WGU D209 TASK 2 REV 1 - MATTINSON

Classification Tree - Telecom Churn Data

Mike Mattinson

Master of Science, Data Analytics, WGU.edu

Task 2 - Decision Classification Tree

November 15, 2021

Configure Notebook. Import and configure packages. All of the code for importing and configuring is contained in a imports .PY file. Also, there is a second helpers .PY file to define a few functions used throughout this notebook.

In [3]:

```python
# import and configure packages
from imports import *
%matplotlib inline
warnings.filterwarnings('ignore')
```

In [4]:

```python
from helpers import *
```

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

A1. Propose one question relevant to a real-world organizational situation that you will answer using one of the following prediction methods: (a) decision trees, (b) random forests or (c) advanced regression (i.e., lasso or ridge regression)

Primary Question. A telecomm company is concerned about customer churn. It has collected customer data on 10,000 customers and the analysis will attempt to determine if the target value 'Churn', which is a yes-nop categorical variable, if 'Churn' can be predicted accurately using only a minimum number of predictor variables. The prediction analysis will use the DecisionClassificationTree to predict the target value.

A2. Define one goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

Primary Goal. Primary goals will look at all predictor variables and then reduce the number of predictor variables while maintaining a reasonable level of accuracy. The company provided data will be split into training and testing data, 70% training and 30% testing. I will using the confusion matrix to show the total number of correct and incorrect predictions in the training and testing datasets.

Part II: Method Justification

B. Explain the reasons for your chosen predictionf method from part A1 by doing the following:

B1. Explain how the prediction method you chose analyzes the selected data set. Include expected outcomes.

Explain Method. I choose to use the DecisionClassificationTree to model the data in order to predict the target value using a minimum number of features. Once the model is trained using 70% of the data, then I will calculate the accuracy of the model. If the model is doing well, anyone will be able to enter the model with unclassified data and the model will accurately predict the target value.

B2. Summarize one assumption of the chosen predition method.

One Assumption. The model is created using the training data and the size of the training data is sufficient large to make the best decision points throughout the tree.

B3. List the packages or libraries you have chosen for Python or R, and justify how each item on the list supports the analysis.

All of the Python packages required for this analysis were loaded at the very top of th notebook. The packages and version numbers are presented. Besides the normal Python packages (numpy, scipy, matplotlib, pandas, etc.) the primary package required to create and view the prediction model comes from sklearn. Also, I use two (2) .PY files instead of including all that code in this notebook, I reuse it in other notebooks.

Imports.py has all of the required packages and Helpers.py has many helpful functions that allow me to standardize my tables, figures and other parts of the notebook. Both of these .PY files will be included with the notebook for reference.

Part III: Data Preparation

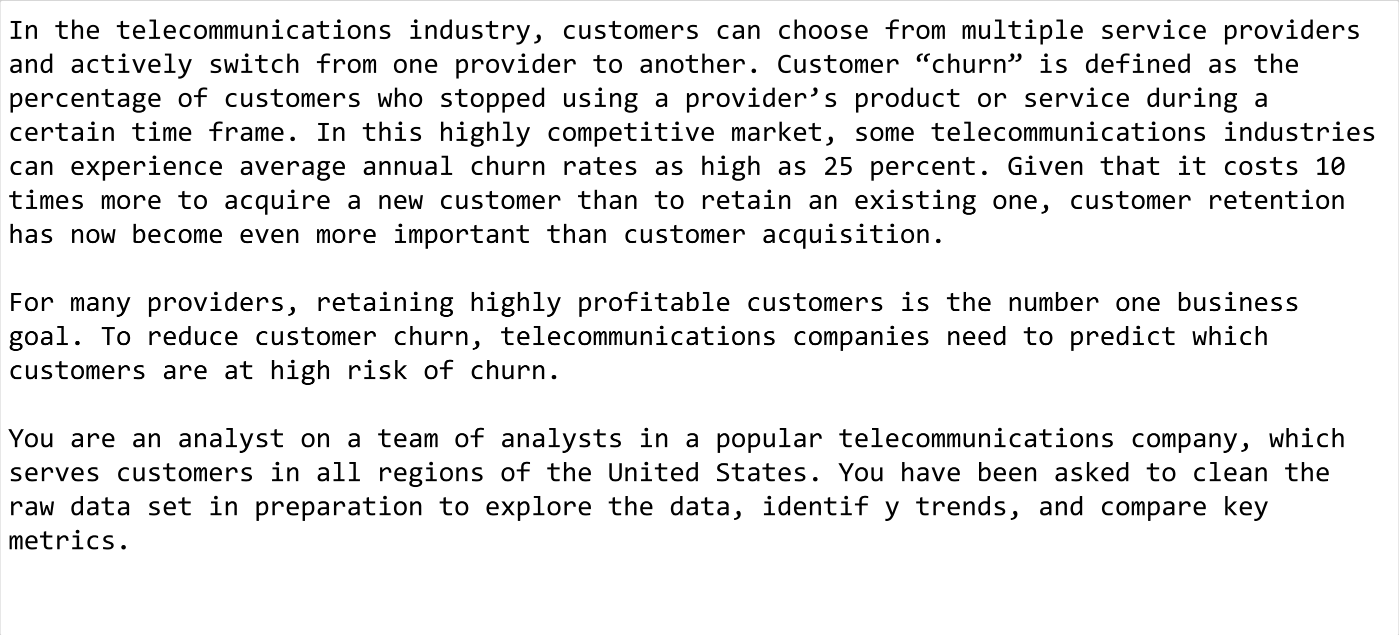
C. Perform data preparation for the chosen data set by doing the following:

C1. Describe one data preprocessing goal relevant to the prediction method from part A1.

Goal. The data provided by the company includes both numeric and categoricald data, the first primary goal will be to convert the categorical data into data that the model can understand. I plan to convert the categorical data using the sklearn.preprocessing LabelEncoder. This will happen in section 'C3' below.

C2. Identify the initial data set variables that you will use to perform the analysis for the classification question from part A1, and classify each variable as continuous or categorical.

TABLE 3-1.DESCRIPTION OF DATA IN TELECO CHURN DATASET



In the telecommunications industry, customers can choose from multiple service providers and actively switch from one provider to another. Customer “churn” is defined as the percentage of customers who stopped using a provider’s product or service during a certain time frame. In this highly competitive market, some telecommunications industries can experience average annual churn rates as high as 25 percent. Given that it costs 10 times more to acquire a new customer than to retain an existing one, customer retention has now become even more important than customer acquisition.

For many providers, retaining highly profitable customers is the number one business goal. To reduce customer churn, telecommunications companies need to predict which customers are at high risk of churn.

You are an analyst on a team of analysts in a popular telecommunications company, which serves customers in all regions of the United States. You have been asked to clean the raw data set in preparation to explore the data, identif y trends, and compare key metrics.

```
In [5]: # load data from .csv
        raw = pd.read_csv('data/churn_clean.csv')
        raw = raw.drop(columns=['CaseOrder','UID',
                'Customer_id','Interaction',
                'City', 'State','County','Zip',
                'Lat','Lng','Job','TimeZone'])
```

TABLE 3-2.RAW DATA

Initial state of dataset before any manipulations.

In [6]:

```
saveTable(data=raw, title='RAW', sect='C2',
    course='D209', task='Task2', caption='3 2')
```

	0	1	2	3
Population	38	10446	3735	13863
Area	Urban	Urban	Urban	Suburban
Children	0	1	4	1
Age	68	27	50	48
Income	28561.99	21704.77	9609.57	18925.23
Marital	Widowed	Married	Widowed	Married
Gender	Male	Female	Female	Male
Churn	No	Yes	No	No
Outage_sec_perweek	7.978	11.699	10.753	14.914
Email	10	12	9	15
Contacts	0	0	0	2
Yearly_equip_failure	1	1	1	0
Techie	No	Yes	Yes	Yes
Contract	One year	Month-to-month	Two Year	Two Year
Port_modem	Yes	No	Yes	No
Tablet	Yes	Yes	No	No
InternetService	Fiber Optic	Fiber Optic	DSL	DSL
Phone	Yes	Yes	Yes	Yes
Multiple	No	Yes	Yes	No
OnlineSecurity	Yes	Yes	No	Yes
OnlineBackup	Yes	No	No	No
DeviceProtection	No	No	No	No
TechSupport	No	No	No	No
StreamingTV	No	Yes	No	Yes
StreamingMovies	Yes	Yes	Yes	No
PaperlessBilling	Yes	Yes	Yes	Yes
PaymentMethod	Credit Card (automatic)	Bank Transfer(automatic)	Credit Card (automatic)	Mailed Check
Tenure	6.796	1.157	15.754	17.087
MonthlyCharge	172.456	242.633	159.948	119.957
Bandwidth_GB_Year	904.536	800.983	2054.707	2164.579
Item1	5	3	4	4
Item2	5	4	4	4
Item3	5	3	2	4
Item4	3	3	4	2
Item5	4	4	4	5

	0	1	2	3
Item6	4	3	3	4
Item7	3	4	3	3
Item8	4	4	3	3

```
shape: (10000, 38)
Table saved to: TABLES/D209_TASK2_C2_TAB_3_2_RAW.CSV
```

C3. Explain each of the steps used to prepare the data for the analysis. Identify the code segment for each step.

Step 1.

The company data has already been cleaned and prepared. The only step required prior to creating the prediction model is to encode the categorical data. So, the raw data will be copied and then the clean data will encode all of the categorical data. The clean data will have all of the original variables but for example, all of the categorical data will have integer numbers representing the raw data, Area = 2 indicates Area=Rural in the raw data, etc.

In [7]:

```
# start with a copy of raw data
clean = raw.copy()
```

In [8]:

```
# define target and features
target = 'Churn'
X = clean.drop(columns=[target])
y = clean[target]
print(type(X))
print(type(y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [9]:

```
# define categorical features
cat_features = X.select_dtypes(include=['object']).columns
print(cat_features)
```

```
Index(['Area', 'Marital', 'Gender', 'Techie', 'Contract', 'Port_modem',
       'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'PaymentMethod'],
      dtype='object')
```

In [10]:

```
# define numerical features
num_features = X.select_dtypes(exclude=['object']).columns
print(num_features)
```

```
Index(['Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
       'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
       'Item6', 'Item7', 'Item8'],
      dtype='object')
```

In [11]:

```
from sklearn.preprocessing import LabelEncoder
categorical_feature_names = []
label_encoders = {}
for categorical in cat_features:
    label_encoders[categorical] = LabelEncoder()
    clean[categorical] = label_encoders[categorical].fit_transform(clean[categorical])
    names = label_encoders[categorical].classes_.tolist()
    print('Label encoder %s - values: %s' % (categorical, names))
    if categorical == target:
        continue
    categorical_feature_names.extend([categorical + '_' + str(name) for name in names])
```

```
Label encoder Area - values: ['Rural', 'Suburban', 'Urban']
Label encoder Marital - values: ['Divorced', 'Married', 'Never Married', 'Separated', 'Wi
dowed']
Label encoder Gender - values: ['Female', 'Male', 'Nonbinary']
Label encoder Techie - values: ['No', 'Yes']
Label encoder Contract - values: ['Month-to-month', 'One year', 'Two Year']
Label encoder Port_modem - values: ['No', 'Yes']
Label encoder Tablet - values: ['No', 'Yes']
Label encoder InternetService - values: ['DSL', 'Fiber Optic', 'None']
Label encoder Phone - values: ['No', 'Yes']
Label encoder Multiple - values: ['No', 'Yes']
Label encoder OnlineSecurity - values: ['No', 'Yes']
Label encoder OnlineBackup - values: ['No', 'Yes']
Label encoder DeviceProtection - values: ['No', 'Yes']
Label encoder TechSupport - values: ['No', 'Yes']
Label encoder StreamingTV - values: ['No', 'Yes']
Label encoder StreamingMovies - values: ['No', 'Yes']
Label encoder PaperlessBilling - values: ['No', 'Yes']
Label encoder PaymentMethod - values: ['Bank Transfer(automatic)', 'Credit Card (automati
c)', 'Electronic Check', 'Mailed Check']
```

In [12]:

```
print(categorical_feature_names)
print(label_encoders)
```

```
['Area_Rural', 'Area_Suburban', 'Area_Urban', 'Marital_Divorced', 'Marital_Married', 'Mar
ital_Never Married', 'Marital_Separated', 'Marital_Widowed', 'Gender_Female', 'Gender_Mal
e', 'Gender_Nonbinary', 'Techie_No', 'Techie_Yes', 'Contract_Month-to-month', 'Contract_O
ne year', 'Contract_Two Year', 'Port_modem_No', 'Port_modem_Yes', 'Tablet_No', 'Tablet_Ye
s', 'InternetService_DSL', 'InternetService_Fiber Optic', 'InternetService_None', 'Phone_
No', 'Phone_Yes', 'Multiple_No', 'Multiple_Yes', 'OnlineSecurity_No', 'OnlineSecurity_Ye
s', 'OnlineBackup_No', 'OnlineBackup_Yes', 'DeviceProtection_No', 'DeviceProtection_Yes',
'TechSupport_No', 'TechSupport_Yes', 'StreamingTV_No', 'StreamingTV_Yes', 'StreamingMovie
s_No', 'StreamingMovies_Yes', 'PaperlessBilling_No', 'PaperlessBilling_Yes', 'PaymentMeth
od_Bank Transfer(automatic)', 'PaymentMethod_Credit Card (automatic)', 'PaymentMethod_Ele
ctronic Check', 'PaymentMethod_Mailed Check']
{'Area': LabelEncoder(), 'Marital': LabelEncoder(), 'Gender': LabelEncoder(), 'Techie': L
abelEncoder(), 'Contract': LabelEncoder(), 'Port_modem': LabelEncoder(), 'Tablet': LabelE
ncoder(), 'InternetService': LabelEncoder(), 'Phone': LabelEncoder(), 'Multiple': LabelEn
coder(), 'OnlineSecurity': LabelEncoder(), 'OnlineBackup': LabelEncoder(), 'DeviceProtect
ion': LabelEncoder(), 'TechSupport': LabelEncoder(), 'StreamingTV': LabelEncoder(), 'Stre
amingMovies': LabelEncoder(), 'PaperlessBilling': LabelEncoder(), 'PaymentMethod': LabelE
ncoder()}
```

C4. Provide Clean Data

Provide a copy of the cleaned data set.

TABLE 3-3. CLEAN DATA

In [13]:
```python
#clean = raw.copy()
saveTable(data=clean, title='CLEAN', sect='C4',
    course='D209', caption='3 3', task='Task2')
```

	0	1	2	3
Population	38	10446	3735	13863
Area	2	2	2	1
Children	0	1	4	1
Age	68	27	50	48
Income	28561.99	21704.77	9609.57	18925.23
Marital	4	1	4	1
Gender	1	0	0	1
Churn	No	Yes	No	No
Outage_sec_perweek	7.978	11.699	10.753	14.914
Email	10	12	9	15
Contacts	0	0	0	2
Yearly_equip_failure	1	1	1	0
Techie	0	1	1	1
Contract	1	0	2	2
Port_modem	1	0	1	0
Tablet	1	1	0	0
InternetService	1	1	0	0
Phone	1	1	1	1
Multiple	0	1	1	0
OnlineSecurity	1	1	0	1
OnlineBackup	1	0	0	0
DeviceProtection	0	0	0	0
TechSupport	0	0	0	0
StreamingTV	0	1	0	1
StreamingMovies	1	1	1	0
PaperlessBilling	1	1	1	1
PaymentMethod	1	0	1	3
Tenure	6.796	1.157	15.754	17.087
MonthlyCharge	172.456	242.633	159.948	119.957
Bandwidth_GB_Year	904.536	800.983	2054.707	2164.579
Item1	5	3	4	4
Item2	5	4	4	4
Item3	5	3	2	4
Item4	3	3	4	2

	0	1	2	3
Item5	4	4	4	5
Item6	4	3	3	4
Item7	3	4	3	3
Item8	4	4	3	3

```
shape: (10000, 38)
Table saved to: TABLES/D209_TASK2_C4_TAB_3_3_CLEAN.CSV
```

Part IV Analysis

D. Perform the data analysis and report on the results by doing the following:

D1. Split the data into training and test data sets and provide the file(s).

In [14]:

```
# define primary feature and target data
target= 'Churn' # target data
X = clean.loc[:, clean.columns != target]
y = clean.loc[:, clean.columns == target]
```

In [15]:

```
# train test split raw data
tts = train_test_split(X, y, test_size=0.3, random_state=13)
(X_train, X_test, y_train, y_test)=tts
print('X_train: {}'.format(X_train.shape))
print('y_train: {}'.format(y_train.shape))
print('X_test: {}'.format(X_test.shape))
print('y_test: {}'.format(y_test.shape))
```

```
X_train: (7000, 37)
y_train: (7000, 1)
X_test: (3000, 37)
y_test: (3000, 1)
```

TABLE 4-1.TRAINING DATA

```
In [16]: saveTable(data=X_train.merge(y_train,
                 left_index=True, right_index=True),
             title='TRAIN', sect='D1',
             course='D209', caption='4 1', task='Task2')
```

	4847	9992	4621	5774
Population	48905	4261	10218	54601
Area	0	1	2	2
Children	0	1	1	4
Age	63	18	27	41
Income	27506.01	35876.21	43148.68	35600.06
Marital	1	0	1	3
Gender	1	1	0	0
Outage_sec_perweek	10.455	9.213	10.526	11.035
Email	14	15	9	17
Contacts	1	2	0	0
Yearly_equip_failure	1	0	0	0
Techie	0	1	1	0
Contract	1	2	2	0
Port_modem	0	0	0	0
Tablet	0	1	1	0
InternetService	2	1	1	0
Phone	1	1	1	1
Multiple	0	0	0	0
OnlineSecurity	0	0	0	0
OnlineBackup	0	0	0	1
DeviceProtection	0	0	0	1
TechSupport	1	0	1	1
StreamingTV	0	1	0	0
StreamingMovies	0	0	0	0
PaperlessBilling	1	1	1	1
PaymentMethod	2	2	2	3
Tenure	9.525	56.472	2.612	58.787
MonthlyCharge	92.488	137.439	124.964	139.983
Bandwidth_GB_Year	820.042	5001.371	320.107	5562.052
Item1	2	3	5	4
Item2	3	2	5	4
Item3	4	4	4	3
Item4	4	3	1	4
Item5	3	4	4	3

	4847	9992	4621	5774
Item6	2	3	5	3
Item7	3	4	4	3
Item8	4	3	3	3
Churn	No	No	No	No

```
shape: (7000, 38)
Table saved to: TABLES/D209_TASK2_D1_TAB_4_1_TRAIN.CSV
```

TABLE 4-2.TEST DATA

```
In [17]: saveTable(data=X_test.merge(y_test,
                 left_index=True, right_index=True),
             title='TEST', sect='D1',
             course='D209', caption='4 2', task='Task2')
```

	5952	1783	4811	145
Population	6399	32000	18951	5157
Area	2	1	2	0
Children	4	4	1	8
Age	59	83	68	75
Income	36343.28	40031.03	39053.35	36342.31
Marital	2	4	2	0
Gender	1	0	0	1
Outage_sec_perweek	8.923	14.75	11.206	15.409
Email	13	10	10	15
Contacts	0	2	7	2
Yearly_equip_failure	0	0	0	0
Techie	1	0	0	0
Contract	2	1	2	0
Port_modem	0	0	1	0
Tablet	0	1	1	0
InternetService	2	2	1	1
Phone	1	1	1	1
Multiple	0	0	0	1
OnlineSecurity	0	1	0	0
OnlineBackup	1	0	1	1
DeviceProtection	0	0	0	1
TechSupport	1	0	0	1
StreamingTV	0	0	1	1
StreamingMovies	0	1	1	0
PaperlessBilling	1	1	1	0
PaymentMethod	3	2	2	0
Tenure	56.633	2.851	5.664	2.733
MonthlyCharge	114.984	117.483	230.105	217.473
Bandwidth_GB_Year	4910.179	479.016	982.717	882.116
Item1	3	2	4	5
Item2	4	3	4	4
Item3	5	3	3	4
Item4	2	3	3	4
Item5	5	3	5	3

	5952	1783	4811	145
Item6	2	2	4	5
Item7	2	4	4	5
Item8	2	4	4	3
Churn	No	No	Yes	Yes

```
shape: (3000, 38)
Table saved to: TABLES/D209_TASK2_D1_TAB_4_2_TEST.CSV
```

D2. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

The data for the model is the training data created above in section 'D1'. I am selected max_depth=2 for the model in order to create a simple model. I want to be able to create a 2D decision boundary plot using a scatter plot.

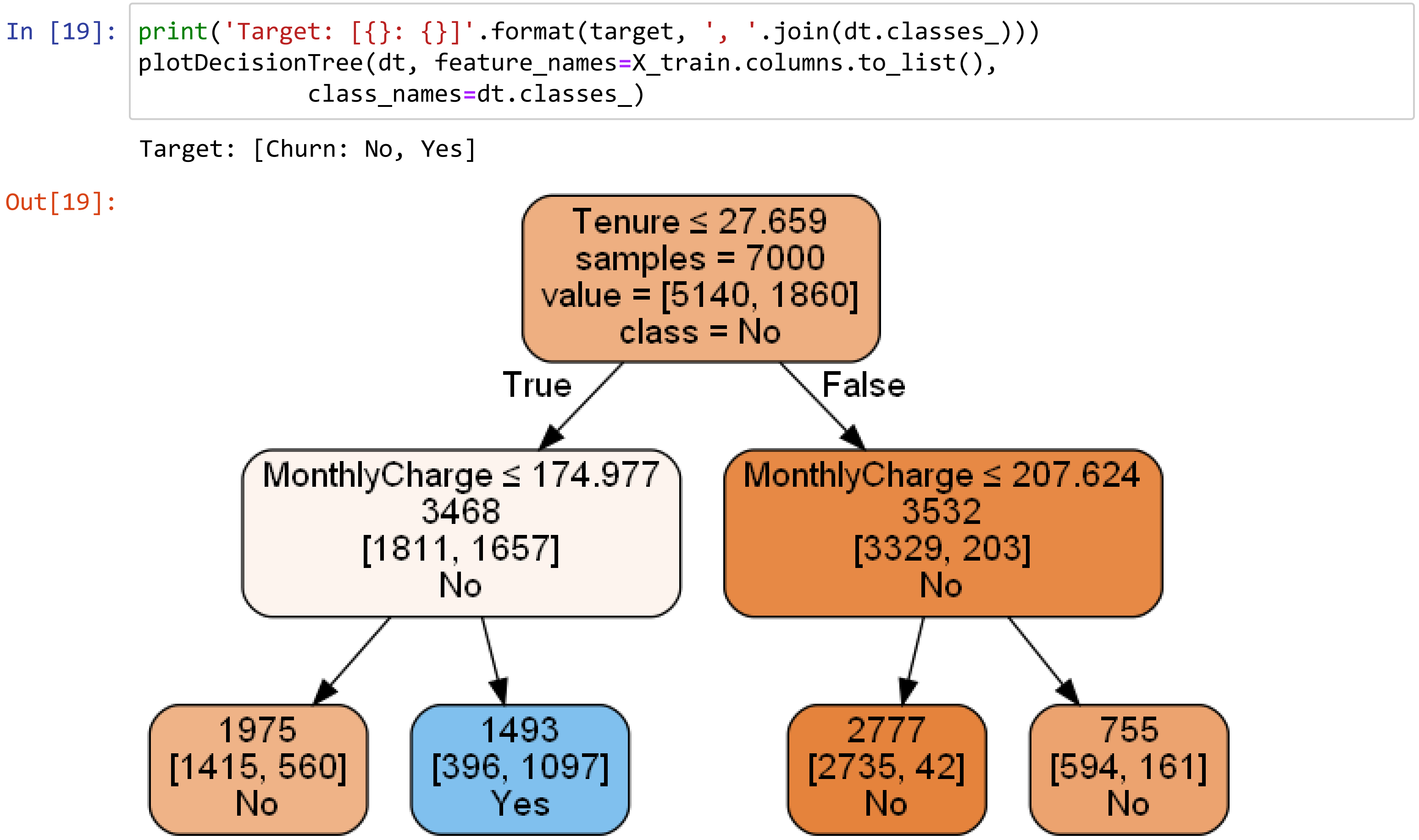
In [18]:

```
# create model
dt = DecisionTreeClassifier(max_depth=2, random_state=13)
dt.fit(X_train, y_train)
```

Out[18]:

```
DecisionTreeClassifier(max_depth=2, random_state=13)
```

FIGURE 4-1.CLASSIFICATION TREE (MAX_DEPTH=2)

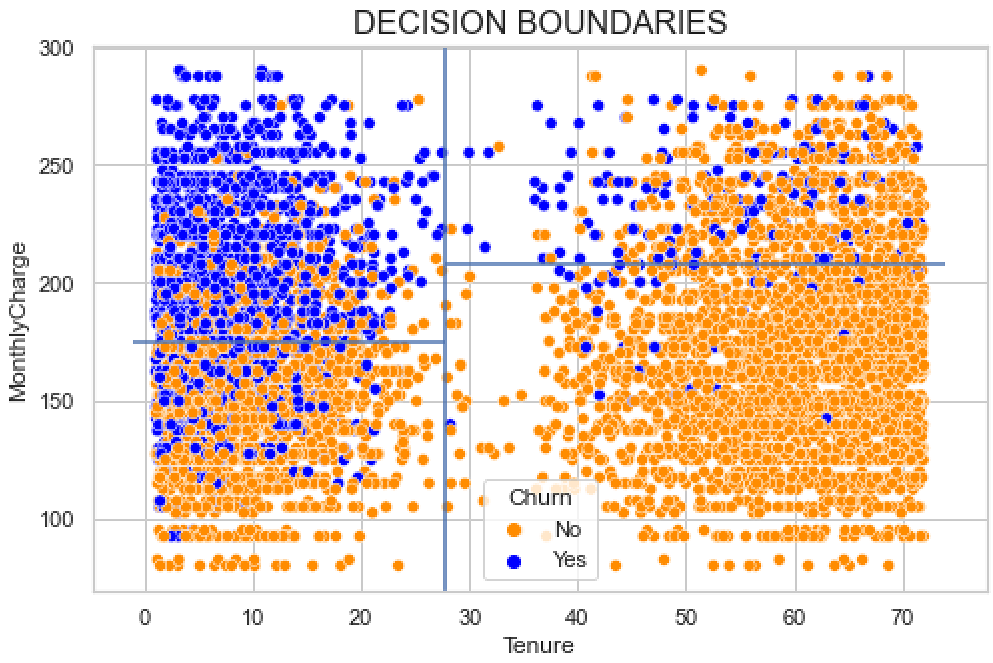


```
In [19]: print('Target: [{}: {}]'.format(target, ', '.join(dt.classes_)))
         plotDecisionTree(dt, feature_names=X_train.columns.to_list(),
                          class_names=dt.classes_)

         Target: [Churn: No, Yes]

Out[19]:
```

FIGURE 4-2.DECISION BOUNDARIES

```
In [20]: # plot decision boundaries
         fig, ax = plt.subplots()
         fig.set_size_inches(8, 5)

         # define plot variables
         x = 'Tenure'
         y='MonthlyCharge'
         title = 'Decision Boundaries'

         sns.scatterplot(x=x, y=y,
             palette=['darkorange','blue'], hue=target,
             data=y_train.merge(X_train, left_index=True, right_index=True))
         ax.axvline(x=27.659)
         ax.hlines(y=174.977, xmin=-1, xmax=27.659) # horizontal line segment
         ax.hlines(y=207.624, xmin=27.659, xmax=74) # horizontal line segment

         ax.set_title(title.upper(), fontsize=16)

         # create filename and save
         f=getFilename(title=title, caption='4 2',
                       course='D209', task='Task2', sect='D2',
                       ftype='PNG', subfolder='figures')
         plt.gcf().text(0, -.05, f, fontsize=14)
         fig.savefig(f, dpi=150, bbox_inches='tight')
         print('Figure saved to: {}'.format(f)) # feedback to notebook
```

```
Figure saved to: FIGURES/D209_TASK2_D2_FIG_4_2_DECISION_BOUNDARIES.PNG
```



FIGURES/D209_TASK2_D2_FIG_4_2_DECISION_BOUNDARIES.PNG

The figure above shows the model from another slightly different vantage point. The scatter plot of the two key predictive features is broken down into four (4) section correlating to the four (4) terminal nodes of the decision tree. Each of the vertical and horizontal line segments are created using the numbers shown in the decision tree decision nodes.

D3. Provide the code used to perform the classification analysis from part D2.

Code. All code and output is contained within this Jupyter notebook. The notebook file is called **D209_2_x.ipynb** and the associated PDF version is called **D209_2_x - Jupyter Notebook.pdf**.

Part V: Data Summary and Implications

E1. Explain the accuracy and the mean squared error (MSE) of your prediction model.

For the classification tree, the measurement of accuracy is found within the confusiion matrix. Below are the confusion matrices for training data (showing 83.44% accuracy) and the test data (showing 83.57% accuracy). The accuracy is calculated by adding the number of True Positives and True Negatives on the diagonal, then divide that number by the total number of records.

```
In [ ]: # training summary
        classificationSummary(y_train, dt.predict(X_train))
```

```
In [ ]: # test summary
        classificationSummary(y_test, dt.predict(X_test))
```

E2. Discuss the results and implications of your prediction analysis.

The final prediction model uses only two (2) of the original features to predict the target class, and it does it correctly 83% of the time, pretty accurate for such a simple model. The simple model and graphical tree will be easy to explain to those individuals who will be using this prediction model.

E3. Discuss one limitation of your data analysis.

Limitations. One limitation of the predictive analysis is that it correctly predicts target class only 83% of the time, that is, there is the 17% chance that the predicted class is incorrect.

E4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

Recommendations. Recommend that company include policy to include the churn prediction for all customers at intervals during the lifespan of the customer with the company, for example, during the first month, then again a 1-3 monthly intervals. Also, the prediction can be used anytime the customer's MonthlyCharge changes. Lastly, the company can have a 'Customer Support' office reachout to customer's with Churned predicted, possibly lowering their current monthly charge by some calculated amount.

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

Video. Panapto video was created and is located at: https://wgu.edu (https://wgu.edu)

G. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

Configure Scrollbars. Disable scrollbars in notebook.

```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

Disable Auto Scroll. Disable automatically scroll to bottom.

```
%%javascript
require("notebook/js/notebook").Notebook.prototype.scroll_to_bottom = function () {}
```

Toggle Notebook Warnings. Use the following code to toggle warning

messages in the notebook. Another piece of code courtesy of stackoverflow (2021).
https://stackoverflow.com/questions/9031783/hide-all-warnings-in-ipython
(https://stackoverflow.com/questions/9031783/hide-all-warnings-in-ipython)

```
from IPython.display import HTML
HTML('''<script>
code_show_err=false;
function code_toggle_err() {
 if (code_show_err){
 $('div.output_stderr').hide();
 } else {
 $('div.output_stderr').show();
 }
 code_show_err = !code_show_err
}
$( document ).ready(code_toggle_err);
</script>
To toggle on/off output_stderr, click <a
href="javascript:code_toggle_err()">here</a>.''')
```

Terminal List Files. List all of the files from the current working directory.

Ref: (1) Fessel, K. (2021). How to save a matplotlib figure and fix text cutting off || Matplotlib Tips (https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel) Retrieved from https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

```
!ls
```

```
!du -h *.*
```

List Installed Packages. List of all installed PIP packages and the versions.

Ref: (1) https://pip.pypa.io/en/stable/cli/pip_list/

```
!pip list
```

Update Package. Update a specific package within notebook.

Ref: (1) https://stackoverflow.com/questions/54453219/why-can-i-see-pip-list-sklearn-but-not-in-jupyter-when-i-run-a-code

```
!python -m pip install -U scikit-learn
```

Merget Two Dataframes. Code to merge two dataframes.

Ref: (1) https://stackoverflow.com/questions/26265819/how-to-merge-a-series-and-dataframe

```
# merge X and y back together, for example
d = X.merge(y, left_index=True, right_index=True)
display(d.head())
```

List .index() Function. The .index() method returns the index of the specified element in the list.

Ref: (1) https://www.programiz.com/python-programming/methods/list/index

```
animals = ['cat', 'dog', 'rabbit', 'horse']
# get the index of 'dog'
index = animals.index('dog')
print(index)
```

Row Index Names in Pandas. Code to get rows/index names in a Pandas dataframe.

Ref: (1) https://www.geeksforgeeks.org/how-to-get-rows-index-names-in-pandas-dataframe/

```
# making data frame
data = cleanData

# calling head() method
# storing in new variable
data_top = data.head()

# iterating the columns
for row in data_top.index:
    print(row, end = " ")
```

Tutorial Python Subplots. Tutorial: Python Subplots

Ref: (1) https://www.kaggle.com/asimislam/tutorial-python-subplots

```
#  Categorical Data
heart_CAT = ['Churn']

#  Categorical Data
a = 2  # number of rows
b = 3  # number of columns
c = 1  # initialize plot counter

fig = plt.figure(figsize=(14,10))

for i in heart_CAT:
    plt.subplot(a, b, c)
    plt.title('{}, subplot: {}{}{}'.format(i, a, b, c))
    plt.xlabel(i)
    sns.countplot(x=i, data=cleanData, palette='hls')
    c = c + 1

plt.show()
```

PASS FIG TO CUSTOM PLOT FUNCTION. A great way to do this is to pass a figure object to your code and have your function add an axis then return the updated figure.

Ref: (1) https://stackoverflow.com/questions/43925337/matplotlib-returning-a-plot-object

```
def plot_hist_overlay(feature, fig, p, bins=8):

    # data
    df_yes = cleanData[cleanData.Churn==True][feature]
    df_no = cleanData[cleanData.Churn==False][feature]

    # plot stacked hist
    ax = f.add_subplot() # here is where you add the subplot to f
    plt.hist([df_yes,df_no], bins=bins, stacked=True)

    # add title
    plt.title(feature + ' grouped by target', size=16)

    # tick marks
    ax.set_xticks([])
    #ax.set_yticks([]) # use default

    # add axis labels
    plt.xlabel(feature)
    plt.ylabel('# Churn')

    # add legend
    ax.legend(['Churn - Yes','Churn - No'])

    return(f)

target = 'Churn'
features = ['MonthlyCharge','Tenure']
bins = 6
for idx,fea in enumerate(features):
    fig_size = (6,5)
    f = plt.figure(figsize=fig_size)
    f = plot_hist_overlay(fea, fig=f, p=idx+1, bins=bins)
    file = getFilename(fea, 'z1','fig 9 ' + str(idx+1)) # getFilename using helper
    plt.gcf().text(0.1, 0, file, fontsize=14)

    # data table
    b = pd.cut(cleanData[fea], bins=bins) # create bins (b) of numeric feature
    dt = pd.crosstab(cleanData[target], b)
    plt.gcf().text(0.1, -.4, dt.T.to_string(), fontsize=14)
    #print(dt.T)

    f.savefig(file, dpi=150, bbox_inches='tight')

#f = plot_hist_overlay('MonthlyCharge', fig=f, p=3)
#f = plot_hist_overlay('Tenure', fig=f, p=2)
```

Enabling Jupyter Notebook extensions.

Ref: (1) https://tljh.jupyter.org/en/latest/howto/admin/enable-extensions.html

```
pip install jupyter_contrib_nbextensions
jupyter contrib nbextension install --sys-prefix
jupyter nbextension enable scratchpad/main --sys-prefix
jupyter nbextension list
```

How to Use HTML to Open a Link in a New Tab.

Ref: (1)https://www.freecodecamp.org/news/how-to-use-html-to-open-link-in-new-tab/

Ref: (1)https://www.freecodecamp.org/news/how-to-use-html-to-open-link-in-new-tab/

```
<p>Check out <a href="https://www.freecodecamp.org/" target="_blank" rel="noopener
noreferrer">freeCodeCamp</a>.</p>
```

CSS Tutorial. This is a great resource for CSS code with many examples.
Ref: (1)https://www.w3schools.com/css/default.asp

HTML Tutorial. This is a great resource for HTML code with many examples.
Ref: (1)https://www.w3schools.com/html/default.asp

Inline Styles in HTML. Usually, CSS is written in a separate CSS file (with file extension .css) or in a 'style' tag inside of the 'head' tag, but there is a third place which is also valid. The third place you can write CSS is inside of an HTML tag, using the style attribute. When CSS is written using the style attribute, it’s called an “inline style”. In general, this is not considered a best practice. However, there are times when inline styles are the right (or only) choice.
Ref: (1) https://www.codecademy.com/articles/html-inline-styles

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Deitel, P. + (2020). Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud

Geron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems

Larose, C.+ (2019). Data Science Using Python and R

Rite, S. (2018). Demystifying ‘Confusion Matrix’ Confusion
https://towardsdatascience.com/demystifying-confusion-matrix-confusion-9e82201592fd

Robinson, S. (2021). K-Nearest Neighbors Algorithm in Python and Scikit-Learn
https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

Sharma, A. (2021). K-Nearest Neighbors (KNN) on Customer Churn Data https://medium.com/data-science-on-customer-churn-data/k-nearest-neighbors-knn-on-customer-churn-data-40e9b2bb9266

Shmueli, G. + (2020). Data Mining for Business Analytics: Concepts, Techniques, and Application in Python

Appendix A - Imports.py

```
# show python environment variables
import sys; import platform
print(sys.executable)
print('python version: {}'.format(platform.python_version()))
import os, sys
from IPython.display import Image

# PIL image
from PIL import Image as png

# import and configure pandas
import pandas as pd
pd.set_option('precision',3)
pd.set_option('max_columns',9)
pd.set_option('display.width', None)
print('pandas version: {}'.format(pd.__version__))

# import and configure scientific computing
import numpy as np
import scipy.stats as stats
import scipy
print('numpy version: {}'.format(np.__version__))
print('scipy version: {}'.format(scipy.__version__))

# import and configure sklearn
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import classification_report
from sklearn import metrics, tree
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import KFold, cross_val_score, train_test_split,
GridSearchCV
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, _tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.pipeline import FeatureUnion, Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.tree import export_graphviz as dt # decisiontree
from sklearn.base import BaseEstimator, TransformerMixin
print('sklearn version: {}'.format(sklearn.__version__))
```

```
# plotDecisionTree from Shmueli (2020) Data Mining for Business Analytics
from dmba import plotDecisionTree, classificationSummary, regressionSummary

# import and configure plotting packages
import graphviz
import matplotlib
#import matplotlib.pyplot as plt
import matplotlib.pylab as plt
import matplotlib.patches as mpatches
plt.rc("font", size=14)

import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
print('matplotlib version: {}'.format(matplotlib.__version__))
print('seaborn version: {}'.format(sns.__version__))
print('graphviz version: {}'.format(graphviz.__version__))

# warnings
import warnings
```

Appendix B - Helpers.py

```
# helper functions

print('getFilename version: {}'.format('1.0'))
def getFilename(title: str, caption: str,
             sect='XX', ftype = 'PNG',
            course = 'D209', task = 'TASK1',
               subfolder='figures') -> str:
    """
    Construct a filename for given figure or table
    Input:
      title:
      sect:
      caption:
      ftype:
      course:
      task:
      subfolder:
    """
    temp = subfolder + '/'  # subfolder for tables and figures, default is 'fig'
    temp += course + '_'
    temp += task + '_'
    temp += sect + '_'
    temp += subfolder[0:3] + " " +caption + '_' #
    temp += title
    temp += '.' + ftype
    return temp.replace(' ','_').upper()


print('saveTable version: {}'.format('1.0'))
def saveTable(data, title: str, caption: str,
              sect='XX',course='D209',
              task='TASK1'):
    """
    Construct a filename for given figure or table
    Input:
      data:
      title:
      sect:
      caption:
      ftype:
      course:
      task:
      subfolder:
    """
    # construct filename based on parameters
    f = getFilename(title=title, sect=sect, task=task,
                caption=caption, ftype='CSV',
                course=course, subfolder='tables')
    data.to_csv(f, index=True, header=True) # create .csv file
    display(data.head(4).T) # display dataframe head data translated for vertical
readibility
    print('shape: {}'.format(data.shape)) # describe dataframe rows and cols
    print('Table saved to: {}'.format(f)) # feedback to notebook


print('describeData version: {}'.format('1.0'))
def describeData(data):
    """
    Describe a set of data as Continuous or Categorical
    Input:
      data: dataframe to be described
    """
    for idx, c in enumerate(data.columns):
        if data.dtypes[c] in ('float', 'int', 'int64'):
            print('\n{}. {} is numerical (CONTINUOUS) - type: {}.'.format(idx+1, c,
data.dtypes[c]))
            if data.dtypes[c] in ('int', 'int64'):
                numbers = data[c].to_numpy()
                print('  Unique: {}'.format(get_unique_numbers(numbers)))
            if data.dtypes[c] in ('float', 'float64'):
                print('  Min: {:.3f}  Max: {:.3f}  Std: {:.3f}'.format(data[c].min(),
data[c].max(),data[c].std()))

        elif data.dtypes[c] == bool:
            print('\n{}. {} is boolean (BINARY): {}.'.format(idx+1,c,data[c].unique()))
        else:
            print('\n{}. {} is categorical (CATEGORICAL):
{}.'.format(idx+1,c,data[c].unique()))


print('createScatter version: {}'.format('1.0'))
def createScatter(data,feature,target,c,edgecolor,title,caption,course,task):
    """
    Create and save a custom scatter plot fiugre
    Input:
    data: dataframe
    feature:
    target:
    c:
    edgecolor:
    title:
    caption:
    course:
    """
    import matplotlib.pyplot as plt

    # define a couple of plot variables
    title = title + ' ' + str(feature) + ' ' + str(target)

    # create fig,ax
    fig,ax = plt.subplots()
    ax.scatter(data[feature],data[target],c=c,edgecolor=edgecolor)

    # set title
    ax.set_title(title.upper(), fontsize=16)

    # create filename and save
    f=getFilename(title=title, caption=caption,
                  course=course, task=task,
                  ftype='PNG', subfolder='figures')
    plt.gcf().text(0, -.05, f, fontsize=14)
    fig.savefig(f, dpi=150, bbox_inches='tight')
    print('Figure saved to: {}'.format(f)) # feedback to notebook


print('createBarplot version: {}'.format('1.1'))
def createBarplot(data,feature,target,title,caption,course,task):
    """
    Create and save a custom bar plot fiugre
    Input:
    feature: feature (Categorical)
    target: target (Numerical)
    title:
    caption:
    course:
    """
    import matplotlib.pyplot as plt
    fig,ax = plt.subplots()
    ax=data.groupby(feature).mean()[target].plot(kind='bar')
    title = title + ' ' + str(feature) + ' ' + str(target)
    ax.set_title(title.upper())
    ax.set_ylabel(('Ave. ' + target).upper())
    f=getFilename(title=title, caption=caption,
                  course=course, task=task,
                  ftype='PNG', subfolder='figures')
    plt.gcf().text(0, -.05, f, fontsize=14)
    fig.savefig(f, dpi=150, bbox_inches='tight')
    print('Figure saved to: {}'.format(f)) # feedback to notebook



print('get_unique_numbers version: {}'.format('1.0'))
```

```
def get_unique_numbers(numbers):
    """
    Input:
    numbers: array

    Ref: https://www.freecodecamp.org/news/python-unique-list-how-to-get-all-the-unique-values-in-a-list-or-array/
    """
    list_of_unique_numbers = []
    unique_numbers = set(numbers)
    for number in unique_numbers:
        list_of_unique_numbers.append(number)
    return list_of_unique_numbers


print('createCorrelationMatrix version: {}'.format('1.0'))
def createCorrelationMatrix(data,title,caption,course,task,highest):
    """
    Create and save custom correlation matrix (heatmap) plot
    Input:
    data: dataframe of corr matrix
    c:
    edgecolor:
    title:
    caption:
    course:
    """
    import matplotlib.pyplot as plt
    import seaborn as sns
    fig,ax = plt.subplots()
    sns.heatmap(data.corr(), annot=True, fmt='.1f',
        cmap='RdBu', center=0, ax=ax)
    ax.set_title(title.upper())
    fig.set_size_inches(8, 5)
    f=getFilename(title=title, caption=caption,
                  course=course, task=task,
                  ftype='PNG', subfolder='figures')
    plt.gcf().text(0, -.05, f, fontsize=14)
    plt.gcf().text(0, -.1, 'Top ' + str(highest) + ' Correlations:', fontsize=14,
              horizontalalignment='left', verticalalignment='top')
    plt.gcf().text(.04, -.15, get_top_abs_correlations(data, n=highest).to_string(),
              fontsize=14,horizontalalignment='left', verticalalignment='top')
    fig.savefig(f, dpi=150, bbox_inches='tight')
    print('Figure saved to: {}'.format(f)) # feedback to notebook


def get_redundant_pairs(df):
    '''Get diagonal and lower triangular pairs of correlation matrix'''
    pairs_to_drop = set()
    cols = df.columns
    for i in range(0, df.shape[1]):
        for j in range(0, i+1):
            pairs_to_drop.add((cols[i], cols[j]))
    return pairs_to_drop

def get_top_abs_correlations(df, n=5):
    au_corr = df.corr().abs().unstack()
    labels_to_drop = get_redundant_pairs(df)
    au_corr = au_corr.drop(labels=labels_to_drop).sort_values(ascending=False)
    return au_corr[0:n]


print('createStackedHistogram version: {}'.format('1.0'))
def
createStackedHistogram(data,feature,target,title,caption,course,task,bins=9,isCat=True):
    """
    Create and save a custom stacked histogram
    Input:
    data:
    feature: feature (Numerical)
    target: target (Yes/No)
    title:
    caption:
    course:
    task:
    bins:
    isCat: bool if target is cat, else num
    """
    import matplotlib.pyplot as plt
    import pandas as pd

    # create fig,ax
    fig,ax = plt.subplots()

    # define couple of plot variables
    if isCat==True:

        yes = data[data[target]=='yes'][feature]; yes_mean = yes.mean();
        no = data[data[target]=='no'][feature]; no_mean = no.mean()

    else:
        yes = data[data[target]==1][feature]; yes_mean = yes.mean();
        no = data[data[target]==0][feature]; no_mean = no.mean()

    plt.hist([yes, no], bins=bins, stacked=True)
    title = title + ' ' + str(feature) + ' ' + str(target)

    # add legend
    ax.legend([str(target)+'= Yes',str(target)+'= No'])

    # add datatable
    b = pd.cut(data[feature], bins=bins) # create bins (b) of numeric feature
    ct = pd.crosstab(data[target], b)
    plt.gcf().text(0.1, -.05, ct.T.to_string(), fontsize=14,
            horizontalalignment='left', verticalalignment='top')

    # set min-max x and y limits
    ymin, ymax = ax.get_ylim()
    xmin, xmax = ax.get_xlim()

    # add title
    plt.title(title.upper(), fontsize=16)

    # axis labesl
    plt.xlabel(feature.upper())
    plt.ylabel(target.upper())

    # create group mean lines
    ax.axvline(yes_mean, color="blue", lw=2)  # yes mean
    ax.axvline(no_mean, color="orangered", lw=2)  # no mean
    #ax.text((xmax - xmin) / 2,
    #        (ymax - ymin) / 2,
    #        "Delta:\n" + str(round(abs(yes_mean - no_mean), 2)),
    #        bbox={"facecolor": "white"} )

    # add filename and save
    f=getFilename(title=title, caption=caption,
                  course=course, task=task,
                  ftype='PNG', subfolder='figures')
    plt.gcf().text(0, -.05, f, fontsize=14)
    fig.savefig(f, dpi=150, bbox_inches='tight')
    print('Figure saved to: {}'.format(f)) # feedback to notebook
```