D209_Performance_Assessment

Task2: Predictive Analysis

December 12th,2021

Table of Contents

Performance Evaluation of Predictive Analysis – NVM2, D209 3
Part I: 3
A. Introduction to Predictive Modeling -NBM2: 3
A1. Analytical Question: 3
A2. Goals and Objectives: 3
Part II: 4
B. Justification for the method 4
B1. Assumptions Summary: Prediction Method 4
B2. Method Assumption Summary: Decision Tree 4
B3. Advantages/Benefit of the Tool 5
Part III: Data Objectives: 5
C1. The following will be part of my strategy: 5
C2. Statistics in Brief: 6
C3. Data Preparation Procedures: 7
C4: Dataset that has been cleaned 26
Part IV: Analysis 26
D. Analysis and Comparison of Models 27
D1. Data Segmentation: 27
D2. Calculations for Output and Intermediate: 27
D3: Execution of Code: 27
Part V: 30
E1. Perform the following MSE & Accuracy model 30
E2. Conclusions and Implications: 32
E3. Constraints/Limitations: 32
E4. Plan of Action: 32
Part VI: Documentary Evidence 33
F. Panopto video recording: 33
G. Third Party Evidence: 33

H. References: 33

Performance Evaluation of Predictive Analysis – NVM2, D209

Name: Rekha Alle
Student ID: 000778673
Course: Masters Data Analytics
Date: 09/22/2021
Program Mentor: Dr. Eric Straw
Contact: 3854282685
Email: eric.straw@wgu.edu

Part I:

A. Introduction to Predictive Modeling -NBM2:

Finding reasons for client loss, gauging customer loyalty, and recovering customers have all become highly essential ideas for many businesses. Companies conduct a variety of studies and efforts to prevent losing consumers rather than gaining new ones.

Due to fast renewable technology, an increase in the number of users, and value-added services, the telecommunications business collects massive amounts of data. Due to the uncontrolled and rapid expansion of this area, considerable losses are incurred because of fraud and technical challenges. As a result, the creation of new analysis methodologies has become a necessity. The number one business goal for many providers is to keep extremely profitable customers. Telecommunications businesses must predict which clients are at greater risk of churning to reduce customer churn.

A1. Analytical Question:

Which consumers are most likely to leave? What are the most important customer features/variables in terms of churn? A **Decision Tree** method will be used to answer this question.

A2. Goals and Objectives:

Knowing something with any degree of certainty will benefit everyone in the organization, which customers are most likely to churn, as this will provide weight to selling enhanced services to consumers with these traits and previous user experiences.

Part II:

B. Justification for the method

B1. Assumptions Summary: Prediction Method

Every branch should represent an attribute, and at the conclusion of each node, the decision tree should determine the best choice. (Plapinger, pg.[1]).

CART algorithms are a set of if/else queries about individual category features with the goal of inferring labels. Nonlinear relationships between characteristics and labels can be captured with trees. They also don't necessitate feature scaling, such as standardization and normalization. "They improve the accuracy, readability, and stability of prediction models" (Corporate Finance Institute, pg[2]).

B2. Method Assumption Summary: Decision Tree

We will use Decision Tree method because we'll be determining whether or not a consumer will leave, which is a classification problem. The logistic model is used in statistics to represent the probability of a specific class or event, such as pass or fail, win, or lose, living or dead, healthy, or sick, occurring.

Decision Tree Method:

Because the dependent variable is binary, it is based on the Bernoulli (also known as binominal or Boolean) Distribution rather than the Gaussian Distribution (in our dataset, to churn or not to churn)

- The anticipated values are limited to a nominal value range: No or Yes
- It predicts the possibility of a given event rather than the actual outcomes.
- There are no predictors with a significant degree of co-relation (multi-collinearity).
- It's the formula for calculating the chances of success: In other words, a regression model with a natural logarithm of the odds output, often known as logit.

Our method will be decision tree, which is a key competitor of logistic regression and is quite resilient. The decision tree's outcomes are simple to comprehend. Before constructing the decision tree, there may be collinear independent variables, as well as extreme values and missing data. The data will be separated into smaller data groups based on the attributes, allowing us to discover which features are the most significant.

B3. Advantages/Benefit of the Tool

Tools will be used:

For this assessment, I'll use Python because the study will be supported by Jupyter notebooks in Python and IPython. Python includes many established data science and machine learning tools, straightforward, and extensible programming style, and grammar. Python is cross-platform, so it will function whether the analysis is viewed on a Windows PC or a MacBook laptop. When compared to other programming languages such as R or MATLAB, it is quick (Massaron, p. 8[3]). In addition, Python is often regarded in popular media as the most widely used programming language for data science and media (CBTNuggets, p. 1). [4]

NumPy used to work with arrays,
Pandas used to load datasets,
Matplotlib used to plot charts,
Scikit-learn used for machine learning model classes,
SciPy used for mathematical problems, specifically linear algebra transformations, and
Seaborn used for a high-level interface and appealing visualizations.
Using the Pandas library and its accompanying "read csv" function to transform our data as a dataframe is a quick, exact example of loading a dataset and constructing a variable efficiently: imported pandas as pd, df(dataframe) = pd.read csv('ChurnData.csv')

Part III: Data Objectives:

C1. The following will be part of my strategy:

1. Using Pandas' read csv command, read the data collection into python programming.
2. Examine the data structure for a better understanding of the data collection process.
3. Using the variable "churn df" to name the dataset, and "df" to name the data frame's subsequent usable slices.
4. Check for misspellings, strange variable names, and data that is missing.
5. Identify outliers that may create or obscure statistical significance using histograms.
6. Computing replaces missing data with relevant central tendency measures (mean, median, or mode) or just Outliers a few standard deviations above the mean are removed.

Using "Bandwidth GB Year" (the average) is a dependent variable of yearly quantity of data consumed, per customer, in GB), it will be our long- term target variable, is the most important to our decision-making process. To construct a model that will give us an indication of data a customer may use given the quantities utilized by known customers given their individual data points for selected predictor variables, on our given dataset, we must first train and then test our machine.

The categorical variables (all binary Predictor except for categorical variable with two values, "Yes"/ "No," were stated) may be shown to be significant: * Churn: Whether the consumer

stopped using the service in the previous month (yes, no)* Techie: Whether or not the customer perceives themselves to be technically savvy (as determined by a customer questionnaire completed * Contract (at the time of signing up for services) (yes, no):The customer's contract term (one year , two years or month-on-month). * Port_modem: consumer using a portable modem (yes/no) * the consumer possesses a tablet, such as or a Surface or an iPad (yes, no) * Internet Service: The internet service provider for the customer (DSL, fiber optic, None) * Phone: Is there a phone service for the consumer (yes, no)? * Multiple: If the customer has more than one line (yes, no) * Online Security: Whether the consumer has an add-on for online security (yes, no) * Online Backup: Whether the consumer has purchased an add-on for internet backup (yes, no) * Device Protection: Is any consumer device protection add-on? (Yes, no) * Tech Support: Is there a technical assistance add-on for the customer (yes, no) * Streaming TV: Whether the consumer has access to streaming television (yes, no) * Streaming Movies: If the customer has access to on-demand movies (yes, no).

In the decisionmaking process, discrete ordinal predictor variables created from consumer surve y responses about various customer service attributes could be valuable. Customers in the surve ys provided ordinal numerical data by rating eight customer service aspects on a scale of 8 to 1 (8 being the most essential and 1 being the least important):

- Item1: Evidence of active listening
- Item2: Courteous exchange
- Item3: Respectful response
- Item4: Options
- Item5: Reliability
- Item6: Timely replacements
- Item7: Timely fixes
- Item8: Timely response

C2. Statistics in Brief:

The dataset has 50 original columns and 10,000 records, as shown in the Python pandas data frame techniques are as follows.

Especial user IDs and statics categorical variables (‘Customer id’,'Case Order, 'Interaction, 'City,'State,'County,'Zip,'Lat,'UID, 'Area,'Lng’,'PaymentMethod','Population, 'TimeZone,'Job,'Marital) not included in the data frame for this research. In addition, binomial "Yes"or "No" / "Male"or"Female" variables encoded to 1 or 0. This left 34 numerical independent predictor factors, including the target variable, to be determined. There appeared to be no nulls, NAs, or missing data points in the dataset, indicating that it had been well cleaned. Ordinary distributions were discovered for "Outage sec per week,” Email" and "Monthly

Charge," using histograms and boxplots as calculate the central tendency. There were no more outliers in the cleaned dataset. In a scatterplot, histograms for "Bandwidth_GB_Year" and "Tenure" displayed bimodal distributions, indicating a straight linear relationship. 53 years old customers are average (with standard deviation of 20 years), had two children (with a standard deviation of two children), had an income of $39,806 (There were 10 outage-seconds every week, with a standard deviation of around 30,000, 12 times email was marked, called technical assistance few times, had fewer than one annual equipment fault, has been with the organization for almost months of 34.5, has a monthly charges of about 173, and uses 3,392 GBs.

C3. Data Preparation Procedures:

- Create a Python data frame from a dataset.
- Rename the survey's columns/variables to make them more clearly identifiable (ex: "Item1" to "Timely_Response").
- Obtain a description of the data frame, including its structure (columns and rows) and data types.
- Look for the summary statistics.
- Remove the data frame's non-vital identifying (ex: "Customer id" and ex: zip code) are demographic columns.
- Search records for missing data and fill in the blanks, Outliers that are several standard deviations above the mean should be removed with the central tendency (mean/median/mode)/ delete the outliers that are more than a standard deviation above the mean.
- Make a list of dummy variables to encode category, yes/no data points into 1/0 number values
- Create a visual representation of univariate and bivariate data.
- At the end of the data frame, add Bandwidth GB Year.
- The prepared dataset will be extracted and delivered as "churn prepared.csv" at the end.

1. **Include standard imports all the required references:**

```
# Increase Jupyter display cell-width
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:75% !important; }</style>"))
```

```
<IPython.core.display.HTML object>
```

```
# Standard data science imports
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Scikit-learn
import sklearn
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
```

2. **Change font and color of the Matplotlib:**

```
# Change color of Matplotlib font
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
```

3. **Imports the Ignore warning codes:**

```
# Ignore Warning Code
import warnings
warnings.filterwarnings('ignore')
```

4. **Load the churn data frame into pandas:**

```
In [8]: # Load data set into Pandas dataframe
        churn_df = pd.read_csv('C:\Rekha\churn_clean.csv', index_col=0)
```

5. **To examine the data frame columns:**

```
# Examine the features of the dataset
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')
```

6. **To List the records & columns of dataset:**

```
# Get an idea of dataset size
churn_df.shape
```

```
(10000, 49)
```

7. **List the churn data set statics:**

```
# Examine first few records of dataset
churn_df.head()
```

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCharge	Bandwidth_GB_Year	Item1	Item2	Item3
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	...	172.455519	904.536110	5	5	5
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	...	242.632554	800.982766	3	4	3
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	...	159.947583	2054.706961	4	4	2
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	...	119.956840	2164.579412	4	4	4
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	...	149.948316	271.493436	4	4	4

5 rows × 49 columns

8. To List the data frame info:

```
# View DataFrame 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  ...
```

```
           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]>
```

9. Dataset with data points:

```
# Provide an initial look at extant dataset
churn_df.head()
```

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCharge	Bandwidth_GB_Year	Item1	Item2	Item3
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	...	172.455519	904.536110	5	5	5
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	...	242.632554	800.982766	3	4	3
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	...	159.947583	2054.706961	4	4	2
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	...	119.956840	2164.579412	4	4	4
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	...	149.948316	271.493436	4	4	4

5 rows × 49 columns

10. To List the churn data set statics:

```
# Get an overview of descriptive statistics
churn_df.describe()
```

	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	...	MonthlyCharge	Bandwidth_GB_Year	
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	10000.000000	10000.000000	10000.0
mean	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	10.001848	12.016000	0.994200	...	172.624816	3392.341550	3.4
std	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.025898	0.988466	...	42.943094	2185.294852	1.0
min	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	0.099747	1.000000	0.000000	...	79.978860	155.506715	1.0
25%	26292.500000	35.341828	-97.082812	738.000000	0.0000	35.000000	19224.717500	8.018214	10.000000	0.000000	...	139.979239	1236.470827	3.0
50%	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.000000	1.000000	...	167.484700	3279.536903	3.0
75%	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.000000	2.000000	...	200.734725	5586.141370	4.0
max	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.000000	7.000000	...	290.160419	7158.981530	7.0

8 rows × 22 columns

11. Matrix of Pearson correlations values

```
# Print a matrix of Pearson correlation values
churn_df.corr()
```

	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	...	MonthlyCharge	Bandwidth_GB_Year	Item1	Item2
Zip	1.000000e+00	-0.042580	-0.901786	0.045138	-0.017205	-0.008135	0.002947	-0.011520	-0.007860	-0.004720	...	-0.008717	-0.002527	-0.026024	-0.003215
Lat	-4.257985e-02	1.000000	-0.100639	-0.220598	-0.000452	-0.002943	0.006563	0.004827	-0.030042	-0.002213	...	0.001406	-0.016949	0.005793	0.005380
Lng	-9.017856e-01	-0.100639	1.000000	-0.048294	0.013634	0.012117	-0.002997	0.005646	0.007872	0.008118	...	0.011765	0.003446	0.024389	0.003024
Population	4.513808e-02	-0.220598	-0.048294	1.000000	-0.005877	0.010538	-0.008639	0.005483	0.017962	0.004019	...	-0.004778	-0.003902	0.000618	-0.002571
Children	-1.720505e-02	-0.000452	0.013634	-0.005877	1.000000	-0.029732	0.009942	0.001889	0.004479	-0.020776	...	-0.009781	0.025585	0.011470	0.013146
Age	-8.135285e-03	-0.002943	0.012117	0.010538	-0.029732	1.000000	-0.004091	-0.008047	0.001588	0.015068	...	0.010729	-0.014724	-0.005972	0.006659
Income	2.946523e-03	0.006563	-0.002997	-0.008639	0.009942	-0.004091	1.000000	-0.010011	-0.009267	0.001233	...	-0.003014	0.003674	-0.004009	0.007718
Outage_sec_perweek	-1.152040e-02	0.004827	0.005646	0.005483	0.001889	-0.008047	-0.010011	1.000000	0.003994	0.015092	...	0.020496	0.004176	-0.023338	-0.008076
Email	-7.860018e-03	-0.030042	0.007872	0.017962	0.004479	0.001588	-0.009267	0.003994	1.000000	0.003040	...	0.001997	-0.014579	0.003709	0.001156
Contacts	-4.719513e-03	-0.002213	0.008118	0.004019	-0.020776	0.015068	0.001233	0.015092	0.003040	1.000000	...	0.004259	0.003299	-0.007364	-0.003003
Yearly_equip_failure	1.104692e-02	-0.007125	-0.005577	-0.004483	0.007321	0.008577	0.005423	0.002909	-0.016354	-0.006032	...	-0.007172	0.012034	0.007518	-0.009773
Tenure	-3.227133e-03	-0.015743	0.003618	-0.003559	-0.005091	0.016979	0.002114	0.002932	-0.014468	0.002820	...	-0.003337	0.991495	-0.006246	0.003073
MonthlyCharge	-8.716780e-03	0.001406	0.011765	-0.004778	-0.009781	0.010729	-0.003014	0.020496	0.001997	0.004259	...	1.000000	0.060406	0.009756	0.003442
Bandwidth_GB_Year	-2.527196e-03	-0.016949	0.003446	-0.003902	0.025585	-0.014724	0.003674	0.004176	-0.014579	0.003299	...	0.060406	1.000000	-0.007314	0.003062
Item1	-2.602437e-02	0.005793	0.024389	0.000618	0.011470	-0.005972	-0.004009	-0.023338	0.003709	-0.007364	...	0.009756	-0.007314	1.000000	0.663069
Item2	-3.215486e-03	0.005380	0.003024	-0.002571	0.013146	0.006659	0.007718	-0.008076	0.001156	-0.003003	...	0.003442	0.003062	0.663069	1.000000
Item3	-1.968673e-02	-0.003421	0.014966	0.001620	0.003394	-0.003778	-0.002707	-0.021366	0.012928	-0.021924	...	-0.008487	0.000978	0.578013	0.520194
Item4	6.196497e-07	0.011015	-0.002398	-0.008272	-0.016768	0.012237	-0.022750	-0.005992	0.000915	0.001071	...	-0.000440	-0.007789	-0.005048	-0.002438
Item5	2.922689e-03	-0.000464	0.001366	0.006970	-0.000240	-0.008305	0.011111	-0.004156	-0.007929	0.004698	...	-0.005562	0.025112	-0.002553	0.001260
Item6	7.375053e-03	-0.008726	-0.008770	0.000834	-0.000547	0.010635	0.001768	-0.014752	0.014468	-0.000994	...	0.002943	-0.010352	0.402804	0.363247
Item7	-5.537247e-03	0.006087	0.001601	-0.013062	0.000687	0.005353	0.016599	-0.020854	0.010010	0.004579	...	-0.006399	-0.001077	0.336782	0.300324
Item8	-2.577073e-03	-0.019583	0.001912	0.008524	-0.005236	0.015193	0.000740	0.008126	-0.005857	-0.012615	...	0.002204	-0.015018	0.292728	0.254259

22 rows × 22 columns

12. List the data types:

```
# Get data types of features
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
Outage_sec_perweek     float64
Email                    int64
Contacts                 int64
Yearly_equip_failure     int64
Techie                  object
Contract                object
Port_modem              object
Tablet                  object
```

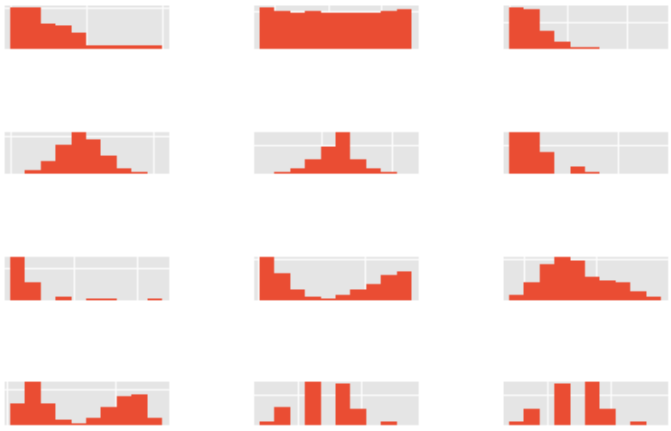
```
InternetService          object
Phone                    object
Multiple                 object
OnlineSecurity           object
OnlineBackup             object
DeviceProtection         object
TechSupport              object
StreamingTV              object
StreamingMovies          object
PaperlessBilling         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
```

12 Change the names of the last eight survey columns to better describe the variables:

```
# Rename last 8 survey columns for better description of variables
churn_df.rename(columns = {'Item1':'Timely_Response',
'Item2':'Timely_Fixes',
'Item3':'Timely_Replacements',
'Item4':'Reliability',
'Item5':'Options',
'Item6':'Respectful_Response',
'Item7':'Courteous_exchange',
'Item8':'Active_Listening'},
inplace=True)
```

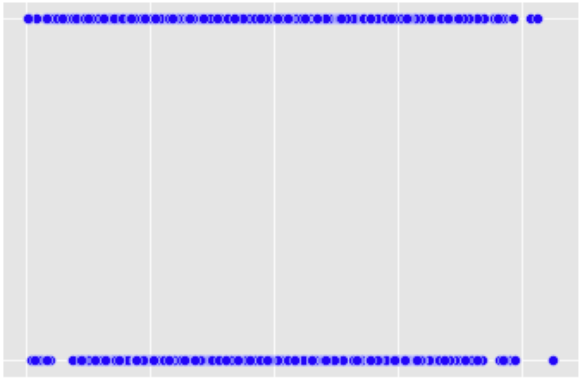
13 Histograms of continuous variables & categorical variables:

```
 # Create histograms of contiuous variables & categorical variables
churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
'Bandwidth_GB_Year','Timely_Response','Courteous_exchange']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```



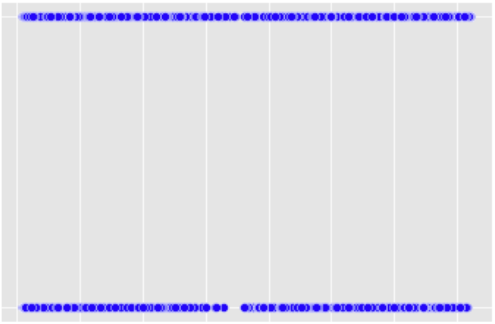
14 Scatterplot:

```
# Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['Outage_sec_perweek'], y=churn_df['Churn'], color='blue')
plt.show();
```



15 Scatterplot:

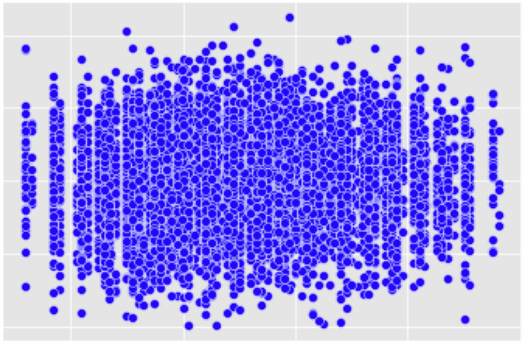
```
 # Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['Churn'], color='blue')
plt.show();
```



---

16 Scatterplot:

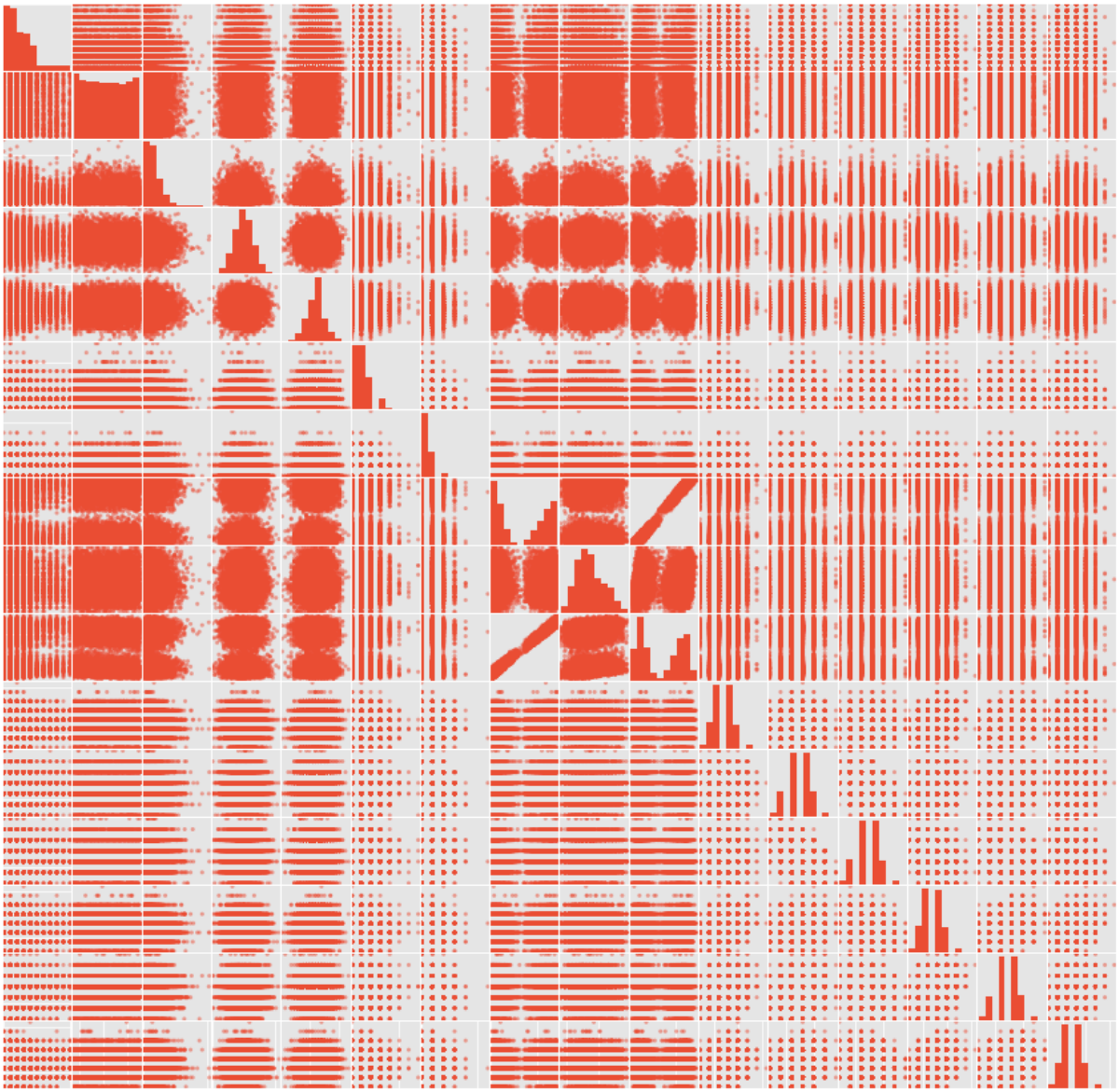
```
 # Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['Outage_sec_perweek'], color='blue')
plt.show();
```



---

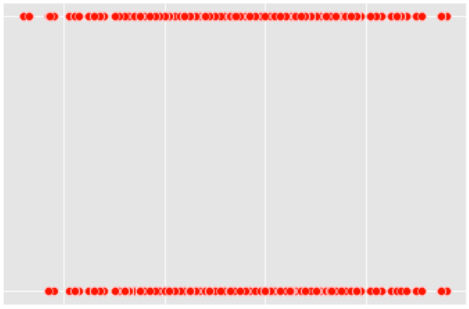
17 Scatter Matrix:

```
# Provide a scatter matrix of numeric variables for high level overview of potential relationships & distributions
churn_numeric = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek',
 'Email', 'Contacts','Yearly_equip_failure', 'Tenure',
 'MonthlyCharge', 'Bandwidth_GB_Year', 'Timely_Replacements',
 'Reliability', 'Options', 'Respectful_Response','Courteous_exchange',
 'Active_Listening']]
pd.plotting.scatter_matrix(churn_numeric, figsize = [15, 15]);
```



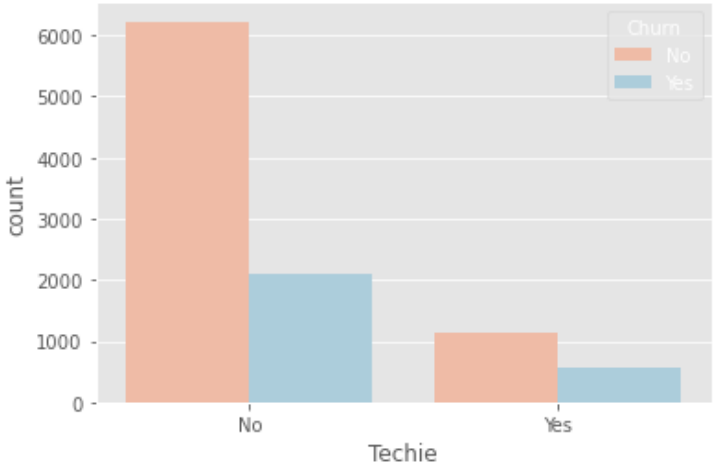
18 Scatterplot

```
# Create individual scatterplot for viewing relationship of key financial featurte against target variable
sns.scatterplot(x = churn_df['MonthlyCharge'], y = churn_df['Churn'], color='red')
plt.show();
```



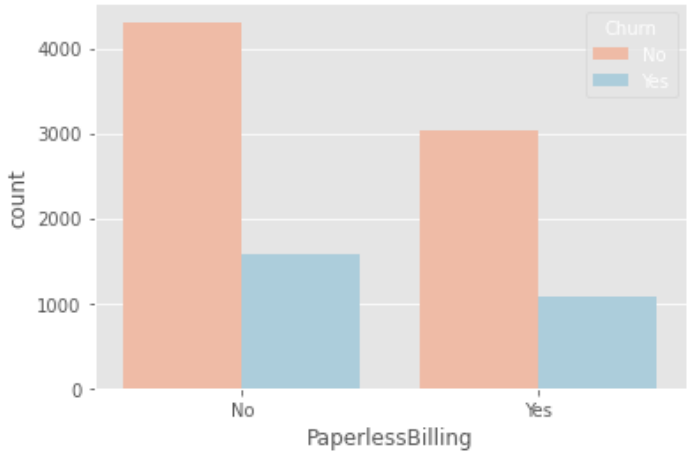
19 Set **plot style to ggplot**

```
# Set plot style to ggplot for aesthetics & R style
plt.style.use('ggplot')
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



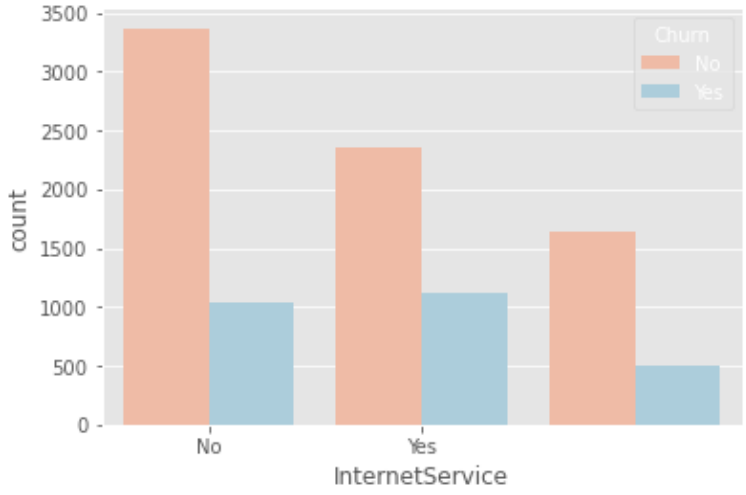
20 Countplot

```
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



21 Countplot

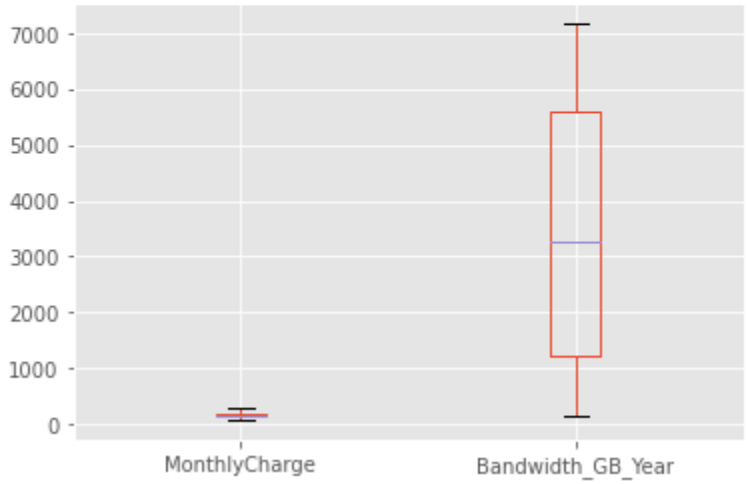
```
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='InternetService', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



22 Boxplot

```
# Create multiple boxplots for continuous & categorical variables
churn_df.boxplot(column=['MonthlyCharge','Bandwidth_GB_Year'])
```

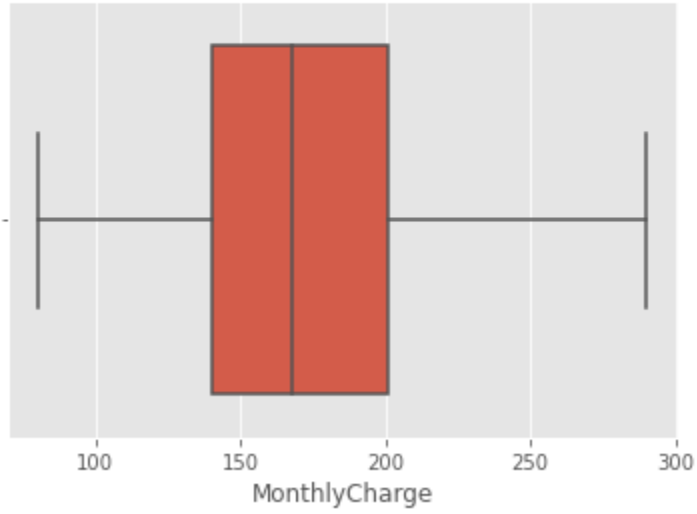
```
<AxesSubplot:>
```



---

23 Boxplot

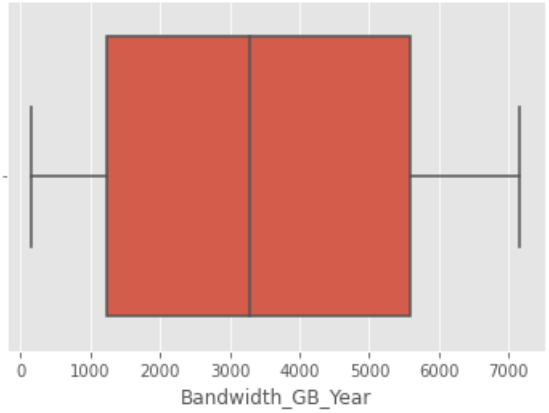
```
# Create Seaborn boxplots for continuous & categorical variables
sns.boxplot('MonthlyCharge', data = churn_df)
plt.show()
```



---

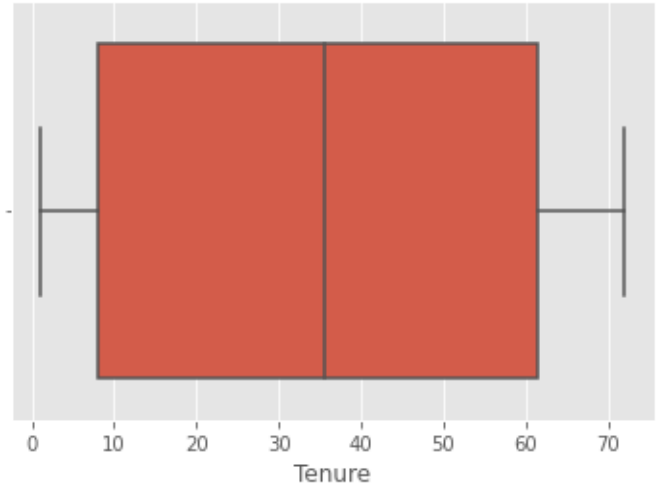
24 Seaborn boxplot

```
# Create Seaborn boxplots for continuous & categorical variables
sns.boxplot('Bandwidth_GB_Year', data = churn_df)
plt.show()
```



25 Seaborn boxplot

```
# Create Seaborn boxplots for continuous variables
sns.boxplot('Tenure', data = churn_df)
plt.show()
```



Anomalies

Anomalies appear to have been removed from the provided dataset, churn clean.csv. There are no more anomalies.

26 Discover missing data points:

```
# Discover missing data points within dataset
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
```

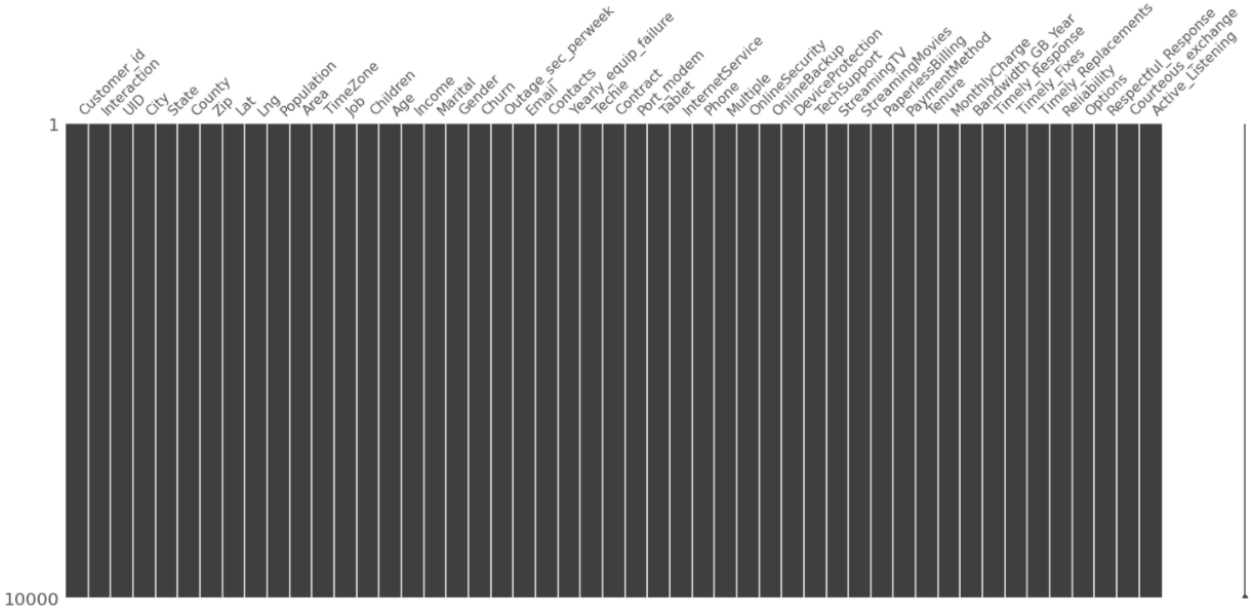
```
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
Timely_Response         0
Timely_Fixes            0
Timely_Replacements     0
Reliability             0
Options                 0
Respectful_Response     0
Courteous_exchange      0
Active_Listening        0
dtype: int64
```

27 Check for missing data

```
# Check for missing data & visualize missing values in dataset
# Install appropriate library
!pip install missingno
# Importing the libraries
import missingno as msno
# Visualize missing values as a matrix
msno.matrix(churn_df);
"""(GeeksForGeeks, p. 1)"""
```

```
Requirement already satisfied: missingno in c:\users\kaila\anaconda3\lib\site-packages (0.5.0)
Requirement already satisfied: scipy in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (1.6.2)
Requirement already satisfied: matplotlib in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (3.3.4)
Requirement already satisfied: seaborn in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (0.11.1)
Requirement already satisfied: numpy in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (1.20.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (8.2.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: six in c:\users\kaila\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->missingno) (1.15.0)
Requirement already satisfied: pandas>=0.23 in c:\users\kaila\anaconda3\lib\site-packages (from seaborn->missingno) (1.2.4)
Requirement already satisfied: pytz>=2017.3 in c:\users\kaila\anaconda3\lib\site-packages (from pandas>=0.23->seaborn->missingno) (2021.1)

'(GeeksForGeeks, p. 1)'
```



28 Encode Binary Categorical

```
 # Encode binary categorical variables with dummies
churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender']]
churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']] ### If the customer left (churned) they get a '1'
churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in churn_df['Contract']]
churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in churn_df['Port_modem']]
churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in churn_df['InternetService']]
churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn_df['Multiple']]
churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineSecurity']]
churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineBackup']]
churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn_df['DeviceProtection']]
churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn_df['TechSupport']]
churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingTV']]
churn_df['StreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingMovies']]
churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn_df['PaperlessBilling']]
```

29 Drop original categorical features

```
# Drop original categorical 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'])
```

30 Churn Head

```
churn_df.head()
```

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

5 rows × 48 columns

31 Remove categorical variables

```
# Remove less meaningful categorical variables from dataset to provide fully numerical dataframe for further analysis
churn_df = churn_df.drop(columns=['Customer_id', 'Interaction', 'UID',
'City', 'State', 'County', 'Zip', 'Lat', 'Lng',
'Area', 'TimeZone', 'Job', 'Marital', 'PaymentMethod'])
churn_df.head()
```

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

5 rows × 34 columns

32 Move DummyChurn

```
# Move DummyChurn to end of dataset to set as target
churn_df = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
'Timely_Response', 'Timely_Fixes', 'Timely_Replacements',
'Reliability', 'Options', 'Respectful_Response', 'Courteous_exchange', 'Active_Listening',
'DummyGender', 'DummyTechie', 'DummyContract',
'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhone',
'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup',
'DummyDeviceProtection', 'DummyTechSupport', 'DummyStreamingTV',
'DummyPaperlessBilling', 'DummyChurn',]]
churn_df.head()
```

CaseOrder	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	Bandwidth_GB_Year	...	DummyInte
1	0	68	28561.99	7.978323	10	0	1	6.795513	172.455519	904.536110	...	
2	1	27	21704.77	11.699080	12	0	1	1.156681	242.632554	800.982766	...	
3	4	50	9609.57	10.752800	9	0	1	15.754144	159.947583	2054.706961	...	
4	1	48	18925.23	14.913540	15	2	0	17.087227	119.956840	2164.579412	...	
5	0	83	40074.19	8.147417	16	2	1	1.670972	149.948316	271.493436	...	

5 rows × 33 columns

33 List features

```
# List features for analysis
features = (list(churn_df.columns[:-1]))
print('Features for analysis include: \n', features)
```

```
Features for analysis include:
 ['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Timely_Response',
'Timely_Fixes', 'Timely_Replacements', 'Reliability', 'Options', 'Respectful_Response', 'Courteous_exchange', 'Active_Listening', 'DummyGender', 'DummyTechie', 'Dumm
yContract', 'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhone', 'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtecti
on', 'DummyTechSupport', 'DummyStreamingTV', 'DummyPaperlessBilling']
```

C4: Dataset that has been cleaned

```
# Extract Clean dataset
churn_df.to_csv('C:\Rekha\churn_prepared.dt.csv')
```

Part IV: Analysis

```
# Re-read fully numerical prepared dataset
churn_df = pd.read_csv('C:\Rekha\churn_prepared.dt.csv')
# Set predictor features & target variable
X = churn_df.drop('DummyChurn', axis=1).values
y = churn_df['DummyChurn'].values
```

```
# Import model, splitting method & metrics from sklearn
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV, KFold, cross_val_predict, train_test_split
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE
```

D. Analysis and Comparison of Models

D1. Data Segmentation:

```
# Set seed for reproducibility
SEED = 1
# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = SEED)
```

```
# Instantiate Decision Tree Regressor model
dt = DecisionTreeRegressor(max_depth = 8,
min_samples_leaf = 0.1,
random_state = SEED)
# Fit dataframe to Decision Tree Regressor model
dt.fit(X_train, y_train)
# Compute y_pred
y_pred = dt.predict(X_test)
# Predict outcomes from test set
y_pred = dt.predict(X_test)
```

D2. Calculations for Output and Intermediate:

```
# Compute test set MSE
mse_dt = MSE(y_test, y_pred)
# Compute test set RMSE
rmse_dt = mse_dt**(1/2)
# Print initial RMSE
print('Initial RMSE score Decison Tree Regressor model: {:.3f}'.format(rmse_dt))
```

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

D3: Execution of Code:

```
# Instantiae Random Random Forest Regressor model
rfr = RandomForestRegressor(n_estimators=500, random_state=1)

# Fit model to dataframe
rfr.fit(X_train, y_train)
```

```
RandomForestRegressor(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)
```

```
# Create vectors of predictions
train_predictions = rfr.predict(X_train)
test_predictions = rfr.predict(X_test)
```

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

```
# Print the accuracy for seen and unseen data
print("Model error on seen data: {0:.2f}.".format(train_error))
print("Model error on unseen data: {0:.2f}.".format(test_error))
```

```
Model error on seen data: 0.07.
Model error on unseen data: 0.20.
```

Vital Feature

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

```
Unnamed: 0: 0.03
CaseOrder: 0.03
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
Timely_Response: 0.01
Timely_Fixes: 0.01
Timely_Replacements: 0.01
Reliability: 0.01
Options: 0.01
Respectful_Response: 0.01
Courteous_exchange: 0.01
Active_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
```

Part V:

E1. Perform the following MSE & Accuracy model

Mean Squared Error

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

- Most widely used regression metric
- Allows outliers to contribute more to overall error

```
# Import cross validation metrics
from sklearn.model_selection import cross_val_score
# Compute the coefficient of determination (R-squared)
scores = cross_val_score(rfr, X, y, scoring='r2')
```

```
# Print R-squared value
print('Cross validation R-squared values: ', scores)
```

```
Cross validation R-squared values:  [0.35180503 0.3727649  0.45407944 0.2164273  0.16632283]
```

```
# Print Mean Squared Error
print('With a manual calculation, the Mean Squared Error: {:.3f} '.format(sum(abs(y_test - y_pred)**2)/len(y_pred)))
# Or
from sklearn.metrics import mean_squared_error as MSE
print('Using scikit-lean, the Mean Squared Error: {:.3f}'.format(MSE(y_test, y_pred)))
```

```
With a manual calculation, the Mean Squared Error: 0.129
Using scikit-lean, the Mean Squared Error: 0.129
```

```
# Calculate & print the Root Mean Squared Error
RMSE = MSE(y_test, y_pred)**(1/2)
# Print the Root Mean Squared Error
print('Root Mean Squared Error: {:.3f} '.format(RMSE))
```

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

Model Comparison

With a low **Mean Squared Error = 0.129** we have a decent model with a high accuracy of prediction.

```
# Get parameters of Random Forest Regression model
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}
```

```
# Import GridSearchCV for cross validation of model
from sklearn.model_selection import GridSearchCV
# Define grid of hyperparameters
params_rfr = {'n_estimators': [300, 400, 500],
'max_depth': [4, 6, 8],
'min_samples_leaf': [0.1, 0.2],
'max_features': ['log2', 'sqrt']}
# Re-intantiate Random Forest Regressor for cross validation
rfr = RandomForestRegressor()
# Instantiate 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 to
rfr_cv.fit(X_train, y_train)

# Print best parameters
print('Best parameters for this Random Forest Regressor model: {}'.format(rfr_cv.best_params_))
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best parameters for this Random Forest Regressor model: {'max_depth': 4, 'max_features': 'log2', 'min_samples_leaf': 0.1, 'n_estimators': 400}
```

```
# # Generate model best score
print('Best score for this Random Forest Regressor model: {:.3f}'.format(rfr_cv.best_score_))
```

```
Best score for this Random Forest Regressor model: -0.136
```

```
# Fit the GridSearch to training data
grid_rf.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   12.6s
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:   52.2s finished
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestRegressor(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=100, n_jobs=None,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [4, 6, 8],
                         'max_features': ['log2', 'sqrt'],
                         'min_samples_leaf': [0.1, 0.2],
                         'n_estimators': [300, 400, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=1)
```

E2. Conclusions and Implications:

Attached the calculations and code outputs.

E3. Constraints/Limitations:

Overfitting is a key issue with all machine learning models. If a model is overfitted, it will not generalize well to fresh data. We delete branches that use low-importance features to prevent the decision tree from overfitting. Pruning, or post-pruning, is the term for this technique. (Yadav, pg. [5])

Furthermore, we must exercise caution while tweaking parameters, as trained trees might exhibit biases if classes predominate. (Yadav, pg. [5])

E4. Plan of Action:

Marketers and decision-makers must be aware that our predictor factors produce a low Mean Squared Error of 0.129. This implies that we should be able to forecast which customers would cancel their telecom services in the next months with a high degree of accuracy.

As a result, we should look at the characteristics that are prevalent among individuals who have left the organization in the past and try to limit their risk of recurrence with any future consumer. This also

means that once a consumer signs up for more services from the provider, they are less likely to leave if they have an additional port modem or an online backup. Certainly, it is in the best interests of customers to provide them with more services and to help customers comprehend all the services available to customers, not just mobile phone service, to enhancing their entire experience with the organization.

Part VI: Documentary Evidence

F. Panopto video recording:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1033977d-f0a5-4e6e-8336-ae070023b2e2

G. Third Party Evidence:

Practiced code: Bivariate plotting with pandas
URL: https://www.kaggle.com

Practiced code: GeeksforGeeks (July 4th, 2019)
URL: https://www.geeksforgeeks.org/python-visualize-missing-values-nanvalues-using-missingno-library/

Practiced code: Dennis. T. (July 25th, 2019)
Confusion Matrix Visualization. Medium.
URL: https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea/

Article: Predict Customer Churn in Python.
URL: https://towardsdatascience.com/

LinkedIn: https://www.linkedin.com/learning/python-statistics-essential-training/introducing-pandas?u=2045532

H. References:

[1] Plapinger, T. (July 29, 2017) Towards Data Science. https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1

[2] Corporate Finance Institute (2021). What is Decision Tree? Corporate Finance Institute: https://corporatefinanceinstitute.com/resources/knowledge/other/decision-tree/

[3] Massaron, L. & Boschetti, A. (2016). Regression Analysis with Python. Packt Publishing.

4CBTNuggets. (2018, September 20). Why Data Scientists Love Python.

[5] Yadav, P.(November13, 2018): Decision Tree in Machine Learning. TowardDataScience. https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96