WGU D209 TASK 1 REV 2 - MATTINSON

KNN Classification Using Churn Data

Mike Mattinson

Master of Science, Data Analytics, WGU.edu

D209: Data Mining I

Task 1 - 1st Submission

Dr. Festus Elleh

October 28, 2021

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 classification methods: (a) k-nearest neighbor (KNN) or (b) Naive Bayes.

Primary Goal. The question has come up for a telecommunications company regarding churn. **Churn** is defined when a customer chooses to stop services. If the company has data on customers that have and have not churned in the past, is it possible to classify a new (or existing) customer based on their similarity to other customers with similar attributes that have and have not churned in the past. This analysis will consider two (2) attributes, **MonthlyCharge** and **Tenure** within the company's customer data of 10,000 customers. In addition, if the prediction is made, the analysis will also attempt to quantify the accuracy of the prediction.

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. The analysis will attempt to predict **Churn** for a new customer with values of **MonthlyCharge** = $170.00 and **Tenure** = 1.0. This goal is within the scope of the company's customer data, both attributes are contained with the data for 10,000 customers and should provide adequate data for the prediction. The analysis will use K-nearest neighbors (KNN) to classify the new customer based on the k-nearest other customers with similar attributes.

In [1]:
```
# define new customer
import pandas as pd
newCustomer = pd.DataFrame([{'MonthlyCharge': 170.0 ,
                             'Tenure': 1.0}])
```

Part II: Method Justification

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

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

Explain Method. KNN classification will look for similar attributes in the closest k-neighbors, that are in close proximity to the target value to be classified. It will decide which classification value occurs most frequently in those k-neighbors and then output a classification prediction based on those values. I would expect the results to show the target variable as it relates to the k-neighbors and accuracy summaries for the model.

B2. Summarize one assumption of the chosen classification method.

Under construction

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

In [2]:
```
# 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)
```

Under construction

In [3]:
```
# import and configure scientific computing
import numpy as np
import scipy.stats as stats
#import statsmodels.api as sm
#import statsmodels.formula.api as smf
```

Under construction

In [4]:
```
# import and configure sklearn
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors
```

Under construction

In [5]:
```
# import and configure matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
plt.rc("font", size=14)
%matplotlib inline
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

Under construction

In [6]:
```python
# helper function to plot hist overlay of feature and target data
def plot_hist_overlay(feature, fig, p, bins=8, target='Churn'):

    # data
    df_yes = df[df[target]==True][feature]
    df_no = df[df[target]==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('Count')

    # add legend
    ax.legend(['True','False'])

    return(f)
```

In [7]:
```python
# helper function to plot a given dataset with the target data
def plotDataset(ax, data, xFeature, yFeature, target, neighbors, showLabel=True, **kwargs)

    # Churn == True
    subset = data.loc[data[target]==True]
    ax.scatter(subset[xFeature], subset[yFeature], marker='o',
              label=str(target)+'=True' if showLabel else None, color='C1', **kwargs)

    # Churn == False
    subset = data.loc[data[target]==False]
    ax.scatter(subset[xFeature], subset[yFeature], marker='D',
              label=str(target)+'=False' if showLabel else None, color='C0', **kwargs)

    # labels
    if len(neighbors) > 0:
        for idx, row in data.iterrows():
            ax.annotate(row.Number, (row[xFeature]+.2, row[yFeature]+.2))
```

In [8]:
```python
# helper function to standardize the format
# a filename for figures and tables
COURSE = 'D209' # global
TASK = 'Task1' # global
FTYPE = 'PNG' # global

def getFilename(title: str, sect: str,
            caption: str, ftype = FTYPE,
            course = COURSE, task = TASK,
               subfolder='figures') -> str:
    """
    Prepare filename for a figure or table
    """
    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()
```

In [9]:
```python
f = getFilename('hello', sect='d2',
    subfolder='tables', caption='4 1', ftype='CSV' )
print(f)
```

```
TABLES/D209_TASK1_D2_TAB_4_1_HELLO.CSV
```

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 classification method from part A1.

One Data Preprocessing Goal. In order to apply the KNN classification analysis to this problem, the company data must be imported into the Python environment and then the raw numerical data must be normalized. In addition, the company data will be broken up into two (2) subsets, 70% in a training dataset, and the remain 30% in a testing or validation dataset. The KNN will then use the training set to build the model, and it will use the test set to validate the model. The main goal for data preparation will be to define these subsets of data is a manner that is as simple and intuitive as possible, to allow anyone to follow the analysis throughout the notebook. The following is a list of the planned data variables for this analysis:

- **df** = the raw set of 10,000 customer records
- **trainData** = a 70% subset of the raw data
- **validData** = a 30% subset of the raw data
- **churnNorm** = the standardized set of 10,000 customer records
- **trainNorm** = a 70% subset of the standardized data. This will be created so that the index of records matches the index for **trainData**
- **validNorm** = a 30% subset of the standardized data. This will be created so that the index of the records matches the index for **validData**
- **X** the feature data from the standardized data (i.e. **MonthlyCharge**, and **Tenure**)
- **y** = the target data from the standardized data (i.e. **Churn**)

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.

In [10]:
```
# read subset of company's customer data from csv file
df = pd.read_csv('data/churn_clean.csv',
        usecols=['MonthlyCharge','Tenure','Churn'])
```

Identify Initial Variables. For this analysis, I will consider two (2) features, **MonthlyCharge** and **Tenure**, and one (1) target, **Churn**. Pandas is used to read the .CSV raw data file, the USECOLS option retrieves only selected data from the file.

- **MonthlyCharge** (FEATURE) the amount charged to the customer monthly, it reflects an average per customer
- **Tenure** (FEATURE) the number of months the customer has stayed with the provider
- **Churn** (TARGET) is whether the customer has discontinued service within the last month (yes, no).

In [11]:
```
# describe target data
target = 'Churn'
print(df[target].describe())
print('Unique values: {}'.format(df[target].unique()))
```

```
count      10000
unique         2
top           No
freq        7350
Name: Churn, dtype: object
Unique values: ['No' 'Yes']
```

TABLE 3.1. SELECTED RAW CUSTOMER DATA. THIS IS THE PRIMARY DATASET IDENTIFIED AS 'DF' OF RAW DATA. NOTICE CHURN VALUES (YES AND NO). Ref. (1) https://stackoverflow.com/questions/15017072/pandas-read-csv-and-filter-columns-with-usecols (https://stackoverflow.com/questions/15017072/pandas-read-csv-and-filter-columns-with-usecols)(Ctrl+click to follow)

In [12]:

```
# describe RAW CUSTOMER data
d = df
f = getFilename('DF RAW', sect='C2',
        subfolder='tables', caption='3 1', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.info())
print('Table saved to: {}'.format(f))
```

	Churn	Tenure	MonthlyCharge
0	No	6.796	172.456
1	Yes	1.157	242.633
2	No	15.754	159.948
3	No	17.087	119.957
4	Yes	1.671	149.948

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Churn          10000 non-null  object
 1   Tenure         10000 non-null  float64
 2   MonthlyCharge  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ KB
None
Table saved to: TABLES/D209_TASK1_C2_TAB_3_1_DF_RAW.CSV
```

Data Cleaning. Take care of a couple minor data cleaning items: (1) convert categorical Churn to numeric boolean, (2) re-index, (3) add row label, and (4) reorder columns. Ref: (1) https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

In [13]:
```python
# convert churn from object [Yes No] to bool [True False]
df['Churn'] = df['Churn'].replace({"No":False, "Yes":True})
df['Churn'] = df['Churn'].astype('bool')

# reset index
df.reset_index(drop=True, inplace=True)

# add a row label called 'Number'
df['Number'] = df.index

# reorder cols
columns=['MonthlyCharge','Tenure','Churn','Number']
df = df[columns]
```

Sample Data. Use the following code to reduce the dataset, use for trouble-shooting, etc.

In [14]:
```python
#df = df.sample(frac=0.01, random_state=13)
#display(df.head())
#print(df.info())

# or use .iloc[]
#df = df.iloc[::100, :] # every 100th row...
```

Initial Variables. Classify each variable as continuous or categorical.

In [15]:
```python
# identify the initial set of variables
for idx, c in enumerate(df.columns):
    if df.dtypes[c] in ('float', 'int', 'int64'):
        print('\n{}. {} is numerical (CONTINUOUS).'.format(idx+1, c))
    elif df.dtypes[c] == bool:
        print('\n{}. {} is boolean (BINARY): {}.'.format(idx+1,c,df[c].unique()))
    else:
        print('\n{}. {} is categorical (CATEGORICAL): {}.'.format(idx+1,c,df[c].unique()))
```

```
1. MonthlyCharge is numerical (CONTINUOUS).

2. Tenure is numerical (CONTINUOUS).

3. Churn is boolean (BINARY): [False  True].

4. Number is numerical (CONTINUOUS).
```

TABLE 3.2. DESCRIBE NUMERIC FEATURE DATA. THESE ARE THE TRADITIONAL STATISTICS FOR THE NUMERICAL DATA

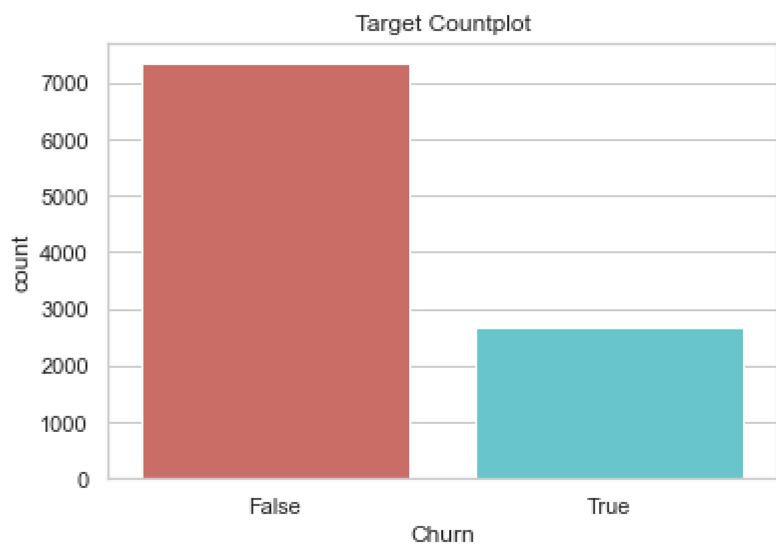
In [16]:
```python
# describe numeric feature data
d = df[['MonthlyCharge','Tenure']].describe()
f = getFilename('DF STATS', sect='c2',
       subfolder='tables', caption='3 2', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d)
print('Table saved to: {}'.format(f))
```

	MonthlyCharge	Tenure
count	10000.000	10000.000
mean	172.625	34.526
std	42.943	26.443
min	79.979	1.000
25%	139.979	7.918
50%	167.485	35.431
75%	200.735	61.480
max	290.160	71.999

```
Table saved to: TABLES/D209_TASK1_C2_TAB_3_2_DF_STATS.CSV
```

Data Visualization. Create plots to visualize target and target-feature data.

In [17]:
```python
# visualize target data
fig, ax = plt.subplots()
sns.countplot(x=target, data=df, palette='hls')
title = 'Target Countplot'
plt.title(title)
f = getFilename(title, sect='c2',
       subfolder='figures', caption='3 1') # getFilename using helper
plt.gcf().text(0, -.2, f, fontsize=14)

# include data table
plt.gcf().text(0, -.1, df[target].value_counts().to_string(), fontsize=14)

fig.savefig(f, dpi=150, bbox_inches='tight')
plt.show()
```

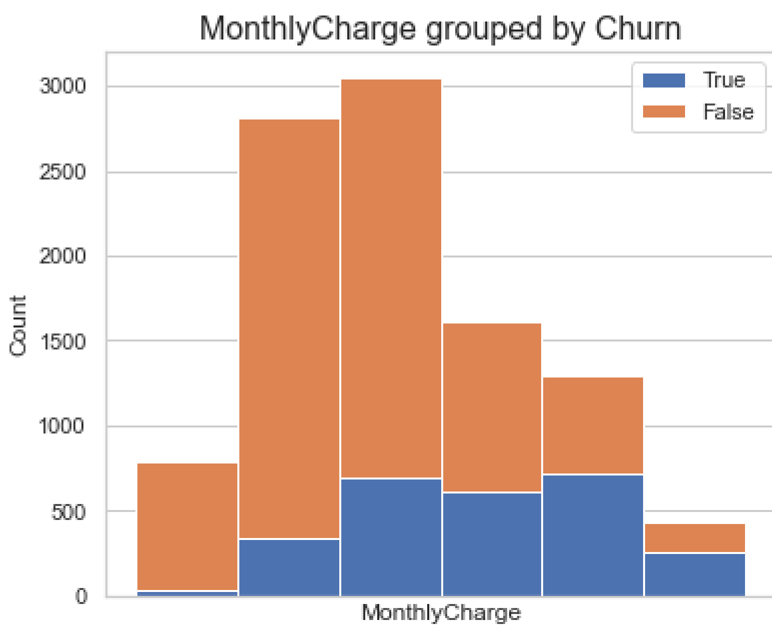


False   7350
True   2650

FIGURES/D209_TASK1_C2_FIG_3_1_TARGET_COUNTPLOT.PNG

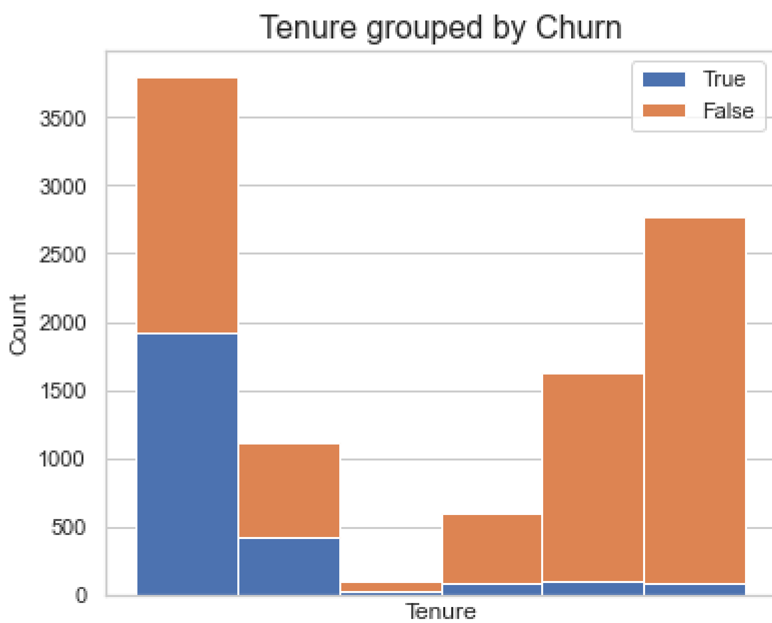
FIGURE 3-1. TARGET COUNTPLOT

In [18]:
```python
# create histogram with target overlay
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, sect='c2',
            subfolder='figures', caption='3 ' + str(idx+2)) # getFilename using helper
    plt.gcf().text(0.1, 0, file, fontsize=14)

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

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



FIGURES/D209_TASK1_C2_FIG_3_2_MONTHLYCHARGE.PNG

Churn	False	True
MonthlyCharge		
(79.769, 115.009]	755	35
(115.009, 150.039]	2477	338
(150.039, 185.07]	2356	693
(185.07, 220.1]	1009	608
(220.1, 255.13]	579	715
(255.13, 290.16]	174	261



FIGURES/D209_TASK1_C2_FIG_3_3_TENURE.PNG

Churn	False	True
Tenure		
(0.929, 12.833]	1876	1924
(12.833, 24.667]	692	418
(24.667, 36.5]	70	28
(36.5, 48.333]	511	83
(48.333, 60.166]	1526	105
(60.166, 71.999]	2675	92

FIGURE 3-2 AND FIGURE 3-3. CHURN GROUPED BY FEATURE. TOP FEATURE IS MONTHLYCHARGE. BOTTOM FEATURE IS TENURE

Summary. The company's customer raw data has been read into the df variable and consists of 10,000 customer records with three (3) variables each. Two (2) of the variables will be used as features and are continuous (numerical) data, and the the third variable is our target, binary variable.

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

Step 1.

Read in selected company data. Applicable customer data (**Churn**, **MonthlyCharge** and **Tenure**) from the company data was read into Python environment using pandas .read_cs() function using the usecols=[] option. This was completed in section C2 [9] above.

Step 2.

Convert cateogrical dataInitially, the **Churn** variable was categorical, each row was Yes or No values, so this step converted the categorical data to boolean data using pandas .replace() function. In Python, boolean data is considered as numerical data, 1 or 0, or type(int). This was completed in section C2 [9] above.

Step 3.

Describe initial set of variablesFor each variable of data, describe the data whether numerical or categorical. I used a function I created to loop through and list each one and a short description. Also, use pandas .describe() method to show descriptive statistics for numerical data. This was completed in section C2 [10] and C2[11] above.

Step 4.

Quick check for null valuesThe company data was previously cleaned and prepared, so I do not expect to find null values, but using the pandas .info() I can observe quickly that there are not any null values for any of the 10,000 customer records. This was completed in section C2 [12] above.

C4. Provide Clean Data

Provide a copy of the cleaned data set.

TABLE 3-3 CLEAN DATA

In [19]:
```python
# provide copy of cleaned data
d = df
f = getFilename('DF CLEAN', sect='c4',
        subfolder='tables', caption='3 3', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.columns.to_series().groupby(df.dtypes).groups)
print(d.info())
print('Shape (rows, cols): {}'.format(d.shape))
print('Table saved to: {}'.format(f))
```

	MonthlyCharge	Tenure	Churn	Number
0	172.456	6.796	False	0
1	242.633	1.157	True	1
2	159.948	15.754	False	2
3	119.957	17.087	False	3
4	149.948	1.671	True	4

```
{bool: ['Churn'], int64: ['Number'], float64: ['MonthlyCharge', 'Tenure']}
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthlyCharge  10000 non-null  float64
 1   Tenure         10000 non-null  float64
 2   Churn          10000 non-null  bool   
 3   Number         10000 non-null  int64  
dtypes: bool(1), float64(2), int64(1)
memory usage: 244.3 KB
None
Shape (rows, cols): (10000, 4)
Table saved to: TABLES/D209_TASK1_C4_TAB_3_3_DF_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 [20]:
```python
# train test split raw data
trainData, validData = train_test_split(df, test_size=0.3, random_state=13)
```

TABLE 4-4 TRAINING DATA

In [21]:
```
# provide TRAIN data
d = trainData
f = getFilename('TRAIN DATA', sect='D1',
        subfolder='tables',caption='4 4', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.index)
print(d.info())
print('Shape (rows, cols): {}'.format(d.shape))
print('Table saved to: {}'.format(f))
```

	MonthlyCharge	Tenure	Churn	Number
4847	92.488	9.525	False	4847
9992	137.439	56.472	False	9992
4621	124.964	2.612	False	4621
5774	139.983	58.787	False	5774
9294	255.120	64.116	False	9294

```
Int64Index([4847, 9992, 4621, 5774, 9294, 1085, 1073,  950, 9512, 3773,
            ...
            6782, 9114, 4026, 8940,  153, 5876,  866, 7696,   74,  338],
           dtype='int64', length=7000)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7000 entries, 4847 to 338
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthlyCharge  7000 non-null   float64
 1   Tenure         7000 non-null   float64
 2   Churn          7000 non-null   bool   
 3   Number         7000 non-null   int64  
dtypes: bool(1), float64(2), int64(1)
memory usage: 225.6 KB
None
Shape (rows, cols): (7000, 4)
Table saved to: TABLES/D209_TASK1_D1_TAB_4_4_TRAIN_DATA.CSV
```

TABLE 4-5 TEST DATA

In [22]:
```
# provide TEST data
d = validData
f = getFilename('TEST DATA', sect='D1',
        subfolder='tables', caption='4 5', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.index)
print(d.info())
print('Shape (rows, cols): {}'.format(d.shape))
print('Table saved to: {}'.format(f))
```

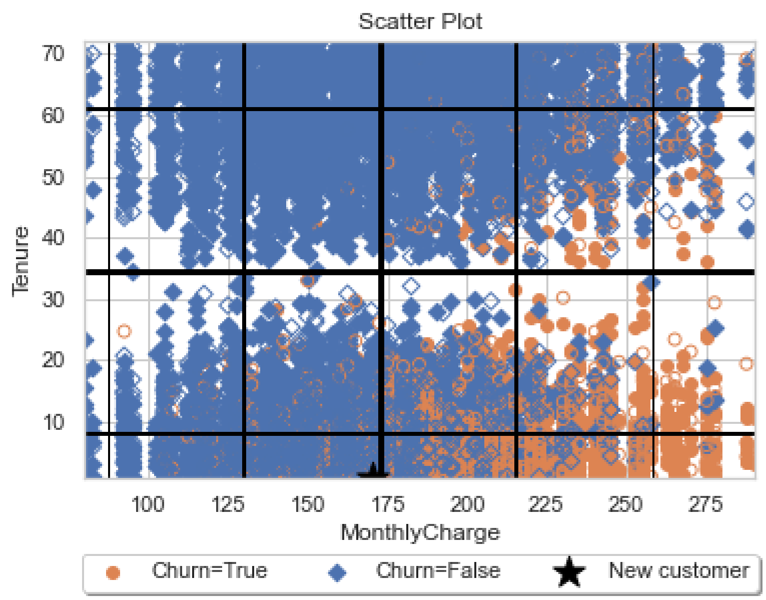
	MonthlyCharge	Tenure	Churn	Number
5952	114.984	56.633	False	5952
1783	117.483	2.851	False	1783
4811	230.105	5.664	True	4811
145	217.473	2.733	True	145
7146	200.132	56.275	True	7146

```
Int64Index([5952, 1783, 4811,  145, 7146, 2452, 4051, 4311, 9715,  303,
            ...
            1442, 5091, 6525, 1241, 1161, 8654, 9777, 3727, 7848, 4977],
           dtype='int64', length=3000)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3000 entries, 5952 to 4977
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthlyCharge  3000 non-null   float64
 1   Tenure         3000 non-null   float64
 2   Churn          3000 non-null   bool   
 3   Number         3000 non-null   int64  
dtypes: bool(1), float64(2), int64(1)
memory usage: 96.7 KB
None
Shape (rows, cols): (3000, 4)
Table saved to: TABLES/D209_TASK1_D1_TAB_4_5_TEST_DATA.CSV
```

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

Data Exploratory Analysis. I will create a scatter plot of the two (2) features showing differences between Churn=True and Churn=False customers. I will plot the new customer in the same plot to see where the new and existing customers are similar. We will then see what we expect the classification results will yield in the end.

In [23]:
```
# scatter plot using the plotDataset() helper function
xFeature = 'MonthlyCharge'
yFeature = 'Tenure'
target = 'Churn'
neighbors = []
fig, ax = plt.subplots()
#fig.set_size_inches(18.5, 10.5)
plotDataset(ax, trainData, xFeature, yFeature, target, neighbors)
plotDataset(ax, validData, xFeature, yFeature, target, neighbors, showLabel=False, facecol

# vertical lines for means for xFeature
c = 'black'
ax.axvline(trainData[xFeature].mean(), color=c, lw=3)
ax.axvline(trainData[xFeature].mean()+1*trainData[xFeature].std(), color=c, lw=2)
ax.axvline(trainData[xFeature].mean()-1*trainData[xFeature].std(), color=c, lw=2)
ax.axvline(trainData[xFeature].mean()+2*trainData[xFeature].std(), color=c, lw=1)
ax.axvline(trainData[xFeature].mean()-2*trainData[xFeature].std(), color=c, lw=1)

# horizontal lines for means for yFeature
c = 'black'
ax.axhline(trainData[yFeature].mean(), color=c, lw=3)
ax.axhline(trainData[yFeature].mean()+1*trainData[yFeature].std(), color=c, lw=2)
ax.axhline(trainData[yFeature].mean()-1*trainData[yFeature].std(), color=c, lw=2)
ax.axhline(trainData[yFeature].mean()+2*trainData[yFeature].std(), color=c, lw=1)
ax.axhline(trainData[yFeature].mean()-2*trainData[yFeature].std(), color=c, lw=1)

# plot new customer as a Star
ax.scatter(newCustomer.MonthlyCharge, newCustomer.Tenure, marker='*',
          label='New customer', color='black', s=270)

title = 'Scatter Plot'
plt.title(title)
plt.xlabel(xFeature)
plt.ylabel(yFeature)
ax.set_xlim(df[xFeature].min(),df[xFeature].max())
ax.set_ylim(df[yFeature].min(),df[yFeature].max())

# configure legend
handles, labels = ax.get_legend_handles_labels()
patch = mpatches.Patch(color='grey', label='Manual Label')
handles.append(patch)
plt.legend(handles, labels, loc='upper center', bbox_to_anchor=(0.5, -0.15),
          fancybox=True, shadow=True, ncol=5)
# add customer data text
plt.gcf().text(0, -.4, newCustomer.to_string(), fontsize=14)

f = getFilename(title, sect='d2',
                subfolder='figures', caption='4 2') # getFilename using helper
plt.gcf().text(0, -.2, f, fontsize=14)
fig.savefig(f, dpi=150, bbox_inches='tight')
plt.show()
```



FIGURES/D209_TASK1_D2_FIG_4_2_SCATTER_PLOT.PNG

	MonthlyCharge	Tenure
0	170.0	1.0

FIGURE 4-2. SCATTER PLOT OF MONTHLYCHARGE VS TENURE FOR TRAINING SET (SOLID MARKERS) AND TEST SET (HOLLOW MARKERS) AND THE NEW CUSTOMER (STAR MARKER) TO BE CLASSIFIED.
Ref: (1) Textbook_____, Chapter 7, KNN.

Scale Data. The z-score method (often called standardization) transforms the info into distribution with a mean of 0 and a typical deviation of 1. Each standardized value is computed by subtracting the mean of the corresponding feature then dividing by the quality deviation. I will use the sklearn .StandardScaler() method to create a standardized data set. Ref: (1) https://www.geeksforgeeks.org/data-normalization-with-pandas/ (https://www.geeksforgeeks.org/data-normalization-with-pandas/)

In [24]:
```
# use training data to learn the transformation
scaler = preprocessing.StandardScaler()
scaler.fit(trainData[['MonthlyCharge','Tenure']])
```

Out[24]:
```
StandardScaler()
```

In [25]:
```
# transform the full dataset
churnNorm = pd.concat([pd.DataFrame(scaler.transform(df[['MonthlyCharge','Tenure']]),
            columns=['zMonthlyCharge','zTenure']),
            df[['Churn','Number']]], axis=1)
trainNorm = churnNorm.iloc[trainData.index]
validNorm = churnNorm.iloc[validData.index]
```

TABLE 4-6 SCALED CUSTOMER DATA INFO

In [26]:
```
# provide SCALED data
d = churnNorm
f = getFilename('SCALED DATA', sect='D2',
        subfolder='tables', caption='4 6', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.index)
print('Table saved to: {}'.format(f))
```

	zMonthlyCharge	zTenure	Churn	Number
0	-0.008	-1.050	False	0
1	1.632	-1.263	True	1
2	-0.300	-0.710	False	2
3	-1.234	-0.660	False	3
4	-0.534	-1.244	True	4

```
RangeIndex(start=0, stop=10000, step=1)
Table saved to: TABLES/D209_TASK1_D2_TAB_4_6_SCALED_DATA.CSV
```

Scale New Customer Data. Use same scaler transformation to normalize the new customer data.

In [27]:
```
# scale new customer data
newCustomerNorm = pd.DataFrame(scaler.transform(newCustomer),
        columns=['zMonthlyCharge', 'zTenure'])
print(newCustomerNorm.round(2))
print(newCustomer.round(2))
```

```
   zMonthlyCharge  zTenure
0           -0.07    -1.27
   MonthlyCharge  Tenure
0          170.0     1.0
```

Find Nearest Training Neighbors. Use KNN and scaled data to find k-nearest neighbors.

In [28]:
```
# list neighbors from raw data
knn = NearestNeighbors(n_neighbors=7)
knn.fit(trainNorm.iloc[:,0:2])
distances, indices = knn.kneighbors(newCustomerNorm)
training_neighbors = df.iloc[indices[0],:]
#display(training_neighbors)
```

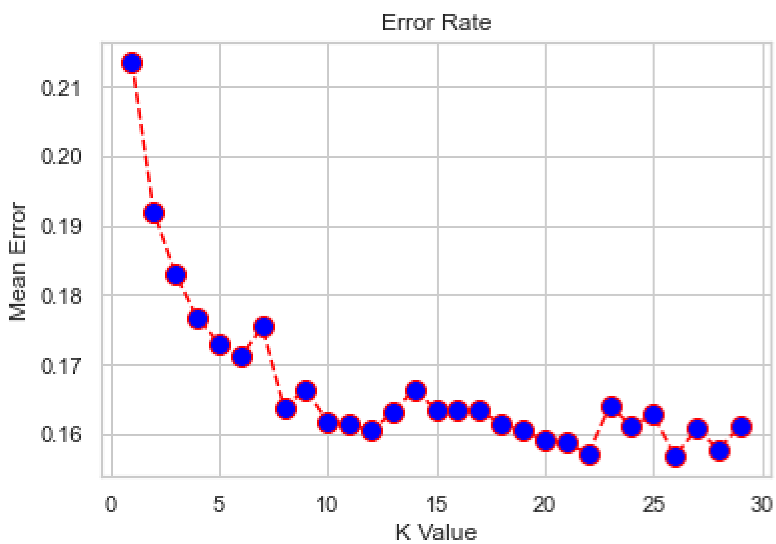
TABLE 4-7 K-NEAREST "TRAINING" NEIGHBORS

In [29]:
```
# provide SCALED data
d = training_neighbors
f = getFilename('TRAINING NEIGHBORS', sect='D2',
        subfolder='tables', caption='4 7', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print('Table saved to: {}'.format(f))
```

	MonthlyCharge	Tenure	Churn	Number
5684	275.120	67.721	False	5684
4485	240.115	10.964	True	4485
3371	139.979	11.327	False	3371
6462	92.455	62.435	False	6462
1943	204.961	24.377	True	1943

```
Table saved to: TABLES/D209_TASK1_D2_TAB_4_7_TRAINING_NEIGHBORS.CSV
```

```python
In [30]: # Calculating and plot error rate for range of k-values
train_X = trainNorm[['zMonthlyCharge','zTenure']]
train_y = trainNorm['Churn']
valid_X = validNorm[['zMonthlyCharge','zTenure']]
valid_y = validNorm['Churn']
error = []
for i in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(train_X, train_y)
    pred_i = knn.predict(valid_X)
    error.append(np.mean(pred_i != valid_y))
fig, ax = plt.subplots()
plt.plot(range(1, 30), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
title = 'Error Rate'
plt.title(title)
plt.xlabel('K Value')
plt.ylabel('Mean Error')
f = getFilename(title, 'd2','fig 4 3') # getFilename using helper
plt.gcf().text(0, -.2, f, fontsize=14)
fig.savefig(f, dpi=150, bbox_inches='tight')
plt.show()
```



FIGURES/D209_TASK1_D2_FIG_FIG_4_3_ERROR_RATE.PNG

FIGURE 4-2. ERROR RATE BY K-VALUE

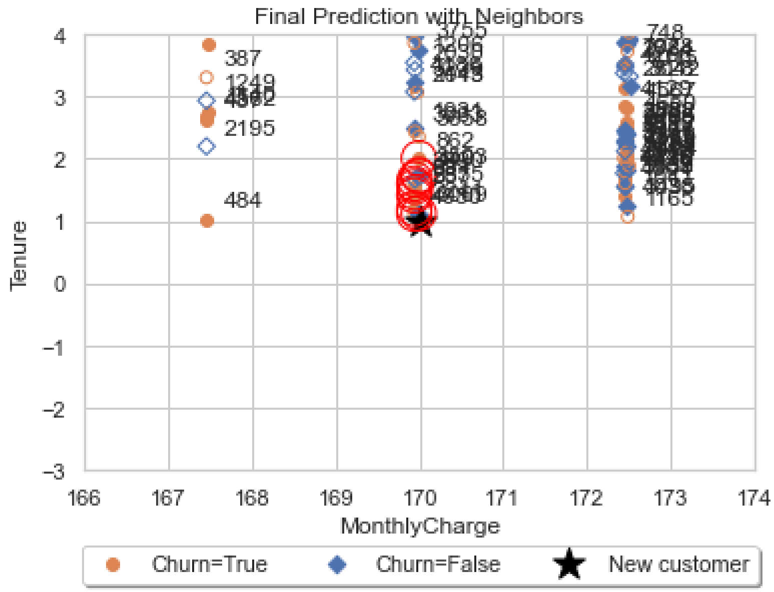
Final Prediction. Calculate final prediction using the complete set of scaled data. Select a value for k from the figure above, let's select k=11 which looks like it should have about 84% accurary. Create a list of the neighbors in order to include highlighted neighbors on the next plot.

In [31]:

```
# retrain with full data.
X = churnNorm[['zMonthlyCharge','zTenure']]
y = churnNorm['Churn']
knn = KNeighborsClassifier(n_neighbors=11).fit(X, y)
distances, indices = knn.kneighbors(newCustomerNorm)
print('Prediction: {}'.format(knn.predict(newCustomerNorm)))
df_neighbors = df.iloc[indices[0],:]
neighbors = df_neighbors.index
neighbors = neighbors.to_list()
```

```
Prediction: [ True]
```

In [32]:
```
# scatter plot using the plotDataset() helper function
xFeature = 'MonthlyCharge'
yFeature = 'Tenure'
fig, ax = plt.subplots()
plotDataset(ax, trainData, xFeature, yFeature, target, neighbors)
plotDataset(ax, validData, xFeature, yFeature, target, neighbors, showLabel=False, facecol

# plot new customer as a Star
ax.scatter(newCustomer.MonthlyCharge, newCustomer.Tenure, marker='*',
          label='New customer', color='black', s=270)

# highlight neighbors with red circles
if len(neighbors) > 0:
    for n in neighbors:
        point = df.iloc[n]
        ax.scatter(point.MonthlyCharge, point.Tenure, marker='o',
                color='red', s=300, facecolors='none')

title = 'Final Prediction with Neighbors'
plt.title(title)
plt.xlabel(xFeature)
plt.ylabel(yFeature)

# set axis limits centered around the new customer
left = float(newCustomer.MonthlyCharge) - 4
right = float(newCustomer.MonthlyCharge) + 4
top = float(newCustomer.Tenure) - 4
bottom = float(newCustomer.Tenure) + 3
ax.set_xlim(left,right)
ax.set_ylim(top,bottom)

handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, loc='upper center', bbox_to_anchor=(0.5, -0.15),
          fancybox=True, shadow=True, ncol=5)
#plt.legend(loc="lower center", bbox_to_anchor=(0.5, -0.15), ncol= 2)
f = getFilename(title, sect='d2',
        subfolder='figures', caption='4 4') # getFilename using helper
plt.gcf().text(0, -.2, f, fontsize=14)

# loop through neighbors and include neighbor as table data
#for idx,n in enumerate(df_neighbors.iloc[:, 0:3]):
#    plt.gcf().text(0, -.5+(.05*idx), n, fontsize=10)
plt.gcf().text(0, -1, df_neighbors.iloc[:, 0:3].to_string(), fontsize=14)

fig.savefig(f, dpi=150, bbox_inches='tight')
plt.show()
```



FIGURES/D209_TASK1_D2_FIG_4_4_FINAL_PREDICTION_WITH_NEIGHBORS.PNG

	MonthlyCharge	Tenure	Churn
4030	169.938	1.115	True
3009	169.993	1.125	False
2241	169.938	1.198	True
557	169.938	1.411	True
2855	169.945	1.462	True
53	169.945	1.553	True
951	169.938	1.600	True
3390	169.945	1.669	False
449	169.993	1.715	False
3503	169.993	1.740	False
862	169.993	2.010	True

FIGURE 4-4. FINAL CLASSIFICATION OF NEW CUSTOMER WITH NEIGHBORS (RED CIRLCES) USED TO CLASSIFY WITH THE NEIGHBOR DATA SORTED BY DISTANCE FROM NEW CUSTOMER. INCLUDE DATA TABLE USING .to_string() method. Adjust plot so that text does not get cut off at bottom.

Ref: (1) https://stackabuse.com/how-to-iterate-over-rows-in-a-pandas-dataframe/, (2) https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

Summary. The KNN model calculated the new customer as Churn=False, with all of the three (3) nearest neighbors having Churn=False, which is what we expected.

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_1_x.ipynb** and the associated PDF version is called **D209_1_x - Jupyter Notebook.pdf**.

Part V: Data Summary and Implications

E1. Explain the accuracy and the area under the curve (AUC) of your classification model.

Confusion and Classification Report. Look at confusion and classification report to determine overall accuracy of the KNN model.

TABLE 5-1. CONFUSION MATRIX AND THE CLASSIFICATION REPORT SHOWING MODEL ACCURARY

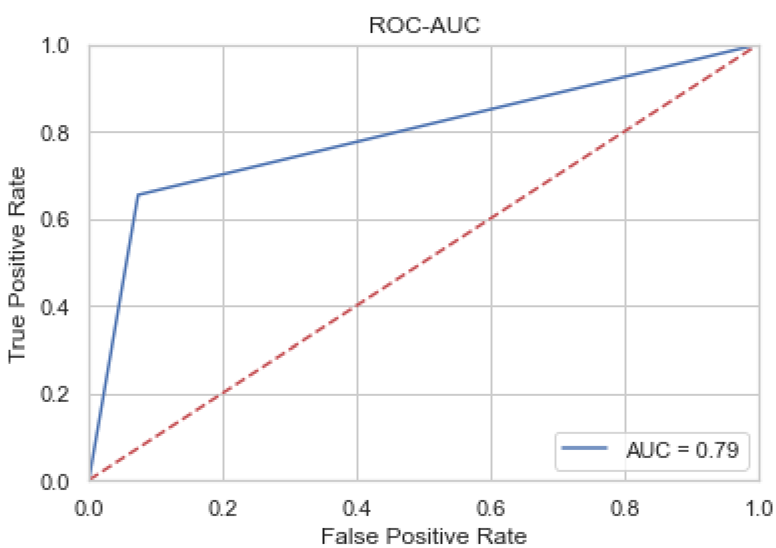
In [33]:
```
# confusion and classification report
classifier = KNeighborsClassifier(n_neighbors=11)
classifier.fit(X, y)
y_pred = classifier.predict(X)
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
[[6810  540]
 [ 917 1733]]
              precision    recall  f1-score   support

       False       0.88      0.93      0.90      7350
        True       0.76      0.65      0.70      2650

    accuracy                           0.85     10000
   macro avg       0.82      0.79      0.80     10000
weighted avg       0.85      0.85      0.85     10000
```

Receiver Operation Characteristic (ROC) ad Area Under Curve (AUC). Calculate and plot ROC and AOUC. Add custom text annotation to the plot. Ref: (1) https://stackoverflow.com/questions/42435446/how-to-put-text-outside-python-plots (https://stackoverflow.com/questions/42435446/how-to-put-text-outside-python-plots), (2) https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html), (3) https://stackoverflow.com/questions/25009284/how-to-plot-roc-curve-in-python (https://stackoverflow.com/questions/25009284/how-to-plot-roc-curve-in-python) and (4) https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5)

In [34]:
```
# calculate the fpr and tpr for all thresholds of the classification
fpr, tpr, threshold = metrics.roc_curve(y, y_pred)
auc = metrics.auc(fpr, tpr)
```

In [35]:
```
# method I: plt
fig, ax = plt.subplots()
title = 'ROC-AUC'
plt.title(title)
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % auc)
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')
plt.xlabel('False Positive Rate')
f = getFilename(title, sect='e1',
    subfolder='figures', caption='5 1') # getFilename using helper
#plt.gcf().text(0, -.1, 'Area Under Curve (AUC): {:.2f}'.format(auc), fontsize=14)
plt.gcf().text(0, -.2, f, fontsize=14)
fig.savefig(f, dpi=150, bbox_inches='tight')
plt.show()
```



FIGURES/D209_TASK1_E1_FIG_5_1_ROC-AUC.PNG

FIGURE 5-1. RECEIVER OPERATION CHARACTERISTIC (ROC)

Discuss Results and Implications

Discuss the results and implications of your classification analysis.

Summary. Looks like $6810 + 1733 = 8543$ predictions on the diagonal were correct for an accuracy of about 85.4%. Analysis predicts 85% that the new customer is Churn=True, so, therefore, there is also the 11% chance that the new customer is actually Churn=False.

E3. Discuss one limitation of your data analysis.

Limitations. It occus to me that a new customer is new, that is, their **Tenure** will always be low compared to other existing customers. The KNN analysis will never make it to the higher **Tenure** numbers. Future study may look at other features instead such as **Income**, **Bandwidth_GB_Year**, or **Outage_sec_perweek** which may provide better insight.

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.

> Under construction

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)](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.

In [36]:
```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

Disable Auto Scroll. Disable automatically scroll to bottom.

In [37]:
```
%%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)](https://stackoverflow.com/questions/9031783/hide-all-warnings-in-ipython)

In [38]:
```
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>.''')
```

Out[38]: To toggle on/off output_stderr, click here.

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)](https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel) Retrieved from https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

In [39]:
```
!ls
```

In [40]:
```
!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/

In [41]:
```
!pip list
```

```
Package                           Version
--------------------------------- ---------
anyio                             3.3.4
argon2-cffi                       21.1.0
attrs                             21.2.0
Babel                             2.9.1
backcall                          0.2.0
bleach                            4.1.0
certifi                           2021.10.8
cffi                              1.15.0
charset-normalizer                2.0.7
colorama                          0.4.4
cycler                            0.10.0
debugpy                           1.5.1
decorator                         5.1.0
defusedxml                        0.7.1
entrypoints                       0.3
idna                              3.3
ipykernel                         6.4.2
ipython                           7.28.0
ipython-genutils                  0.2.0
jedi                              0.18.0
Jinja2                            3.0.2
joblib                            1.1.0
json5                             0.9.6
jsonschema                        4.1.2
jupyter-client                    7.0.6
jupyter-contrib-core              0.3.3
jupyter-contrib-nbextensions      0.5.1
jupyter-core                      4.8.1
jupyter-highlight-selected-word   0.2.0
jupyter-latex-envs                1.4.6
jupyter-nbextensions-configurator 0.4.1
jupyter-server                    1.11.1
jupyterlab                        3.2.1
jupyterlab-pygments               0.1.2
jupyterlab-server                 2.8.2
kiwisolver                        1.3.2
lxml                              4.6.3
MarkupSafe                        2.0.1
matplotlib                        3.4.3
matplotlib-inline                 0.1.3
mistune                           0.8.4
nbclassic                         0.3.3
nbclient                          0.5.4
nbconvert                         6.2.0
nbformat                          5.1.3
nest-asyncio                      1.5.1
notebook                          6.4.5
numpy                             1.21.3
packaging                         21.0
pandas                            1.3.4
pandocfilters                     1.5.0
parso                             0.8.2
pickleshare                       0.7.5
Pillow                            8.4.0
```

```
pip                               21.3.1
prometheus-client                 0.11.0
prompt-toolkit                    3.0.21
pycparser                         2.20
Pygments                          2.10.0
pyparsing                         3.0.1
pyrsistent                        0.18.0
python-dateutil                   2.8.2
pytz                              2021.3
pywin32                           302
pywinpty                          1.1.4
PyYAML                            6.0
pyzmq                             22.3.0
requests                          2.26.0
requests-unixsocket               0.2.0
scikit-learn                      1.0.1
scipy                             1.7.1
seaborn                           0.11.2
Send2Trash                        1.8.0
setuptools                        57.4.0
six                               1.16.0
sklearn                           0.0
sniffio                           1.2.0
terminado                         0.12.1
testpath                          0.5.0
threadpoolctl                     3.0.0
tornado                           6.1
traitlets                         5.1.1
urllib3                           1.26.7
wcwidth                           0.2.5
webencodings                      0.5.1
websocket-client                  1.2.1
```

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

In [42]:
```
!python -m pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in p:\code\venv\lib\site-packages (1.0.1)
Requirement already satisfied: joblib>=0.11 in p:\code\venv\lib\site-packages (from sciki
t-learn) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in p:\code\venv\lib\site-packages (from sciki
t-learn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in p:\code\venv\lib\site-packages (fr
om scikit-learn) (3.0.0)
Requirement already satisfied: numpy>=1.14.6 in p:\code\venv\lib\site-packages (from scik
it-learn) (1.21.3)
```

Merget Two Dataframes. Code to merge two dataframes. Ref: (1) https://stackoverflow.com/questions/26265819/how-to-merge-a-series-and-dataframe

In [43]:
```python
# merge X and y back together, for example
d = X.merge(y, left_index=True, right_index=True)
display(d.head())
```

	zMonthlyCharge	zTenure	Churn
0	-0.008	-1.050	False
1	1.632	-1.263	True
2	-0.300	-0.710	False
3	-1.234	-0.660	False
4	-0.534	-1.244	True

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

In [44]:
```python
animals = ['cat', 'dog', 'rabbit', 'horse']
# get the index of 'dog'
index = animals.index('dog')
print(index)
```

```
1
```

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/

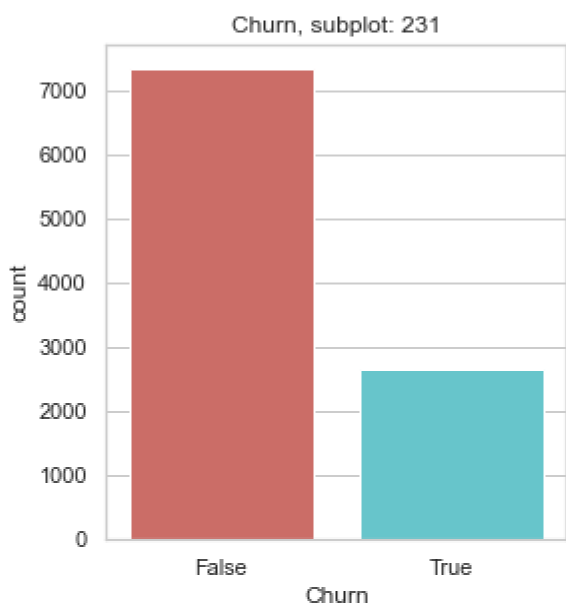
In [45]:
```python
# making data frame
data = df

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

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

```
0 1 2 3 4
```

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

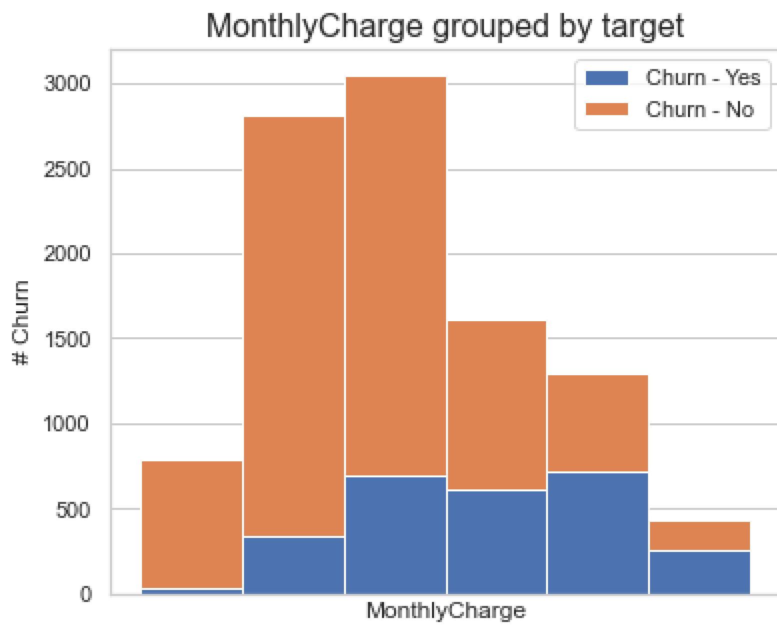
In [46]:

```
#  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=df, palette='hls')
    c = c + 1

plt.show()
```



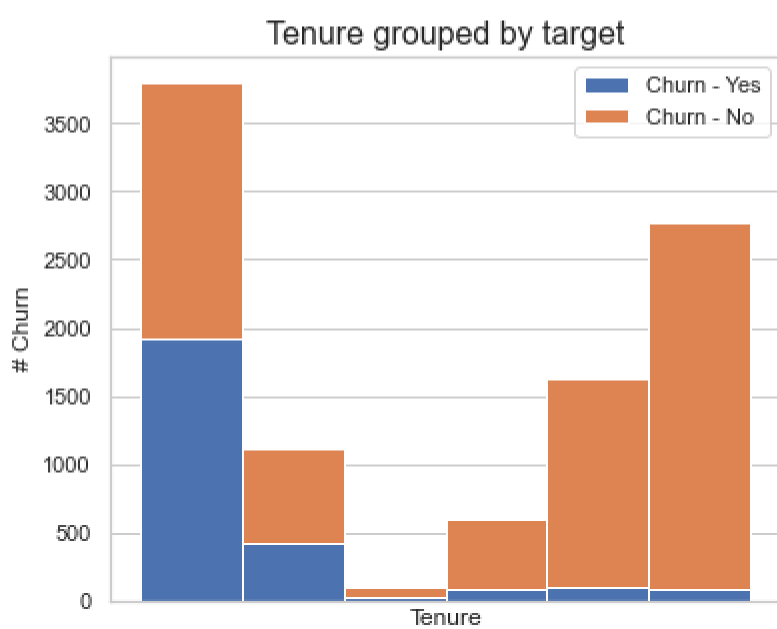
PASS FIG TO CUSTOM PLOT FUNCTION. A great way to do this is to pass a figure object to your code and have your function add an axis then return the updated figure. Here is an example: Ref: (1) https://stackoverflow.com/questions/43925337/matplotlib-returning-a-plot-object

In [47]:
```
def plot_hist_overlay(feature, fig, p, bins=8):

    # data
    df_yes = df[df.Churn==True][feature]
    df_no = df[df.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(df[fea], bins=bins) # create bins (b) of numeric feature
    dt = pd.crosstab(df[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)
```



FIGURES/D209_TASK1_Z1_FIG_FIG_9_1_MONTHLYCHARGE.PNG

Churn	False	True
MonthlyCharge		
(79.769, 115.009]	755	35
(115.009, 150.039]	2477	338
(150.039, 185.07]	2356	693
(185.07, 220.1]	1009	608
(220.1, 255.13]	579	715
(255.13, 290.16]	174	261



FIGURES/D209_TASK1_Z1_FIG_FIG_9_2_TENURE.PNG

Churn	False	True
Tenure		
(0.929, 12.833]	1876	1924
(12.833, 24.667]	692	418
(24.667, 36.5]	70	28
(36.5, 48.333]	511	83
(48.333, 60.166]	1526	105
(60.166, 71.999]	2675	92

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/

```
<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.

Robinson, S. (2021, October). K-Nearest Neighbors Algorithm in Python and Scikit-Learn

https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/ The K-nearest neighbors (KNN) algorithm is a type of supervised machine learning algorithms. KNN is extremely easy to implement in its most basic form, and yet performs quite complex classification tasks. It is a lazy learning algorithm since it doesn't have a specialized training phase. Rather, it uses all of the data for training while classifying a new data point or instance. KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data. This is an extremely useful feature since most of the real world data doesn't really follow any theoretical assumption e.g. linear-separability, uniform distribution, etc.

In [ ]:

```

```