In [8]:

```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
import matplotlib.pyplot as pl
import math
```

In [9]:

```
data=pd.read_csv("C:/Users/Admin/Downloads/Social_Network_Ads.csv")
```

In [10]:
```
data
```

Out[10]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19.0	19000.0	0
1	15810944	Male	35.0	20000.0	0
2	15668575	Female	26.0	43000.0	0
3	15603246	Female	27.0	57000.0	0
4	15804002	Male	19.0	76000.0	0
5	15728773	Male	27.0	58000.0	0
6	15598044	Female	27.0	84000.0	0
7	15694829	Female	32.0	150000.0	1
8	15600575	Male	25.0	33000.0	0
9	15727311	Female	35.0	65000.0	0
10	15570769	Female	26.0	80000.0	0
11	15606274	Female	26.0	52000.0	0
12	15746139	Male	20.0	86000.0	0
13	15704987	Male	32.0	18000.0	0
14	15628972	Male	18.0	82000.0	0
15	15697686	Male	29.0	80000.0	0
16	15733883	Male	47.0	25000.0	1
17	15617482	Male	45.0	26000.0	1
18	15704583	Male	46.0	28000.0	1
19	15621083	Female	48.0	29000.0	1
20	15649487	Male	45.0	22000.0	1
21	15736760	Female	47.0	49000.0	1
22	15714658	Male	48.0	41000.0	1
23	15599081	Female	45.0	22000.0	1
24	15705113	Male	46.0	23000.0	1
25	15631159	Male	47.0	20000.0	1
26	15792818	Male	49.0	28000.0	1
27	15633531	Female	47.0	30000.0	1
28	15744529	Male	29.0	43000.0	0
29	15669656	Male	31.0	18000.0	0
...	...	...	...	...	...
370	15611430	Female	60.0	46000.0	1
371	15774744	Male	60.0	83000.0	1
372	15629885	Female	39.0	73000.0	0
373	15708791	Male	59.0	130000.0	1
374	15793890	Female	37.0	80000.0	0
375	15646091	Female	46.0	32000.0	1

	User ID	Gender	Age	EstimatedSalary	Purchased
376	15596984	Female	46.0	74000.0	0
377	15800215	Female	42.0	53000.0	0
378	15577806	Male	41.0	87000.0	1
379	15749381	Female	58.0	23000.0	1
380	15683758	Male	42.0	64000.0	0
381	15670615	Male	48.0	33000.0	1
382	15715622	Female	44.0	139000.0	1
383	15707634	Male	49.0	28000.0	1
384	15806901	Female	57.0	33000.0	1
385	15775335	Male	56.0	60000.0	1
386	15724150	Female	49.0	39000.0	1
387	15627220	Male	39.0	71000.0	0
388	15672330	Male	47.0	34000.0	1
389	15668521	Female	48.0	35000.0	1
390	15807837	Male	48.0	33000.0	1
391	15592570	Male	47.0	23000.0	1
392	15748589	Female	45.0	45000.0	1
393	15635893	Male	60.0	42000.0	1
394	15757632	Female	39.0	59000.0	0
395	15691863	Female	46.0	41000.0	1
396	15706071	Male	51.0	23000.0	1
397	15654296	Female	50.0	20000.0	1
398	15755018	Male	36.0	33000.0	0
399	15594041	Female	49.0	36000.0	1

400 rows × 5 columns

In [11]:
```
data.shape
```

Out[11]:
```
(400, 5)
```

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
User ID            400 non-null int64
Gender             400 non-null object
Age                393 non-null float64
EstimatedSalary    390 non-null float64
Purchased          400 non-null int64
dtypes: float64(2), int64(2), object(1)
memory usage: 15.7+ KB
```

In [13]:
```
data.describe()
```

Out[13]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	393.000000	390.000000	400.000000
mean	1.569154e+07	37.758270	69758.974359	0.357500
std	7.165832e+04	10.534689	34063.427288	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	30.000000	43000.000000	0.000000
50%	1.569434e+07	37.000000	69500.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [14]:
```
data.isnull().sum()
```

Out[14]:
```
User ID             0
Gender              0
Age                 7
EstimatedSalary    10
Purchased           0
dtype: int64
```

In [16]:
```
data['Age'].fillna(data['Age'].mean(),inplace=True)
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
User ID             0
Gender              0
Age                 0
EstimatedSalary    10
Purchased           0
dtype: int64
```

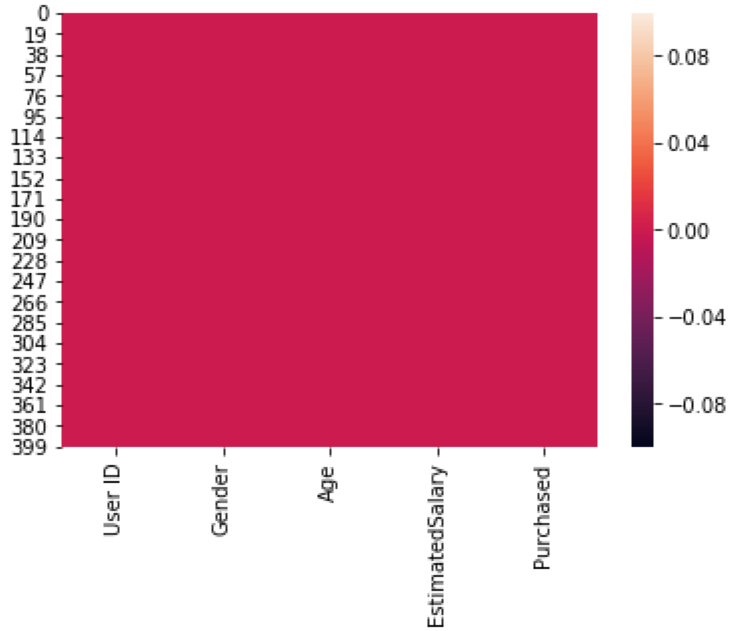
In [18]:
```
data['EstimatedSalary'].fillna(data['EstimatedSalary'].mean(),inplace=True)
```

In [19]:
```
data.isnull().sum()
```

Out[19]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

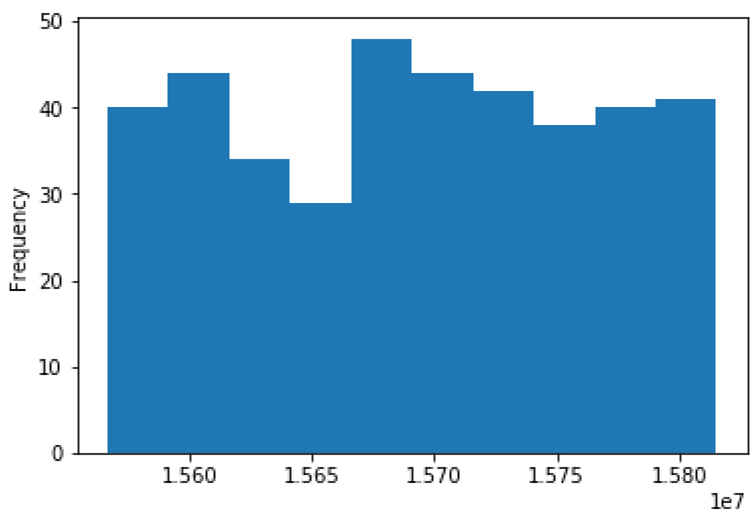
In [20]:
```python
sns.heatmap(data.isnull())
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0xb281278>

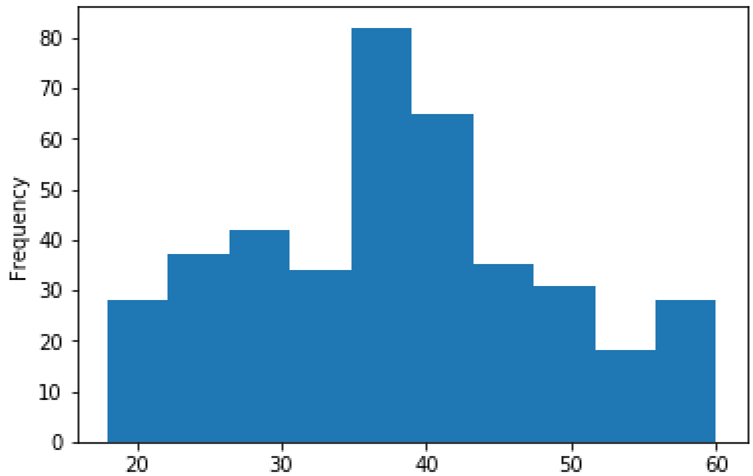


In [21]:
```python
data['User ID'].plot.hist()
```

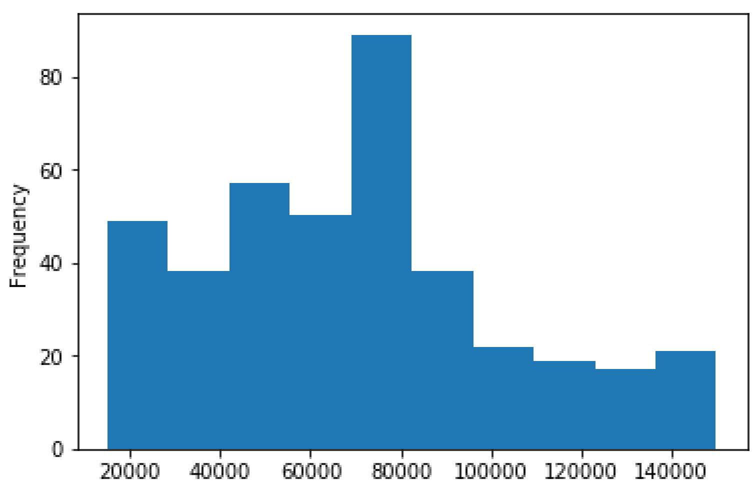
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x5c1a438>



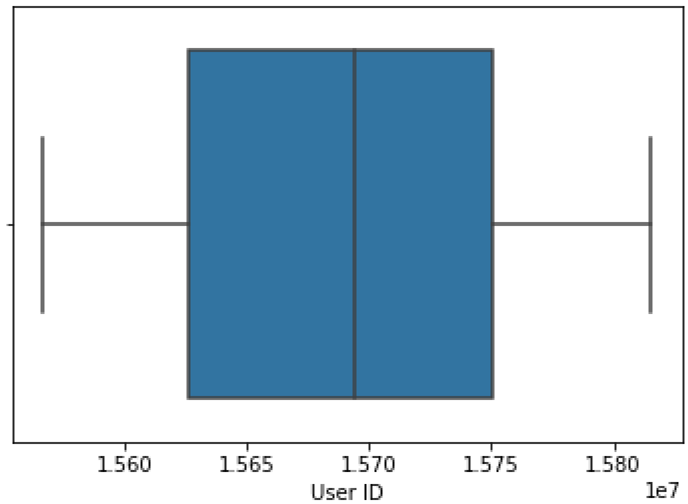
In [22]:
```
data['Age'].plot.hist()
```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x5c9e518>



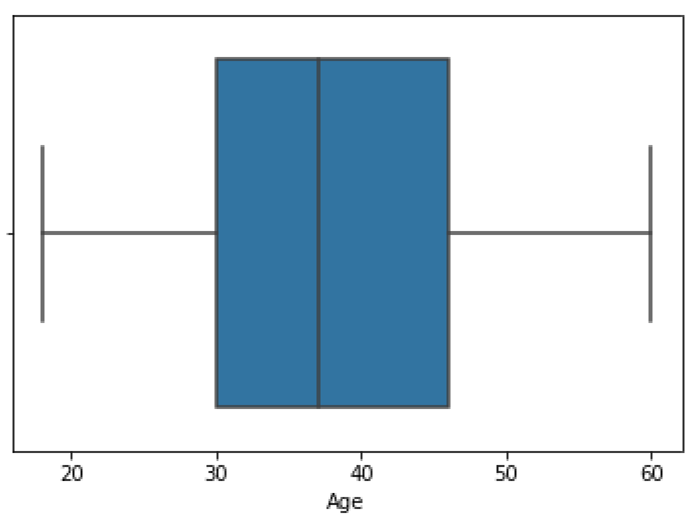
In [23]:
```
data['EstimatedSalary'].plot.hist()
```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x5d22320>



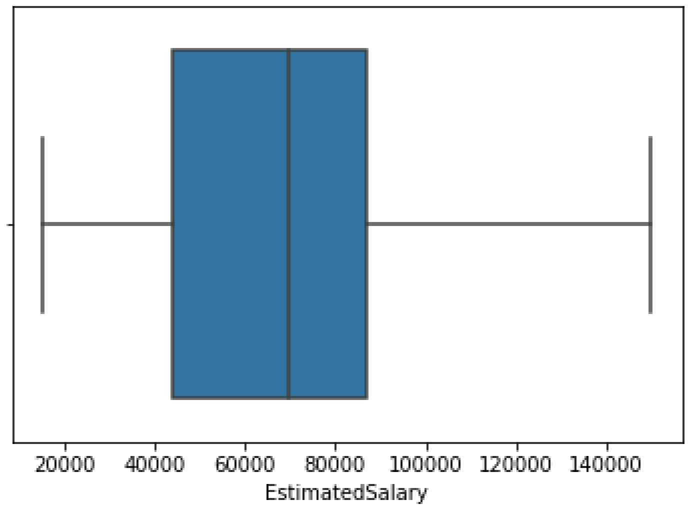
In [25]:
```
sns.boxplot("User ID",data=data)
```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0xb11b2e8>



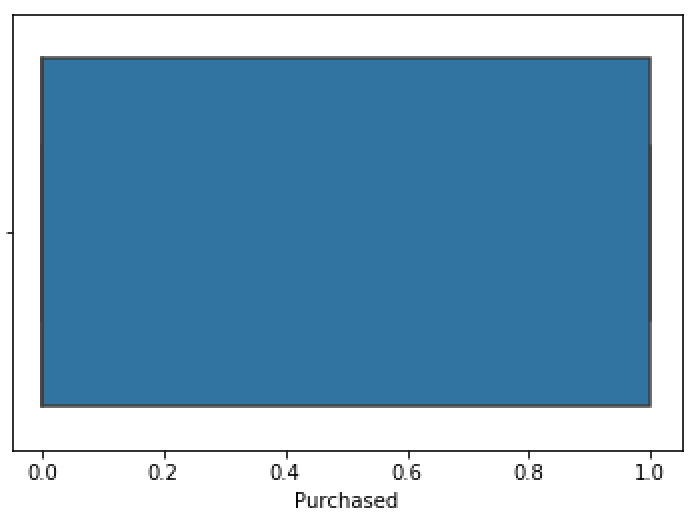
In [24]:
```
sns.boxplot(x="Age",data=data)
```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x5d8f940>



In [28]:
```
sns.boxplot(x="EstimatedSalary",data=data)
```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0xbbb75c0>

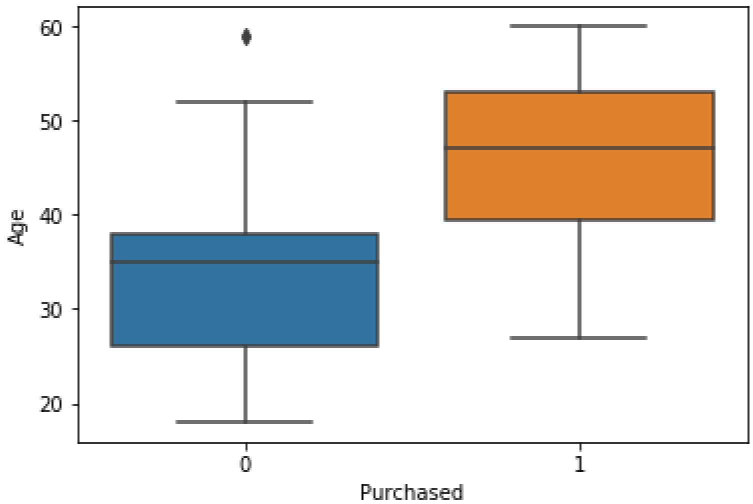


In [29]:
```
sns.boxplot(x="Purchased",data=data)
```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x5d8fb38>



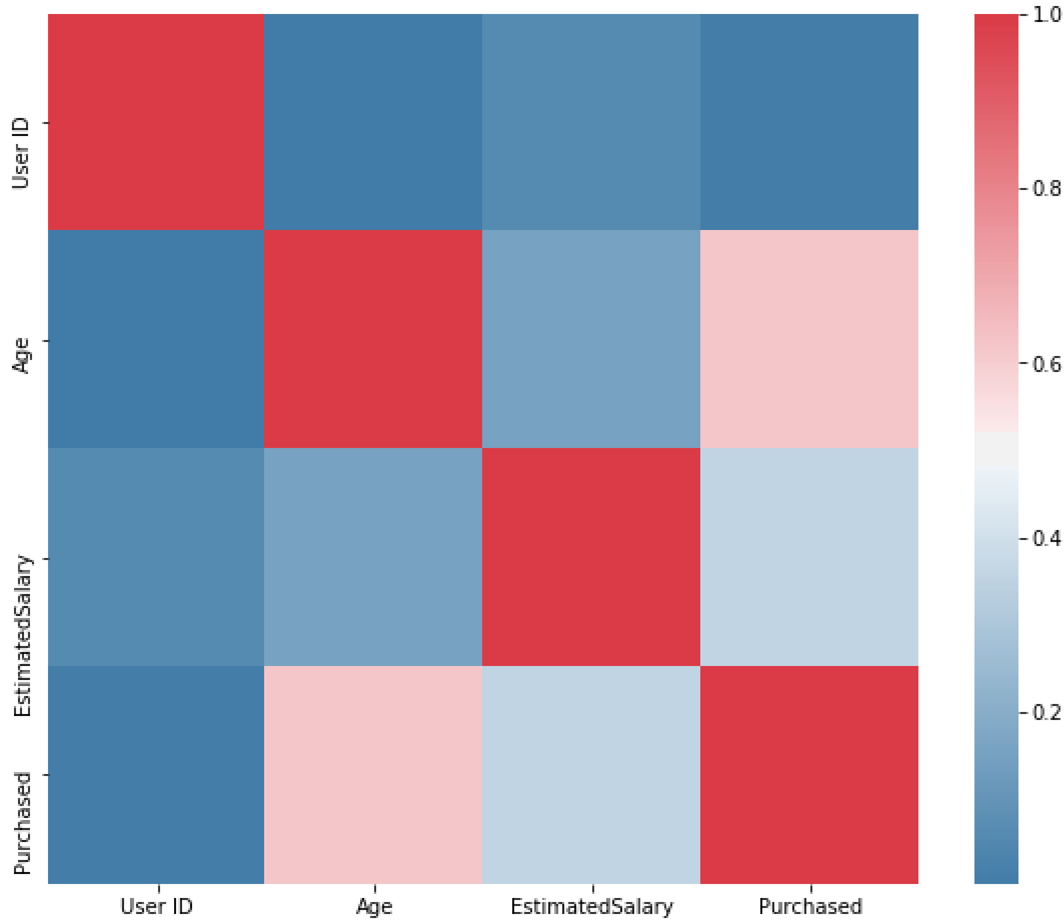
In [30]:
```
sns.boxplot(x="Purchased",y="Age",data=data)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0xbb0eeb8>



In [34]:
```
f, ax=pl.subplots(figsize=(10,8))
corr=data.corr()
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(240,10,
```

Out[34]: `<matplotlib.axes._subplots.AxesSubplot at 0xbc63898>`



In [36]:

```
Gender=data['Gender']=pd.get_dummies(data['Gender'],drop_first=True)
```

In [37]:
```
Gender
```

Out[37]:

	Male
0	1
1	1
2	0
3	0
4	1
5	1
6	0
7	0
8	1
9	0
10	0
11	0
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	0
20	1
21	0
22	1
23	0
24	1
25	1
26	1
27	0
28	1
29	1
...	...
370	0
371	1
372	0
373	1
374	0
375	0

	Male
376	0
377	0
378	1
379	0
380	1
381	1
382	0
383	1
384	0
385	1
386	0
387	1
388	1
389	0
390	1
391	1
392	0
393	1
394	0
395	0
396	1
397	0
398	1
399	0

400 rows × 1 columns

In [38]:
```
data=pd.concat([data,Gender],axis=1)
```

In [39]:
```
data
```

Out[39]:

	User ID	Gender	Age	EstimatedSalary	Purchased	Male
0	15624510	1	19.0	19000.0	0	1
1	15810944	1	35.0	20000.0	0	1
2	15668575	0	26.0	43000.0	0	0
3	15603246	0	27.0	57000.0	0	0
4	15804002	1	19.0	76000.0	0	1
5	15728773	1	27.0	58000.0	0	1
6	15598044	0	27.0	84000.0	0	0
7	15694829	0	32.0	150000.0	1	0
8	15600575	1	25.0	33000.0	0	1
9	15727311	0	35.0	65000.0	0	0
10	15570769	0	26.0	80000.0	0	0
11	15606274	0	26.0	52000.0	0	0
12	15746139	1	20.0	86000.0	0	1
13	15704987	1	32.0	18000.0	0	1
14	15628972	1	18.0	82000.0	0	1
15	15697686	1	29.0	80000.0	0	1
16	15733883	1	47.0	25000.0	1	1
17	15617482	1	45.0	26000.0	1	1
18	15704583	1	46.0	28000.0	1	1
19	15621083	0	48.0	29000.0	1	0
20	15649487	1	45.0	22000.0	1	1
21	15736760	0	47.0	49000.0	1	0
22	15714658	1	48.0	41000.0	1	1
23	15599081	0	45.0	22000.0	1	0
24	15705113	1	46.0	23000.0	1	1
25	15631159	1	47.0	20000.0	1	1
26	15792818	1	49.0	28000.0	1	1
27	15633531	0	47.0	30000.0	1	0
28	15744529	1	29.0	43000.0	0	1
29	15669656	1	31.0	18000.0	0	1
...	...	...	...	...	...	...
370	15611430	0	60.0	46000.0	1	0
371	15774744	1	60.0	83000.0	1	1
372	15629885	0	39.0	73000.0	0	0
373	15708791	1	59.0	130000.0	1	1
374	15793890	0	37.0	80000.0	0	0
375	15646091	0	46.0	32000.0	1	0

	User ID	Gender	Age	EstimatedSalary	Purchased	Male
376	15596984	0	46.0	74000.0	0	0
377	15800215	0	42.0	53000.0	0	0
378	15577806	1	41.0	87000.0	1	1
379	15749381	0	58.0	23000.0	1	0
380	15683758	1	42.0	64000.0	0	1
381	15670615	1	48.0	33000.0	1	1
382	15715622	0	44.0	139000.0	1	0
383	15707634	1	49.0	28000.0	1	1
384	15806901	0	57.0	33000.0	1	0
385	15775335	1	56.0	60000.0	1	1
386	15724150	0	49.0	39000.0	1	0
387	15627220	1	39.0	71000.0	0	1
388	15672330	1	47.0	34000.0	1	1
389	15668521	0	48.0	35000.0	1	0
390	15807837	1	48.0	33000.0	1	1
391	15592570	1	47.0	23000.0	1	1
392	15748589	0	45.0	45000.0	1	0
393	15635893	1	60.0	42000.0	1	1
394	15757632	0	39.0	59000.0	0	0
395	15691863	0	46.0	41000.0	1	0
396	15706071	1	51.0	23000.0	1	1
397	15654296	0	50.0	20000.0	1	0
398	15755018	1	36.0	33000.0	0	1
399	15594041	0	49.0	36000.0	1	0

400 rows × 6 columns

In [40]:
```
data=data.drop(['Gender'],axis=1)
```

In [41]:
```
data
```

Out[41]:

	User ID	Age	EstimatedSalary	Purchased	Male
0	15624510	19.0	19000.0	0	1
1	15810944	35.0	20000.0	0	1
2	15668575	26.0	43000.0	0	0
3	15603246	27.0	57000.0	0	0
4	15804002	19.0	76000.0	0	1
5	15728773	27.0	58000.0	0	1
6	15598044	27.0	84000.0	0	0
7	15694829	32.0	150000.0	1	0
8	15600575	25.0	33000.0	0	1
9	15727311	35.0	65000.0	0	0
10	15570769	26.0	80000.0	0	0
11	15606274	26.0	52000.0	0	0
12	15746139	20.0	86000.0	0	1
13	15704987	32.0	18000.0	0	1
14	15628972	18.0	82000.0	0	1
15	15697686	29.0	80000.0	0	1
16	15733883	47.0	25000.0	1	1
17	15617482	45.0	26000.0	1	1
18	15704583	46.0	28000.0	1	1
19	15621083	48.0	29000.0	1	0
20	15649487	45.0	22000.0	1	1
21	15736760	47.0	49000.0	1	0
22	15714658	48.0	41000.0	1	1
23	15599081	45.0	22000.0	1	0
24	15705113	46.0	23000.0	1	1
25	15631159	47.0	20000.0	1	1
26	15792818	49.0	28000.0	1	1
27	15633531	47.0	30000.0	1	0
28	15744529	29.0	43000.0	0	1
29	15669656	31.0	18000.0	0	1
...	...	...	...	...	...
370	15611430	60.0	46000.0	1	0
371	15774744	60.0	83000.0	1	1
372	15629885	39.0	73000.0	0	0
373	15708791	59.0	130000.0	1	1
374	15793890	37.0	80000.0	0	0
375	15646091	46.0	32000.0	1	0

	User ID	Age	EstimatedSalary	Purchased	Male
376	15596984	46.0	74000.0	0	0
377	15800215	42.0	53000.0	0	0
378	15577806	41.0	87000.0	1	1
379	15749381	58.0	23000.0	1	0
380	15683758	42.0	64000.0	0	1
381	15670615	48.0	33000.0	1	1
382	15715622	44.0	139000.0	1	0
383	15707634	49.0	28000.0	1	1
384	15806901	57.0	33000.0	1	0
385	15775335	56.0	60000.0	1	1
386	15724150	49.0	39000.0	1	0
387	15627220	39.0	71000.0	0	1
388	15672330	47.0	34000.0	1	1
389	15668521	48.0	35000.0	1	0
390	15807837	48.0	33000.0	1	1
391	15592570	47.0	23000.0	1	1
392	15748589	45.0	45000.0	1	0
393	15635893	60.0	42000.0	1	1
394	15757632	39.0	59000.0	0	0
395	15691863	46.0	41000.0	1	0
396	15706071	51.0	23000.0	1	1
397	15654296	50.0	20000.0	1	0
398	15755018	36.0	33000.0	0	1
399	15594041	49.0	36000.0	1	0

400 rows × 5 columns

In [42]:
```
x=data.drop(['Purchased'],axis=1)
```

In [43]:
```
y=data.Purchased
```

In [44]:
```
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import cohen_kappa_score as kappa
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [45]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=10)
```

In [46]:
```
from sklearn.linear_model import LinearRegression
```

In [47]:
```
 classifier=(LogisticRegression())
    #fitting training data to the model
```

In [48]:
```
classifier.fit(x_train,y_train)
```

Out[48]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [49]:
```
y_pred=classifier.predict(x_test)
```

In [50]:
```
print(list(zip(y_test,y_pred)))
```

```
[(0, 0), (0, 0), (1, 0), (0, 0), (0, 0), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(0, 1), (1, 1), (1, 0), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0),
(0, 0), (1, 0), (1, 0), (0, 0), (0, 0), (1, 1), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(1, 0), (1, 1), (0, 0), (1, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 0), (0, 0), (0, 0),
(0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (1, 1), (1, 0), (0, 0), (0, 0),
(0, 0), (1, 0), (0, 0), (1, 0), (1, 1), (0, 0), (1, 1), (0, 0), (1, 1), (1, 1), (0, 0),
(0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(0, 0), (0, 0), (1, 0), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(0, 0), (0, 0), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0),
(0, 0), (0, 0), (0, 0), (0, 0), (1, 0), (0, 0), (1, 0), (0, 0), (0, 0), (1, 1), (0, 0),
(0, 0), (0, 1), (0, 0), (0, 0), (1, 0), (1, 0), (0, 0), (1, 1), (0, 1), (0, 0)]
```

In [51]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [52]:
```
confusion_matrix=confusion_matrix(y_test,y_pred)
```

In [53]:
```
print(confusion_matrix)
```

```
[[80  3]
 [20 17]]
```

In [54]:
```
accuracy_score=accuracy_score(y_test,y_pred)
```

In [55]:
```
print("Accuracy of the model:",accuracy_score)
```

```
Accuracy of the model: 0.8083333333333333
```

In [56]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [57]:
```
cfm=confusion_matrix(y_test,y_pred)
```

In [58]:
```
print(cfm)
```

```
[[80  3]
 [20 17]]
```

In [59]:
```
print("classification_report:")
```

```
classification_report:
```

In [60]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.96      0.87        83
           1       0.85      0.46      0.60        37

   micro avg       0.81      0.81      0.81       120
   macro avg       0.82      0.71      0.74       120
weighted avg       0.82      0.81      0.79       120
```

In [61]:
```
acc=accuracy_score(y_test,y_pred)
```

In [62]:
```
print('acc',acc)
```

```
acc 0.8083333333333333
```

In [83]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [86]:
```
model_DecisionTree=DecisionTreeClassifier()
```

In [87]:
```
model_DecisionTree.fit(x_train,y_train)
```

Out[87]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [88]:
```
y_pred=model_DecisionTree.predict(x_test)
```

In [89]:
```
print(list(zip(y_test,y_pred)))
```

```
[(0, 0), (0, 1), (1, 1), (0, 1), (0, 0), (1, 1), (0, 0), (0, 1), (0, 0), (0, 0), (0, 0),
(0, 1), (1, 1), (1, 1), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0),
(0, 0), (1, 1), (1, 1), (0, 0), (0, 0), (1, 1), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0),
(1, 1), (1, 1), (0, 0), (1, 1), (1, 1), (0, 0), (0, 0), (0, 1), (1, 1), (0, 0), (0, 0),
(0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (1, 1), (1, 1), (0, 0), (0, 0),
(0, 0), (1, 1), (0, 0), (1, 1), (1, 1), (0, 0), (1, 0), (0, 0), (1, 1), (1, 1), (0, 1),
(0, 0), (1, 1), (0, 0), (0, 0), (0, 1), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(0, 0), (0, 1), (1, 1), (1, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0),
(0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 1), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0),
(0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 1), (1, 1), (0, 0), (0, 0), (1, 1), (0, 0),
(0, 0), (0, 1), (0, 0), (0, 0), (1, 1), (1, 1), (0, 0), (1, 1), (0, 1), (0, 1)]
```

In [90]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [91]:
```
#confusion matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[70 13]
 [ 2 35]]
```

In [92]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.875
```

In [93]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.84      0.90        83
           1       0.73      0.95      0.82        37

   micro avg       0.88      0.88      0.88       120
   macro avg       0.85      0.89      0.86       120
weighted avg       0.90      0.88      0.88       120
```

In [94]:
```
from sklearn import tree
```

In [95]:
```
with open("model_DecisionTree.txt","w")as f:
    f=tree.export_graphviz(model_DecisionTree,out_file=f)
```

In [71]:
```
#http://www.webgraphviz.com
#go to C drive->Users->Admin->open model_DecisionTree(txt doc)->copy and paste the text on
#DecisionTree will be formed
```

In [96]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [97]:
```
model_RandomForest=RandomForestClassifier(501)
```

In [98]:
```
model_RandomForest.fit(x_train,y_train)
```

Out[98]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=501, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [99]:
```
y_pred=model_RandomForest.predict(x_test)
```

In [100]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [101]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[73 10]
 [ 1 36]]
```

In [102]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.9083333333333333
```

In [103]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.88      0.93        83
           1       0.78      0.97      0.87        37

   micro avg       0.91      0.91      0.91       120
   macro avg       0.88      0.93      0.90       120
weighted avg       0.92      0.91      0.91       120
```

In [ ]:
```

```

In [ ]:
```

```