In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
dataset = pd.read_csv('bikebuyer1.csv')
```

In [3]:
```
dataset
```

Out[3]:

	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 [4]:
```
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 [5]:
```
dataset.isnull().any()
dataset.isnull().sum()
```

Out[5]:
```
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 [6]:
```
dataset['Marital Status'].unique()
```

Out[6]:
```
array(['Single', 'Married', nan], dtype=object)
```

In [7]:
```
p=dataset['Marital Status'].value_counts()
```

In [8]:
```
p[0]
```

Out[8]:
```
4133
```

In [9]:
```
dataset['Gender'].value_counts()
```

Out[9]:
```
Male      3527
Female    3441
Name: Gender, dtype: int64
```

In [10]:
```
dataset['Marital Status'].fillna(dataset['Marital Status'].mode()[0],inplace=True
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=T
```

In [11]:
```
dataset.isnull().any()
```

Out[11]:
```
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 [12]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

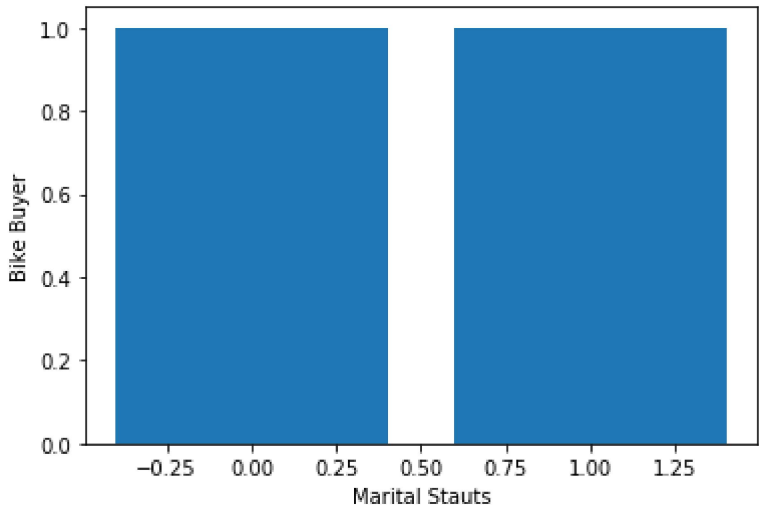
In [13]:
```
dataset['Marital Status']=le.fit_transform(dataset['Marital Status'])
dataset['Gender']=le.fit_transform(dataset['Gender'])
dataset['Education']=le.fit_transform(dataset['Education'])
dataset['Occupation']=le.fit_transform(dataset['Occupation'])
dataset['Home Owner']=le.fit_transform(dataset['Home Owner'])
dataset['Region']=le.fit_transform(dataset['Region'])
dataset['Bike Buyer']=le.fit_transform(dataset['Bike Buyer'])
```

In [14]:
```
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
```

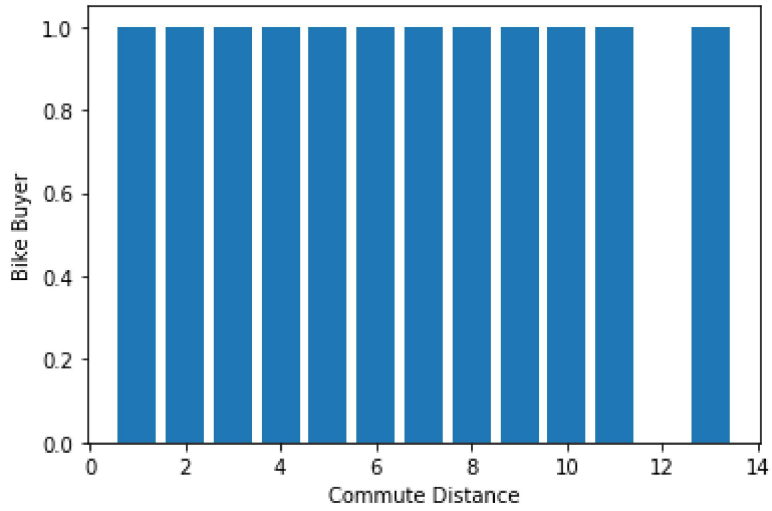
In [58]:
```
plt.bar(dataset['Marital Status'],dataset['Bike Buyer'])
plt.xlabel('Marital Stauts')
plt.ylabel('Bike Buyer')
```

Out[58]:
```
Text(0, 0.5, 'Bike Buyer')
```



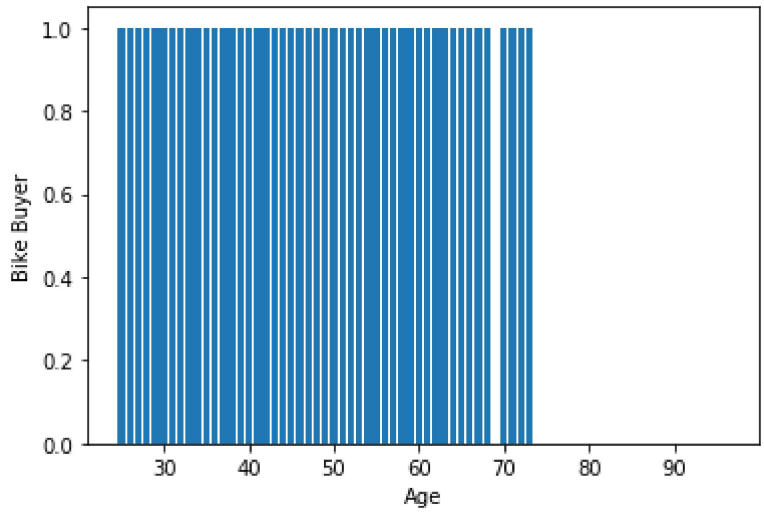
In [59]:
```
plt.bar(dataset['Commute Distance'],dataset['Bike Buyer'])
plt.xlabel('Commute Distance')
plt.ylabel('Bike Buyer')
```

Out[59]: Text(0, 0.5, 'Bike Buyer')



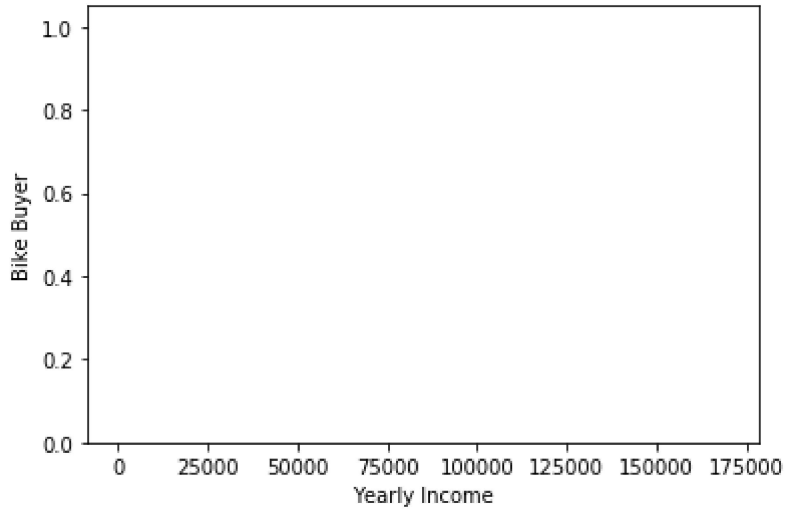
In [60]:
```
plt.bar(dataset['Age'],dataset['Bike Buyer'])
plt.xlabel('Age')
plt.ylabel('Bike Buyer')
```

Out[60]: Text(0, 0.5, 'Bike Buyer')



In [62]:
```
plt.bar(dataset['Yearly Income'],dataset['Bike Buyer'])
plt.xlabel('Yearly Income')
plt.ylabel('Bike Buyer')
```

Out[62]:
```
Text(0, 0.5, 'Bike Buyer')
```

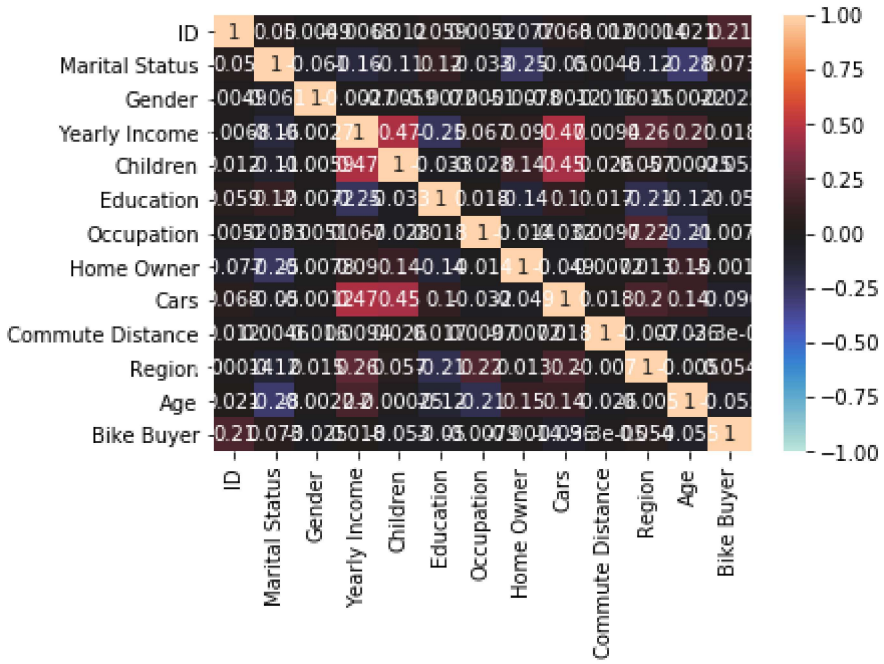


In [61]:
```
import seaborn as sns
sns.pairplot(dataset, hue = 'Bike Buyer')
```

Out[61]: <seaborn.axisgrid.PairGrid at 0x29096134048>

In [15]:
```
import seaborn as sns
sns.heatmap(dataset.corr(),annot=True,vmin=-1,vmax=1,center=0)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x290953830c8>
```



In [16]:
```
dataset.drop(['ID','Education'],axis=1,inplace=True)
```

In [17]:
```
dataset.shape
```

Out[17]:
```
(6997, 11)
```

In [18]:
```
dataset['Occupation'].value_counts()
```

Out[18]:
```
3    2031
4    1748
1    1265
2     990
0     963
Name: Occupation, dtype: int64
```

In [19]:
```
dataset['Region'].value_counts()
```

Out[19]:
```
1    3728
0    2096
2    1173
Name: Region, dtype: int64
```

In [20]:
```
dataset['Gender'].value_counts()
```

Out[20]:
```
1    3556
0    3441
Name: Gender, dtype: int64
```

In [21]:
```
dataset.head(1)
```

Out[21]:

	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

In [22]:
```
x = dataset.iloc[:,[0,4,5,7]].head(2)
x
```

Out[22]:

	Marital Status	Occupation	Home Owner	Commute Distance
0	1	0	0	1.0
1	0	0	1	1.0

In [23]:
```
#input
x = dataset.iloc[:,0:10].values
x
```

Out[23]:
```
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 [24]:
```
#target
y=dataset.iloc[:,-1:].values
y
```

Out[24]:
```
array([[1],
       [1],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [25]:
```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
```

In [26]:
```
z=oh.fit_transform(x[:,4:5]).toarray()
t=oh.fit_transform(x[:,8:9]).toarray()
```

In [27]:
```
z
```

Out[27]:
```
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 [28]:
```
t
```

Out[28]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

In [29]:
```
x=np.delete(x,[4,8],axis=1)
```

In [30]:
```
x.shape
```

Out[30]:
```
(6997, 8)
```

In [31]:
```
#region,occupation
x=np.concatenate((t,z,x),axis=1)
```

In [32]:
```
x.shape
```

Out[32]:
```
(6997, 16)
```

In [33]:
```
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_state=0)
```

In [34]:
```
x_train.shape
```

Out[34]:
```
(5597, 16)
```

In [35]:
```
x_test.shape
```

Out[35]:
```
(1400, 16)
```

In [36]:
```
y_train.shape
```

Out[36]:
```
(5597, 1)
```

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

Out[37]:
```
(1400, 1)
```

In [38]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [39]:
```
x_train
```

Out[39]:
```
array([[-0.64906073,  0.93007795, -0.4484116 , ...,  0.36401125,
        -0.75940145, -1.26244529],
       [-0.64906073, -1.0751787 ,  2.23009397, ...,  2.10312435,
        -0.75940145, -0.8428397 ],
       [ 1.54068787, -1.0751787 , -0.4484116 , ..., -1.37510184,
         1.63267038, -0.25539189],
       ...,
       [ 1.54068787, -1.0751787 , -0.4484116 , ..., -1.37510184,
        -1.101126  , -0.339313  ],
       [-0.64906073, -1.0751787 ,  2.23009397, ...,  1.2335678 ,
         1.29094583, -0.92676082],
       [-0.64906073,  0.93007795, -0.4484116 , ...,  2.10312435,
         1.63267038,  2.09439939]])
```

In [40]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy')
dtc.fit(x_train,y_train)
```

Out[40]:
```
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 [41]:
```
y_pred = dtc.predict(x_test)
y_pred
```

Out[41]:
```
array([0, 0, 0, ..., 1, 0, 0])
```

In [42]:
```
y_test
```

Out[42]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [43]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)
```

Out[43]:
```
0.8271428571428572
```

In [44]:
```
from sklearn.metrics import confusion_matrix
```

In [45]:
```
cm=confusion_matrix(y_test,y_pred)
```

In [46]:
```
cm
```

Out[46]:
```
array([[1073,  131],
       [ 111,   85]], dtype=int64)
```

In [47]:
```
dataset.head(1)
```

Out[47]:

	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

In [48]:
```
#1  1   30000   0.0 0   0   1   1.0 0   33

dtc.predict(sc.transform([[1., 0., 0.,1., 0., 0., 0., 0.,1,1,30000,0,0,1,0,33]]))
```

Out[48]:
```
array([1])
```

In [49]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=10,ma
```

In [50]:
```
rfc.fit(x_train,y_train)
```

```
C:\Users\PRUDHVI\anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  """Entry point for launching an IPython kernel.
```

Out[50]:
```
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 [51]:
```
y_pred_rfc = rfc.predict(x_test)
```

In [52]:
```
y_pred_rfc
```

Out[52]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [53]:
```
y_test
```

Out[53]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [54]:
```
accuracy_score(y_test,y_pred_rfc)
```

Out[54]:
```
0.86
```

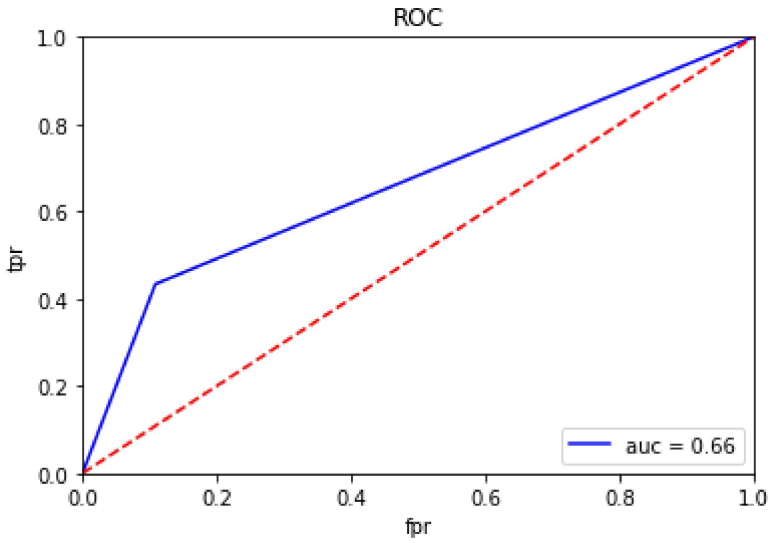
In [55]:
```
confusion_matrix(y_test,y_pred_rfc)
```

Out[55]:
```
array([[1204,    0],
       [ 196,    0]], dtype=int64)
```

In [56]:
```
#auc, #roc
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')
```

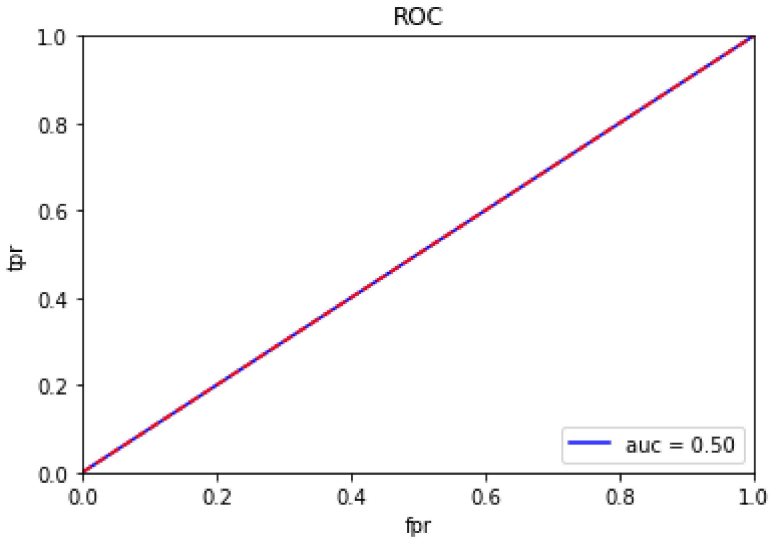
Out[56]: Text(0, 0.5, 'tpr')



In [57]:
```
#auc, #roc
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[57]:
```
Text(0, 0.5, 'tpr')
```



In [ ]:
```

```