DESC 624 - EDA and Decision Tree Notebook

HW #1

This notebook will contain both R and nearly equivalent Python code. Pick the code you prefer to work with and make adjustments to that code as you move through the notebook.

1. Remember to execute code as you go along by hitting the play button.
2. Do not run things out of order. You need to run things in sequence
3. If you pick Python, do **NOT** execute R blocks, if you pick R, do **NOT** execute the Python blocks
4. Exception to 3., run everything in Section 1 for both R and Python (this way you have the environments properly set up for both)
5. Please make a copy of this notebook to **YOUR** Google Drive. File -> Save a Copy in Drive
6. You will submit the actual notebook, so I can run it and you will submit a PDF version (File -> Print) of the complete notebook in which you have executed each coding block.
7. Please name your notebook and PDF version in the following way:

   HW1_lastname_firstname.ipynb for the notebook

   HW1_lastname_firstname.pdf for the PDF

You must use %%R at the top of each notebook block in order for the code to be interpreted as R code.

"#" before or after code is a comment

If you do not specify %%R at the top of the notebook block, the notebook will interpret the code as Python code.

You are welcome to use whatever your prefer or both Python and R to do the work. If you pick one or the other, before you finalize and submit your workbook, please delete the blocks of code you did not use (i.e., If you pick Python, delete all of the R code blocks)

You MUST use "#!pip install rpy2" "%load_ext rpy2.ipython"

Before any R code is executed. Notice how this block of code does NOT contain a %%R at the top.

```
#!pip3 install rpy2
%load_ext rpy2.ipython
```

```
The rpy2.ipython extension is already loaded. To reload it, use:
  %reload_ext rpy2.ipython
```

```
!apt-get install default-jre
!java -version
```

```
Reading package lists... Done
Building dependency tree
                                    ne
                                   st version (2:1.11-68ubuntu1~18.04.1).
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.
openjdk version "11.0.11" 2021-04-20
```

Saved successfully! ×

```
    OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.18.04)
    OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.18.04, mixed mode, sharing)
```

I. R Setup Sections

[ ] ↳ *2 cells hidden*

I. Python Setup Section

```
##Python Packages
import pandas as pd
from sklearn import tree
import pydotplus
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import matplotlib.image as pltimg
import seaborn as sn
import numpy as np
from scipy import stats
```

```
#Install python datatable which is the same as data.table() in R
!pip install datatable
import datatable as dt

from datatable import (dt, f, by, ifelse, update, sort,
                       count, min, max, mean, sum, rowsum)
```

```
    Requirement already satisfied: datatable in /usr/local/lib/python3.7/dist-packages (0.11.1)
```

```
#Python h2o

#h2o is the Machine Learning package we will be using. It has both an R implementation and a Python implemenation.
#The h2o.init() is commented out below because we are not going to use it for this HW.  You can only have on instance (
#of h2o running in any one session.

! pip install h2o

import h2o

#h2o.init()
#h2o.shutdown()
```

```
    Requirement already satisfied: h2o in /usr/local/lib/python3.7/dist-packages (3.32.1.3)
    Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from h2o) (0.16.0)
    Requirement already satisfied: colorama>=0.3.8 in /usr/local/lib/python3.7/dist-packages (from h2o) (0.4.4)
    Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from h2o) (0.8.9)
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from h2o) (2.23.0)
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->h2o)
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests-
```

Saved successfully! ×

II. R Data Loading

[ ] ↳ *3 cells hidden*

II. Python Data Loading

[ ] ↳ *1 cell hidden*

III. R Data Exploration

[ ] ↳ *1 cell hidden*

III. Python Data Exploration

[ ] ↳ *17 cells hidden*

IV. R Data Processing

[ ] ↳ *1 cell hidden*

IV. Python Data Processing

```
#IV. Data Processing ---------------------------------------------
#A. Adjust values based on the review of the data
import numpy as np
bank_pd = bank_p.to_pandas()
bank_pd['pdays'] = bank_pd['pdays'].replace({999: np.NaN})
```

```
bank_pd.head()
```

```
   age           job  marital  education  default  balance  housing  loan  contact  day  month  duration  cam
0   58    management  married   tertiary       no     2143      yes    no  unknown    5    may       261
1   44    technician   single  secondary       no       29      yes    no  unknown    5    may       151
2   33  entrepreneur  married  secondary       no        2      yes   yes  unknown    5    may        76
3   47   blue-collar  married    unknown       no     1506      yes    no  unknown    5    may        92
4   33       unknown   single    unknown       no        1       no    no  unknown    5    may       198
```

```
bank_p=dt.Frame(bank_pd)
```

```
bank_p.head()
```

Saved successfully! ×

```
bank_p=bank_p[f.duration>0,:]   ### post hoc
```

```
del bank_p[:,'duration']  #Variable no good.  We do not know the duration of the call BEFORE the call occurs
```

```
bank_p.head()
```

```
#C. Build additional features
from scipy import stats
bank_p['previous_norm'] = stats.zscore(bank_p['previous'])
```

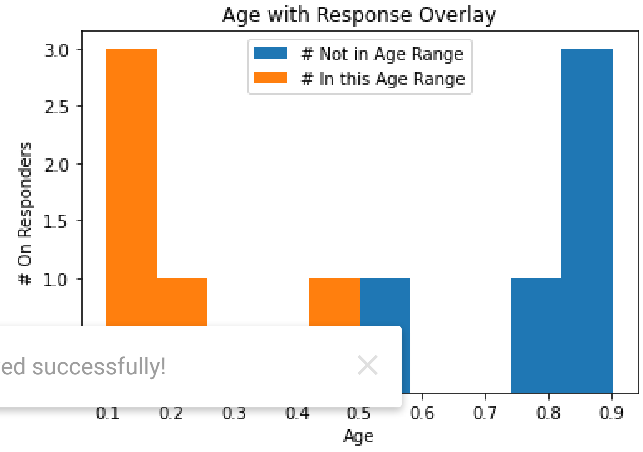
```
from scipy import stats
bank_p[:,'campaign_norm'] = stats.zscore(bank_p['campaign'])
```

```
#C. (cont) Build additional features
#Feature 2 - Age
bank_p[f.age<=17, 'Age_CAT'] = '<=17'
bank_p[(f.age>=18) & (f.age<=24), 'Age_CAT'] = ' 1)  18 to 24'
bank_p[(f.age>24) & (f.age<=30), 'Age_CAT'] = ' 2)  25 to 30'
bank_p[(f.age>30) & (f.age<=50), 'Age_CAT'] = ' 3)  31 to 50'
bank_p[(f.age>50) & (f.age<=64), 'Age_CAT'] = ' 4)  51 to 65'
bank_p[f.age>=65, 'Age_CAT']  = ' 5)  >= 65'
bank_p[:, sum(f.count), by('Age_CAT') ]
```

```
#C. (cont) Build additional features
#Feature 2 - Education
bank_p[f.education,'Education'] = 'primary'
# bank_p[(f.education=='unknown'), 'Education_CAT'] = 'Unknown or Student'
# bank_p[:, sum(f.count), by('Education') ]
```

```
#Feature 1 - Age in the Crosstabs
bank_pd=bank_p.to_pandas()
crosstab_01 = pd.crosstab(bank_pd['Age_CAT'], bank_pd['y'])
crosstab_norm = crosstab_01.div(crosstab_01.sum(1), axis = 0)
print(crosstab_norm)
crosstab_norm.plot(kind='hist', stacked = False)

plt.legend(['# Not in Age Range', '# In this Age Range'])
plt.title('Age with Response Overlay')
plt.xlabel('Age'); plt.ylabel('# On Responders'); plt.show()
```

```
y                     no       yes
Age_CAT
 1)  18 to 24  0.744129  0.255871
 2)  25 to 30  0.849220  0.150780
 3)  31 to 50  0.902123  0.097877
 4)  51 to 65  0.884889  0.115111
 5)  >= 65     0.579012  0.420988
```

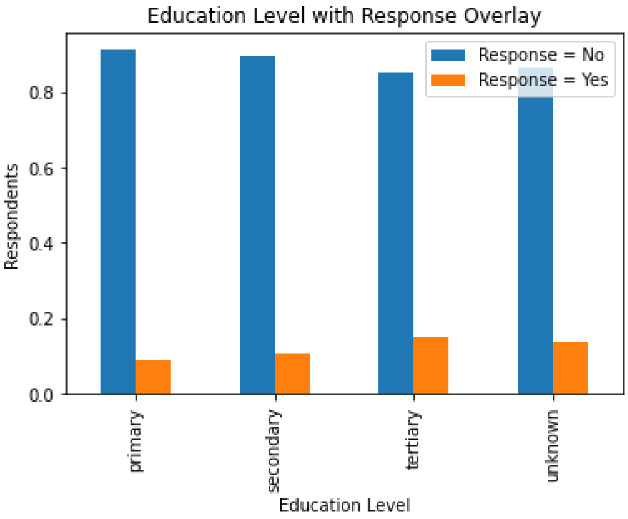


This is where you are going to do some EDA work. You need to plot your data with graphics. I want to see 4 plots of the data against the response variable. After each plot you need to summarize why the plot demonstrates or does not demonstrate that the feature could be useful in the model.

You also need to create 2 contingency tables against the response. Again, you need to summarize why the contingency tables could be useful in the model.

```
#C. Build additional features
#Feature 2 - Education in the Crosstabs
bank_pd=bank_p.to_pandas()
crosstab_02 = pd.crosstab(bank_pd['education'], bank_pd['y'])
print(crosstab_norm)
crosstab_norm = crosstab_02.div(crosstab_02.sum(1), axis = 0)
crosstab_norm.plot(kind='bar', stacked = False)

plt.legend(['Response = No', 'Response = Yes'])
plt.title('Education Level with Response Overlay')
plt.xlabel('Education Level'); plt.ylabel('Respondents'); plt.show()
```

```
y                    no       yes
Age_CAT
 1)  18 to 24  0.744129  0.255871
 2)  25 to 30  0.849220  0.150780
 3)  31 to 50  0.902123  0.097877
 4)  51 to 65  0.884889  0.115111
 5)  >= 65     0.579012  0.420988
```



V. R Data Analysis

[ ] ↳ *25 cells hidden*

V. Python Data Analysis

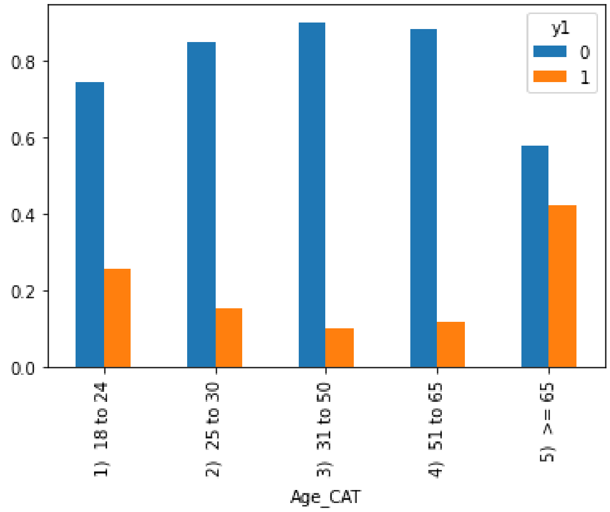
Saved successfully! ×

```
import h2o

h2o.init()
#h2o.shutdown()
```

```
# Dataframe
bank_pd = bank_p.to_pandas()
```

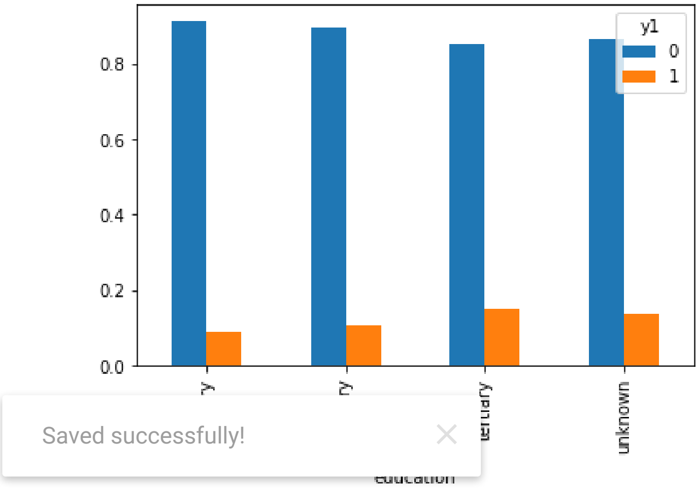
```
#Plot 1
# numeric_col = ['y1', 'age']

# corr_matrix = bank_pd.loc[:,numeric_col].corr()
# print(corr_matrix)

#Using heatmap to visualize the correlation matrix
# sn.heatmap(corr_matrix, annot=True)

crosstab_01 = pd.crosstab(bank_pd['Age_CAT'], bank_pd['y1'])
crosstab_norm = crosstab_01.div(crosstab_01.sum(1), axis = 0)
crosstab_norm.plot(kind='bar', stacked = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f224aa28750>
```



```
#Plot 2
crosstab_02 = pd.crosstab(bank_pd['education'], bank_pd['y1'])
crosstab_norm = crosstab_02.div(crosstab_02.sum(1), axis = 0)
crosstab_norm.plot(kind='bar', stacked = False)
```

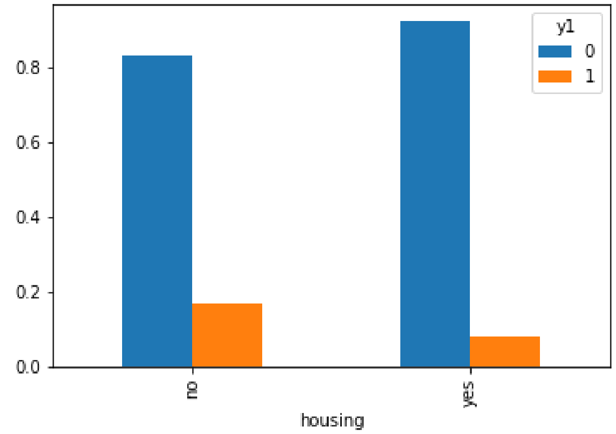
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2241bd8090>
```



Saved successfully!

```
#Plot 3
crosstab_03 = pd.crosstab(bank_pd['housing'], bank_pd['y1'])
crosstab_norm = crosstab_03.div(crosstab_03.sum(1), axis = 0)
crosstab_norm.plot(kind='bar', stacked = False)
crosstab_norm.plot(kind='bar', stacked=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f22415544d0>
```



```
#Plot 4
crosstab_01 = pd.crosstab(bank_pd['age'], bank_pd['marital'])
crosstab_01['Total'] = crosstab_01.sum(axis=1)  #JJG - Changed t1 to crosstab_01
crosstab_01.loc['Total'] = crosstab_01.sum()    #JJG - Changed t1 to crosstab_01
crosstab_01
```

marital	divorced	married	single	Total
age				
18	0	0	12	12
19	0	0	35	35
20	0	3	47	50
21	0	5	74	79
22	0	9	120	129
...	...	...	...	...
92	0	2	0	2
93	0	2	0	2
94	1	0	0	1
95	1	1	0	2
Total	5207	27211	12790	45208

```
#Contingency Table 1
crosstab_01 = pd.crosstab(bank_pd['age'], bank_pd['marital'])
crosstab_01['Total'] = crosstab_01.sum(axis=1)  #JJG - Changed t1 to crosstab_01
crosstab_01.loc['Total'] = crosstab_01.sum()    #JJG - Changed t1 to crosstab_01
crosstab_01
```

Saved successfully! ×

```
marital  divorced  married  single  Total
    age
     18         0        0      12     12
     19         0        0      35     35
     20         0        3      47     50
     21         0        5      74     79
     22         0        9     120    129
```

```
#Contingency Table 2
```

```
crosstab_02 = pd.crosstab(bank_pd['Education'], bank_pd['y1'])

crosstab_02 = pd.crosstab(bank_pd['y1'], bank_pd['Education'])
round(crosstab_02.div(crosstab_02.sum(0), axis = 1)*100, 1)
```

```
Education     0     1     2     3
       y1
        0  91.4  89.4  85.0  86.4
        1   8.6  10.6  15.0  13.6
```

```
# Creating a dataframe
bank_pd = bank_p.to_pandas()

#Convert variables to factor variables for modeling
bank_pd['job'],_=pd.factorize(bank_pd['job'], sort=True)
bank_pd['marital'],_=pd.factorize(bank_pd['marital'], sort=True)
bank_pd['education'],_=pd.factorize(bank_pd['education'], sort=True)
bank_pd['default'],_=pd.factorize(bank_pd['default'], sort=True)
bank_pd['housing'],_=pd.factorize(bank_pd['housing'], sort=True)
bank_pd['loan'],_=pd.factorize(bank_pd['loan'], sort=True)
bank_pd['poutcome'],_=pd.factorize(bank_pd['poutcome'], sort=True)
bank_pd['contact'],_=pd.factorize(bank_pd['contact'], sort=True)
bank_pd['month'],_=pd.factorize(bank_pd['month'], sort=True)
bank_pd['Age_CAT'],_=pd.factorize(bank_pd['age'], sort=True)
bank_pd['Education'],_=pd.factorize(bank_pd['education'], sort=True)

# Creating a dataframe with X%
# values of original dataframe
bank_pd_train = bank_pd.sample(frac = 0.8)  #Put the percentage split here

# Creating dataframe with
# rest of the X% values
bank_pd_tst = bank_pd.drop(bank_pd_train.index)

bank_pd_tst.describe()
```

Saved successfully! ×

	age	job	marital	education	default	balance	housing	loa
count	9042.000000	9042.000000	9042.000000	9042.000000	9042.000000	9042.000000	9042.000000	9042.00000
mean	40.949016	4.404667	1.168104	1.232581	0.017363	1348.965052	0.559500	0.16279
std	10.536223	3.290196	0.604931	0.742464	0.130629	3118.705150	0.496475	0.36920
25%	33.000000	1.000000	1.000000	1.000000	0.000000	71.000000	0.000000	0.00000

Now that the EDA is done for this round, I want you to prepare a decision tree model.

1. Explain why you need to split your data into training and test sets?

###

> #> Splitting a dataset helps to eliminate bias in classification models. In analysis we want to have an unbias model. By splitting the data in training set and test set is part of the preparation in modeling. In classification models, an analyst can apply methods and algorithms to test the data within a model. It is an important gauge and measure to how an algorithm is performing. sklearn is used in machine learning and predictive analytics tools in part of the splitting process.

2. What percentage split are you going to use? ###

   #> Typical split is round 25 / 75 or 20 / 80 according the textbook and standards. For this I used 20 / 80 split

3. How does that percentage affect your model?

   #> The analyst must ensure that the training set represents the entire dataset (sampling), so it is important to confirm there is a selection of balanced evenly-distributed data covering all of categories properly and accurately depending on the question the model in set to help answer.

4. If you select a higher or lower percentage for your training / test split what affect will it have on your validation of the model.

   #> If an analysts selects to low a percentage of data for the training there is a risk of not having enough data to test and validate the model. If the analyst has too high a percentage of test data the risk may result in not having enough model not being able to perform well to provide accurate outcome for the analysis.

5. Why do you need to validate your model?

   #> It is very important to validate the model. The modeling process must be able to provide [illegible] holders outcomes that they depend on. There are several ways to validate model and its outcomes to provide a more accurate outcome. It is super important to evaluate the model and its outcomes by establishing a baseline model of accuracy and

Saved successfully! ×

comparing that to other regression models and classification algorithms. These methods help visualize and structure the data output for making predictions.

This is the Evaluation of the Model

```
train=bank_pd_train.loc[:,['Age_CAT', 'Education', 'marital']]  #I added my predictors here!
tar=bank_pd_train.loc[:,['y1']]
tst=bank_pd_tst.loc[:,['Age_CAT', 'Education', 'marital']] # I added my predictors here!
```

```
clf = tree.DecisionTreeClassifier() # defining decision tree classifier
clf=clf.fit(train,tar)
```

```
prediction = clf.predict(tst) #  Run Predictions
```

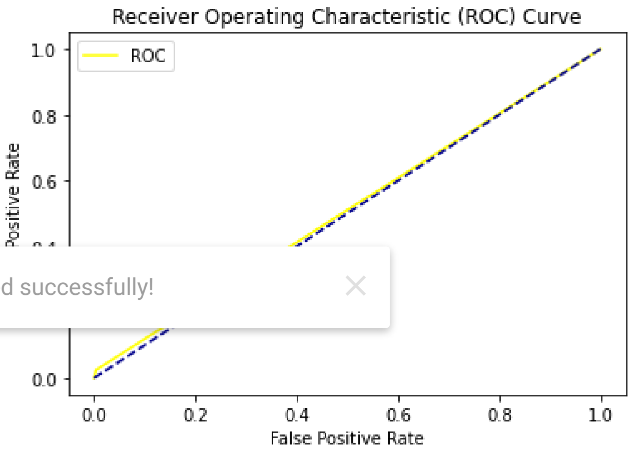
```
from sklearn.metrics import accuracy_score  #  accuracy is good at 88%.  But it doesn't tell how model will perform
from sklearn import metrics

y=bank_pd_tst.loc[:,['y1']]

fpr, tpr, thresholds = metrics.roc_curve(y, prediction)
accuracy_score(y, prediction)
```

```
0.8814421588144216
```

```
metrics.auc(fpr, tpr)
```

```
0.5103422601326248
```

```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='yellow', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

plot_roc_curve(fpr, tpr)
```



Saved successfully!

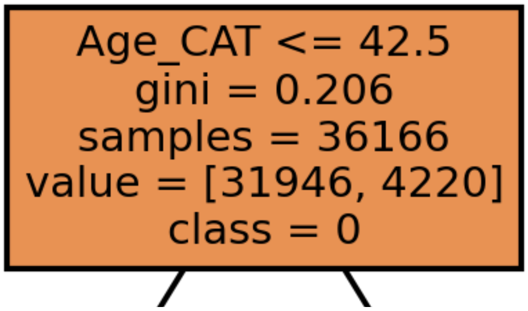
Adjust some tuning paramters. Fill in the Xs with values. The values next to each one are the defaults

```
clf = tree.DecisionTreeClassifier(
    max_depth= 25  #Default=None
    , min_samples_leaf = 10 #Default=1
    , min_samples_split = 15 #Default=2
    , min_impurity_decrease = 0.001)  #Default=0
clf=clf.fit(train,tar)
prediction = clf.predict(tst) #  Run Predictions
```

```
#Display your tree
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz

X_names = ['Age_CAT', 'Education', 'marital', 'housing']
y_names = ['0','1']

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(clf,feature_names = X_names, class_names=y_names, filled = True)

```

Saved successfully! ×

```
[Text(558.0, 755.0, 'Age_CAT <= 42.5\ngini = 0.206\nsamples = 36166\nvalue = [31946, 4220]\nclass = 0'),
 Text(372.0, 453.0, 'marital <= 1.5\ngini = 0.194\nsamples = 35233\nvalue = [31398, 3835]\nclass = 0'),
 Text(186.0, 151.0, 'gini = 0.168\nsamples = 25010\nvalue = [22695, 2315]\nclass = 0'),
 Text(558.0, 151.0, 'gini = 0.253\nsamples = 10223\nvalue = [8703, 1520]\nclass = 0'),
 Text(744.0, 453.0, 'gini = 0.485\nsamples = 933\nvalue = [548, 385]\nclass = 0')]
```



1. Describe what your tree looks like after you adjusted tuning parameters?

#> The tree provides insight to the variables. The top, is the root node and the rest descend through the tree based on each decision node within the tree. The top node, Age category (Age_CAT) tells us less that 42 is mean age in these records. Additionally, the records in the Age Category show a higher greater number of marital category in the married status, than those who are not married. Of the records in the range of Age 42.5 (mean) and show up as married, the class = 0 is all categories. So, it is important to find variables that show up with a class = 1. Class is report as 0 or 1, a binary number that typicially mean yes or no or true or false. None of the variables indicated a TRUE or Yes as a valid variable with the other variables provided.

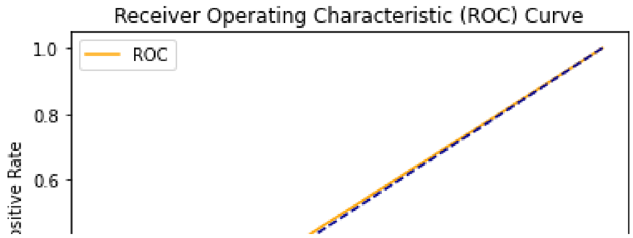
2. Generally, why are tuning parameters important in models?

# > Also, important to note education was add to this analysis. It was not significant enough to be reported. So, as an analyst it is important to go in the process and loop back to find higher predictor variable.s

```
y=bank_pd_tst.loc[:,['y1']]

fpr, tpr, thresholds = metrics.roc_curve(y, prediction)
accuracy_score(y, prediction)
```

```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

plot_roc_curve(fpr, tpr)
```

Saved successfully!



```
#D. Evaluate

```

1. Please describe AUC.

   #> In Machine Learning it is important to understand the performance of the classification modeling you are planning to use. It is recommended that analysts use the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve to analyze the model performance with use of various variables before going to test. These tools provide the important visualization of the model's performance to make appropriate adjustments to the dataset and approached to the analysis.

It is a key evaluation metrics for validation of a classification model's performance. #

2. What is it used to measure?

   #> In Machine Learning it is important to understand the performance of the classification modeling. So it is recommend that analyst use the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve.

#> It is a key evaluation metrics for validation of a classification model's performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics)

#> The AUC ROC provides in this analysis various threshold settings for gauging model performance. The ROC is a probability curve, while the AUC allows the analyst to visual the degree or measure of separability.

#> These tools will demonstrate what the model is capable of distinguishing between classes of the variables.

#> Note: The higher the amount in the AUC, the better performance of the model can be expected, especiall with predicting binary amounts.

#> An example from the website, Towards Data indicates that higher AUC ROC score of the records of patients with the disease and no disease.

While the ROC curve plotting of the True Positive Rate(TPR) against the False Positive Rate FPR is where the TPR is on the y-axis and FPR is on the x-axis.

#> https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

#

Saved successfully! ×

#> My model did not present a solid performance. The initial analysis of the variables showed 88% accuracy, while the AUC was only 51%. That's not a reliable model. Yet, with more iterations of the variables, this analysts believes a prediction can be made to support this bank's marketing campaigns.

4. Do you think your model has a reliable prediction? Why or Why Not?

   #> At this writing, the model for analyzing this data for the bank's marketing campaign is not yet ready for release. Yet, with more time and education this analyst will be able to use the tools for modeling in Machine Learning to provide a better performing model.

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import mutual_info_classif

feat_importance = clf.tree_.compute_feature_importances(normalize=True)
feat_importance  #Outputs in order of the predictors you listed
```

```
array([0.78588389, 0.        , 0.21411611])
```

1. Where any of the features you developed important? Please describe.

>\# This is important to understand the process, the details and how to management expectations. The variables initially chosen for the assignment did not perform well, in the gini and AUC / ROC tests of the variables. Like any scientist knows, the elimination of the variables is important to creating a reliable model. These variables together are not showing high predictability at 51%, but there are more tests to apply. The loop part of the process is surely accurate to this models performance and reliability.

#>

✓ 0s completed at 10:23 PM ● ×

Saved successfully! ×