Decision Tree - Python

Data set: bankloan.csv

Set Working Directory

```
In [1]: # Jesus is my Saviour!!

In [2]: import os

In [3]: os.getcwd()
Out[3]: 'C:\\Users\\Dr Vinod\\Desktop\\WD_python'
```

Import Libraries

```
In [4]: import pandas as pd

In [5]: import matplotlib.pyplot as plt

In [6]: import pylab as pl

In [7]: import sklearn

In [8]: from sklearn import tree


In [9]: from sklearn import metrics

In [10]: from sklearn.tree import DecisionTreeClassifier

In [11]: from sklearn.model_selection import train_test_split
```

Import Libraries

```
In [12]: from sklearn.metrics import confusion_matrix

In [13]: from sklearn.metrics import roc_curve, auc, roc_auc_score

In [14]: from sklearn.metrics import accuracy_score

In [15]: from sklearn.metrics import classification_report

In [16]: import graphviz
```

```
!pip install graphviz

import graphviz
```

Import Data Set & Explore

```
In [17]: bankloan = pd.read_csv("C:/Users/Dr Vinod/Desktop/DataSets1/bankloan.csv")

In [18]: bankloan = pd.DataFrame(bankloan)

In [19]: bankloan.shape
Out[19]: (700, 9)

In [20]: bankloan.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 9 columns):
age         700 non-null int64
ed          700 non-null int64
employ      700 non-null int64
address     700 non-null int64
income      700 non-null int64
debtinc     700 non-null float64
creddebt    700 non-null float64
othdebt     700 non-null float64
default     700 non-null int64
dtypes: float64(3), int64(6)
memory usage: 49.3 KB
```

bankloan - DataFrame

Index	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.36	5.01	1
1	27	1	10	6	31	17.3	1.36	4	0
2	40	1	15	14	55	5.5	0.86	2.17	0
3	41	1	15	14	120	2.9	2.66	0.82	0
4	24	2	2	0	28	17.3	1.79	3.06	1
5	41	2	5	5	25	10.2	0.39	2.16	0
6	39	1	20	9	67	30.6	3.83	16.67	0
7	43	1	12	11	38	3.6	0.13	1.24	0
8	24	1	3	4	19	24.4	1.36	3.28	1
9	36	1	0	13	25	19.7	2.78	2.15	0
10	27	1	0	1	16	1.7	0.18	0.09	0

Format Resize Background color Column min/max Save and Close Close

Explore

```
In [21]: bl = bankloan.ix[:, (0,4,6,7,8)]
__main__:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated

In [22]: bl.shape
Out[22]: (700, 5)

In [23]: bl.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 5 columns):
age         700 non-null int64
income      700 non-null int64
creddebt    700 non-null float64
othdebt     700 non-null float64
default     700 non-null int64
dtypes: float64(2), int64(3)
memory usage: 27.4 KB
```

Predictors

```
In [24]: X = bl.ix[:, (0,1,2,3)]
__main__:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated

In [25]: X.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 4 columns):
age         700 non-null int64
income      700 non-null int64
creddebt    700 non-null float64
othdebt     700 non-null float64
dtypes: float64(2), int64(2)
memory usage: 22.0 KB
```

Target Variable

```
In [26]: y = bl.ix[:, 4]
__main__:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated

In [27]: y.head(4)
Out[27]:
0    1
1    0
2    0
3    0
Name: default, dtype: int64
```

Train & Test Data

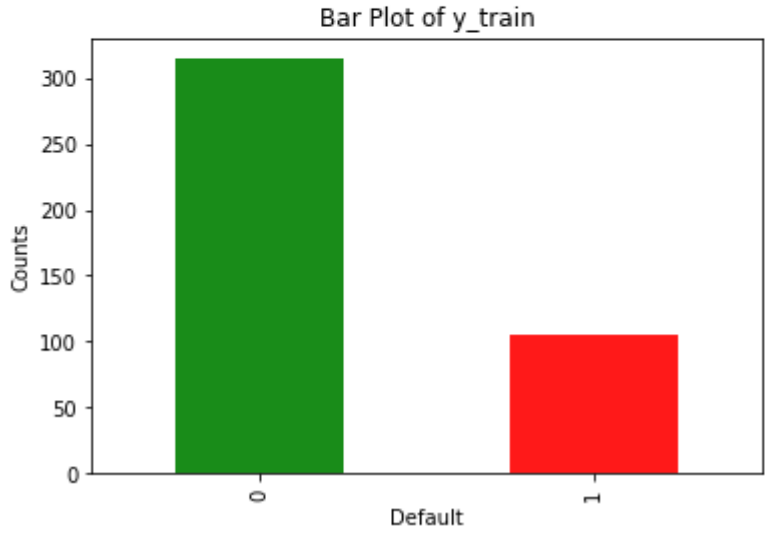
```
In [28]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4)

In [29]: len(X_train) # 420
Out[29]: 420

In [30]: len(y_train)
Out[30]: 420

In [31]: len(X_test) # 280
Out[31]: 280

In [32]: len(y_test)
Out[32]: 280
```

Check Proportions

```
In [33]: #_________Proportions checking

In [34]: import matplotlib.pyplot as plt

In [35]: trn = y_train.value_counts()

In [36]: trn # 0= 308, 1= 112
Out[36]:
0    314
1    106
Name: default, dtype: int64

In [37]: 106/314 # 112/308 = 36.36%
Out[37]: 0.3375796178343949
```

Train Set

```
In [38]: trn.plot.bar(color = ('g', 'r'), alpha = 0.9)
    ...: plt.title('Bar Plot of y_train')
    ...: plt.xlabel("Default")
    ...: plt.ylabel("Counts")
    ...: plt.show
    ...: ##_____run in block
Out[38]: <function matplotlib.pyplot.show(*args, **kw)>
```

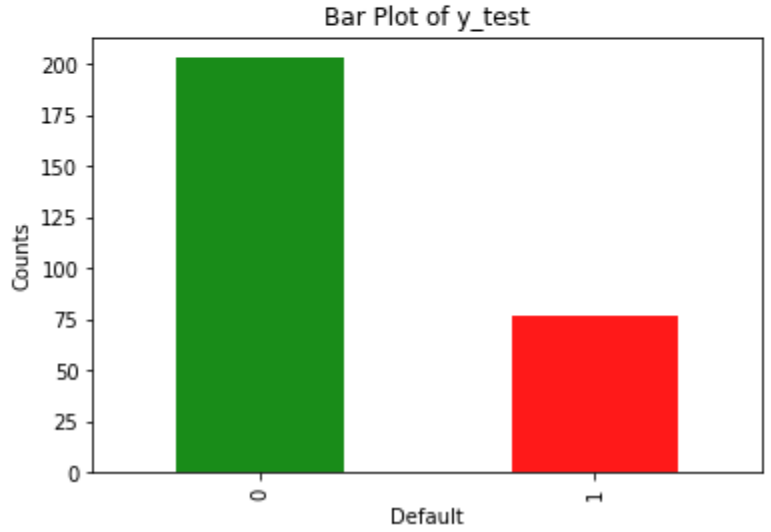


Test Set

```
In [39]: tst = y_test.value_counts()

In [40]: tst # 0= 209, 1= 71
Out[40]:
0    203
1     77
Name: default, dtype: int64

In [41]: 77/203 # 71/209 = 33.97%
Out[41]: 0.3793103448275862
```

Test Set

```
In [42]: tst.plot.bar(color = ('g', 'r'), alpha = 0.9)
    ...: plt.title('Bar Plot of y_test')
    ...: plt.xlabel("Default")
    ...: plt.ylabel("Counts")
    ...: plt.show
    ...: ##________run in block
    ...:
Out[42]: <function matplotlib.pyplot.show(*args, **kw)>
```

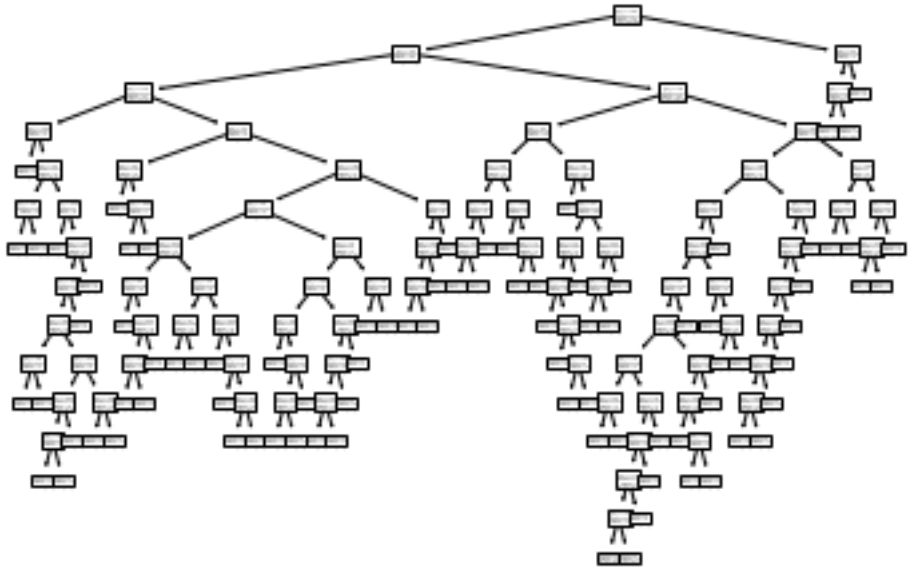


Build Model

```
In [43]: #_____________________________________Model

In [44]: clf = tree.DecisionTreeClassifier()

In [45]: clfFit = clf.fit(X_train, y_train)

In [46]: clfFit
Out[46]:
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')
```

Plot Tree

```
In [47]: #_____________________plot tree

In [48]: tree.plot_tree(clfFit)
Out[48]:
[Text(231.83008474576272, 210.192, 'X[2] <= 5.945\nentropy = 0.377\nsamples =
420\nvalue = [314, 106]'),
 Text(148.95762711864407, 195.696, 'X[1] <= 29.5\nentropy = 0.356\nsamples =
406\nvalue = [312, 94]'),
```

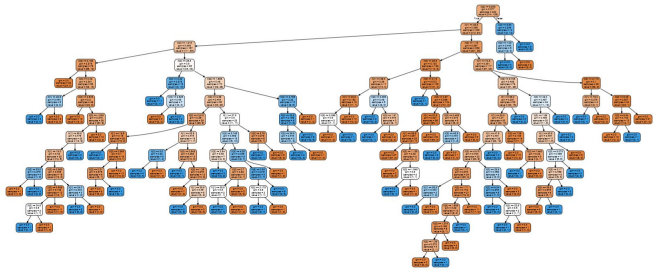


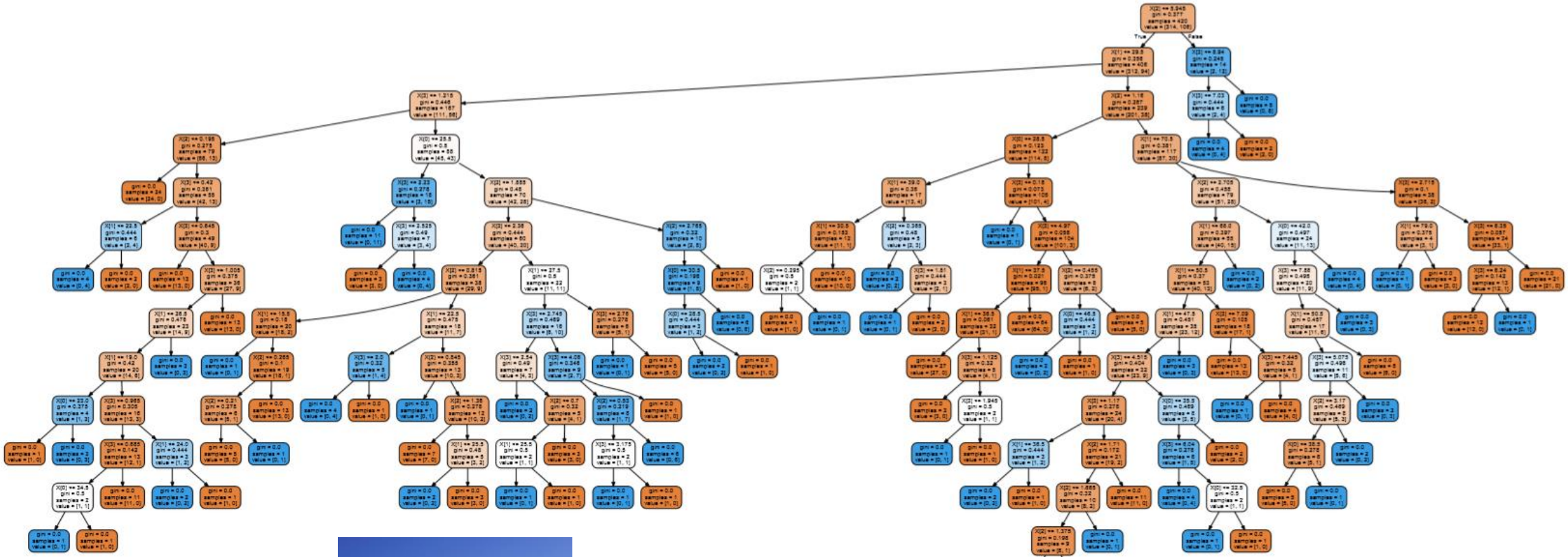
Tree Plot

```
In [49]: #_____graphviz plot

In [50]: import graphviz

In [51]: dot_data = tree.export_graphviz(clf, out_file = None)

In [52]: graph = graphviz.Source(dot_data)

In [53]: graph.render("bl")
Out[53]: 'bl.pdf'
```

Colored Tree

```
In [54]: dot_data = tree.export_graphviz(clf, out_file = None,
    ...:                                 filled = True, rounded = True)
    ...: ##_________run in block

In [55]: graph = graphviz.Source(dot_data)

In [56]: graph.render("bl_1")
Out[56]: 'bl_1.pdf'
```





Prediction


```
In [57]: #__________Prediction

In [58]: y_predict = clfFit.predict(X_test)

In [59]: # dont run y_predict, all 280 output will come

In [60]: # confusion matrix

In [61]: cm_tree = pd.crosstab(y_test,
    ...:                       y_predict,
    ...:                       rownames = ["Actual"],
    ...:                       colnames = ["Predicted"],
    ...:                       margins = True)
    ...: ##_________run in block
    ...: cm_tree
Out[61]:
Predicted    0   1  All
Actual
0          149  54  203
1           40  37   77
All        189  91  280
```

Result


```
In [62]: #_____________Direct from sklearn

In [63]: confusion_matrix(y_test, y_predict)
Out[63]:
array([[149,  54],
       [ 40,  37]], dtype=int64)

In [64]: print(classification_report(y_test, y_predict))
              precision    recall  f1-score   support

           0       0.79      0.73      0.76       203
           1       0.41      0.48      0.44        77

    accuracy                           0.66       280
   macro avg       0.60      0.61      0.60       280
weighted avg       0.68      0.66      0.67       280
```

```
In [65]: #________accuracy

In [66]: from sklearn.metrics import accuracy_score

In [67]: accuracy_score(y_test, y_predict)
Out[67]: 0.6642857142857143
```

ROC Curve

```
In [68]: #_______ROC Curve Preperation, fpr, tpr, auc

In [69]: predictedProbability_tree = clfFit.predict_proba(X_test)[:, 1]

In [70]: # mind : and ;

In [71]: fpr, tpr, thresholds = metrics.roc_curve(y_test, predictedProbability_tree)

In [72]: fpr
Out[72]: array([0.        , 0.26600985, 1.        ])

In [73]: tpr
Out[73]: array([0.        , 0.48051948, 1.        ])

In [74]: thresholds
Out[74]: array([2., 1., 0.])
```

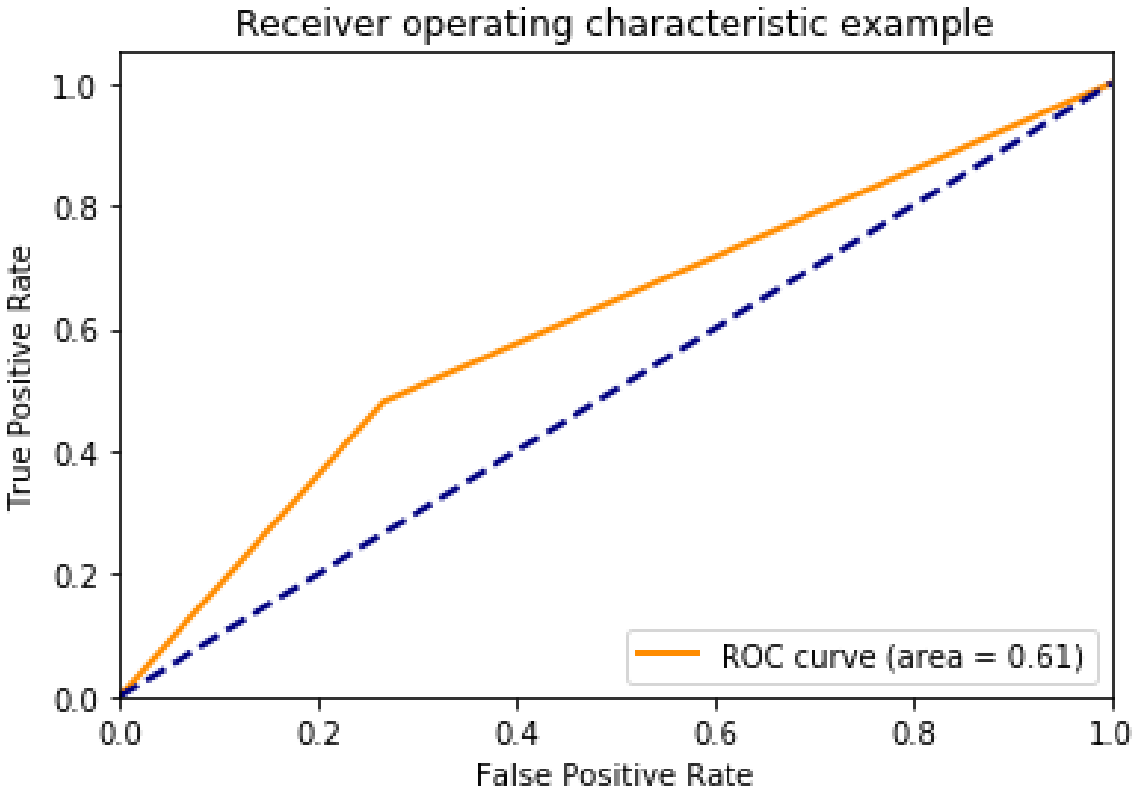
ROC Curve

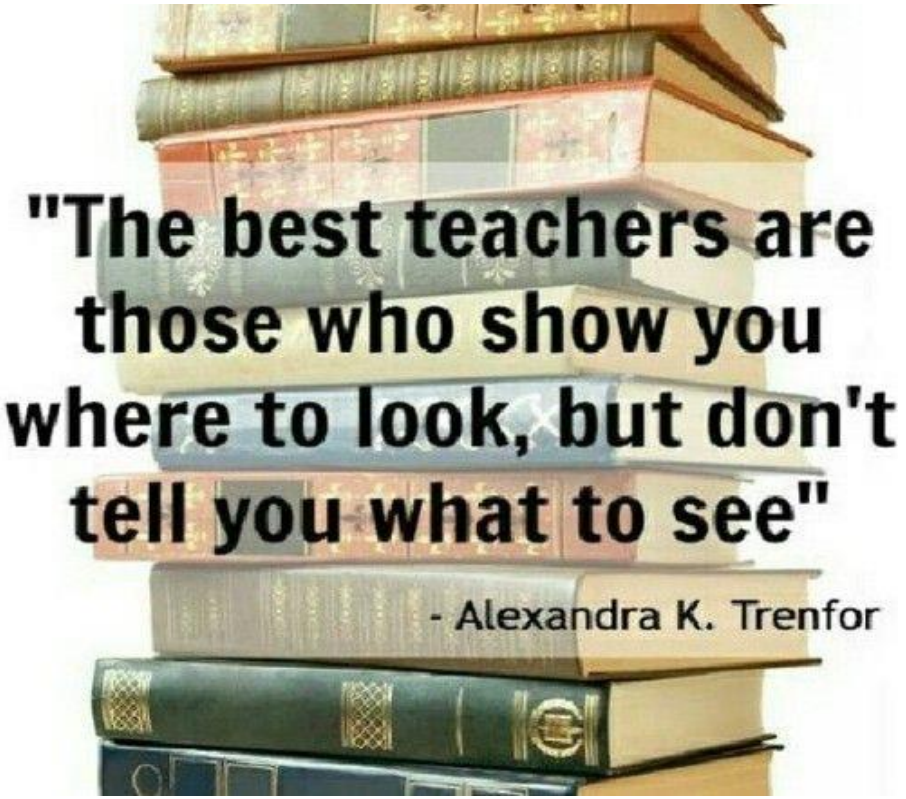
```
In [75]: # put fpr and tpr in a data frame

In [76]: df = pd.DataFrame(dict(fpr = fpr, tpr = tpr))

In [77]: auc = auc(fpr, tpr)

In [78]: auc
Out[78]: 0.607254814151366
```

ROC Curve

```
In [79]: #_________ROC Curve

In [80]: ##________run in block

In [81]: plt.figure()
    ...: lw = 2
    ...: plt.plot(fpr, tpr, color='darkorange',
    ...:          lw=lw, label='ROC curve (area = %0.2f)' % auc)
    ...: plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    ...: plt.xlim([0.0, 1.0])
    ...: plt.ylim([0.0, 1.05])
    ...: plt.xlabel('False Positive Rate')
    ...: plt.ylabel('True Positive Rate')
    ...: plt.title('Receiver operating characteristic example')
    ...: plt.legend(loc="lower right")
    ...: plt.show()
    ...: ##________run in block
```

ROC Curve


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.61)


"The best teachers are those who show you where to look, but don't tell you what to see"
- Alexandra K. Trenfor