WGU D209 TASK 1 REV 1 - 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 18, 2021

Abstract. ___ .

Keywords. Data Mining. KNN. Classification.

PART I: RESEARCH QUESTION (I)

A1. PROPOSE ONE QUESTION

Propose **one** question relevant to a real-world organizational situation that you will answer using one of the following classification methods: • k-nearest neighbor (KNN) • Naive Bayes

Primary Question. The question has come up for a telecommunicatiosn 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

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 the new customer
import pandas as pd
newCustomer = pd.DataFrame([{'MonthlyCharge': 200.0 ,'Tenure': 24.0}])
```

PART II: METHOD JUSTIFICATION (II)

B1. EXPLAIN METHOD AND OUTCOMES

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

Explain KNN classification and expected outcomes_______

B2. SUMMARIZE ONE ASSUMPTION

Summarize **one** assumption of the chosen classification method.

Summarize one KNN assumption_______

B3. JUSTIFY PACKAGES

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

Data Manipulation. The pandas package enables common data analytics.

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)
```

Scientific Comptuing. Standard packages enable scientific computing and number crunching.

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
```

Modeling and Metrics. Standard packages that enable modeling and metrics..

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
```

Plotting. Matplotlib is a standard plotting library for Python that enables custom visualizations of the data.

In [5]:
```
# import and configure matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
plt.rc("font", size=14)
%matplotlib inline
```

Jupyter Notebook and IPython. These libraries enable the Jupyter notebook to work with HTML code. And, I am able to apply custom CSS styles to the notebook.

In [6]:
```
# import and configure IPython.display
from IPython.core.display import HTML
from IPython.display import Image
from IPython.display import display
```

In [7]:
```
HTML(open('custom.css', 'r').read()) # apply custom styles
```
Out[7]:

Helper Functions. In addition to above packages, the following functions are defined within the notebook to help with common tasks.

In [8]:
```
# helper function
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))
```

PART III: DATA PREPARATION (III)

C1. DESCRIBE ONE GOAL

Describe **one** data preprocessing goal relevant to the classification method from part A1.

Data 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.

A. **df** = the raw set of 10,000 customer records

B. **trainData** = a 70% subset of the raw data

C. **validData** = a 30% subset of the raw data

D. **churnNorm** = the standardized set of 10,000 customer records

E. **trainNorm** = a 70% subset of the standardized data. This will be created so that the index of records matches the index for **trainData**

F. **validNorm** = a 30% subset of the standardized data. This will be created so that the index of the records matches the index for **validData**

G. **X** the feature data from the standardized data (i.e. **MonthlyCharge**, and **Tenure**)

H. **y** = the target data from the standardized data (i.e. **Churn**)

C2. DESCRIBE VARIABLES

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.

For this analysis, I will consider two (2) features, **MonthlyCharge** and **Tenure**, and one (1) target, **Churn.**

Churn (Target). Whether the customer discontinued service within the last month (yes, no)

Tenure. Number of months the customer has stayed with the provider

MonthlyCharge. The amount charged to the customer monthly. This valu e reflects an average per customer.

Table III-1. SELECTED CUSTOMER DATA

In [9]:
```
# read subset of company's customer data from csv file
df = pd.read_csv('churn_clean.csv',
        usecols=['MonthlyCharge','Tenure','Churn'])
# 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')
#df = df.iloc[::100, :] # temp smaller data for trouble shooting...
df.reset_index(drop=True, inplace=True)
df['Number'] = df.index
columns=['MonthlyCharge','Tenure','Churn','Number']
df = df[columns]
df.head(10)
```

Out[9]:

	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
5	185.008	7.001	False	5
6	200.119	13.237	True	6
7	114.951	4.264	True	7
8	117.469	8.221	False	8
9	162.483	3.422	False	9

In [10]:
```
# 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 III-2. DESCRIBE NUMERIC FEATURE DATA

In [11]:
```python
# describe numeric feature data
df[['MonthlyCharge','Tenure']].describe()
```

Out[11]:

	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 III-3. CUSTOMER DATA INFO

In [12]:
```python
# use .info to show the structure of the data
df.info()
print('\nshape (rows,cols): {}'.format(df.shape))
```

```
<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

shape (rows,cols): (10000, 4)
```

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 STEPS

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 data`

Initially, 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 variables`

For 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 values`

The 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.

In [13]:
```
# provide copy of cleaned data
df.to_csv('d209_task1_c4_clean_data.csv', index=False, header=True)
print(df.columns.to_series().groupby(df.dtypes).groups)
```

```
{bool: ['Churn'], int64: ['Number'], float64: ['MonthlyCharge', 'Tenure']}
```

PART IV: ANALYSIS (IV)

D1. SPLIT DATA

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

In [14]:
```
# train test split raw data
trainData, validData = train_test_split(df, test_size=0.3, random_state=13)
trainData.to_csv('d209_task1_d1_trainData.csv', index=False, header=True)
validData.to_csv('d209_task1_d1_validData.csv', index=False, header=True)
```

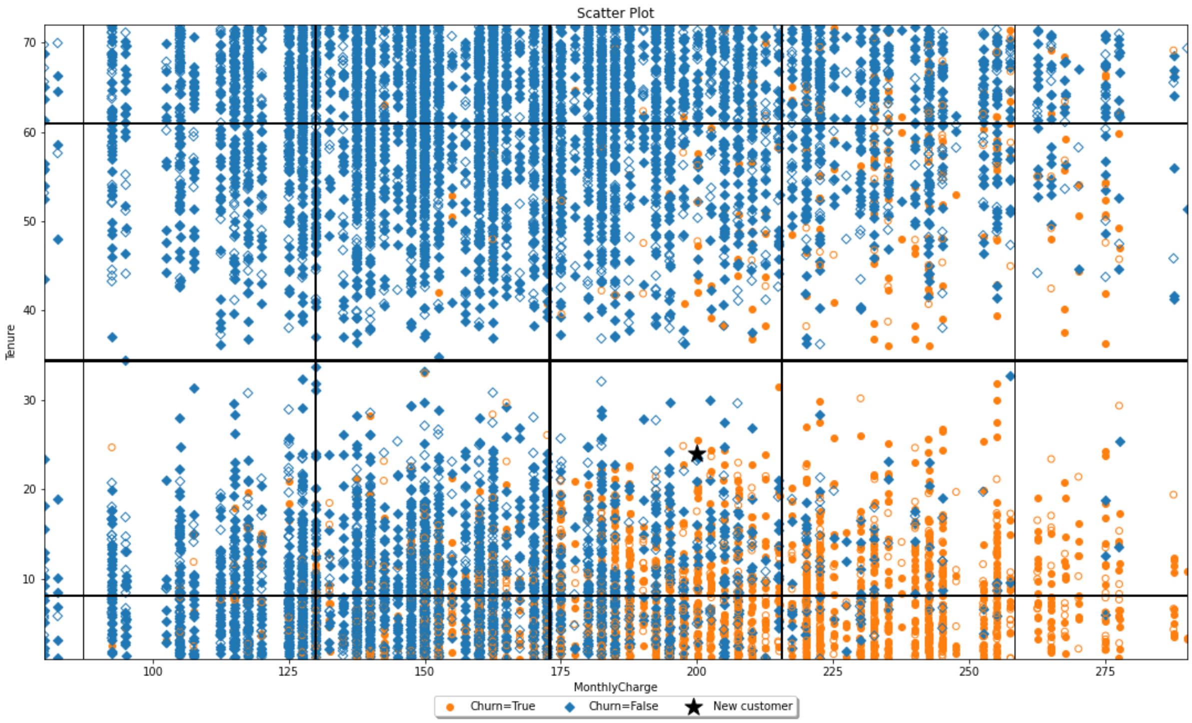
D2. DESCRIBE ANALYSIS

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 [15]:

```
# 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, -0.05),
          fancybox=True, shadow=True, ncol=5)

file = 'd209_task1_d2_fig_4_1_' + title.replace(' ','_') + '.png'
plt.gcf().text(0, -.2, file, fontsize=14)
fig.savefig(file, dpi=150, bbox_inches='tight')
plt.show()
```



d209_task1_d2_fig_4_1_Scatter_Plot.png

Figure IV-1. 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.

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. https://www.geeksforgeeks.org/data-normalization-with-pandas/

In [16]:
```
# use training data to learn the transformation
scaler = preprocessing.StandardScaler()
scaler.fit(trainData[['MonthlyCharge','Tenure']])
```

Out[16]:
```
StandardScaler()
```

In [17]:
```
# 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 IV-4. SCALED CUSTOMER DATA INFO

In [18]:
```
# describe scaled customer data
churnNorm.info()
print('\nshape (rows,cols): {}'.format(df.shape))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   zMonthlyCharge  10000 non-null  float64
 1   zTenure         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

shape (rows,cols): (10000, 4)
```

In [19]:
```
# scale new customer data
newCustomerNorm = pd.DataFrame(scaler.transform(newCustomer),
        columns=['zMonthlyCharge', 'zTenure'])
print(newCustomerNorm.round(2))
print(newCustomer.round(2))
```

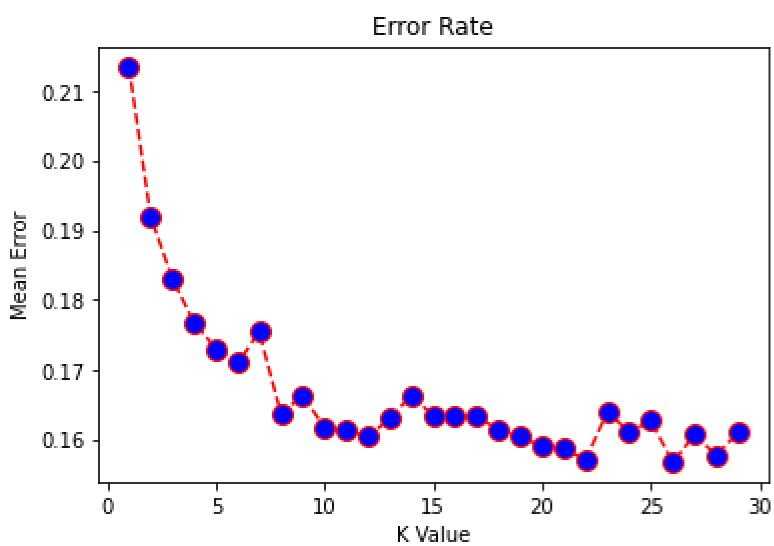
```
   zMonthlyCharge  zTenure
0            0.64     -0.4
   MonthlyCharge  Tenure
0          200.0    24.0
```

Table IV-5. CLOSEST TRAINING NEIGHBORS

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

	MonthlyCharge	Tenure	Churn	Number
2611	140.005	2.333	False	2611
4133	149.945	14.610	False	4133
1813	220.162	12.706	False	1813
4772	150.021	7.355	True	4772
5974	227.475	61.510	False	5974
4065	242.633	20.850	True	4065
814	95.006	4.455	False	814

```
In [21]: # 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')
file = 'd209_task1_d2_fig_4_2_' + title.replace(' ','_') + '.png'
plt.gcf().text(0, -.2, file, fontsize=14)
fig.savefig(file, dpi=150, bbox_inches='tight')
plt.show()
```



d209_task1_d2_fig_4_2_Error_Rate.png

Figure IV-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.

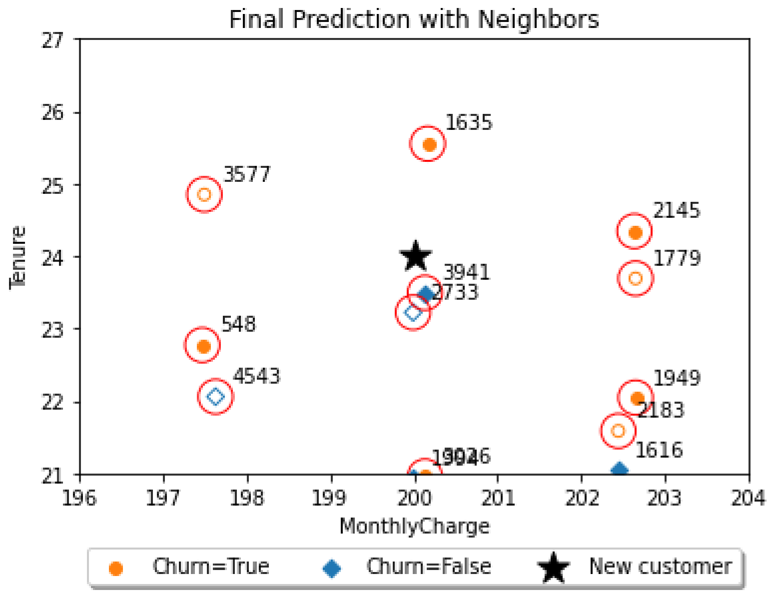
In [22]:

```
# 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 [39]:

```
# 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
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) - 3
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)
file = 'd209_task1_d2_fig_4_3_' + title.replace(' ','_') + '.png'
plt.gcf().text(0, -.2, file, 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(file, dpi=150, bbox_inches='tight')
plt.show()
#display(df_neighbors.iloc[:, 0:3] )
```



d209_task1_d2_fig_4_3_Final_Prediction_with_Neighbors.png

	MonthlyCharge	Tenure	Churn
3941	200.132	23.491	False
2733	199.990	23.220	False
1635	200.165	25.550	True
2145	202.636	24.343	True
1779	202.650	23.690	True
3577	197.494	24.848	True
548	197.470	22.770	True
4543	197.629	22.057	False
1949	202.650	22.044	True
2183	202.443	21.584	True
3026	200.132	20.956	True

Figure IV-3. FINAL CLASSIFICATION OF NEW CUSTOMER WITH NEIGHBORS (RED CIRLCES) USED TO CLASSIFY WITH THE NEIGHBOR DATA SORTED BY DISTANCE FROM NEW CUSTOMER.

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 CODE

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

All code and output is contained within this Jupyter notebook. The notebook file is called **D209_1_1.ipynb** and the associated PDF version is called **D209_1_1 - Jupyter Notebook.pdf.**

PART V: DATA SUMMARY AND IMPLICATIONS (V)

E1. EXPLAIN ACCURACY

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 V-6. CONFUSION AND CLASSIFICATION

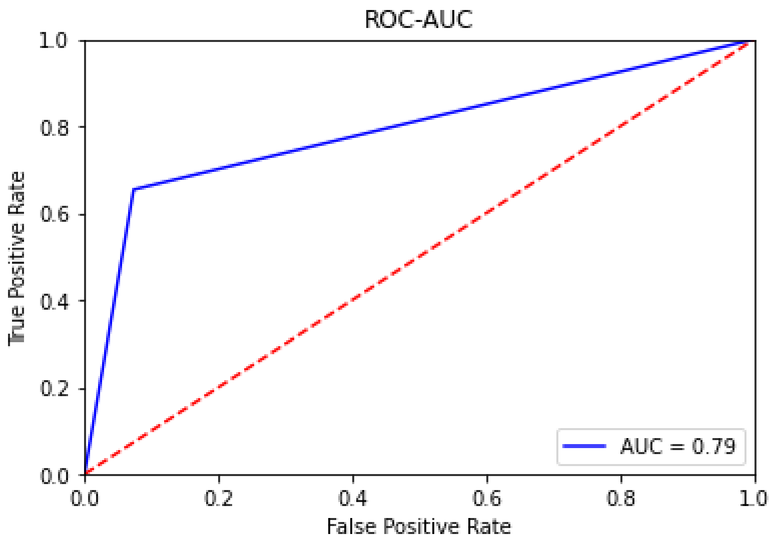
In [24]:
```
# confusion and classification report
classifier = KNeighborsClassifier(n_neighbors=11)
classifier.fit(X, y)
y_pred = classifier.predict(X)
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
[[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.

In [25]:
```
# 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 [26]:
```python
# 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')
file = 'd209_task1_e1_fig_5_4_' + title.replace(' ','_') + '.png'
plt.gcf().text(0, -.1, 'Area Under Curve (AUC): {:.2f}'.format(auc), fontsize=14)
plt.gcf().text(0, -.2, file, fontsize=14)
fig.savefig(file, dpi=150, bbox_inches='tight')
plt.show()
```



Area Under Curve (AUC): 0.79

d209_task1_e1_fig_5_4_ROC-AUC.png

Figure V-4. RECEIVER OPERATION CHARACTERISTIC (ROC)

E2. 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

Discuss **one** limitation of your data analysis.

Limitation. 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. RECOMMENDATIONS

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

Type *Markdown* and LaTeX: α^2

PART VI: DEMONSTRATION (VI)

F. PROVIDE PANAPTO VIDEO

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

G. WEB SOURCES

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

Type *Markdown* and LaTeX: α^2

H. ACKNOWLEDGE SOURCES

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

Type *Markdown* and LaTeX: α^2

List Files. List all of the files from the current working directory.
https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

In [27]:
```
!ls
```

```
churn_clean.csv
churn_clean_20.csv
churn_clean_21.csv
custom.css
d209_1_0.ipynb
D209_1_1.html
D209_1_1.ipynb
d209_task1_c4_clean_data.csv
d209_task1_d1_trainData.csv
d209_task1_d1_validData.csv
d209_task1_d2_fig_4_1_Scatter_Plot.png
d209_task1_d2_fig_4_2_Error_Rate.png
d209_task1_d2_fig_4_3_Final_Prediction_with_Neighbors.png
d209_task1_e1_fig_5_4_ROC-AUC.png
```

List Files and Sizes. List all of the files from the current working directory and show the filesize for each file.

In [28]:
```
!du -h *.*
```

```
1.4M    D209_1_1.html
616K    D209_1_1.ipynb
3.5M    churn_clean.csv
504K    churn_clean_20.csv
8.0K    churn_clean_21.csv
4.0K    custom.css
564K    d209_1_0.ipynb
300K    d209_task1_c4_clean_data.csv
208K    d209_task1_d1_trainData.csv
92K     d209_task1_d1_validData.csv
856K    d209_task1_d2_fig_4_1_Scatter_Plot.png
40K     d209_task1_d2_fig_4_2_Error_Rate.png
68K     d209_task1_d2_fig_4_3_Final_Prediction_with_Neighbors.png
52K     d209_task1_e1_fig_5_4_ROC-AUC.png
```

Open .SVG File. Open one of the .SVG file images.

In [29]:
```
!open file
```

```
'open' is not recognized as an internal or external command,
operable program or batch file.
```

In [ ]:
```
```