D209 DATA MINING PREDICTIVE ANALYSIS [Task 2]

Performance Assessment Task

WGU - MSDA

Data Mining Predictive 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 Decision Tree Prediction Method in this analysis.

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 Prediction Method

One of the various supervised Machine Learning algorithms available in Data Science is a Decision Tree. A Decision Tree models logical human thinking in an easy-to-read format to aid in making a decision. A Decision Tree is similar looking to a tree with branches as they contain nodes, edges, roots, and leaves.

Nodes are areas that split according to attributes or features of the dataset. Edges direct the resulting outcomes of a split to underlying nodes. A Root is the first node and contains the first split. Leaves describe terminal nodes and predict the outcome of the decision tree (Vidhya 2021).

B2. Summary of Method Assumption

This analysis assumes the Decision Tree method will begin with the dataset as the root, before using features to create branches. The features should be categorical in order to make best use of the decision tree. Continuous features should be quantized into discrete values for the analysis.

The underlying algorithm on which this decision tree is modeled uses Attribute Selection Measure (ASM) and the two techniques Information Gain and Gini Index (Vidhya 2021).

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 Decision Tree Prediction 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)

```python
# 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

```python
# 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
```

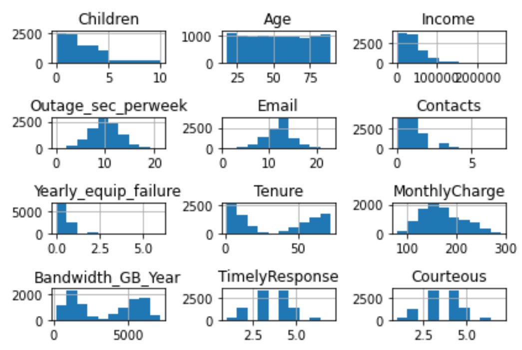
```
Customer_id              object
Interaction              object
UID                      object
City                     object
State                    object
County                   object
Zip                       int64
Lat                     float64
Lng                     float64
Population                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
```

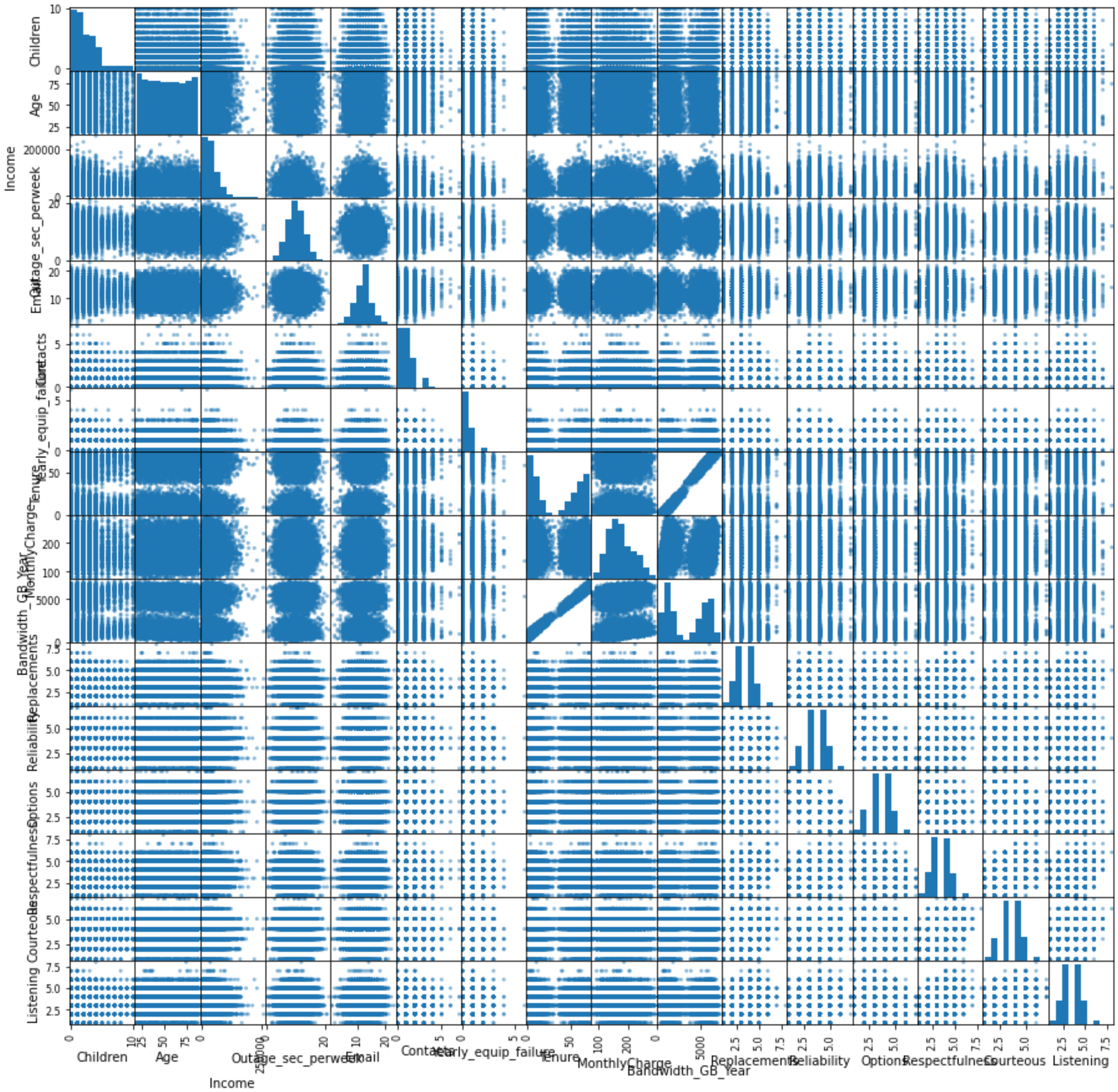
```
# Rename 8 customer survey features to represent descriptions for clarity
churn_df.rename(columns = {'Item1':'TimelyResponse',
                   'Item2':'Fixes',
                    'Item3':'Replacements',
                    'Item4':'Reliability',
                    'Item5':'Options',
                    '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()
```



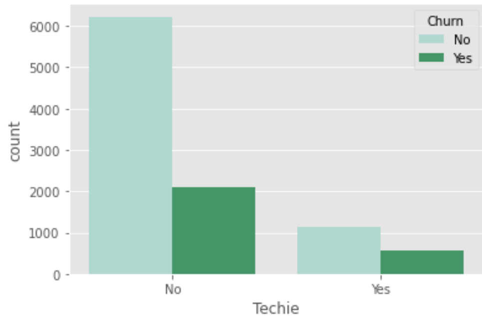
(DataCamp 2021)

```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]);
```

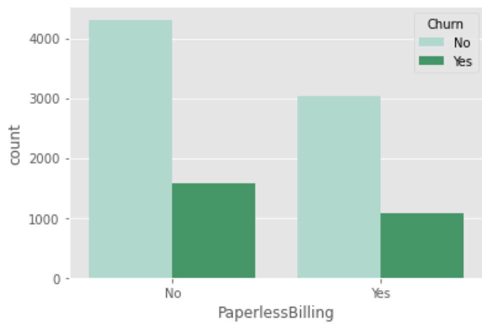


(DataCamp 2021)

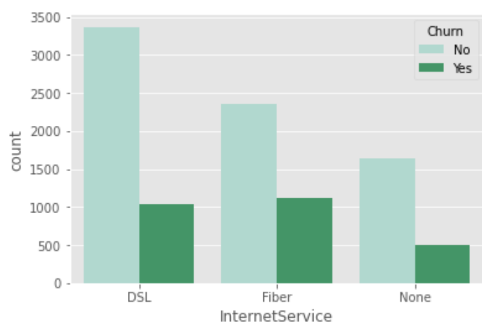
```python
# Enable ggplot
plt.style.use('ggplot')

# 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()
```




```python
# 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()
```




```python
# 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()
```



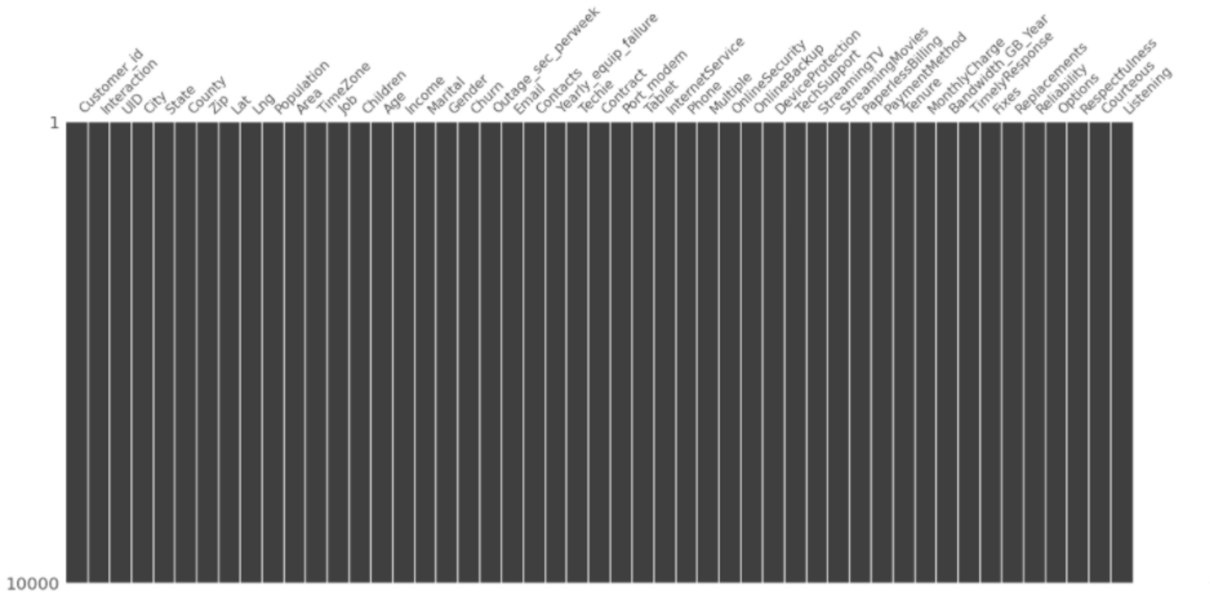
(DataCamp 2021)

```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)

```
# 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);
```




```
# 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']]
```

```
# 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_decisiontree.csv')
```

Part IV: Analysis

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

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

```
# From SKLearn import libraries for decision tree predicion model

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV, KFold, cross_val_predict, train_test_split
from sklearn.tree import DecisionTreeRegressor
```

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 Decision Tree model, fit data, and outcomes
dt = DecisionTreeRegressor(max_depth = 8, min_samples_leaf = 0.1, random_state = SEED)

dt.fit(X_train, y_train)

y_pred = dt.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]
```

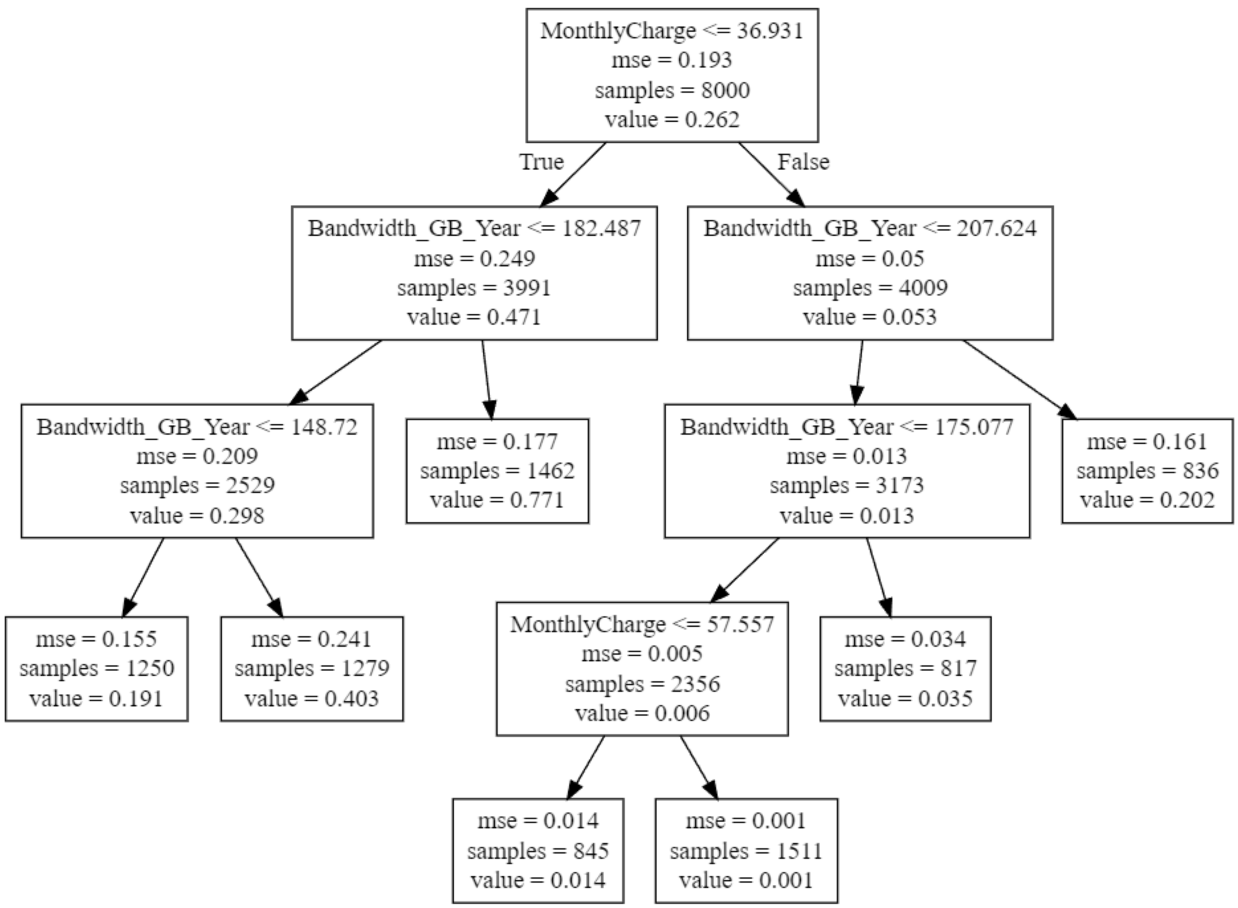
```
# Extract Training Set to CSV
print(y_pred)
pd.DataFrame(y_pred).to_csv("training_set.csv")
```

```
[0.01420118 0.40265833 0.77086183 ... 0.01420118 0.77086183 0.1912    ]
```

(GeeksforGeeks 2021)

D2. Output and Intermediate Calculations

```
# Export Decision Tree Graph as .dot
from sklearn.tree import export_graphviz
export_graphviz(dt, out_file ='tree.dot',
                feature_names =['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
        'TimelyResponse', 'Fixes', 'Replacements',
       'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening',
       'DummyGender', 'DummyTechie', 'DummyContract',
       'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhone',
       'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup',
       'DummyDeviceProtection', 'DummyTechSupport', 'DummyStreamingTV',
       'DummyPaperlessBilling', 'DummyChurn',])
```




```
# Calculate MSE and RMSE test sets
mse_dt = MSE(y_test, y_pred)

rmse_dt = mse_dt**(1/2)

print('Decison Tree Initial RMSE model score: {:.3f}'.format(rmse_dt))
```

```
Decison Tree Initial RMSE model score: 0.359
```

(GeeksforGeeks 2021)

D3. Code Execution

```
# Create Random Forest and fit model
rfr = RandomForestRegressor(n_estimators=500, random_state=1)

rfr.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=500, random_state=1)
```

```
# Train and test predictions
train_predictions = rfr.predict(X_train)
test_predictions = rfr.predict(X_test)
```

```
# Train and Test Errors
train_error = MAE(y_true=y_train, y_pred=train_predictions)
test_error = MAE(y_true=y_test, y_pred=test_predictions)
```

```
# Accuracy of seen and unseen data
print("Seen data error: {0:.2f}.".format(train_error))
print("Unseen data error: {0:.2f}.".format(test_error))
```

```
Seen data error: 0.07.
Unseen data error: 0.20.
```

```
# Comparison of each feature to the model
for i, item in enumerate(rfr.feature_importances_):
    print('{0:s}: {1:.2f}'.format(churn_df.columns[i], item))
```

```
CaseOrder: 0.05
Children: 0.02
Age: 0.03
Income: 0.04
Outage_sec_perweek: 0.04
Email: 0.03
Contacts: 0.01
Yearly_equip_failure: 0.01
Tenure: 0.28
MonthlyCharge: 0.23
Bandwidth_GB_Year: 0.04
TimelyResponse: 0.01
Fixes: 0.01
Replacements: 0.01
Reliability: 0.01
Options: 0.01
Respectfulness: 0.01
Courteous: 0.01
Listening: 0.01
DummyGender: 0.00
DummyTechie: 0.01
DummyContract: 0.04
DummyPort_modem: 0.00
DummyTablet: 0.00
DummyInternetService: 0.03
DummyPhone: 0.00
DummyMultiple: 0.00
DummyOnlineSecurity: 0.00
DummyOnlineBackup: 0.00
DummyDeviceProtection: 0.00
DummyTechSupport: 0.00
DummyStreamingTV: 0.00
DummyPaperlessBilling: 0.00
```

(GeeksforGeeks 2021)

Part V: Data Summary and Implications

E1. Accuracy and MSE

```
# From SKLearn import cross validation scoring and calculate coefficient of determination
from sklearn.model_selection import cross_val_score

scores = cross_val_score(rfr, X, y, scoring='r2')
```

```
# R-squared value
print('R-Squared Cross Validation values: ', scores)
```

```
R-Squared Cross Validation values:  [0.35182042 0.37141884 0.45388691 0.21454753 0.16200604]
```

```
# MSE versus ScikitLearn MSE
from sklearn.metrics import mean_squared_error as MSE
print('Mean Squared Error: {:.3f} '.format(sum(abs(y_test - y_pred)**2)/len(y_pred)))

print('ScikitLearn Mean Squared Error: {:.3f}'.format(MSE(y_test, y_pred)))
```

```
Mean Squared Error: 0.129
ScikitLearn Mean Squared Error: 0.129
```

```
# Root Mean Squared Error
RMSE = MSE(y_test, y_pred)**(1/2)

print('Root Mean Squared Error: {:.3f} '.format(RMSE))
```

```
Root Mean Squared Error: 0.359
```

```
# Random Forest Parameters
rfr.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 500,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 1,
 'verbose': 0,
 'warm_start': False}
```

Root Mean Square Error (RMSE) is calculated with the formula:

$$\mathrm{RMSE}_{fo} = [\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N]^{1/2}$$

Where:

- Σ = summation ("add up")
- $(z_{fi} - Z_{oi})^2$ = differences, squared
- N = sample size.

(Zach 2021)

```python
# From SKLearn import GridSearchCV for cross validation
from sklearn.model_selection import GridSearchCV

# Defined hyperparameters
params_rfr = {'n_estimators': [300, 400, 500],
            'max_depth': [4, 6, 8],
            'min_samples_leaf': [0.1, 0.2],
            'max_features': ['log2', 'sqrt']}

# Random Forest Regressor Cross Validation
rfr = RandomForestRegressor()

# GridSearch Cross Validation
rfr_cv = GridSearchCV(estimator=rfr,
                       param_grid=params_rfr,
                       scoring='neg_mean_squared_error',
                       cv=5,
                       verbose=1,
                      n_jobs=-1)
# Fit model
rfr_cv.fit(X_train, y_train)

print('Optimal Parameters for Random Forest Regressor model: {}'.format(rfr_cv.best_params_))
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Optimal Parameters for Random Forest Regressor model: {'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 0.1, 'n_esti
mators': 300}
```

```python
# Optimal Model Score
print('Optimal score for Random Forest Regressor model: {:.3f}'.format(rfr_cv.best_score_))
```

```
Optimal score for Random Forest Regressor model: -0.138
```

(SciKit 2021)

E2. Results and Implications

For our Decision Tree Prediction Model, the most important indicators of reliable results are verifying Root Mean Squared Error (RMSE) and the Adjusted R-Squared values.

Root Mean Squared Error calculates the standard deviation of prediction errors or residuals. As we have seen, residuals measure the various distances points are from the regression line. The closer RMSE is to 0, the better the model. Generally, RMSE values that lie between 0.5 and 0.2 indicate the model can accurately predict data. In our Decision Tree Prediction Model, the RMSE scored a value of 0.359 which proves that our machine learning algorithm can produce reliable results (Zach 2021).

R-Squared value can be thought of as the explanation for percent of variance. R-Squared can be defined as the ratio by which the variance of errors is less than the variance of a dependent variable. In our Decision Tree Prediction Model, we are focused on verifying the result of the Adjusted R-Squared Value to observe the standard error of regression (Frost 2018). In the Decision Tree Machine Learning model, we saw R-Squared values of 0.3518, 0.3714, 0.4538, 0.2145, and 0.1620 signifying that our scores accurately reflect the dependent variable variation explained by the linear model.

E3. Limitation

One limitation of a Decision Tree-based Machine Learning Prediction Model is that small changes to the data can result in major changes in the structure of the tree model. In comparison to the various decision predictors available, decision trees' can be largely unstable when altering data and re-creating the analysis. The resulting changes can greatly impact the message delivered from the data (Decision Tree 2021).

Fortunately, our dataset has static data that has not been changed or altered once the preparation phase was completed, thus ensuring that our resulting insights are safe and the seed can allow other researchers to recreate the data on separate systems.

E4. Course of Action

For this analysis, the reliability of the machine learning model's ability to accurately predict our target variable depends on the calculated value of Mean Square Error. In our Decision Tree Prediction Model, we calculated Mean Square Error (MSE) to a degree of 0.129. This score demonstrates the Decision Tree Model is able to accurately able to predict the target variable to a high degree of accuracy based on the values of features in the dataset.

From our Decision Tree, we can see certain features have a high degree of correlation to customer churn and our prediction model suggests we should focus on two features; Monthly Charge and Bandwidth Gigabytes used per year. The Decision Tree model suggests, based on MSE and the sample size, that customers with lower monthly charges and those who use a significant amount of data each year are less likely to churn or discontinue services with the company. The company should strive to offer lower and competitive monthly rate plans as well as subscriptions and offers for features that increase the data consumed by customers.

From our exploratory analysis we strongly recommend 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=655d0529-255e-45f3-b148-ae2c005e708c

G. Sources for Third-Party Code

Python decision tree classification with Scikit-Learn Decisiontreeclassifier. DataCamp Community. (n.d.). Retrieved January 29, 2022, from https://www.datacamp.com/community/tutorials/decision-tree-classification-python

1.10. decision trees. scikit. (n.d.). Retrieved January 28, 2022, from https://scikit-learn.org/stable/modules/tree.html#tree

Python: Decision tree regression using sklearn. GeeksforGeeks. (2021, October 20). Retrieved January 28, 2022, from https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/

Python: Mean squared error. GeeksforGeeks. (2019, June 30). Retrieved January 28, 2022, from https://www.geeksforgeeks.org/python-mean-squared-error/

I. Sources

Decision Tree Classification: Guide to Decision Tree Classification. Analytics Vidhya. (2021, April 29). Retrieved January 24, 2022, from https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-decision-tree-classification-using-python/

Zach. (2021, May 10). *How to interpret root mean square error (RMSE)*. Statology. Retrieved January 28, 2022, from https://www.statology.org/how-to-interpret-rmse/

Frost, J., Kaitlyn, Suski, K., Kembhootha, & Nugroho, A. (2018, November 12). *How high does R-squared need to be?* Statistics By Jim. Retrieved January 28, 2022, from https://statisticsbyjim.com/regression/how-high-r-squared/

Business, F. S. of. (n.d.). What's a good value for R-squared? Retrieved January 28, 2022, from https://people.duke.edu/~rnau/rsquared.htm

Decision tree. Corporate Finance Institute. (2021, September 2). Retrieved January 28, 2022, from https://corporatefinanceinstitute.com/resources/knowledge/other/decision-tree