= 0
```

```
)
```

In [32]:

```
x_train.shape
```

Out[32]:

```
(280, 3)
```

In [33]:

```
y_train.shape
```

Out[33]:

```
(280,)
```

Feature Scaling

In [44]:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [45]:

```
print(x_train[0:10])
```

```
[[-1.1631724  -1.5849703  -0.99288247]
 [ 2.17018137  0.93098672 -0.99288247]
 [ 0.0133054   1.22017719  1.00716855]
 [ 0.20938504  1.07558195 -0.99288247]
 [ 0.40546467 -0.48604654  1.00716855]
 [-0.28081405 -0.31253226 -0.99288247]
 [ 0.99370357 -0.8330751  -0.99288247]
 [ 0.99370357  1.8563962   1.00716855]
 [ 0.0133054   1.24909623  1.00716855]
 [-0.86905295  2.26126285 -0.99288247]]
```

Training the Random Forest Classification model on the Training set

In [46]:

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators= 5,criterion = 'entropy',random_state=0
)

classifier.fit(x_train,y_train)
```

Out[46]:

```
RandomForestClassifier(criterion='entropy', n_estimators=5, random_state=0)
```

Predicting a new result

In [66]:

```
print(classifier.predict([[55,50000,0]]))
```

```
[1]
```

Predicting the Test set results

In [69]:

```
y_pred = classifier.predict(x_test)
len(y_pred)
```

Out[69]:

```
120
```

Making the confusion matrix

In [57]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[70  9]
 [ 4 37]]
```

Out[57]:

```
0.8916666666666667
```