D209 DATA MINING CLASSIFICATION ANALYSIS [Task 1]

Performance Assessment Task

WGU - MSDA

Data Mining Classification Analysis using Cleaned Churn Dataset

Richard Flores
Rflo147@wgu.edu
Student ID: 006771163

Table of Contents

Part I: Research Question

A1. Proposal of Question

As the telecommunications market becomes increasingly competitive with new and improved technologies including free applications like META (Facebook) messenger, Telegram, and TikTok the need for customer retention is becoming critically important.

The question answered in this research project is:

How do we identify customers at risk of churn and what telecom services or features are correlated?

I will be using the K-Nearest Neighbor classification method.

A2. Defined Goal

The goal of the research question is to provide stakeholders direct and actionable insight to create a plan for operations personnel, officers, and managers to increase customer satisfaction through targeted services observed in the dataset and to reduce customer churn and protect long-term profits.

Part II: Method Justification

B1. Explanation of Classification Method

In this project I will be using K-Nearest Neighbor (KNN) classification. KNN is a simple, supervised machine learning algorithm which excels in solving classification problems. The KNN algorithm functions by finding distances, specifically distances between a query and all examples in a dataset, and then selecting a pre-determined number of examples "K" nearest the query and then chooses the most frequent label or classification (Vatsal 2021).

The expected outcome from KNN is experimental data points will be classified based on ranking with their nearest neighbor.

B2. Summary of Method Assumption

The method assumption is that the majority of calculations will show that similar data points are close to each other by defining a specific Euclidean distance calculated with:

$$\textbf{distance} = \sqrt{} \, a^2 + b^2$$

By using K-Nearest Neighbor algorithm the function finds classification of points. The model assumes two groups and classifies points into Group 0 or else into Group 1 (Kohli 2019).

B3. Packages or Libraries List

For this Data Mining analysis, I will be using the Python language and the following packages or libraries:

Data Science Libraries

- NumPy
- Pandas

Visualization Libraries

- Seaborn
- Matplotlib

Predictive Analysis

- Scikit-Learn

Justification for libraries and packages in support of the Data Mining Analysis

NumPy – NumPy is integral for performing mathematical and logical operations on arrays. It provides many of the functions needed to manipulate n-arrays and matrices in Python. This includes how to create NumPy arrays, broadcasting, accessing values, and managing arrays.

Pandas – Pandas is used to infer and analyze data in Python. Pandas is used for data cleanup, transformation, management and analysis of the cleaned churn dataset.

Seaborn – Seaborn takes each data frame or array that contains information and performs internal functions necessary to integrate semantic mapping and statistics to turn the data into visual representations.

Matplotlib – Matplotlib is a plotting library for creating 2D plots in Python. It consists of a set of graphing plots such as line plots, bar plots, frequency distribution plots, and histograms and can display different types of data.

Scikit-Learn - Scikit-learn is a library that provides many supervised and unsupervised learning algorithms in Python. Functions provided by Scikit-learn include Regression, linear and logistic regression as well as classification including K-Nearest Neighbors.

Part III: Data Preparation

C1. Data Preprocessing

As with the previous Multiple and Logistic regression analysis, a preprocessing data goal is to convert binary responses in the dataset i.e. ‘Yes’ or ‘No’ into dummy variables using numerical ‘1’ or ‘0’ variables in order to enable statistical analysis.

For example, converting customer responses if they have “TechSupport” from ‘No’ to ‘0’ and changing ‘Yes’ to ‘1’.

C2. Data Set Variables

This analysis will use the following 9 continuous variables and 13 categorical variables.

Continuous variables include:

- Bandwidth_GB_Year
- Children
- Contacts
- Email
- Income
- MonthlyCharge
- Outage_sec_perweek
- Tenure
- Yearly_equip_failure

Categorical variables include:

- Contract
- DeviceProtection
- InternetService
- Multiple
- OnlineBackup
- OnlineSecurity
- Phone
- Port_modem
- StreamingMovies
- StreamingTV
- Tablet
- TechSupport
- Techie

In addition, the customer survey responses represent ordinal predictors, listed as follows:

Item1 - Timely response
Item2 - Timely fixes
Item3 - Timely replacements
Item4 - Reliability
Item5 - Options
Item6 - Respectful Response
Item7 - Courteous Exchange
Item8 – Evidence of Active Listening

C3. Steps for Analysis

▪ Import the 'clean_churn' dataset into a Pandas dataframe for analysis.
▪ Rename features in the survey responses to better describe the items.
▪ Describe the various features and data to prepare relevant items.
▪ Create a view of the summary statistics.
▪ After review, remove features that are not relevant to analyzing the target variable.
▪ Review record data to check for anomalies, outliers, missing data and other data that could become obstacles in the analysis.
▪ Utilize dummy variables in order to numerically analyze data by changing "Yes/No" responses to binary "1/0" responses.
▪ Export manipulated Dataframe to .CSV for analysis in K-Nearest Neighbor (KNN) model.

```
# Standard library imports, and Visualization, Statistics, SciKit libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import sklearn
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Ignore Warning messages
import warnings
warnings.filterwarnings('ignore')

import matplotlib as mpl
COLOR = 'white'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR
```

```
# Load churn dataset into a Pandas dataframe
churn_df = pd.read_csv('churn_clean.csv', index_col=0)
```

```
# List columns in the dataframe
churn_df.columns
```

```
Index(['Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip',
       'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children',
       'Age', 'Income', 'Marital', 'Gender', 'Churn', 'Outage_sec_perweek',
       'Email', 'Contacts', 'Yearly_equip_failure', 'Techie', 'Contract',
       'Port_modem', 'Tablet', 'InternetService', 'Phone', 'Multiple',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'PaymentMethod',
       'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
# Verify the number of records and columns in the dataset
churn_df.shape
```

```
(10000, 49)
```

(DataCamp 2021)

```
# Verify headers of imported dataset
churn_df.head()
```

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	...	MonthlyChar
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	...	172.4555
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	...	242.6325
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	...	159.9475
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	...	119.9568
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	...	149.9483

5 rows × 49 columns

```
# Verify dataset info
churn_df.info
```

```
<bound method DataFrame.info of           Customer_id                           Interaction  \
CaseOrder
1             K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
2             S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
3             K191035  344d114c-3736-4be5-98f7-c72c281e2d35
4              D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
5             K662701  68a861fd-0d20-4e51-a587-8a90407ee574
...               ...                                   ...
9996          M324793  45deb5a2-ae04-4518-bf0b-c82db8dbe4a4
9997          D861732  6e96b921-0c09-4993-bbda-a1ac6411061a
9998          I243405  e8307ddf-9a01-4fff-bc59-4742e03fd24f
9999          I641617  3775ccfc-0052-4107-81ae-9657f81ecdf3
10000          T38070  9de5fb6e-bd33-4995-aec8-f01d0172a499

                                        UID          City State  \
CaseOrder
1          e885b299883d4f9fb18e39c75155d990   Point Baker    AK
2          f2de8bef964785f41a2959829830fb8a   West Branch    MI
3          f1784cfa9f6d92ae816197eb175d3c71       Yamhill    OR
4          dc8a365077241bb5cd5ccd305136b05e       Del Mar    CA
5          aabb64a116e83fdc4befc1fbab1663f9     Needville    TX
...                                     ...           ...   ...
9996       9499fb4de537af195d16d046b79fd20a   Mount Holly    VT
9997       c09a841117fa81b5c8e19afec2760104   Clarksville    TN
9998       9c41f212d1e04dca84445019bbc9b41c      Mobeetie    TX
9999       3e1f269b40c235a1038863ecf6b7a0df    Carrollton    GA
10000      0ea683a03a3cd544aefe8388aab16176   Clarkesville    GA

                          County    Zip       Lat        Lng  Population  ...  \
CaseOrder                                                                 ...
1          Prince of Wales-Hyder  99927  56.25100 -133.37571          38  ...
2                         Ogemaw  48661  44.32893  -84.24080       10446  ...
3                        Yamhill  97148  45.35589 -123.24657        3735  ...
4                      San Diego  92014  32.96687 -117.24798       13863  ...
5                      Fort Bend  77461  29.38012  -95.80673       11352  ...
...                          ...    ...       ...        ...         ...  ...
9996                     Rutland   5758  43.43391  -72.78734         640  ...
9997                  Montgomery  37042  36.56907  -87.41694       77168  ...
9998                     Wheeler  79061  35.52039 -100.44180         406  ...
9999                     Carroll  30117  33.58016  -85.13241       35575  ...
10000                  Habersham  30523  34.70783  -83.53648       12230  ...
```

(DataCamp 2021)

```
           MonthlyCharge Bandwidth_GB_Year Item1  Item2  Item3  Item4 Item5  \
CaseOrder
1             172.455519        904.536110     5      5      5      3     4
2             242.632554        800.982766     3      4      3      3     4
3             159.947583       2054.706961     4      4      2      4     4
4             119.956840       2164.579412     4      4      4      2     5
5             149.948316        271.493436     4      4      4      3     4
...                  ...               ...   ...    ...    ...    ...   ...
9996          159.979400       6511.252601     3      2      3      3     4
9997          207.481100       5695.951810     4      5      5      4     4
9998          169.974100       4159.305799     4      4      4      4     4
9999          252.624000       6468.456752     4      4      6      4     3
10000         217.484000       5857.586167     2      2      3      3     3

           Item6 Item7  Item8
CaseOrder
1              4     3      4
2              3     4      4
3              3     3      3
4              4     3      3
5              4     4      5
...          ...   ...    ...
9996           3     2      3
9997           5     2      5
9998           4     4      5
9999           3     5      4
10000          3     4      1

[10000 rows x 49 columns]>
```

```
# Describe Churn dataset
churn_df.describe()
```

	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	..
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	..
mean	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	10.001848	12.016000	0.994200	..
std	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.025898	0.988466	..
min	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	0.099747	1.000000	0.000000	..
25%	26292.500000	35.341828	-97.082812	738.000000	0.0000	35.000000	19224.717500	8.018214	10.000000	0.000000	..
50%	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.000000	1.000000	..
75%	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.000000	2.000000	..
max	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.000000	7.000000	..

8 rows × 22 columns

```
# List features available in the dataset
churn_df.dtypes
```

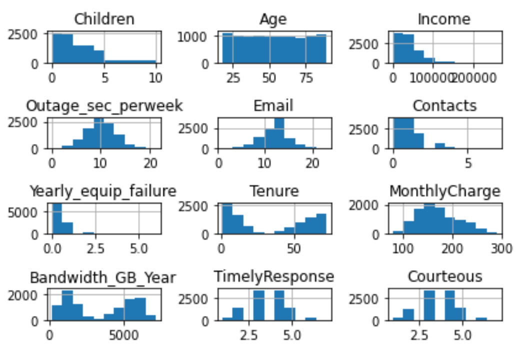
```
Customer_id           object
Interaction           object
UID                   object
City                  object
State                 object
County                object
Zip                    int64
Lat                  float64
Lng                  float64
Population             int64
Area                  object
TimeZone              object
Job                   object
Children               int64
Age                    int64
Income               float64
Marital               object
Gender                object
Churn                 object
```

(DataCamp 2021)

```
Outage_sec_perweek      float64
Email                     int64
Contacts                  int64
Yearly_equip_failure      int64
Techie                   object
Contract                 object
Port_modem               object
Tablet                   object
InternetService          object
Phone                    object
Multiple                 object
OnlineSecurity           object
OnlineBackup             object
DeviceProtection         object
TechSupport              object
StreamingTV              object
StreamingMovies          object
PaperlessBilling         object
PaymentMethod            object
Tenure                  float64
MonthlyCharge           float64
Bandwidth_GB_Year       float64
Item1                     int64
Item2                     int64
Item3                     int64
Item4                     int64
Item5                     int64
Item6                     int64
Item7                     int64
Item8                     int64
dtype: object
```

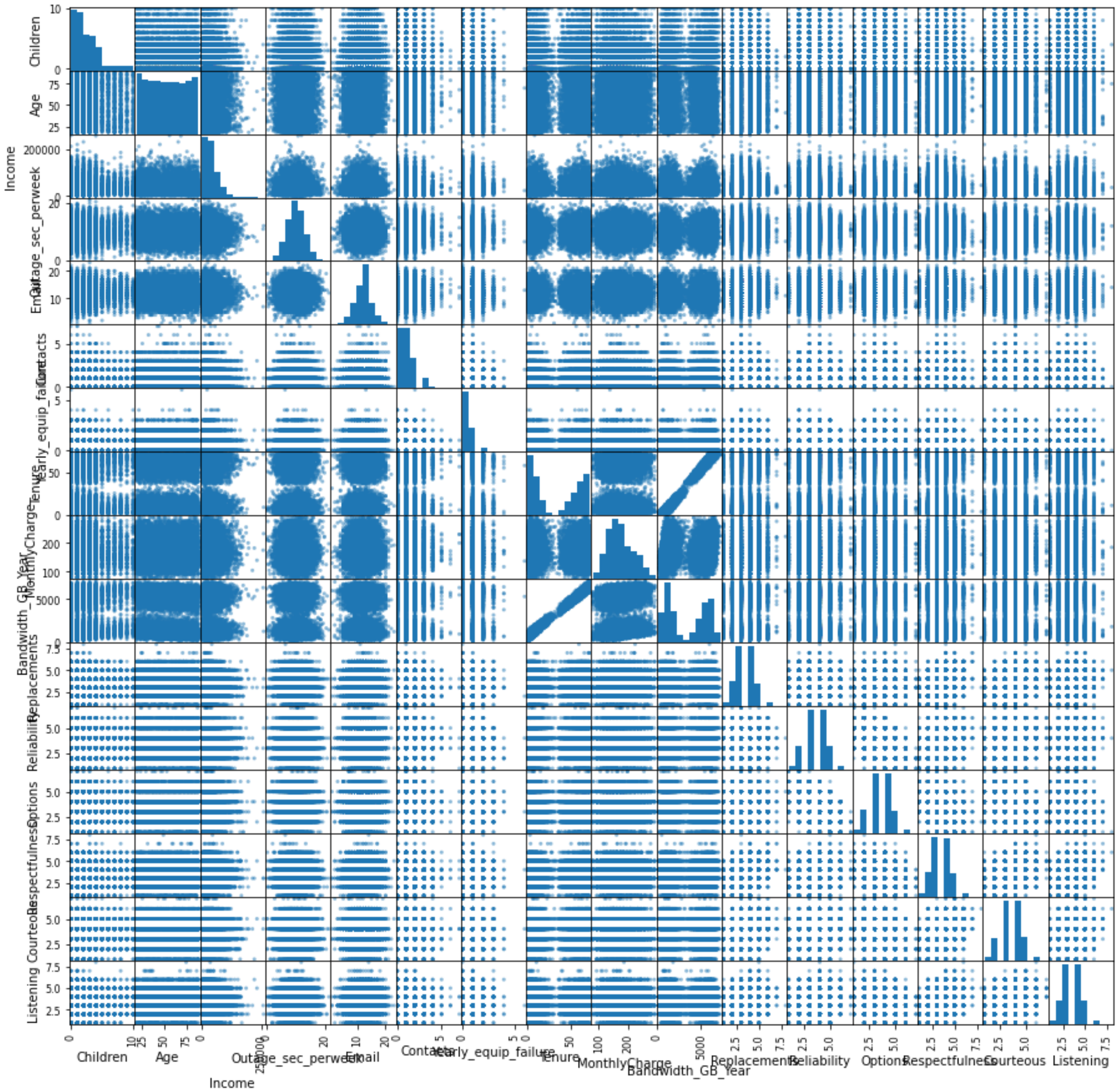
```
# Rename 8 customer survey features to represent descriptions for clarity
churn_df.rename(columns = {'Item1':'TimelyResponse',
                   'Item2':'Fixes',
                    'Item3':'Replacements',
                    'Item4':'Reliability',
                    'Item5':'Options',
                    'Item6':'Respectfulness',
                    'Item7':'Courteous',
                    'Item8':'Listening'},
          inplace=True)
```

```
# Display histograms of continuous & categorical variables
churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure',
          'MonthlyCharge', 'Bandwidth_GB_Year', 'TimelyResponse', 'Courteous']].hist()
plt.savefig('classification_pyplot.jpg')
plt.tight_layout()
```



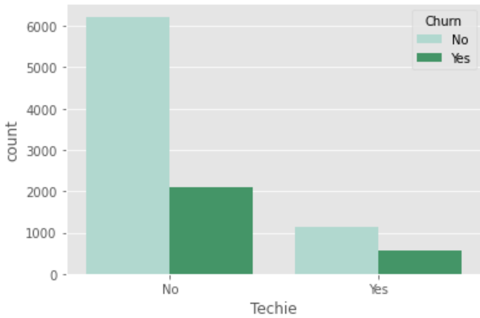
(DataCamp 2021)

```python
# Scatter matrixes of numeric variables for a broad overview of possible relationships.
churn_numeric = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek',
                          'Email', 'Contacts','Yearly_equip_failure', 'Tenure',
                          'MonthlyCharge', 'Bandwidth_GB_Year', 'Replacements',
                          'Reliability', 'Options', 'Respectfulness', 'Courteous',
                          'Listening']]

pd.plotting.scatter_matrix(churn_numeric, figsize = [15, 15]);
```

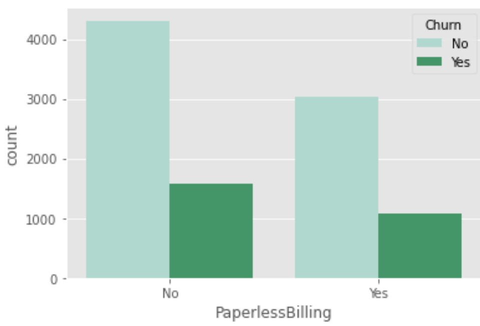


(DataCamp 2021)

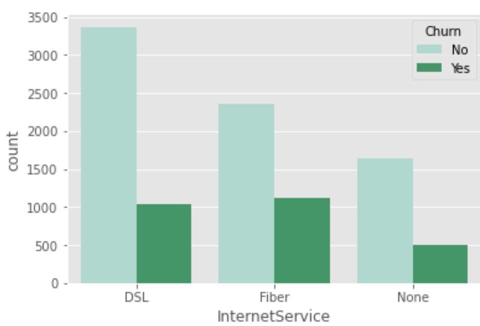
```
# Enable ggplot
plt.style.use('ggplot')

# Countplot to show relationship of binary feature techie and churn
plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```
# Countplot to show relationship of binary feature PaperlessBilling and churn
plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```
# Countplot to show relationship of binary feature InternetService and churn
plt.figure()
sns.countplot(x='InternetService', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1,2], ['DSL', 'Fiber', 'None'])
plt.show()
```



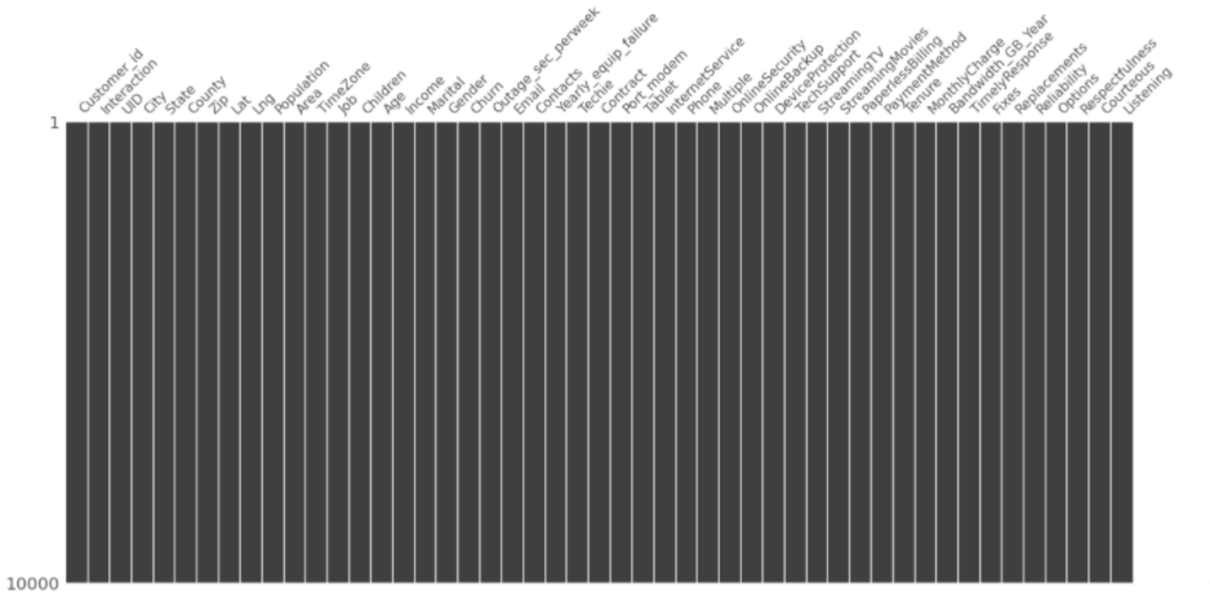
(DataCamp 2021)

```python
# Verify missing data points
data_nulls = churn_df.isnull().sum()
print(data_nulls)
```

```
Customer_id             0
Interaction             0
UID                     0
City                    0
State                   0
County                  0
Zip                     0
Lat                     0
Lng                     0
Population              0
Area                    0
TimeZone                0
Job                     0
Children                0
Age                     0
Income                  0
Marital                 0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
TimelyResponse          0
Fixes                   0
Replacements            0
Reliability             0
Options                 0
Respectfulness          0
Courteous               0
Listening               0
dtype: int64
```

(DataCamp 2021)

```python
# Visualize missing values in dataset using missingno

!pip install missingno
import missingno as msno

# Display matrix to visualize any missing values
msno.matrix(churn_df);
```




```python
# Convert all "Yes/No" data into binary "1/0" representation
churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']]
churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in churn_df['Contract']]
churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn_df['DeviceProtection']]
churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender']]
churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in churn_df['InternetService']]
churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn_df['Multiple']]
churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineBackup']]
churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineSecurity']]
churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn_df['PaperlessBilling']]
churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in churn_df['Port_modem']]
churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingTV']]
churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn_df['TechSupport']]
churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
churn_df['StreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingMovies']]
```

```python
# Remove redundant 'yes/no' features from dataframe
churn_df = churn_df.drop(columns=['Gender', 'Churn', 'Techie', 'Contract', 'Port_modem', 'Tablet',
                                  'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
                                  'OnlineBackup', 'DeviceProtection', 'TechSupport',
                                  'StreamingTV', 'StreamingMovies', 'PaperlessBilling'])
```

(DataCamp 2021)

```
# Remove features not relevant to the proposed analysis question
churn_df = churn_df.drop(columns=['Customer_id', 'Interaction', 'UID',
                                  'City', 'State', 'County', 'Zip', 'Lat', 'Lng',
                                  'Area', 'TimeZone', 'Job', 'Marital', 'PaymentMethod'])
churn_df.head()
```

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	...	DummyMultiple	Du
CaseOrder													
1	38	0	68	28561.99	7.978323	10	0	1	6.795513	172.455519	...	0	
2	10446	1	27	21704.77	11.699080	12	0	1	1.156681	242.632554	...	1	
3	3735	4	50	9609.57	10.752800	9	0	1	15.754144	159.947583	...	1	
4	13863	1	48	18925.23	14.913540	15	2	0	17.087227	119.956840	...	0	
5	11352	0	83	40074.19	8.147417	16	2	1	1.670972	149.948316	...	0	

5 rows × 34 columns

```
# Display features in churn dataframe
features = (list(churn_df.columns[:-1]))
print('Analysis Features: \n', features)
```

```
Analysis Features:
 ['CaseOrder', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'Month
lyCharge', 'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options', 'Respectfulness', 'Courteo
us', 'Listening', 'DummyGender', 'DummyTechie', 'DummyContract', 'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'Dum
myPhone', 'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection', 'DummyTechSupport', 'DummyStrea
mingTV', 'DummyPaperlessBilling']
```

(DataCamp 2021)

C4. Cleaned Data Set

```
# Extract cleaned dataset to CSV
churn_df.to_csv('churn_classification.csv')
```

Part IV: Analysis

```
# Import SKLearn KNN model, Classifier, accuracy
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split
```

```
# Import previously prepared dataset
churn_df = pd.read_csv('churn_classification.csv')

# Set DummyChurn predictor features & target
X = churn_df.drop('DummyChurn', axis=1).values
y = churn_df['DummyChurn'].values
```

D1. Splitting the Data

```
# Enable seed to objectionally verify results later and create training/test sets
SEED = 1

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = SEED)
```

```
# Load KNN model, fit data, and outcomes
knn = KNeighborsClassifier(n_neighbors = 7)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

```
# Extract Test Set to CSV
print(y_test)
pd.DataFrame(y_test).to_csv("test_set.csv")
```

```
[0 0 1 ... 0 1 1]
```

```
# Extract Training Set to CSV
print(y_pred)
pd.DataFrame(y_pred).to_csv("training_set.csv")
```

```
[0 0 0 ... 0 1 0]
```

D2. Output and Intermediate Calculations

```
# KNN model accuracy score
print('KNN model accuracy: ', accuracy_score(y_test, y_pred))
```

```
KNN model accuracy:  0.7145
```

```
# Display classification metrics
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81      1442
           1       0.49      0.40      0.44       558

    accuracy                           0.71      2000
   macro avg       0.63      0.62      0.62      2000
weighted avg       0.70      0.71      0.71      2000
```

(DataCamp 2021)

D3. Code Execution

```
# From SKLearn import scaler, pipeline, and accuracy
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# Define steps
steps = [('scaler', StandardScaler()),
        ('knn', KNeighborsClassifier())]

# Create pipeline instance
pipeline = Pipeline(steps)

# Separate dataframe
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(X, y, test_size = 0.2, random_state = SEED)

# Fit pipeline
knn_scaled = pipeline.fit(X_train_scaled, y_train_scaled)

# Pipepline Prediction
y_pred_scaled = pipeline.predict(X_test_scaled)
```

```
# Updated KNN model accuracy score
print('Updated KNN model accuracy: {:0.3f}'.format(accuracy_score(y_test_scaled, y_pred_scaled)))
```

```
Updated KNN model accuracy: 0.790
```

```
# Display classification report
print(classification_report(y_test_scaled, y_pred_scaled))
```

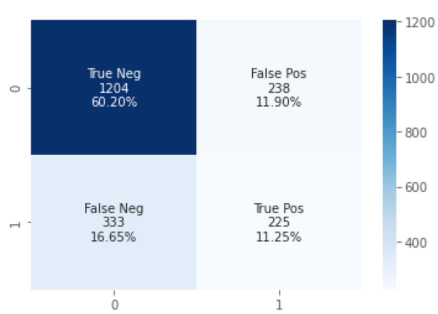
```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1442
           1       0.64      0.56      0.60       558

    accuracy                           0.79      2000
   macro avg       0.74      0.72      0.73      2000
weighted avg       0.78      0.79      0.79      2000
```

```
# Import SKLearn confusion matrix library and print results
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(y_test, y_pred)
print(cf_matrix)
```

```
[[1204  238]
 [ 333  225]]
```

```
# Display confusion matrix
group_names = ['True Neg', 'False Pos', 'False Neg', 'True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues')
```

```
<AxesSubplot:>
```



(DataCamp 2021)

Part V: Data Summary and Implications

E1. Accuracy and AUC

```
# Import SKLearn GridSearchCV Library in order to use cross validation
from sklearn.model_selection import GridSearchCV

# New parameter grid
param_grid = {'n_neighbors': np.arange(1, 50)}

# New KNN instance for cross validation
knn = KNeighborsClassifier()

# Use GridSearch library for calculation
knn_cv = GridSearchCV(knn , param_grid, cv=5)

# Setup model using fit
knn_cv.fit(X_train, y_train)

# Display parameters
print('KNN model Parameters: {}'.format(knn_cv.best_params_))
```

```
KNN model Parameters: {'n_neighbors': 6}
```

```
# KNN Model optimal score
print('KNN model Optimal Score: {:.3f}'.format(knn_cv.best_score_))
```

```
KNN model Optimal Score: 0.735
```

```
# Import Roc_auc_score to calculate integral
from sklearn.metrics import roc_auc_score

# Fit model and calculate probabilities
knn_cv.fit(X, y)

y_pred_prob = knn_cv.predict_proba(X_test)[:,1]

# Print ROC Auc score
print("Area Under Curve validation: {:.4f}".format(roc_auc_score(y_test, y_pred_prob)))
```

```
Area Under Curve validation: 0.7959
```

```
# Cross Validate AUC Score
cv_auc = cross_val_score(knn_cv, X, y, cv=5, scoring='roc_auc')

# Display AUC scores
print("AUC scores using 5x cross-validation: {}".format(cv_auc))
```

```
AUC scores using 5x cross-validation: [0.68120909 0.17406045 0.96370684 0.96560711 0.58834745]
```

From our model comparison, the initial Accuracy for Class 0 scored 0.7145 and initial Precision scored 0.78. After scaling the model improved Class 0 Accuracy by 0.08 for a score of 0.790 and Precision by 0.06 for a score of 0.84. The AUC or Area Under the Curve calculation made an excellent scoring at 0.795.

The accuracy score shows the KNN classifier is able to reliably distinguish points in the data. While a score of 1 is ideal, the model can be improved upon using a larger dataset and more powerful processors (Vatsal 2021).

An Area Under the Curve or AUC score of 0.7959 also proves a reliable result as scoring for AUC ranges from 0 to 1. An AUC score of 0.79 falls into the category of moderate accuracy and in the future, we should aim to achieve high accuracy above 0.85.

E2. Results and Implications

From our machine learning analysis, we can breakdown the results into seven categories: Precision, Recall, F1 Score, Support, Accuracy, Macro Average, and Weighted Average (Harrison 2019).

Precision – indicates the proportion of positive identifications which were actually correct. A perfect model with no false positives has a score of 1.0. From our analysis we scored 0.78 in precision, which while not perfect, is a great score for reliable metrics.

Recall – indicates the proportion of actual positives correctly classified. Like precision, a model with a score of 1.0 is ideal. In our analysis Recall scored 0.83 which is close to a perfect score.

F1 Score – combines precision and recall. A perfect model scores an F1 Score of 1.0. In our analysis given the precision score of 0.78 and 0.83 the F1 Score averages to 0.81 which shows the analysis is reliable.

Support – shows the number of samples on which each metric is calculated. For our analysis Class 0 has 1442 samples and Class 1 558 samples for a total of 2000.

Accuracy – shows the accuracy of the model with a perfect model having a score equal to 1.0. Our analysis has an accuracy score of 0.71.

Macro Average – the average between precision, recall, and F1 score between classes. In our analysis the macro average scored 0.62 and 0.63.

Weighted Average – the weighted average of precision, recall, and F1. Calculated with respect to samples in each class. With a total of 2000 samples for Class 0 and 1, the weighted average scores 0.70 and 0.71.

The scores from our data mining analysis were successful and can be reliably used to give insight. The implication, or conclusion, we can draw from the results are as follows.

A precision score, measuring accuracy of positive predictions, of 0.84 shows a great deal of True Positive cases showing the model is reliable, and is calculated by measuring True Positive (TP) and False Positive (FP) using the formula:
Precision = TP/ (TP + FP)

In our analysis Recall scored a 0.88 showing a good fraction of True Positives correctly identified. Recall is calculated using True Positives (TP) and False Negatives (FN) using the formula:
Recall = TP/ (TP + FN)

F1 score is a weighted harmonic mean of precision and recall used to compare classifier models as opposed to global accuracy. A weighted average from our data mining analysis of 0.79 is a great score for a reliable model. F1 is calculated with the formula:

F1 = 2 * (Recall * Precision) / (Recall + Precision)

A deeper look into the underlying mechanics of Precision, Recall, and F1 Score show that the data mining analysis is a reliable model as it maximizes True Negatives and True Positives while minimizing False Negatives and False Positives (Kohli 2019).

E3. Limitation

One limitation of the data analysis is the choice of “K” value for the model. Choosing an incorrect K-value can cause the model to be under or over fit. If the K-value chosen is small this will cause noise in the data, and conversely if the K-value is too large the expense for computation becomes very large (Harrison 2019).

In this nearest neighbor data analysis, we have K = 7 which performs well by not under or over fitting the data. From our calculations the Precision score achieved was 0.84 which is great, however decreasing the K-value would also reduce precision. Increasing the K-value may also increase precision, however with computation time currently taking 30 minutes it may not be justified to spend increased time calculating for a minor or negligible increase in precision.

E4. Course of Action

From an initial exploratory data analysis, we can draw conclusion that the majority of subscribers are not tech savvy and only a small minority of customers who churn are tech savvy. This observation correlates to the fact that while receiving a bill via email is available, many customers who continue to pay for services prefer receiving the bill in traditional paper format. We also see that customers who subscribe to internet service, either DSL or Fiber, are less likely to leave the company.

From our machine learning model, our accuracy scored an impressive 0.79 and precision 0.84. The model can be improved by increasing the amount of data available in the dataset as well as increasing the K-value and running the calculations on an increasingly powerful computer. However, this model still produces meaningful metrics for providing insight.

Based on the information available, we strongly recommend improving the data mining model with an improved dataset and access to computers capable of complex computations.

For stakeholders and decision makers, the data shows that although the world’s technology is becoming increasingly complex and innovative, most customers in the telecommunications company are not tech savvy and prefer simpler communication methods such as having a paper bill versus using an app or email. The best recommendation to reduce churn for this particular company is ensuring that customer issues are resolved quickly and that the equipment provided is reliable and of quality, with fewer equipment replacements. Additionally, the data shows that customers with more services added to their accounts such as tech support and online backups are more likely to stay with the company so these optional services should be advertised and promoted in future marketing campaigns.

Part VI: Demonstration

F. Panopto Recording

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a606f43d-2426-4683-81cf-ae2500975dcd

G. Sources for Third-Party Code

KNN classification using Sklearn Python. DataCamp Community. (n.d.). Retrieved January 23, 2022, from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

K-Nearest Neighbors: Fit. Python. (n.d.). Retrieved January 23, 2022, from https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/classification?ex=7

Supervised learning with scikit-learn course. DataCamp. (n.d.). Retrieved January 23, 2022, from https://www.datacamp.com/courses/supervised-learning-with-scikit-learn/

I. Sources

Vatsal. (2021, November 3). *K nearest neighbours explained*. Medium. Retrieved January 23, 2022, from https://towardsdatascience.com/k-nearest-neighbours-explained-7c49853633b6

Harrison, O. (2019, July 14). *Machine learning basics with the K-nearest neighbors algorithm*. Medium. Retrieved January 23, 2022, from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Kohli, S. (2019, November 18). *Understanding a classification report for your machine learning model*. Medium. Retrieved January 23, 2022, from https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397