In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset=pd.read_csv(r"D:\ML_Course\Works_on_python\Decision tree & Random Forest Calssification\bikebuyer1.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	ID	Marital Status	Gender	Yearly Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Bike Buyer
0	22711.0	Single	Male	30000	0.0	Partial College	Clerical	No	1	1.0	Europe	33	Yes
1	13555.0	Married	Female	40000	0.0	Graduate Degree	Clerical	Yes	0	1.0	Europe	37	Yes
2	NaN	Married	Male	160000	5.0	Partial College	Professional	No	3	2.0	Europe	55	No
3	2.0	Single	Male	160000	0.0	Graduate Degree	Management	Yes	2	5.0	Pacific	47	No
4	25410.0	NaN	Female	70000	2.0	Bachelors	Skilled Manual	No	1	1.0	North America	38	Yes

In [4]:
```
dataset.isnull().any()
```

Out[4]:
```
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 [5]:
```
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) #Don't take mean, else you might get 2.5 children etc
dataset['Commute Distance'].fillna(dataset['Commute Distance'].median(), inplace =True) #You can choose any method here
```

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

Out[6]:
```
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 [7]:
```
dataset.drop(["ID","Education","Home Owner"],axis=1,inplace=True)
```

In [8]:
```
dataset.head()
```

Out[8]:

	Marital Status	Gender	Yearly Income	Children	Occupation	Cars	Commute Distance	Region	Age	Bike Buyer
0	Single	Male	30000	0.0	Clerical	1	1.0	Europe	33	Yes
1	Married	Female	40000	0.0	Clerical	0	1.0	Europe	37	Yes
2	Married	Male	160000	5.0	Professional	3	2.0	Europe	55	No
3	Single	Male	160000	0.0	Management	2	5.0	Pacific	47	No
4	Married	Female	70000	2.0	Skilled Manual	1	1.0	North America	38	Yes

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

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

In [10]:
```
dataset["Gender"].unique()
```

Out[10]:
```
array(['Male', 'Female'], dtype=object)
```

In [11]:
```
dataset["Occupation"].unique()
```

Out[11]:
```
array(['Clerical', 'Professional', 'Management', 'Skilled Manual',
       'Manual'], dtype=object)
```

In [12]:
```
dataset["Region"].unique()
```

Out[12]:
```
array(['Europe', 'Pacific', 'North America'], dtype=object)
```

In [13]:
```
dataset["Bike Buyer"].unique()
```

Out[13]:
```
array(['Yes', 'No'], dtype=object)
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['Marital Status']=le.fit_transform(dataset['Marital Status'])
dataset['Gender']=le.fit_transform(dataset['Gender'])
dataset['Occupation']=le.fit_transform(dataset['Occupation'])
dataset['Region']=le.fit_transform(dataset['Region'])
dataset['Bike Buyer']=le.fit_transform(dataset['Bike Buyer'])
```

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

Out[15]:

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

In [16]:
```
x=dataset.iloc[:,0:9].values
y=dataset.iloc[:,9:10].values
```

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

Out[17]:
```
(6997, 9)
```

In [18]:
```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
z = one.fit_transform(x[:,4:5]).toarray()
t = one.fit_transform(x[:,7:8]).toarray()
x = np.delete(x,[4,7],axis=1)
x = np.concatenate((t,z,x), axis=1)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data
will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future
they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneH
otEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data
will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future
they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneH
otEncoder directly.
  warnings.warn(msg, FutureWarning)
```

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

Out[19]:
```
(6997, 15)
```

In [20]:
```
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)
```

For classification need to consider feature scaling

In [21]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

DecisionTreeClassifier

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion="entropy",random_state=0)
dtc.fit(x_train,y_train)
```

Out[22]:
```
DecisionTreeClassifier(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=False, random_state=0,
            splitter='best')
```

In [23]:
```
y_pred=dtc.predict(x_test)
```

In [24]:
```
y_pred
```

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

In [25]:
```
y_test
```

Out[25]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [26]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[26]:
```
0.8164285714285714
```

In [27]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

In [28]:
```
cm # tpr o.88 #fpr 0.5
```

Out[28]:
```
array([[1060,  144],
       [ 113,   83]], dtype=int64)
```

auc and roc curve

In [29]:
```
import sklearn.metrics as metrics
fpr,tpr,threshold=metrics.roc_curve(y_test,y_pred)
roc_auc=metrics.auc(fpr,tpr)
```

In [30]:
```
threshold
```

Out[30]:
```
array([2, 1, 0])
```

In [31]:
```
fpr
```

Out[31]:
```
array([0.        , 0.11960133, 1.        ])
```

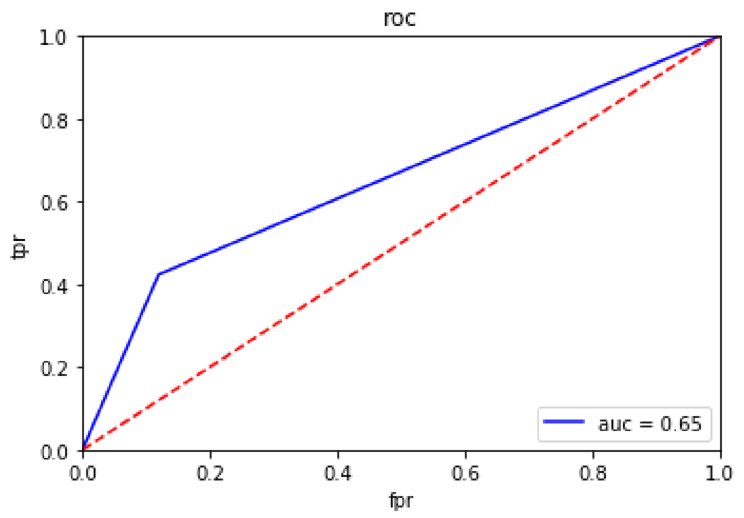
In [32]:
```
tpr
```

Out[32]:
```
array([0.        , 0.42346939, 1.        ])
```

here for threshold 0 fpr 1 and tpr 1 taken

In [34]:
```
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.ylabel('tpr')
plt.xlabel('fpr')
```

Out[34]:
```
Text(0.5, 0, 'fpr')
```



if the auc roc curve <0.75 then not a very good model if the auc roc curve >0.75 then model is tuned in a proper way if the auc roc curve 90-95 then model is perfectly tuned

In [35]:
```
dataset.head(1)
```

Out[35]:

	Marital Status	Gender	Yearly Income	Children	Occupation	Cars	Commute Distance	Region	Age	Bike Buyer
0	1	1	30000	0.0	0	1	1.0	0	33	1

In [36]:
```
#region then occu then married then male then income then child then cars then dist then age
y=dtc.predict(sc.transform([[0,1,0,1,0,0,0,0,0,1,100000,4,2,5,32]]))#
```

In [37]:
```
y
```

Out[37]:
```
array([0])
```

RandomForestClassifier

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 30,criterion = 'entropy',random_state = 1)#hyper tuning parameters
```

In [39]:
```
rfc.fit(x_train,y_train)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConversionWarning: 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[39]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            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=30, n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False)
```

In [40]:
```
y_pred1 = rfc.predict(x_test)
```

In [41]:
```
y_pred1
```
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_pred1,y_test)
```
Out[43]:
```
0.8464285714285714
```

In [44]:
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_pred1)
```

In [45]:
```
cm1
```
Out[45]:
```
array([[1131,   73],
       [ 142,   54]], dtype=int64)
```

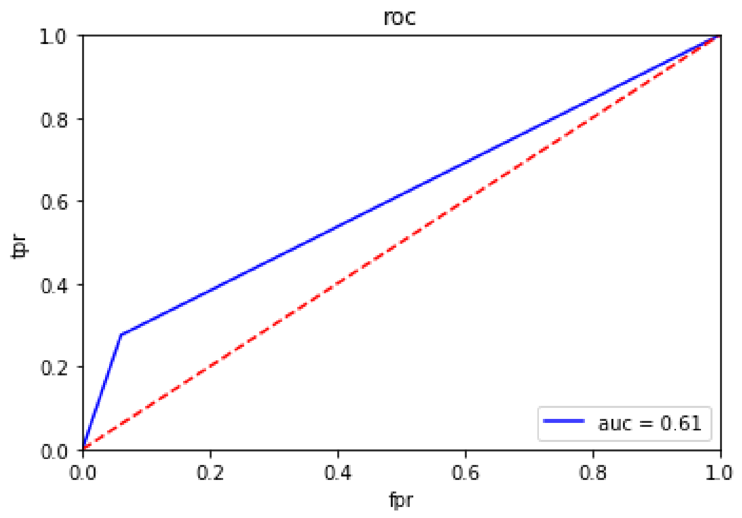
In [46]:
```
import sklearn.metrics as metrics
fpr1,tpr1 , threshold = metrics.roc_curve(y_test,y_pred1)
roc_auc1 = metrics.auc(fpr1,tpr1)
```

In [72]:
```
fpr1
```
Out[72]:
```
array([0.        , 0.06063123, 1.        ])
```

In [73]:
```
tpr1
```
Out[73]:
```
array([0.       , 0.2755102, 1.       ])
```

In [47]:
```
plt.title("roc")
plt.plot(fpr1,tpr1,'b',label = 'auc = %0.2f'%roc_auc1)
plt.legend(loc = 'lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('tpr')
plt.xlabel('fpr')
```

Out[47]:
```
Text(0.5, 0, 'fpr')
```



Logistic Regresion

In [48]:
```
from sklearn.linear_model import LogisticRegression
log=LogisticRegression()
log.fit(x_train,y_train)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

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_pred2=log.predict(x_test)
```

In [50]:
```
y_pred2
```

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

In [51]:
```
y_test
```

Out[51]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

In [52]:
```
cm2=confusion_matrix(y_test,y_pred2)
```

In [53]:
```
cm2
```

Out[53]:
```
array([[1204,    0],
       [ 195,    1]], dtype=int64)
```

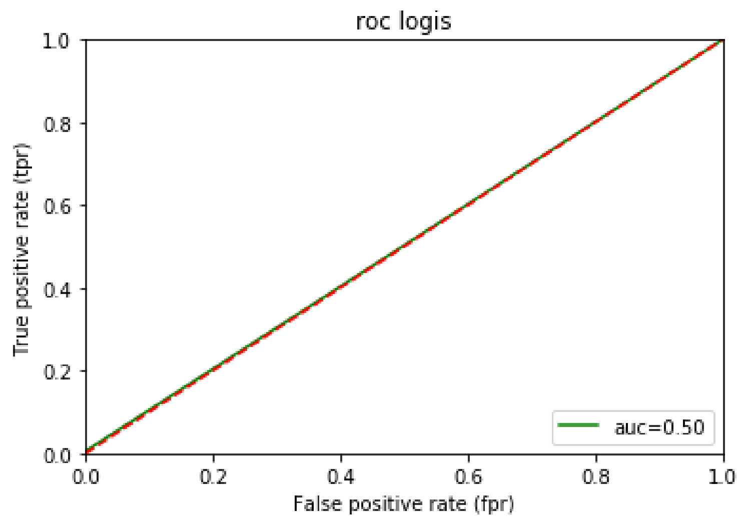
```
In [54]: accuracy_score(y_test,y_pred2)

Out[54]: 0.8607142857142858
```

```
In [66]: fpr3,tpr3,threshold3=metrics.roc_curve(y_test,y_pred2)
         roc_auc3=metrics.auc(fpr3,tpr3)
```

```
In [67]: import matplotlib.pylab as plt
         plt.title("roc logis")
         plt.plot(fpr3,tpr3,'g',label='auc=%0.2f'%roc_auc3)
         plt.legend(loc = 'lower right')
         plt.plot([0,1],[0,1],'r--')
         plt.xlim([0,1])
         plt.ylim([0,1])
         plt.ylabel('True positive rate (tpr)')
         plt.xlabel('False positive rate (fpr)')

Out[67]: Text(0.5, 0, 'False positive rate (fpr)')
```



worst curve

```
In [69]: threshold3# why 2 1 0 always

Out[69]: array([2, 1, 0])
```

In [70]:
```
fpr3
```

Out[70]:
```
array([0., 0., 1.])
```

In [71]:
```
tpr3
```

Out[71]:
```
array([0.        , 0.00510204, 1.        ])
```

K Nearest Neighbor

In [75]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski')
knn.fit(x_train,y_train)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: 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().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[75]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [76]:
```
ypred3=knn.predict(x_test)
```

In [77]:
```
ypred3
```

Out[77]:
```
array([0, 0, 0, ..., 0, 1, 0])
```

In [78]:
```
cm3=confusion_matrix(y_test,ypred3)
```

In [79]:
```
cm3
```

Out[79]:
```
array([[1161,   43],
       [ 162,   34]], dtype=int64)
```

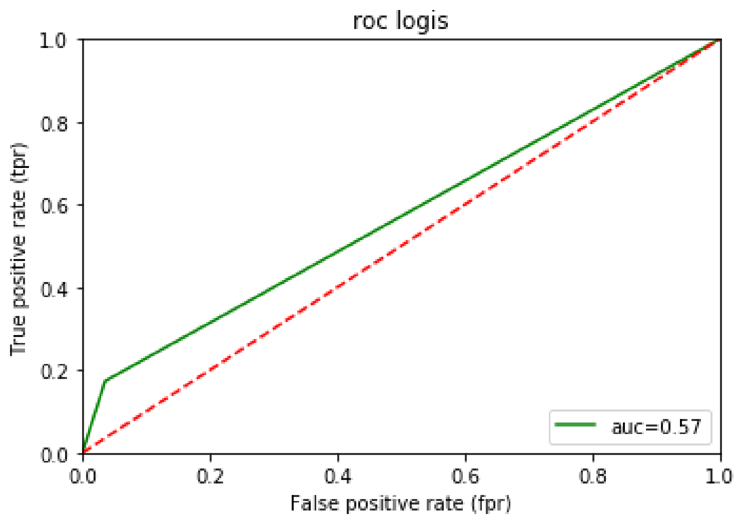
In [80]:
```
accuracy_score(y_test,ypred3)
```

Out[80]:
```
0.8535714285714285
```

In [81]:
```
fpr4,tpr4,threshold4=metrics.roc_curve(y_test,ypred3)
roc_auc4=metrics.auc(fpr4,tpr4)
```

In [82]:
```
import matplotlib.pylab as plt
plt.title("roc logis")
plt.plot(fpr4,tpr4,'g',label='auc=%0.2f'%roc_auc4)
plt.legend(loc = 'lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True positive rate (tpr)')
plt.xlabel('False positive rate (fpr)')
```

Out[82]:
```
Text(0.5, 0, 'False positive rate (fpr)')
```



this is also worst