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 (1)

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
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

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]:
```
def plot_histogram(c) -> plt:
    """
    plot histogram grouped by target 'Churn'
    """
    df_yes = df[df.Churn==True][c]
    df_no = df[df.Churn==False][c]
    yes_mean = df_yes.mean();
    no_mean = df_no.mean();
    fig,ax = plt.subplots(figsize=(6,6))
    ax.hist([df_yes,df_no], bins=5, stacked=True)
    ax.legend(['Churn - Yes','Churn - No'])
    ymin, ymax = ax.get_ylim();
    xmin, xmax = ax.get_xlim()
    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'})
    plt.xlabel(c);
    plt.ylabel('# Churn');
    return plt
```

In [9]:
```
# plotDataset()
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 [10]:
```
COURSE = 'D209' # global
TASK = 'Task1' # global
FTYPE = 'PNG' # global

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

    return temp.replace(' ','_').upper()
```

In [11]:
```
f = getFilename('hello', 'd2','fig 4 1')
print(f)
```

```
D209_TASK1_D2_FIG_4_1_HELLO.PNG
```

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.

Selected Customer Data. Read in selected parts of the company's customer data.

In [12]:
```
# read subset of company's customer data from csv file
df = pd.read_csv('churn_clean.csv',
        usecols=['MonthlyCharge','Tenure','Churn'])
```

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)

In [13]:
```
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']
```

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 RAW CUSTOMER DATA. THIS IS THE PRIMARY DATASET IDENTIFIED AS 'DF' OF RAW DATA. NOTICE CHURN VALUES (YES AND NO). Utilize the USECOLS option to only include selected data.

Ref. (1) https://stackoverflow.com/questions/15017072/pandas-read-csv-and-filter-columns-with-usecols

In [14]:
```
# describe RAW CUSTOMER data
d = df
f = getFilename('DF RAW', 'c2','tab 3 1', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.info())
```

	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
```

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. If needed to debug, use the .iloc[::100, :] option to slice the data

Ref: (1) https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

In [15]:
```
# 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')

# use this line if you want to debug with a smaller set of data
#df = df.iloc[::100, :] # every 100th row...

# 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]
```

Table III-2. SELECTED CLEANED CUSTOMER DATA. THIS IS THE PRIMARY DATASET IDENTIFIED AS 'DF' OF CLEAN DATA. NOTICE CHURN VALUES ARE BOOL (TRUE AND FALSE) AND, COLUMNS ARE RE-ORDERED. USE .INFO() TO SHOW DATA STRUCTURE. USE .SHAPE TO COUNT ROWS AND COLS

In [16]:
```
# provide selected cleaned customer data
d = df
f = getFilename('DF CLEANED', 'c2','tab 3 2', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.info())
print('Shape (rows, cols): {}'.format(d.shape))
```

	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

```
<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)
```

Inital Variables. Classify each variable as continuous or categorical.

In [17]:
```
# 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-3. DESCRIBE NUMERIC FEATURE DATA. THESE ARE THE TRADITIONAL STATISTICS FOR THE NUMERICAL DATA

In [18]:
```
# describe numeric feature data
d = df[['MonthlyCharge','Tenure']].describe()
f = getFilename('DF STATS', 'c2','tab 3 3', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d)
```

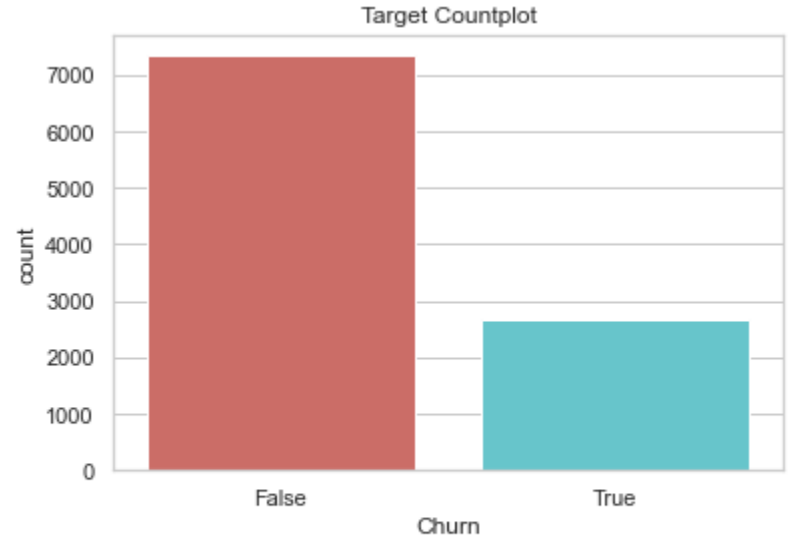
	MonthlyCharge	Tenure
count	10000.000	10000.000
mean	172.625	34.526
std	42.943	26.443
min	79.979	1.000
25%	139.979	7.918
50%	167.485	35.431
75%	200.735	61.480
max	290.160	71.999

Visualize Data. Create plot to visualize data.

In [19]:
```
# visualize target data
fig, ax = plt.subplots()
sns.countplot(x=target, data=df, palette='hls')
title = 'Target Countplot'
plt.title(title)
f = getFilename(title, 'c2','fig 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

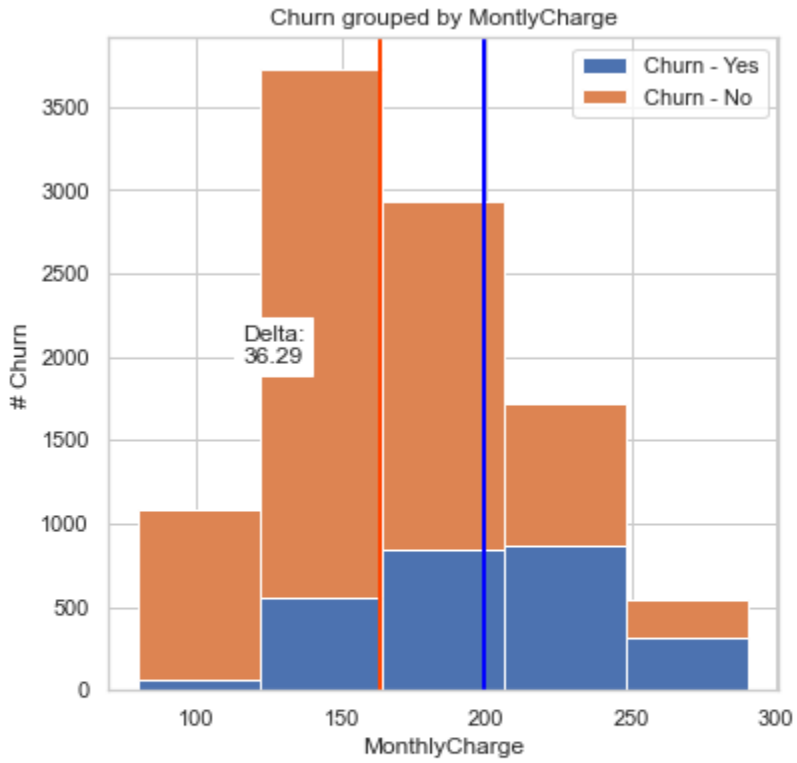
D209_TASK1_C2_FIG_3_1_TARGET_COUNTPLOT.PNG

Figure III-1. TARGET COUNT PLOT

In [20]:
```
# create histogram with target overlay
plt = plot_histogram('MonthlyCharge')
title = 'Churn grouped by MontlyCharge'
plt.title(title)
f = getFilename(title, 'c2','fig 3 2') # 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

D209_TASK1_C2_FIG_3_2_CHURN_GROUPED_BY_MONTLYCHARGE.PNG

Figure III-2. CHURN GROUPED BY MONTHLYCHARGE

```
In [21]: d = df.groupby(pd.cut(df['MonthlyCharge'], bins=5))
         d.Churn.count()
```

```
Out[21]: MonthlyCharge
         (79.769, 122.015]     1083
         (122.015, 164.051]    3729
         (164.051, 206.088]    2931
         (206.088, 248.124]    1717
         (248.124, 290.16]      540
         Name: Churn, dtype: int64
```

```
In [22]: d.groupby("Churn")["MonthlyCharge"].count()
```

```
---------------------------------------------
AttributeError Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_10756/844735128.py in <module>
----> 1 d.groupby("Churn")["MonthlyCharge"].count()

p:\code_wgu\5\v\lib\site-packages\pandas\core\groupby\groupby.py in __getattr__(self, att
r)
    908             return self[attr]
    909
--> 910         raise AttributeError(
    911             f"'{type(self).__name__}' object has no attribute '{attr}'"
    912         )

AttributeError: 'DataFrameGroupBy' object has no attribute 'groupby'
```

Figure III-3. CHURN GROUPED BY 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 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.

Table III-4. CLEAN DATA

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

PART IV: ANALYSIS (IV)

D1. SPLIT DATA

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

In [ ]:
```
# train test split raw data
trainData, validData = train_test_split(df, test_size=0.3, random_state=13)
```

Table IV-5. TRAINING DATA

In [ ]:
```
# provide TRAIN data
d = trainData
f = getFilename('TRAIN DATA', 'D1','tab 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))
```

Table IV-6. TEST DATA

In [ ]:
```
# provide TEST data
d = validData
f = getFilename('TEST DATA', 'D1','tab 4 6', 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))
```

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 [ ]:
```
# 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, facecolo

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

Figure IV-4. 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/

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

In [ ]:
```
# 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-7. SCALED CUSTOMER DATA INFO

In [ ]:
```
# provide SCALED data
d = churnNorm
f = getFilename('SCALED DATA', 'D2','tab 4 7', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
print(d.index)
```

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

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

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

In [ ]:
```
# 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 IV-8. K-NEAREST "TRAINING" NEIGHBORS

In [ ]:
```
# provide SCALED data
d = training_neighbors
f = getFilename('TRAINING NEIGHBORS', 'D2','tab 4 8', ftype='CSV')
d.to_csv(f, index=True, header=True)
display(d.head())
```

In [ ]:
```
# 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()
```

Figure IV-5. 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 [ ]:
```
# 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()
```

In [ ]:
```
# 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, facecolo

# 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)
f = getFilename(title, 'd2','fig 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()
```

Figure IV-6. 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 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-9. CONFUSION AND CLASSIFICATION

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

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, (2) 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, (4) https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

In [ ]:
```
# 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 [ ]:
```
# 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, 'e1','fig 5 5') # 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()
```

Figure V-7. 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.

List Files. List all of the files from the current working directory.
Ref: (1) https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

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

List Files and Sizes. List all of the files from the current working directory and show the filesize for each file.
Ref: (1) https://www.youtube.com/watch?v=C8MT-A7Mvk4&ab_channel=KimberlyFessel

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

PIP LIST. List of all installed PIP packages and the versions.
Ref: (1) https://pip.pypa.io/en/stable/cli/pip_list/

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

UPDATE PACKAGE WITHIN NOTEBOOK. 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 [ ]:
```
!python -m pip install -U scikit-learn
```

MERGE TWO DATAFRAMES. Code to merge two dataframes.
Ref: (1) https://stackoverflow.com/questions/26265819/how-to-merge-a-series-and-dataframe

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

LIST INDEX. 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 [ ]:
```
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/

In [ ]:
```
# 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 = " ")
```

PYTHON SUBPLOTS. Tutorial: Python Subplots

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

In [ ]:
```
#  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()
```

H. ACKNOWLEDGE SOURCES

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. Retrieved from: 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 [ ]: