In [1]:
```
#7-7-21
#Mohith Saran
#assignment-5 decision tree and random forest classification
```

In [54]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [55]:
```
dataset = pd.read_csv('Churn_Modelling1.csv')
dataset
```

Out[55]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.
1	2	15647311	Hill	608	Spain	Female	41.0	1	83807.
2	3	15619304	Onio	502	France	Female	42.0	8	159660.
3	4	15701354	Boni	699	France	Female	39.0	1	0.
4	5	15737888	Mitchell	850	Spain	Female	NaN	2	125510.
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39.0	5	0.
9996	9997	15569892	Johnstone	516	France	Male	35.0	10	57369.
9997	9998	15584532	Liu	709	France	Female	36.0	7	0.
9998	9999	15682355	Sabbatini	772	Germany	Male	42.0	3	75075.
9999	10000	15628319	Walker	792	France	Female	28.0	4	130142.

10000 rows × 14 columns

In [56]:
```
type(dataset)
```

Out[56]:
```
pandas.core.frame.DataFrame
```

In [57]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        9997 non-null   object 
 5   Gender           9997 non-null   object 
 6   Age              9996 non-null   float64
 7   Tenure           10000 non-null  int64  
 8   Balance          9998 non-null   float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

In [58]:
```
dataset.isnull().any()
```

Out[58]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography           True
Gender              True
Age                 True
Tenure             False
Balance             True
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [59]:
```
dataset.isnull().sum()
```

Out[59]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          3
Gender             3
Age                4
Tenure             0
Balance            2
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [60]:
```
dataset[dataset['Age'].isnull()].index.tolist()
```

Out[60]:
```
[4, 28, 43, 59]
```

In [61]:
```
dataset[4:5]
```

Out[61]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
4	5	15737888	Mitchell	850	Spain	Female	NaN	2	125510.82

In [62]:
```
dataset['Age'].fillna(dataset['Age'].mean(),inplace=True)
print(dataset['Age'].isnull().sum())
```

```
0
```

In [63]:
```
dataset['Balance'].fillna(dataset['Balance'].mean(),inplace=True)
```

In [64]:
```
dataset.isnull().any()
```

Out[64]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography           True
Gender              True
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [65]:
```
dataset['Geography'].unique()
```

Out[65]:
```
array(['France', 'Spain', 'Germany', nan], dtype=object)
```

In [66]:
```
dataset['Gender'].unique()
```

Out[66]:
```
array(['Female', 'Male', nan], dtype=object)
```

In [67]:
```
dataset['Gender'].mode()
```

Out[67]:
```
0    Male
dtype: object
```

In [68]:
```
dataset['Geography'].mode()
```

Out[68]:
```
0    France
dtype: object
```

In [69]:
```
dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)
dataset['Geography'].fillna(dataset['Geography'].mode()[0],inplace=True)
```

In [70]:
```
dataset.isnull().any()
```

Out[70]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [71]:
```
dataset
```

Out[71]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42.000000	2	
1	2	15647311	Hill	608	Spain	Female	41.000000	1	8
2	3	15619304	Onio	502	France	Female	42.000000	8	15
3	4	15701354	Boni	699	France	Female	39.000000	1	
4	5	15737888	Mitchell	850	Spain	Female	38.918768	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39.000000	5	
9996	9997	15569892	Johnstone	516	France	Male	35.000000	10	5
9997	9998	15584532	Liu	709	France	Female	36.000000	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42.000000	3	7
9999	10000	15628319	Walker	792	France	Female	28.000000	4	13

10000 rows × 14 columns

In [72]:
```
dataset['Age'] = dataset['Age'].round()
```

In [73]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['Gender'] = le.fit_transform(dataset['Gender'])
dataset['Geography'] = le.fit_transform(dataset['Geography'])
```

In [74]:
```
print(dataset['Gender'].unique())
print(dataset['Geography'].unique())
```

```
[0 1]
[0 2 1]
```

In [75]:
```
dataset.nunique()
```

Out[75]:
```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6381
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

In [76]:
```
dataset['Surname'].value_counts()
```

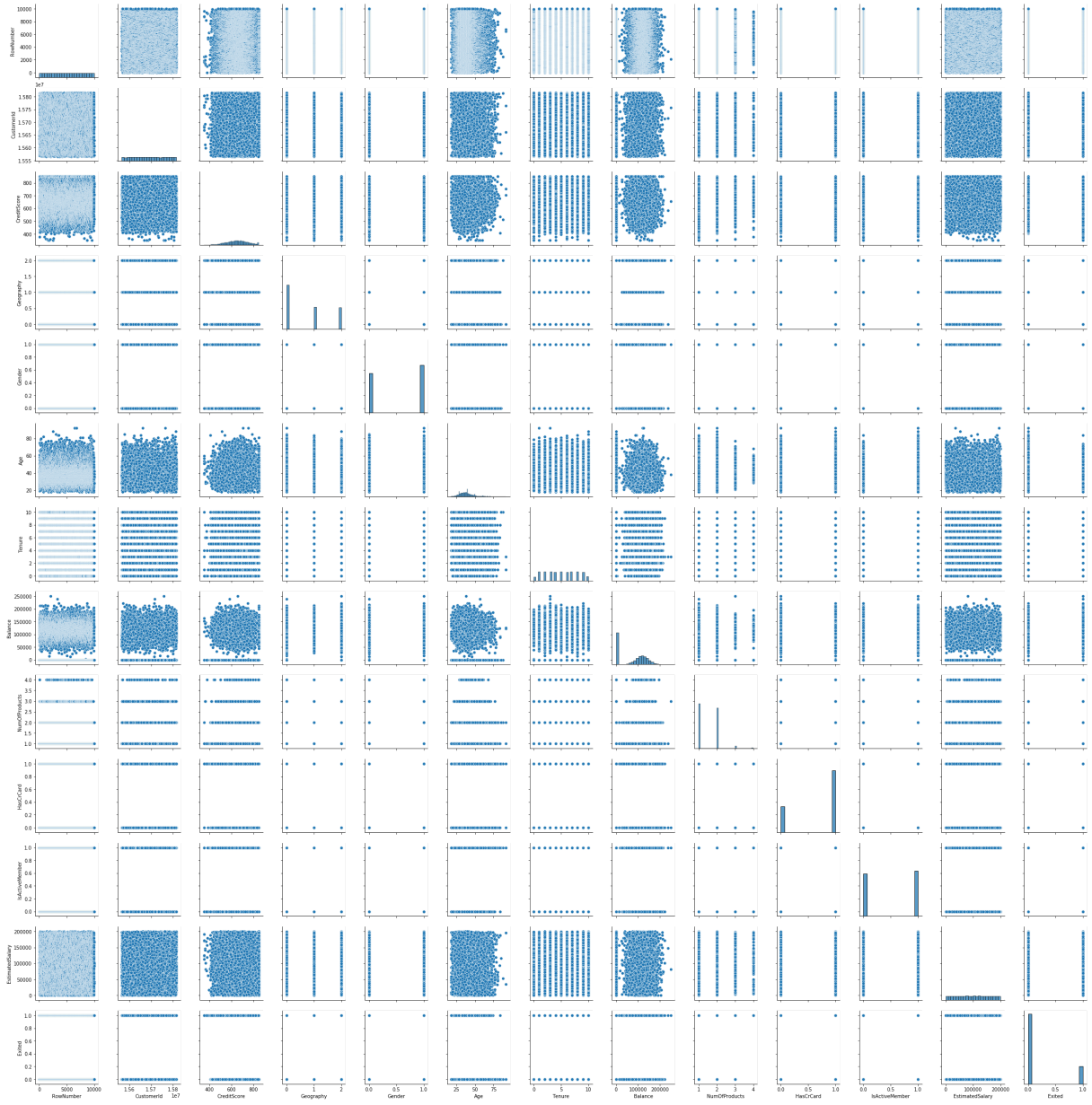
Out[76]:
```
Smith         32
Martin        29
Scott         29
Walker        28
Brown         26
              ..
Weller         1
Shoebridge     1
Creel          1
Fox            1
Aksenova       1
Name: Surname, Length: 2932, dtype: int64
```

In [77]:
```
dataset.isnull().any()
```

Out[77]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

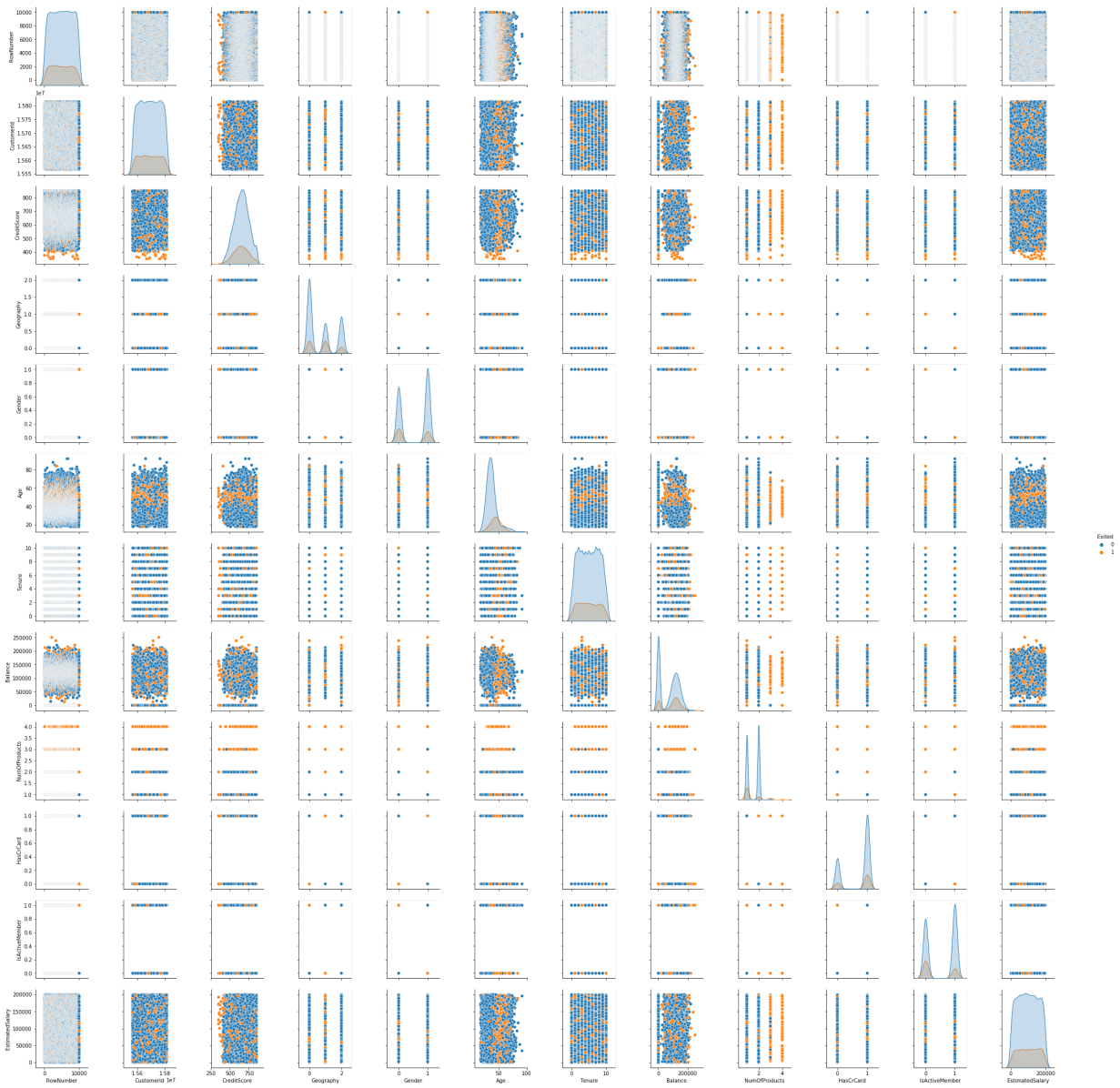
In [241]:
```
import seaborn as sns
sns.pairplot(dataset)
```

Out[241]: <seaborn.axisgrid.PairGrid at 0x1d8be3caf98>



In [242]:
```
import seaborn as sns
sns.pairplot(dataset,hue='Exited')
```

Out[242]: <seaborn.axisgrid.PairGrid at 0x1d8c7099390>



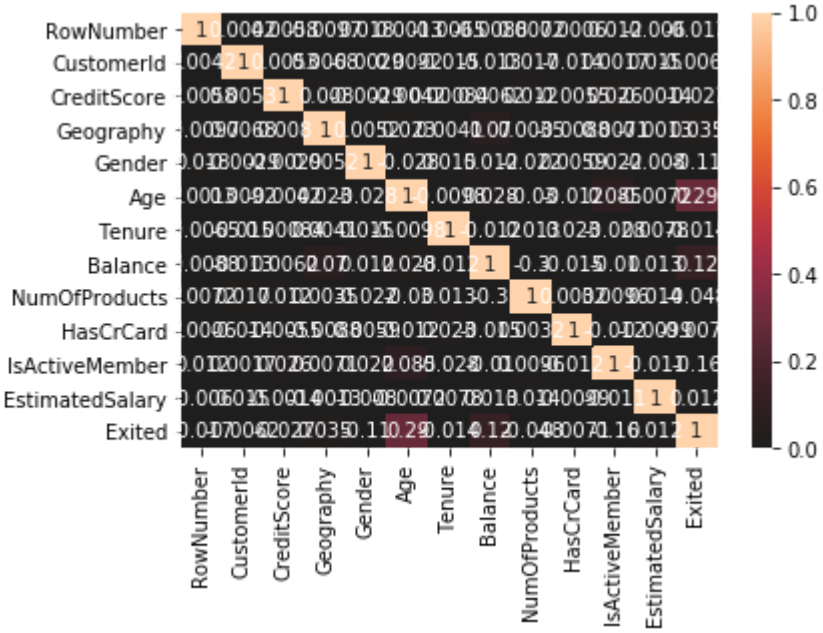
In [78]:
```
dataset.corr()
```

Out[78]:

lumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProd
000000	0.004202	0.005840	-0.009734	0.017850	0.001274	-0.006495	-0.008830	0.007
004202	1.000000	0.005308	0.006779	-0.002901	0.009187	-0.014883	-0.012525	0.016
005840	0.005308	1.000000	0.008034	-0.002887	-0.004152	0.000842	0.006215	0.012
009734	0.006779	0.008034	1.000000	0.005178	0.022713	0.004075	0.069609	0.003
017850	-0.002901	-0.002887	0.005178	1.000000	-0.027597	0.014941	0.011716	-0.021
001274	0.009187	-0.004152	0.022713	-0.027597	1.000000	-0.009792	0.028283	-0.030
006495	-0.014883	0.000842	0.004075	0.014941	-0.009792	1.000000	-0.012168	0.013
008830	-0.012525	0.006215	0.069609	0.011716	0.028283	-0.012168	1.000000	-0.304
007246	0.016972	0.012238	0.003472	-0.021697	-0.030423	0.013444	-0.304141	1.000
000599	-0.014025	-0.005458	-0.008757	0.005896	-0.011698	0.022583	-0.014858	0.003
012044	0.001665	0.025651	0.007098	0.022338	0.085396	-0.028362	-0.010136	0.009
005988	0.015271	-0.001384	-0.001339	-0.007978	-0.007238	0.007784	0.012695	0.014
016571	-0.006248	-0.027094	0.035227	-0.106616	0.285267	-0.014001	0.118609	-0.047

In [79]:
```
import seaborn as sns
sns.heatmap(dataset.corr(),annot=True,vmin=0,vmax=1,center=0)
```

Out[79]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d8bc8ae7b8>
```



In [80]:
```
y = dataset.iloc[:,-1:].values
y
```

Out[80]:
```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]], dtype=int64)
```

In [81]:
```
x = dataset.iloc[:,0:-1].values
x
```

Out[81]:
```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

In [82]:
```
x.shape
```

Out[82]:
```
(10000, 13)
```

In [83]:
```
#deleting rownumber , customerId, Surname
x = np.delete(x,0,axis=1)
x
```

Out[83]:
```
array([[15634602, 'Hargrave', 619, ..., 1, 1, 101348.88],
       [15647311, 'Hill', 608, ..., 0, 1, 112542.58],
       [15619304, 'Onio', 502, ..., 1, 0, 113931.57],
       ...,
       [15584532, 'Liu', 709, ..., 0, 1, 42085.58],
       [15682355, 'Sabbatini', 772, ..., 1, 0, 92888.52],
       [15628319, 'Walker', 792, ..., 1, 0, 38190.78]], dtype=object)
```

In [84]:
```
x = np.delete(x,0,axis=1)
x = np.delete(x,0,axis=1)
```

In [85]:
```
x
```

Out[85]:
```
array([[619, 0, 0, ..., 1, 1, 101348.88],
       [608, 2, 0, ..., 0, 1, 112542.58],
       [502, 0, 0, ..., 1, 0, 113931.57],
       ...,
       [709, 0, 0, ..., 0, 1, 42085.58],
       [772, 1, 1, ..., 1, 0, 92888.52],
       [792, 0, 0, ..., 1, 0, 38190.78]], dtype=object)
```

In [86]:
```
x.shape
```

Out[86]:
```
(10000, 10)
```

In [87]:
```
# 2nd(Geography) and 3rd(Gender) cols are categorical cols converted to numerical
```

In [88]:
```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
```

In [89]:
```
z = oh.fit_transform(x[:,1:2]).toarray()
z
```

Out[89]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])
```

In [90]:
```
x = np.delete(x,1,axis=1)
```

In [91]:
```
x = np.concatenate((x,z),axis=1)
```

In [92]:
```
x.shape
```

Out[92]:
```
(10000, 12)
```

In [93]:
```
z = oh.fit_transform(x[:,1:2]).toarray()
z
```

Out[93]:
```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [0., 1.],
       [1., 0.]])
```

In [94]:
```
x = np.delete(x,1,axis=1)
x = np.concatenate((x,z),axis=1)
x.shape
```

Out[94]:
```
(10000, 13)
```

In [95]:
```
x
```

Out[95]:
```
array([[619, 42.0, 2, ..., 0.0, 1.0, 0.0],
       [608, 41.0, 1, ..., 1.0, 1.0, 0.0],
       [502, 42.0, 8, ..., 0.0, 1.0, 0.0],
       ...,
       [709, 36.0, 7, ..., 0.0, 1.0, 0.0],
       [772, 42.0, 3, ..., 0.0, 0.0, 1.0],
       [792, 28.0, 4, ..., 0.0, 1.0, 0.0]], dtype=object)
```

In [96]:
```
# cols in x are CreditScore,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMe
#France,Germany,Spain,Female,Male
```

In [97]:
```
y.shape
```

Out[97]:
```
(10000, 1)
```

In [98]:
```
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,random_
```

In [99]:
```
x_train.shape
```

Out[99]:
```
(8000, 13)
```

In [100]:
```
x_test.shape
```

Out[100]:
```
(2000, 13)
```

In [101]:
```
y_train.shape
```

Out[101]:
```
(8000, 1)
```

In [102]:
```
y_test.shape
```

Out[102]:
```
(2000, 1)
```

In [103]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [104]:
```
x_test
```

Out[104]:
```
array([[-0.56129438, -0.39401698,  0.9869706 , ..., -0.57427105,
         1.11339196, -1.11339196],
       [-1.33847768,  0.07611425, -1.08432132, ..., -0.57427105,
         1.11339196, -1.11339196],
       [ 0.58347561,  0.26416674,  0.9869706 , ...,  1.74133801,
         1.11339196, -1.11339196],
       ...,
       [-0.76084144, -0.29999074, -1.42953664, ...,  1.74133801,
        -0.8981563 ,  0.8981563 ],
       [-0.0046631 , -0.48804323, -0.39389068, ..., -0.57427105,
        -0.8981563 ,  0.8981563 ],
       [-0.81335383, -0.86414821,  0.9869706 , ..., -0.57427105,
        -0.8981563 ,  0.8981563 ]])
```

In [105]:
```
x_train
```

Out[105]:
```
array([[ 0.16958176, -0.46432479,  0.00666099, ...,  1.74367544,
         1.09196221, -1.09196221],
       [-2.30455945,  0.30143362, -1.37744033, ..., -0.57350122,
        -0.91578261,  0.91578261],
       [-1.19119591, -0.94292379, -1.031415  , ..., -0.57350122,
         1.09196221, -1.09196221],
       ...,
       [ 0.9015152 , -0.36860499,  0.00666099, ..., -0.57350122,
        -0.91578261,  0.91578261],
       [-0.62420521, -0.08144559,  1.39076231, ...,  1.74367544,
         1.09196221, -1.09196221],
       [-0.28401079,  0.87575242, -1.37744033, ..., -0.57350122,
         1.09196221, -1.09196221]])
```

In [200]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion = 'entropy',max_depth=10,max_features=10,r
dtc.fit(x_train,y_train)
```

Out[200]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=10, max_features=10, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [201]:
```
y_pred = dtc.predict(x_test)
y_pred
```

Out[201]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [202]:
```
y_test
```

Out[202]:
```
array([[0],
       [1],
       [0],
       ...,
       [0],
       [0],
       [0]], dtype=int64)
```

In [203]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[203]:
```
0.8605
```

In [198]:
```
depth = 0
feature = 0
node = 0
max = 0
for i in range(5,20):
    for j in range(5,14):
        for k in range(5,20):
            dtc = DecisionTreeClassifier(criterion = 'entropy',max_depth=i,max_f
            dtc.fit(x_train,y_train)
            y_pred = dtc.predict(x_test)
            ac_score = accuracy_score(y_test,y_pred)
            if(max<ac_score):
                max = ac_score
                depth = i
                feature = j
                node = k
print('max_depth= ',i,' max_features= ',j,' max_leaf_nodes= ',k)
print(max)
```

```
max_depth=  19  max_features=  13  max_leaf_nodes=  19
0.8655
```

In [205]:
```
dtc = DecisionTreeClassifier(criterion = 'entropy',max_depth=depth,max_features=
dtc.fit(x_train,y_train)
y_pred = dtc.predict(x_test)
accuracy_score(y_test,y_pred)
```

Out[205]:
```
0.856
```

In [220]:
```
dtc = DecisionTreeClassifier(criterion = 'entropy',max_depth=10,max_features=10,
dtc.fit(x_train,y_train)
y_pred = dtc.predict(x_test)
accuracy_score(y_test,y_pred)
```

Out[220]:
```
0.856
```

In [221]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test , y_pred)
cm
```

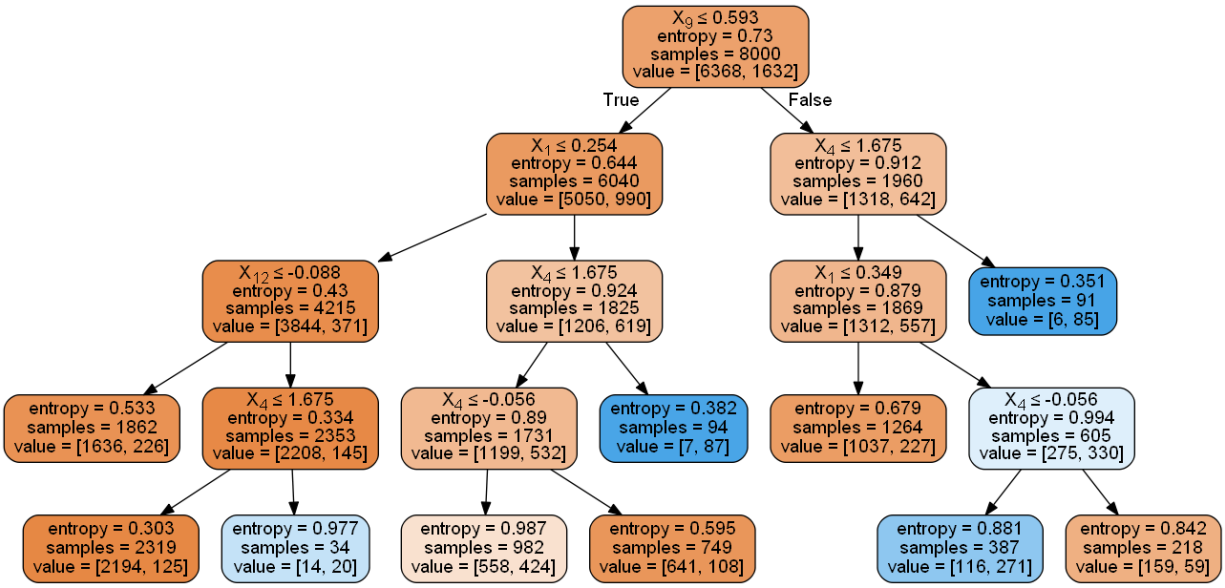
Out[221]:
```
array([[1494,  101],
       [ 187,  218]], dtype=int64)
```

In [ ]:
```
```

In [153]:
```
from sklearn import tree
tree.export_graphviz(dtc)

from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data=StringIO()
export_graphviz(dtc,out_file=dot_data,
                filled=True,rounded=True,
                special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

```
C:\Users\hp\AppData\Roaming\Python\Python37\site-packages\sklearn\externals\si
x.py:31: FutureWarning: The module is deprecated in version 0.21 and will be re
moved in version 0.23 since we've dropped support for Python 2.7. Please rely o
n the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
```

Out[153]:



In [235]:
```
#random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=30,criterion='entropy',max_depth=11,ma
rfc.fit(x_train,y_train)
y_pred_rfc = rfc.predict(x_test)
y_pred_rfc
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConvers
ionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.
```

Out[235]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [236]:
```
y_test
```

Out[236]:
```
array([[0],
       [1],
       [0],
       ...,
       [0],
       [0],
       [0]], dtype=int64)
```

In [237]:
```
accuracy_score(y_test,y_pred_rfc)
```

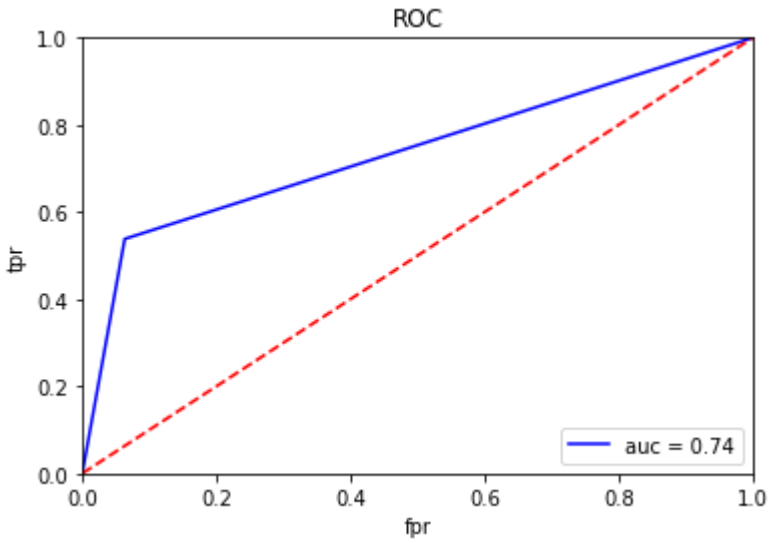
Out[237]:
```
0.8415
```

In [238]:
```
confusion_matrix(y_test,y_pred_rfc)
```

Out[238]:
```
array([[1587,    8],
       [ 309,   96]], dtype=int64)
```

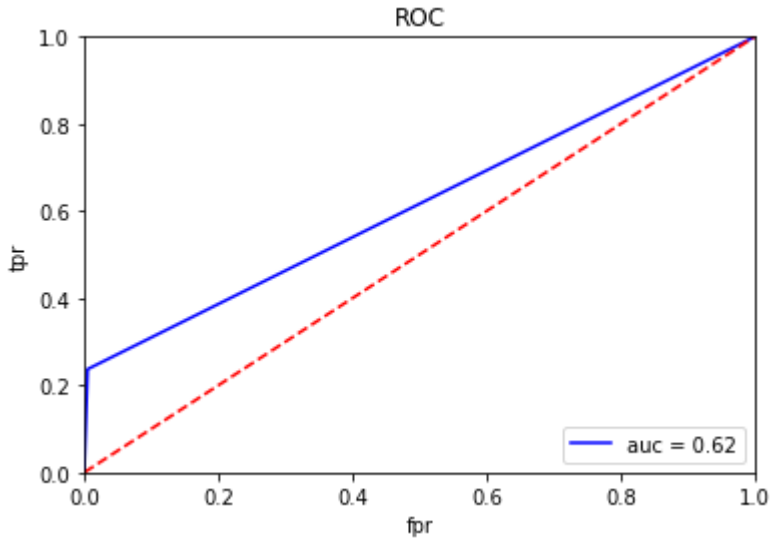
In [239]:
```
import sklearn.metrics as metrics
fpr,tpr,threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

Out[239]:
```
Text(0, 0.5, 'tpr')
```



In [240]:
```
#auc, #roc
#area under curve
import sklearn.metrics as metrics
fpr,tpr,threshold = metrics.roc_curve(y_test,y_pred_rfc)
roc_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

Out[240]:
```
Text(0, 0.5, 'tpr')
```



visualization of bike buyer dataset

6-7-21

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
dataset = pd.read_csv('bikebuyer1.csv')
```

In [4]:
```
dataset
```

Out[4]:

	ID	Marital Status	Gender	Yearly Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance
0	22711.0	Single	Male	30000	0.0	Partial College	Clerical	No	1	1.(
1	13555.0	Married	Female	40000	0.0	Graduate Degree	Clerical	Yes	0	1.(
2	NaN	Married	Male	160000	5.0	Partial College	Professional	No	3	2.(
3	2.0	Single	Male	160000	0.0	Graduate Degree	Management	Yes	2	5.(
4	25410.0	NaN	Female	70000	2.0	Bachelors	Skilled Manual	No	1	1.(
...	...	...	...	...	...	...	...	...	...	..
6992	22820.0	Married	Male	100000	4.0	High School	Professional	Yes	3	1.(
6993	22821.0	Married	Female	130000	4.0	Partial College	Professional	Yes	4	2.(
6994	22823.0	Married	Female	160000	5.0	Bachelors	Management	Yes	2	1.(
6995	22825.0	Single	Female	120000	5.0	Partial College	Professional	Yes	3	1.(
6996	22826.0	Married	Male	130000	5.0	High School	Professional	Yes	3	2.(

6997 rows × 13 columns

In [5]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6997 entries, 0 to 6996
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ID                6996 non-null   float64
 1   Marital Status    6981 non-null   object 
 2   Gender            6968 non-null   object 
 3   Yearly Income     6997 non-null   int64  
 4   Children          6979 non-null   float64
 5   Education         6997 non-null   object 
 6   Occupation        6997 non-null   object 
 7   Home Owner        6997 non-null   object 
 8   Cars              6997 non-null   int64  
 9   Commute Distance  6968 non-null   float64
 10  Region            6997 non-null   object 
 11  Age               6997 non-null   int64  
 12  Bike Buyer        6997 non-null   object 
dtypes: float64(3), int64(3), object(7)
memory usage: 710.8+ KB
```

In [6]:
```
dataset.isnull().any()
```

Out[6]:
```
ID                 True
Marital Status     True
Gender             True
Yearly Income     False
Children           True
Education         False
Occupation        False
Home Owner        False
Cars              False
Commute Distance   True
Region            False
Age               False
Bike Buyer        False
dtype: bool
```

In [7]:
```
dataset.isnull().sum()
```

Out[7]:
```
ID                   1
Marital Status      16
Gender              29
Yearly Income        0
Children            18
Education            0
Occupation           0
Home Owner           0
Cars                 0
Commute Distance    29
Region               0
Age                  0
Bike Buyer           0
dtype: int64
```

In [8]:
```
# id is unique and it does not affect so it can be dropped
#handling null values using mode/mean/median as appropriate
```

In [9]:
```
dataset['Marital Status'].unique()
```

Out[9]:
```
array(['Single', 'Married', nan], dtype=object)
```

In [10]:
```
dataset['Marital Status'].value_counts()
```

Out[10]:
```
Married    4133
Single     2848
Name: Marital Status, dtype: int64
```

In [11]:
```
dataset['Marital Status'].fillna(dataset['Marital Status'].mode()[0],inplace=Tru
dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)
dataset['Children'].fillna(dataset['Children'].median(),inplace=True)
dataset['Commute Distance'].fillna(dataset['Commute Distance'].median(),inplace=
```

In [12]:
```
dataset.isnull().any()
```

Out[12]:
```
ID                   True
Marital Status      False
Gender              False
Yearly Income       False
Children            False
Education           False
Occupation          False
Home Owner          False
Cars                False
Commute Distance    False
Region              False
Age                 False
Bike Buyer          False
dtype: bool
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

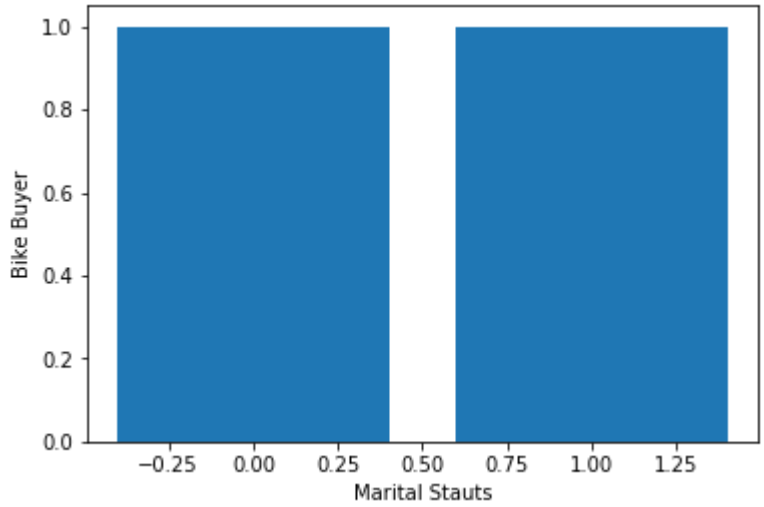
In [14]:
```
dataset['Marital Status']=le.fit_transform(dataset['Marital Status'])
dataset['Gender']=le.fit_transform(dataset['Gender'])
dataset['Education']=le.fit_transform(dataset['Education'])
dataset['Occupation']=le.fit_transform(dataset['Occupation'])
dataset['Home Owner']=le.fit_transform(dataset['Home Owner'])
dataset['Region']=le.fit_transform(dataset['Region'])
dataset['Bike Buyer']=le.fit_transform(dataset['Bike Buyer'])
```

In [15]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6997 entries, 0 to 6996
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                6996 non-null   float64
 1   Marital Status    6997 non-null   int32
 2   Gender            6997 non-null   int32
 3   Yearly Income     6997 non-null   int64
 4   Children          6997 non-null   float64
 5   Education         6997 non-null   int32
 6   Occupation        6997 non-null   int32
 7   Home Owner        6997 non-null   int32
 8   Cars              6997 non-null   int64
 9   Commute Distance  6997 non-null   float64
 10  Region            6997 non-null   int32
 11  Age               6997 non-null   int64
 12  Bike Buyer        6997 non-null   int32
dtypes: float64(3), int32(7), int64(3)
memory usage: 519.4 KB
```

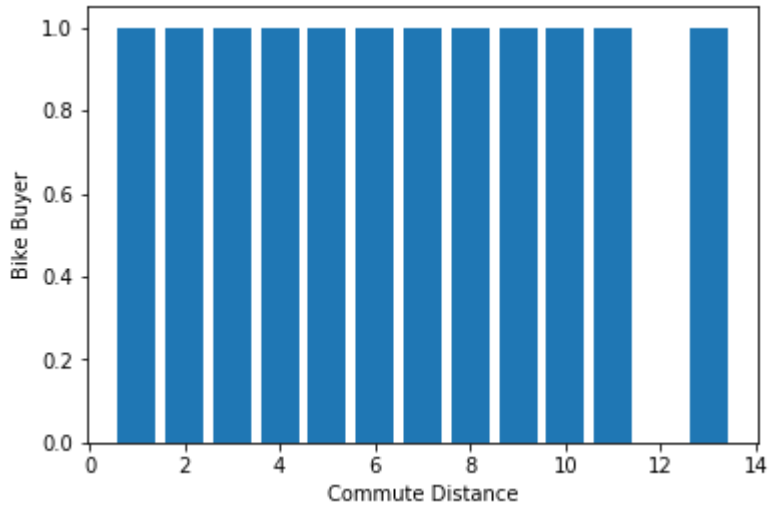
In [25]:
```
plt.bar(dataset['Marital Status'],dataset['Bike Buyer'])
plt.xlabel('Marital Stauts')
plt.ylabel('Bike Buyer')
```

Out[25]: Text(0, 0.5, 'Bike Buyer')



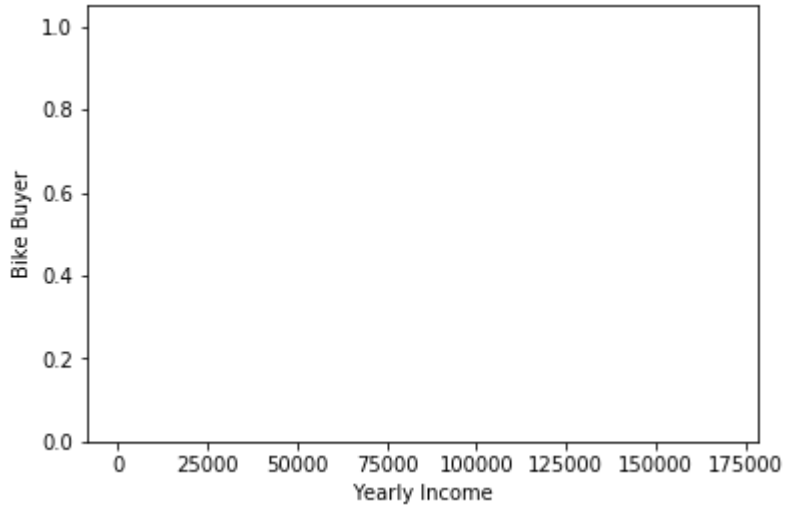
In [26]:
```
plt.bar(dataset['Commute Distance'],dataset['Bike Buyer'])
plt.xlabel('Commute Distance')
plt.ylabel('Bike Buyer')
```

Out[26]: Text(0, 0.5, 'Bike Buyer')



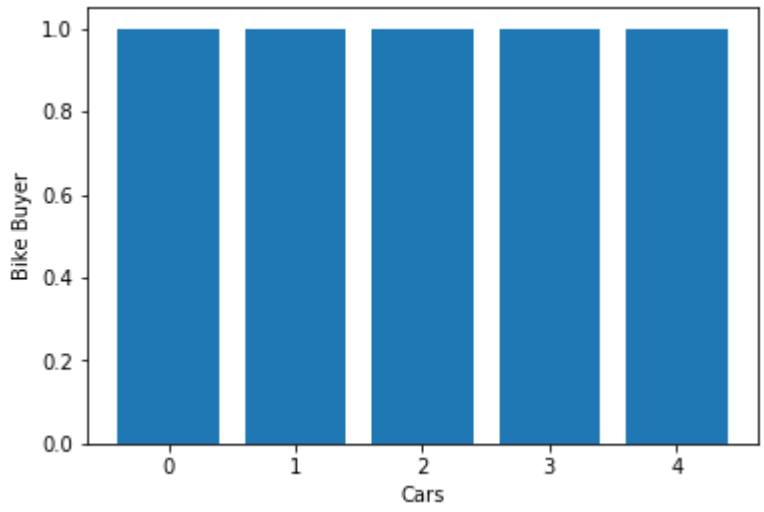
In [28]:
```
plt.bar(dataset['Yearly Income'],dataset['Bike Buyer'])
plt.xlabel('Yearly Income')
plt.ylabel('Bike Buyer')
```

Out[28]:
```
Text(0, 0.5, 'Bike Buyer')
```



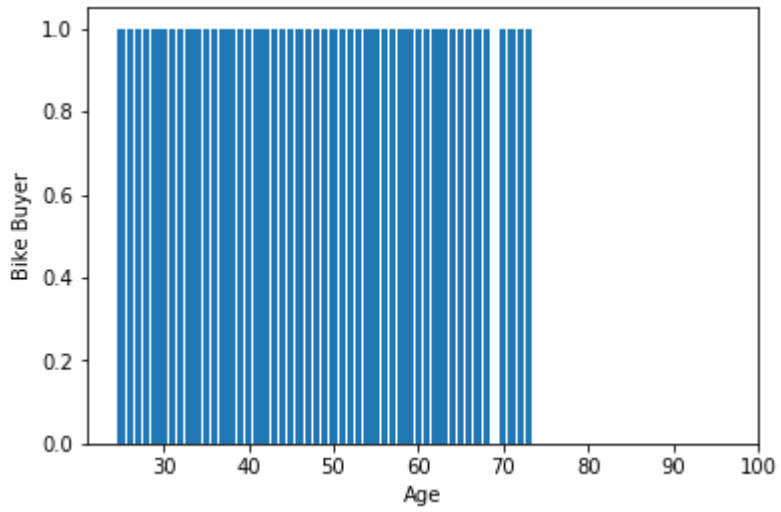
In [33]:
```
plt.bar(dataset['Cars'],dataset['Bike Buyer'])
plt.xlabel('Cars')
plt.ylabel('Bike Buyer')
```

Out[33]:
```
Text(0, 0.5, 'Bike Buyer')
```



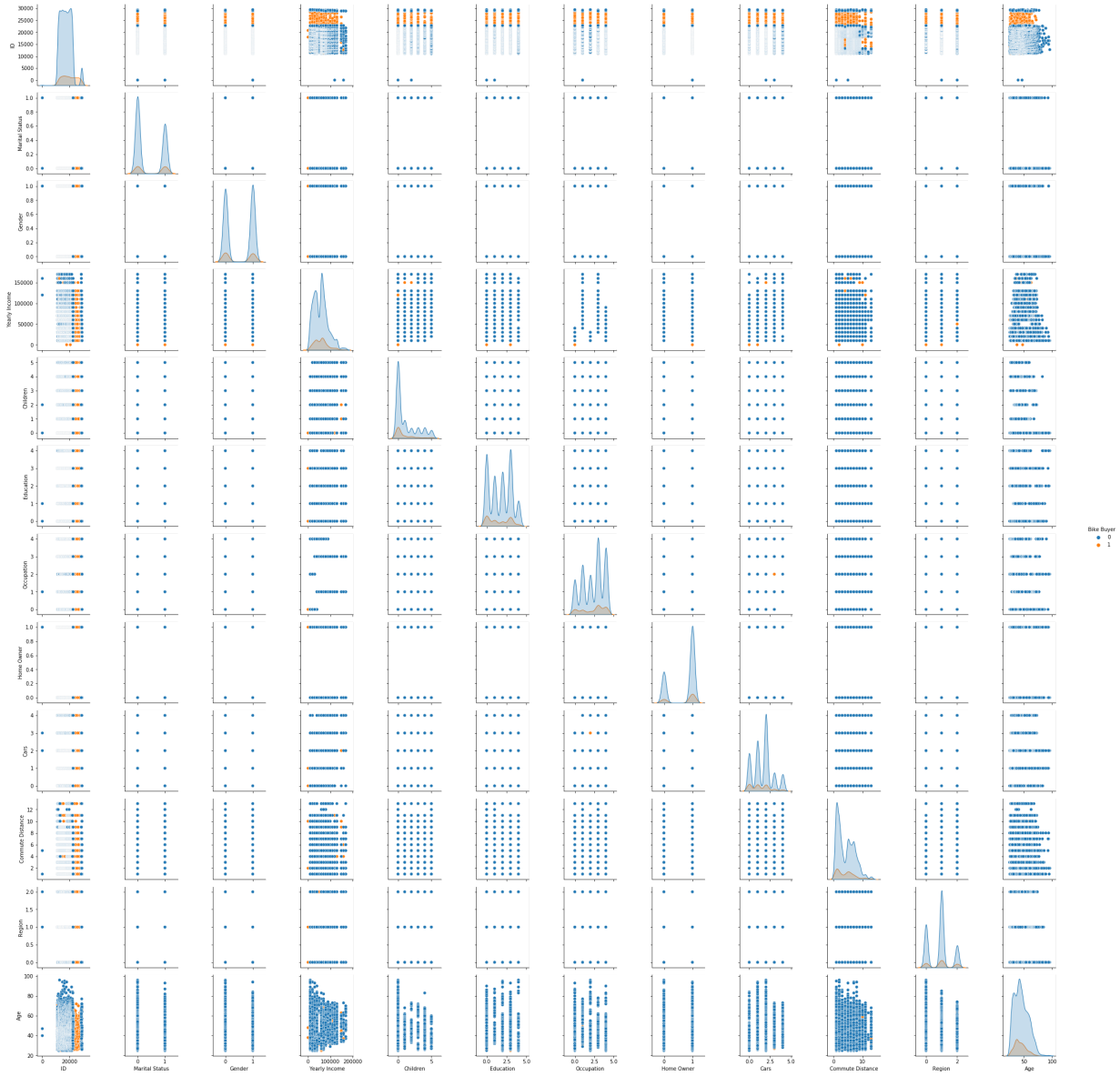
In [30]:
```
plt.bar(dataset['Age'],dataset['Bike Buyer'])
plt.xlabel('Age')
plt.ylabel('Bike Buyer')
```

Out[30]:
```
Text(0, 0.5, 'Bike Buyer')
```



In [36]:
```
import seaborn as sns
sns.pairplot(dataset, hue = 'Bike Buyer')
```

Out[36]: <seaborn.axisgrid.PairGrid at 0x16a67fc67f0>



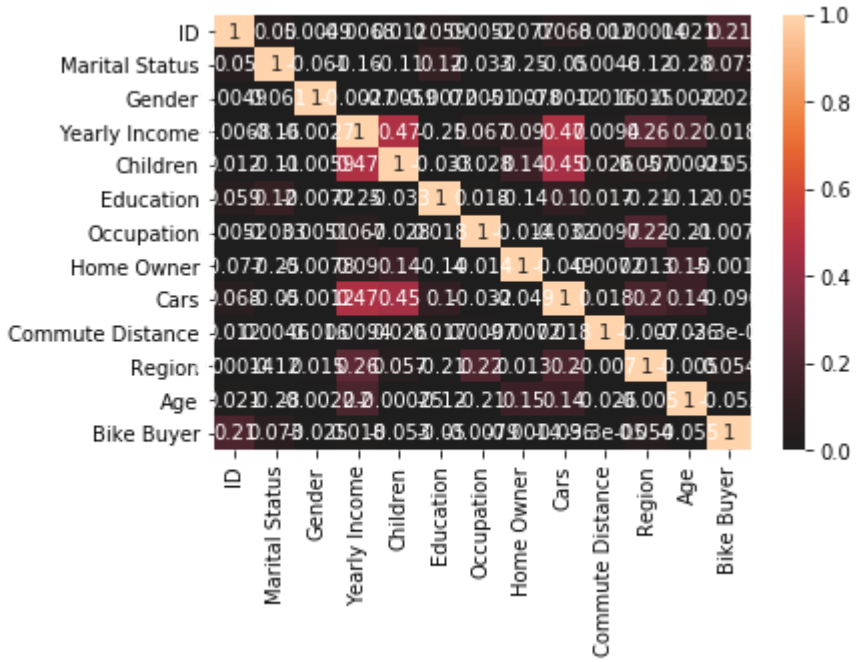
In [16]:
```
dataset.corr()
```

Out[16]:

	ID	Marital Status	Gender	Yearly Income	Children	Education	Occupation	Home Owne
ID	1.000000	0.050063	0.004949	-0.006802	0.011751	0.058976	0.005225	-0.07687
Marital Status	0.050063	1.000000	-0.060753	-0.159823	-0.107973	0.123656	-0.032806	-0.25468
Gender	0.004949	-0.060753	1.000000	-0.002706	-0.005853	-0.007174	0.005131	-0.00783
Yearly Income	-0.006802	-0.159823	-0.002706	1.000000	0.474231	-0.249386	0.067132	0.08985
Children	0.011751	-0.107973	-0.005853	0.474231	1.000000	-0.033362	-0.028387	0.14168
Education	0.058976	0.123656	-0.007174	-0.249386	-0.033362	1.000000	0.018243	-0.13841
Occupation	0.005225	-0.032806	0.005131	0.067132	-0.028387	0.018243	1.000000	-0.01376
Home Owner	-0.076878	-0.254684	-0.007835	0.089856	0.141689	-0.138411	-0.013761	1.00000
Cars	0.068315	-0.049793	-0.001182	0.472089	0.448015	0.103111	-0.032003	-0.04918
Commute Distance	0.011544	0.004593	-0.015961	0.009403	0.025816	0.017238	0.009749	-0.00721
Region	0.000143	-0.117683	0.014955	0.256088	0.056532	-0.210062	0.215309	0.01307
Age	0.021135	-0.283900	-0.002175	0.195351	-0.000245	-0.123425	-0.212152	0.14801
Bike Buyer	0.210813	0.073119	-0.024682	0.018456	-0.053034	-0.049899	-0.007937	-0.00143

In [17]:
```
import seaborn as sns
sns.heatmap(dataset.corr(),annot=True,vmin=0,vmax=1,center=0)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x167c99786a0>



In [18]:
```
#dropping cols
dataset.drop(['ID','Education'],axis=1,inplace=True)
```

In [19]:
```
dataset.shape
```

Out[19]: (6997, 11)

In [20]:
```
dataset.head()
```

Out[20]:

	Marital Status	Gender	Yearly Income	Children	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Bike Buyer
0	1	1	30000	0.0	0	0	1	1.0	0	33	1
1	0	0	40000	0.0	0	1	0	1.0	0	37	1
2	0	1	160000	5.0	3	0	3	2.0	0	55	0
3	1	1	160000	0.0	1	1	2	5.0	2	47	0
4	0	0	70000	2.0	4	0	1	1.0	1	38	1

In [22]:
```
dataset['Occupation'].value_counts()
```

Out[22]:
```
3    2031
4    1748
1    1265
2     990
0     963
Name: Occupation, dtype: int64
```

In [24]:
```
dataset['Region'].value_counts()
```

Out[24]:
```
1    3728
0    2096
2    1173
Name: Region, dtype: int64
```

In [28]:
```
#splitting data
x = dataset.iloc[:,0:10].values
x
```

Out[28]:
```
array([[1.0e+00, 1.0e+00, 3.0e+04, ..., 1.0e+00, 0.0e+00, 3.3e+01],
       [0.0e+00, 0.0e+00, 4.0e+04, ..., 1.0e+00, 0.0e+00, 3.7e+01],
       [0.0e+00, 1.0e+00, 1.6e+05, ..., 2.0e+00, 0.0e+00, 5.5e+01],
       ...,
       [0.0e+00, 0.0e+00, 1.6e+05, ..., 1.0e+00, 0.0e+00, 5.3e+01],
       [1.0e+00, 0.0e+00, 1.2e+05, ..., 1.0e+00, 0.0e+00, 5.4e+01],
       [0.0e+00, 1.0e+00, 1.3e+05, ..., 2.0e+00, 0.0e+00, 5.4e+01]])
```

In [30]:
```
y = dataset.iloc[:,-1:].values
y
```

Out[30]:
```
array([[1],
       [1],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [31]:
```
# as marital_stauts, home_owner and gender is binary
#no need to use OneHotEncoder on it
# applying onehotencoder on occupation,region
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
```

In [32]:
```
z = oh.fit_transform(x[:,4:5]).toarray()
t = oh.fit_transform(x[:,8:9]).toarray()
```

In [33]:
```
z
```

Out[33]:
```
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.]])
```

In [34]:
```
t
```

Out[34]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

In [36]:
```
x = np.delete(x,[4,8],axis=1)
```

In [37]:
```
x.shape
```

Out[37]:
```
(6997, 8)
```

In [38]:
```
# first region then occupation
x = np.concatenate((t,z,x),axis=1)
x.shape
```

Out[38]:
```
(6997, 16)
```

In [39]:
```
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,random_
```

In [41]:
```
x_test.shape
```

Out[41]:
```
(1400, 16)
```

In [42]:
```
# scaling values to range from -1 to 1
# scaling is used only for classification
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [43]:
```
x_train  = sc.fit_transform(x_train)
x_test  =sc.fit_transform(x_test)
```

In [44]:
```
x_test
```

Out[44]:
```
array([[-0.67363955,  0.96214456, -0.45027674, ..., -0.53708308,
         0.29126659,  0.22132995],
       [-0.67363955, -1.03934486,  2.22085645, ...,  0.3454478 ,
        -1.09101553,  1.22873799],
       [-0.67363955,  0.96214456, -0.45027674, ..., -0.53708308,
         0.63683712,  1.14478732],
       ...,
       [ 1.48447342, -1.03934486, -0.45027674, ..., -1.41961397,
        -0.745445  , -0.70212743],
       [-0.67363955, -1.03934486,  2.22085645, ..., -0.53708308,
         0.63683712,  0.30528062],
       [-0.67363955,  0.96214456, -0.45027674, ..., -1.41961397,
         0.29126659, -0.45027542]])
```

In [47]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion = 'entropy')
dtc.fit(x_train,y_train)
```

Out[47]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       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='deprecated',
                       random_state=None, splitter='best')
```

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

Out[49]:
```
array([0, 0, 0, ..., 1, 0, 0])
```

In [50]:
```
y_test
```

Out[50]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [51]:
```
# evaluation metrics are different
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[51]:
```
0.8235714285714286
```

In [52]:
```
# plotting confusion matrix
from sklearn.metrics import confusion_matrix
```

In [53]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[53]:
```
array([[1071,  133],
       [ 114,   82]], dtype=int64)
```

In [78]:
```
#predicting a random value
dtc.predict([[1,0,0,1,0,0,0,0,1,1,30000,0,0,1,0,33]])
```

Out[78]:
```
array([0])
```

In [77]:
```
# as we scaled inputs for model fitting random prediction should also be scaled
dtc.predict(sc.transform([[1,0,0,1,0,0,0,0,1,1,30000,0,0,1,0,33]]))
```

Out[77]:
```
array([1])
```

In [69]:
```
#random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=10,ma
```

In [70]:
```
rfc.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConvers
ionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  """Entry point for launching an IPython kernel.
```

Out[70]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=10, max_features='auto',
                       max_leaf_nodes=7, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [71]:
```
y_pred_rfc = rfc.predict(x_test)
y_pred_rfc
```

Out[71]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [72]:
```
y_test
```

Out[72]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [73]:
```
accuracy_score(y_test,y_pred_rfc)
```

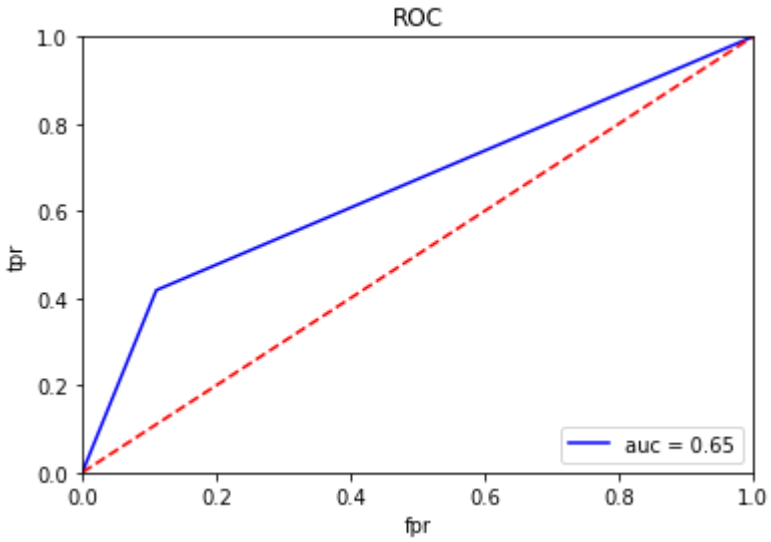
Out[73]:
```
0.86
```

In [74]:
```
confusion_matrix(y_test,y_pred_rfc)
```

Out[74]:
```
array([[1204,    0],
       [ 196,    0]], dtype=int64)
```

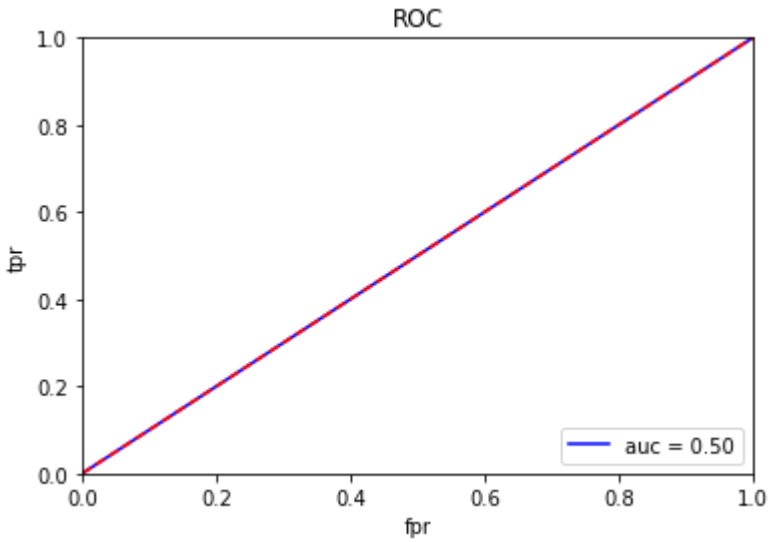
In [79]:
```
#auc, #roc
#area under curve
import sklearn.metrics as metrics
fpr,tpr,threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

Out[79]:
```
Text(0, 0.5, 'tpr')
```



In [80]:
```
#auc, #roc
#area under curve
import sklearn.metrics as metrics
fpr,tpr,threshold = metrics.roc_curve(y_test,y_pred_rfc)
roc_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

Out[80]:
```
Text(0, 0.5, 'tpr')
```



In [ ]: