PART I- RESEARCH QUESTION

1. Research Question

This research paper intends to evaluate our ability to clean complex data sets and get them ready for exploration using different programming languages. We opted to clean the churn data CSV file using the python programming language. We chose the churn data set because of our familiarity with it since we have used the same data set in our previous assignment and the python programming language because we are more familiar with it than the R programming language. After a deep study of the data set, we decided to analyze the individual customers that are at high risk of churn and identify which characteristics of the data set are significant to these individual customers making telecommunication service churn decisions.

Researching the factors contributing to customers making telecommunication service churn decisions is very important for the telecommunication industry because it helps the industry to identify the different factors contributing to customers’ decision-making process in order to set up targeted customers retention marketing strategies, given that customers retention in the telecommunication industry is as much important as new customers acquisition.

1. Description of all variables in the data set

This paper examines the individual customers that are at high risk of churn and identifies which characteristics of the data set are significant to these individual customers making telecommunication service churn decisions. The churn data set provided to us details 10,000 customers data from a telecommunication firm. In our research, the dependent variable depicts customers who responded “Yes” or “No” to the question of whether they discontinued their telecommunication service in the last month, which is a qualitative variable.

The independent variables in the data set consist of qualitative and quantitative data.

The qualitative variables are as follows: City, State, County, Area, Timezone, Job, Education, Employment, Marital, Gender, Techie, Contract, Port\_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling, PaymentMethod.

The quantitative variables are as follows: Population, Children, Age, Income, Outage\_sec\_perweek, Email, Contacts, Yearly\_equip\_failure, Tenure, MonthlyCharge, Bandwidth\_GB\_Year, item1, item2, item3, item4, item5, item6, item7, item8.

The CaseOrder column aims at maintaining the order of the attributes in the data set (example: 1, 2, 3, or 4). The Consumer\_id column depicts unique set of characters for each consumer (example: K409198 or S120509). The interaction column depicts unique set of characters for each interaction with a customer for their various needs (example: aa90260b-4141-4a24-8e36-b04ce1f4f77b). The City column describes the towns customers live in based on their billing addresses (example: Detroit). The State column describes the states the consumers live in based on their billing addresses (example: MI). The County column describes the counties the consumers are located in based on their billing addresses (example: Wayne). The Zip column describes the zip codes the consumers are located in based on their billing addresses (example: 48211). The Lat column describes the latitudes of the addresses provided by the consumers based on their billing statements (example: 56.251). The Lng column describes the longitudes of the addresses provided by the consumers based on their billing statements (example: -133.376). The Population column describes the number of people that live within a mile radius of the consumers’ billing addresses (example: 76). The Area column describes the type of area the customers live in based on their billing statements and using the census data (example: urban). The Timezone column describes the time zones the consumers' homes are located in at the time of initial service sign-up and using the billing addresses provided on the billing statements (example: EST). The Job column describes the job titles of the consumers responsible for paying the telecommunication service bills (example: Cashier). The Children column describes the number of children in the consumers’ homes (example: 3). The Age column describes the consumers’ ages at the time of telecommunication services sign-up (example: 27). The Education column describes the highest levels of education reported by the customers at the time of telecommunication services sign-up (example: High School). The Employment column describes the consumers’ reported employment statuses at the time of telecommunication services sign-up (example: Full Time). The Income column describes the yearly incomes reported by consumers at the time of telecommunication services sign-up (example: 57,159). The Marital column describes the consumers’ reported marital statuses at the time of telecommunication services sign-up (example: Single). The Gender column describes the consumers’ reported genders at the time of telecommunication services sign-up (example: male). The Outage\_sec\_perweek column describes the weekly average number of seconds the telecommunication system was down in the consumers’ area (example: 397). The Email column counts the number of emails sent to the consumers in the last year (example: 72). The Contacts column counts the number of times the consumers call technical support in the last year (example: 72). The Yearly\_equip\_failure column counts the number of times the equipment failed and resulted in a reset of replacement in the last year (example:7). The Techie column indicates whether the consumers self-describe as technically savvy (example: yes). The Contract column reports consumers’ contract terms (example: month-to-month). The Port\_modem column indicates whether the consumers own portable modems (example: yes). The Tablet column indicates whether the consumers own tablets (example: yes). The InternetService column describes the type of internet providers the consumers subscribed to (example: fiber optic). The Phone column indicates whether the consumers subscribed to phone services (example: yes). The Multiple column indicates whether the consumers subscribed to multiple lines (example: yes). The OnlineSecurity column indicates whether the consumers subscribed to the online security feature (example: yes). The OnlineBackup column indicates whether the consumers subscribed to the online backup feature (example: yes). The DeviceProtection column indicates whether the consumers subscribed to device protection plans (example: yes). The TechSupport column indicates whether the consumers subscribed to the tech support feature (example: yes). The StreamingTV column indicates whether the consumers subscribed to TV plans (example: Yes). The StreamingMovies column indicates whether the consumers subscribed to the movies streaming feature (example: yes). The PaperlessBilling column indicates whether the consumers signed up for paperless billing (example: yes). The PaymentMethod column describes the consumers’ payment methods (example: credit card). The Tenure column indicates the number of months the consumers spent with the telecommunication service provider (example: 17). The MonthlyCharge column indicates the monthly charges billed to the consumers, computed as per customer average (example: 148.18). The Bandwidth\_GB\_Year column describes the consumers’ average GB in data used per year (example: 800.779). The item1 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important is a timely response (example: 1, 8, 9). The item2 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important are timely fixes (example: 1, 8, 9). The item3 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important are timely replacements (example: 1, 8, 9). The item4 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important is reliability (example: 1, 8, 9). The item5 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important are options (example: 1, 8, 9). The item6 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important is a respectful response (example: 1, 8, 9). The item7 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important is a courteous exchange (example: 1, 8, 9). The item8 column indicates on a scale of 1 ‘Most Important’ – 8 ‘Least Important’, how important is active listening (example: 1, 8, 9).

PART II- DATA CLEANING PLAN

C-1-

The following steps are my data cleaning plan:

1. Indexing the CaseOrder column,
2. Provide new names to ambiguous columns for a better understanding of the variables,
3. Evaluate the data structure using descriptive statistics,
4. Locate and analyze outliers in the data set: we computed the z-scores for each variable by subtracting each value in the column by its mean and then dividing the result by its standard deviation (Larose & Larose, 2019). The criteria used to identify outliers is that any computed z-score greater than 3 or less than -3 displays rows with outliers and we run a code provided in section C-4 to show the total number of rows with outliers,
5. Locate missing values in the data set: we used python’s library pandas syntaxes dataframe.isnull() and dataframe.isnull().sum() to locate the columns with missing values and display the number of values missing in each of these columns. The dataframe.isnull() syntax describes the columns with NA values and empty cells as having missing values. Its output yields boolean responses True or False with True indicating missing values and False otherwise. The dataframe.isnull().sum() syntax counts the total number of empty cells and NA fields in specific columns. We decided to impute these missing values with the median values. The median is defined as the middlemost value in any given variable when the attributes in that variable are arranged in ascending or descending order,
6. Detecting and removing duplicate values in rows: we used python’s library pandas syntax data\_duplicates = df.loc[df.duplicated()] print(data\_duplicates) to identify, flag, and possibly drop any duplicate values in rows. The output of this syntax displays the total number of rows with duplicate values,
7. Export the cleaned data set in a CSV format.

C-2-

The churn data set depicts several columns that allow for different data sorting options. For data cleaning purposes, maintaining the order of the variables in the original data set is very important for data analysis. Doing so would require the use of an index field. After examining our data set, we opted to use the column CaseOrder as the index field. The CaseOrder column displays a unique number for each row in the data set, sorted in ascending order.

The next step in our churn data cleaning process is to rename the ambiguous columns for a better understanding of the variables. We identified these columns as the columns depicting the consumers' satisfaction survey responses. These columns were named in the data set as item1, item2, and so on. We decided to rename these columns with names that better describe their different attributes.

As defined by Larose & Larose (2019), outliers are identified by analyzing the z-scores in variables. In this paper, we measured the z-scores and analyzed them in order to detect whether outliers are present in the variables. The z-scores are computed by subtracting each value in the column by its mean and then dividing by its standard deviation.

A quick review of the churn data set displays many columns, for instance Children and Age, with cells containing missing values or the character NA that indicates missing values. We used python’s library pandas syntaxes dataframe.isnull() and dataframe.isnull().sum() to locate the columns with missing values and display the number of values missing in each of these columns. Our justification for using these two syntaxes is that they are pretty straightforward, and their outputs return precisely the number of missing values in each column in the data set. The wide range of literature on data cleaning suggests using programmatic values such as the median, the mode, or the mean to impute the missing values (Lianne & Justin @ Just into Data, 2021). That is why in this paper, we decided to use the median to impute the missing values. In the children column, we imputed the missing values with the value zero because we assumed that the customers who did not report values for that column did not probably have children.

Detecting and removing duplicate rows is very important in our data cleaning process for ensuring the integrity of the data set. We used python’s library pandas syntax data\_duplicates = df.loc[df.duplicated()] print(data\_duplicates) to identify, flag, and possibly drop any duplicate values in rows. The justification for using this syntax is that its output displays the exact number of rows with duplicate values.

C-3-

In this research, we opted to use the python programming language over the R programming language because we had taken the introduction to python class on Udemy and had used python previously as part of a professional project at our place of employment. Python is one of the most popular programming languages in the world and has a diverse range of libraries for data analytics. Its syntaxes are also very easy to understand and use and can handle large amounts of data.

We installed python from the anaconda distribution package and will be using the Jupyter Notebook integrated development environment. To support our data cleaning process, we chose to use the following libraries:

1. Pandas: we used this library to import the data set into the python dataframe using its read attribute. We also used this library for its loc attribute which helped us to access rows and columns by the label in the dataframe's array,
2. Numpy: we used this library to perform mathematical operations such as computing descriptive statistics on arrays and to compute z-scores for different columns in order to detect outliers in the data set,
3. SKlearn: we used this library to perform the principal component analysis,
4. Matplotlib: We used this library to create and edit plots and for interactive visualizations,
5. Seaborn: we used this library concurrently with the Matplotlib library to visualize the data and detect outliers,
6. Scipy: We used this library for different technical analyses of the data set.

C-4-

Please find the following steps:

Step 1: Import needed libraries, the raw churn data set, and indexing the CaseOrder column

***Import certain python libraries***

import numpy as np

import pandas as pd

from scipy import stats

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import seaborn as sns

import matplotlib.pyplot as plt

***Code to import the csv data file***

df = pd.read\_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn\_raw\_data.csv')

print(df)

***indexing CaseOrder column***

df\_index = pd.read\_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn\_raw\_data.csv', usecols = ['CaseOrder',

'Customer\_id'])

print(df\_index)

print(df\_index["CaseOrder"])

type(df["CaseOrder"])

Step 2: Rename columns for better description of the variables

We rename certain columns in order to better describe them, the code used is the following:

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)

print(df)

***Display the list of Dataframe Columns***

df.columns

Step 3: Evaluate the data structure using descriptive statistics

***Describe Churn dataset statistics***

df.describe()

***Calculate Churn Rate***

df.Churn.value\_counts() / len(df)

***Identify the standard deviation of every numeric column in the data set***

df.std()

***Review data type***

df.dtypes

Step 4: Identify outliers in certain columns

We used the following codes:

***Detecting outliers in Email column code***

**Print histogram**

data\_frame = pd.read\_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn\_raw\_data.csv')

data\_frame['Email\_z'] = stats.zscore(data\_frame['Email'])

graph = pd.DataFrame({"Email\_z":data\_frame.loc[:,"Email\_z"]})

graph.hist()

**Displays rows with outliers**

Email\_z = (df['Email'] - df['Email'].mean()) /df['Email'].std()

data = df.loc[(Email\_z > 3) | (Email\_z < -3)]

print(data)

***Detecting outliers in Population column code***

**Print histogram**

data\_frame['Population\_z'] = stats.zscore(data\_frame['Population'])

graph = pd.DataFrame({"Population\_z":data\_frame.loc[:,"Population\_z"]})

graph.hist()

**Displays rows with outliers**

Population\_z = (df['Population'] - df['Population'].mean()) /df['Population'].std()

data = df.loc[(Population\_z > 3) | (Population\_z < -3)]

print(data)

***Detecting outliers in Children column code***

**Print histogram**

data\_frame['Children\_z'] = stats.zscore(data\_frame['Children'])

graph = pd.DataFrame({"Children\_z":data\_frame.loc[:,"Children\_z"]})

graph.hist()

**Displays rows with outliers**

Children\_z = (df['Children'] - df['Children'].mean()) /df['Children'].std()

data = df.loc[(Children\_z > 3) | (Children\_z < -3)]

print(data)

***Detecting outliers in Age column code***

**Print histogram**

data\_frame['Age\_z'] = stats.zscore(data\_frame['Age'])

graph = pd.DataFrame({"Age\_z":data\_frame.loc[:,"Age\_z"]})

graph.hist()

**Displays rows with outliers**

Age\_z = (df['Age'] - df['Age'].mean()) /df['Age'].std()

data = df.loc[(Age\_z > 3) | (Age\_z < -3)]

print(data)

***Detecting outliers in Income column code***

**Print histogram**

data\_frame['Income\_z'] = stats.zscore(data\_frame['Income'])

graph = pd.DataFrame({"Income\_z":data\_frame.loc[:,"Income\_z"]})

graph.hist()

**Displays rows with outliers**

Income\_z = (df['Income'] - df['Income'].mean()) /df['Income'].std()

data = df.loc[(Income\_z > 3) | (Income\_z < -3)]

print(data)

***Detecting outliers in Contacts column code***

**Print histogram**

data\_frame['Contacts\_z'] = stats.zscore(data\_frame['Contacts'])

graph = pd.DataFrame({"Contacts\_z":data\_frame.loc[:,"Contacts\_z"]})

graph.hist()

**Displays rows with outliers**

Contacts\_z = (df['Contacts'] - df['Contacts'].mean()) /df['Contacts'].std()

data = df.loc[(Contacts\_z > 3) | (Contacts\_z < -3)]

print(data)

***Detecting outliers in*** ***Yearly\_equip\_failure column code***

**Print histogram**

data\_frame['Yearly\_equip\_failure \_z'] = stats.zscore(data\_frame['Yearly\_equip\_failure'])

graph = pd.DataFrame({"Yearly\_equip\_failure \_z":data\_frame.loc[:,"Yearly\_equip\_failure \_z"]})

graph.hist()

**Displays rows with outliers**

Yearly\_equip\_failure\_z = (df['Yearly\_equip\_failure'] - df['Yearly\_equip\_failure'].mean()) /df['Yearly\_equip\_failure'].std()

data = df.loc[(Yearly\_equip\_failure\_z > 3) | (Yearly\_equip\_failure\_z < -3)]

print(data)

***Detecting outliers in Tenure column code***

**Print histogram**

data\_frame['Tenure\_z'] = stats.zscore(data\_frame['Tenure'])

graph = pd.DataFrame({"Tenure\_z":data\_frame.loc[:,"Tenure\_z"]})

graph.hist()

**Displays rows with outliers**

Tenure\_z = (df['Tenure'] - df['Tenure'].mean()) /df['Tenure'].std()

data = df.loc[(Tenure\_z > 3) | (Tenure\_z < -3)]

print(data)

***Detecting outliers in MonthlyCharge column code***

**Print histogram**

data\_frame['MonthlyCharge\_z'] = stats.zscore(data\_frame['MonthlyCharge'])

graph = pd.DataFrame({"MonthlyCharge\_z":data\_frame.loc[:,"MonthlyCharge\_z"]})

graph.hist()

**Displays rows with outliers**

MonthlyCharge\_z = (df['MonthlyCharge'] - df['MonthlyCharge'].mean()) /df['MonthlyCharge'].std()

data = df.loc[(MonthlyCharge\_z > 3) | (MonthlyCharge\_z < -3)]

print(data)

***Detecting outliers in Bandwidth\_GB\_Year outlier column code***

**Print histogram**

data\_frame['Bandwidth\_GB\_Year\_z'] = stats.zscore(data\_frame['Bandwidth\_GB\_Year'])

graph = pd.DataFrame({"Bandwidth\_GB\_Year\_z":data\_frame.loc[:,"Bandwidth\_GB\_Year\_z"]})

graph.hist()

**Displays rows with outliers**

Bandwidth\_GB\_Year\_z = (df['Bandwidth\_GB\_Year'] - df['Bandwidth\_GB\_Year'].mean()) /df['Bandwidth\_GB\_Year'].std()

data = df.loc[(Bandwidth\_GB\_Year\_z > 3) | (Bandwidth\_GB\_Year\_z < -3)]

print(data)

The histograms generate odd distributions,

***let us generate box plots for outliers analysis for Tenure, MonthlyCharge, Bandwidth\_GB\_Year columns***

df.boxplot(['Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year'])

plt.savefig('df\_boxplots.jpg')

A graph with lines and numbers

Description automatically generated with medium confidence

We can see clear indication of the presence of outliers,

***Let us box plot the MonthlyCharge seperately***

df.boxplot(['MonthlyCharge'])

plt.savefig('df\_boxplots.jpg')

A graph with a line and a square

Description automatically generated

Step 5: Locate missing values in the data set and impute these missing values

***Find missing values***

df.isnull()

***Locate rows from containing missing values***

df.isnull().any(axis=1)

We noticed several empty fields as they returned “True”, and most of these empty fields are in columns "Children", "Age", "Income", "Techie", "Phone", "Tenure".

***Display the columns with missing values***

df.isna().any()

The output displayed eight different columns with missing values. These columns are: Children, Age, Techie, Phone, TechSupport, Income, Tenure and Bandwidth\_GB\_Year. The missing values in these columns are less than 40 % of the size of the attributes in these columns. So, we decided to estimate these missing values by replacing them with the median values.

***Count the number of missing values in columns***

data\_nulls = df.isnull().sum()

print(data\_nulls)

***Impute the missing values for the variables Children, Age, Income, Tenure and Bandwidth\_GB\_Year with the median and the variables Techie, Phone, TechSupport with "No"***

df['Children'].fillna(0, inplace=True)

df['Age'].fillna(df['Age'].median(), inplace=True)

df['Income'].fillna(df['Income'].median(), inplace=True)

df['Bandwidth\_GB\_Year'].fillna(df['Bandwidth\_GB\_Year'].median(), inplace=True)

df['Tenure'].fillna(df['Tenure'].median(), inplace=True)

df['Techie'].fillna('No', inplace=True)

df['Phone'].fillna('No', inplace=True)

df['TechSupport'].fillna('No', inplace=True)

print(df)

Step 6: Detecting and removing duplicate values in rows

data\_duplicates = df.loc[df.duplicated()]

print(data\_duplicates)

Step 7: Export the cleaned data

This is our final step in the cleaning process of the churn data set, we used the following code to export the cleaned data as a CSV file:

df.to\_csv(r'C:\Users\richa\OneDrive\Desktop\d206\cleaned\_churn\_data.csv', index=False, header=True)

print(df)

PART-III – DATA CLEANING

D-1- The first step in cleaning up the data set was to set up an index field using the CaseOrder column, our intention in doing so was to try to keep the order in the data set in the event we use another variable to filter the data. Our next step was to rename the columns that had unclear names for a better understanding of the data set. These columns were mainly the columns depicting customers’ satisfaction survey responses. After that, we provided a few descriptive statistics for the data set.

Next, we identified and explored outliers in the data set. From the calculated standard deviation in the descriptive statistics step, we observed that certain columns displayed rows with possible outliers. We further analyzed these outliers by computing the z-scores for these columns. Also, we displayed rows that contain outliers in these columns using the code provided in C-4. The literature suggests that any row with z-score that is greater than 3 or less than -3 contains outliers. The computed z-scores showed that the Population, Income, MonthlyCharge, Tenure, bandwidth\_GB\_Year and other columns contain outliers. We further explored these outliers by generating the histograms, box plots, and the Seaborn boxplot bandwidth. For the purpose of this paper, we decided not to remove these outliers because these columns constitute very important independent variables for our analysis.

The next step was to identify columns with missing values and to impute these missing values. The result in this step yields 8 different columns with missing values. The column Children has 2495 missing values. Age has 2475 missing values. Income has 2490 missing values. Techie has 2477 missing values. Phone has 1026 missing values. TechSupport has 991 missing values. Tenure has 931 missing values and Banwidth\_GB\_Year has 1021 missing values. We decided to impute the columns Age, Tenure, Income, Banwidth\_GB\_Year with the median values of these columns because they depict quantitative variables and the columns Techie, Phone, TechSupport with the value “No” because they depict qualitative variables taking either the values “Yes” or “No”; “No” being the most frequent value observed in these columns. We imputed the missing values in the column Children with the value “0” because the assumed that customers who did not report values for this column did not probably have children.

The final step before exporting the cleaned data set in a CSV format was to detect and remove duplicate values in rows. The result yielded no rows with duplicate values.

D-2- The main reason the CaseOrder column was used as an index field is due to the size of the data set. The data set depicts 10 000 unique records in ascending order. Indexing the CaseOrder column is an adequate and efficient manner of keeping the data organized. The other columns that had missing values were imputed with their median values. Using the median as an estimator to impute missing values is a very popular tool suggested by the literature and used by many scientists because it helps to better comprehend the different patterns in the data set. The variables that display outliers were retained for our analysis and they did not seem to significantly impact our study.

D-3-

The first step, which involves indexing the CaseOrder column helped us to maintain order in the data set.

The second step involves renaming the columns for a better understanding of the data set. Doing so was very important because certain columns, especially the consumers’ survey responses have unclear names that could lead to misinterpretation of the data set.

In the third step, we evaluated the data structure using descriptive statistics. We also computed the standard deviation of every numerical column in the data set. Doing so helped us to explore the presence of outliers in these columns.

In the fourth step, we identified and explored outliers by running codes that display the number of rows with outliers and also by using histograms. As we know it, outliers not properly dealt with could lead to faulty predictive modeling of the data set. We decided to keep the variables that display outliers because these variables are essential variables for our analysis and keeping them did not seem to be significant.

Finally, the last two steps involved identifying the columns with missing values and the rows with duplicate values. We found eight columns that have missing values and no rows with duplicate values. The missing values in these columns represented less than 40 % of the attributes, so we decided to impute these missing values with their median values.

D-4- See Section C-4, the pdf attachment and the panopto video

D-5- see the attached CSV file

D-6- In our data cleaning process, we used the median to impute the missing values. One of the problems that come with using the median to impute the values of the missing variables is its inaccuracy. The median is defined as the middle value in the list of attributes in a given column when arranged in ascending or descending order. It is critical to understand that imputing the missing values with the median could lead to interpretation bias as it assumes that the missing values are similar to the non-missing values.

In the churn data set, the use of the median values to impute missing values means the middlemost values in each column with missing values were used. This can cause serious predictive modeling interpretation issues because the use of the median may overstate certain attributes in the data set and could lead to inaccurate predictive modeling outcomes and misleading dashboard insights provided to stakeholders.

In spite of the disadvantages of using the median values to impute the missing values in the data set, it is still our best methodology for cleaning the data set given that the next alternative would be to drop the columns with missing values, which may lead to missing out on the attributes of the variables depicted in these columns in the predictive modeling.

D-7- It is common knowledge that most databases have data quality issues and one of the issues most frequently encountered by data analysts is the imputation of the missing values, especially when the missing values are located in columns that are seen important to the analysis. In the case of the churn data set, the missing values in the columns amounted to less than 40 % of the data, so imputing these missing values with the median values was our best option. However, certain columns containing missing values may prove very important to the predictive modeling. For instance, the age column may be very important for our analysis because certain customers of a certain age may have been making their churn decisions based on their past experiences with telecommunication service providers. Younger customers may be dependent on their parents when it comes to churn decision making while older customers may be making their churn decision based on past experiences. Imputing the missing values in the age column based on its median value may lead to a misleading interpretation of churn predictive modeling and inaccurate dashboard insights to stakeholders.

One of the disadvantages of using the median to impute missing values is that it could skew the results. Using the median imputes the missing values by selecting the middlemost attributes in the variables and can be seen as a limitation because it could overstate the attributes that are located in the middle of the data set when they are arranged in ascending or descending order.

E- Principal Component Analysis

E-1 – Please find the following codes depicting the different steps in establishing our principal component analysis:

***Step 1 : We reload the clean data set***

df = pd.read\_csv(r'C:\Users\richa\OneDrive\Desktop\d206\cleaned\_churn\_data.csv')

print(df)

***Step 2 : We selected the last eleven service related columns***

data = df.loc[:, 'Tenure':'Active\_listening']

print(data)

***Step 3 : We Normalize the data***

churn\_normalized = (data - data.mean()) / data.std()

***Step 4 : We select number of components to extract***

pca = PCA(n\_components = data.shape[1])

***Step 5 : We create a list of PCA names***

churn\_numeric = data[['Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year', 'Timely\_response',

'Timely\_fixes', 'Timely\_replacements', 'Reliability', 'options',

'Respectful\_response', 'Courteous\_exchange', 'Active\_listening']]

pcs\_names = []

for i, col in enumerate(churn\_numeric.columns):

pcs\_names.append('PC' + str(i + 1))

print(pcs\_names)

['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11']

***Step 6 : We Call PCA application and convert the dataset of 11 variables into a dataset of 11 components***

pca.fit(churn\_normalized)

churn\_pca = pd.DataFrame(pca.transform(churn\_normalized),

columns = pcs\_names)

***Step 7 : We run the scree plot***

plt.plot(pca.explained\_variance\_ratio\_)

plt.xlabel('Number of Components')

plt.ylabel('Explained Variance')

plt.show();

A graph with a line

Description automatically generated

***Step 8 : We Extract the eigenvalues***

cov\_matrix = np.dot(churn\_normalized.T, churn\_normalized) / data.shape[0]

eigenvalues = [np.dot(eigenvector.T, np.dot(cov\_matrix, eigenvector)) for eigenvector in pca.components\_]

***Step 9: We plot the eigenvalues***

plt.plot(eigenvalues)

plt.xlabel('Number of Components')

plt.ylabel('Eigenvalue')

plt.show();

A graph with a line

Description automatically generated

***Step 10 : We select the fewest components***

for pc, var in zip(pcs\_names, np.cumsum(pca.explained\_variance\_ratio\_)):

print(pc, var)

A screenshot of a computer program

Description automatically generated

As we can see from the above screenshot, 86 % of the variance is explained by 7 components

***Step 11 : We created a rotation***

rotation = pd.DataFrame(pca.components\_.T, columns = pcs\_names, index = churn\_numeric.columns)

print(rotation)

A screenshot of a computer

Description automatically generated

***Step 12 : We generated output loadings for components***

loadings = pd.DataFrame(pca.components\_.T,

columns = pcs\_names,

index = data.columns)

loadings

A table with numbers and symbols

Description automatically generated

***Step 13 : We extract reduced dataset & print 3 components***

churn\_reduced = churn\_pca.iloc[ : , 0:3]

print(churn\_reduced)

A screenshot of a computer program

Description automatically generated

E-2-

We selected the principal components for our analysis based on our understanding of the data set from the consumers’ viewpoint. Telecommunication service consumers highly rate the quality of services provided to them. Since the customers satisfaction survey columns depict numeric predictors of customers' churn decision-making process, we selected all of them as variables for our principal component analysis. We also included Tenure, MonthlyCharge and Yearly GB in our analysis because we anticipated that these variables are significant in consumers’ churn decision-making process.

In our principal component analytical process, we selected 11 variables from the original data set and visually represented these variables using eigenvalues scree plots. Lewith et al., (2010) defined a scree plot as a line plot of the eigenvalues of principal components in an analysis. In general, scree plots are used to identify the number of components to retain in a principal component analysis. Dmitrienko et al., (2017) emphasized that the "elbow" of the graph where the eigenvalues seem to level off is observed and components to the left of this point should be retained as significant. In the case of our analysis, the elbow bent at about 3. Therefore, we retained the first three components and printed them out. These components are: Tenure, MonthlyCharge, and Bandwidth\_GB\_Year. Also, these 3 components we selected satisfy the Kaiser rule of picking components with eigenvalues of at least 1.

E-3- We identified two main benefits of principal component analysis. The first one is the reduction of the number of variables in the predictive modeling. Most databases have vast amounts of data that encompass several variables depicting a wide range of attributes and the data analyst’s main job is selecting relevant dependent and independent variables for analysis; principal components can come handy in helping in variable reduction. Our principal component analysis eigenvalues scree plot suggests that the variables Tenure, MonthlyCharge, and Bandwidth\_GB\_Year are significant.

The second benefit of principal component analysis is agile machine learning. The machine learning algorithm is more agile when the process involves the use of principal component analysis.

F- Panopto video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7c06e9b0-9189-4b86-aa67-b0590117a661>

G- Web Source References

Lianne & Justin @ Just into Data. (2021, May 3). Data Cleaning in python: The ultimate guide (2020). Medium. Retrieved September 23, 2021, from <https://towardsdatascience.com/data-cleaning-in-python-the-ultimate-guide-2020-c63b88bf0a0d>.

H- References

Dmitrienko, Alex; Chuang-Stein, Christy; D'Agostino, Ralph B. (2007). Pharmaceutical Statistics Using SAS: A Practical Guide. SAS Institute. p. 380. ISBN 978-1-59994-357-2.

Larose, C. D., & Larose, D. T. (2019). Data Science using python and R. Wiley.

Lewith, George Thomas; Jonas, Wayne B.; Walach, Harald (2010). Clinical Research in Complementary Therapies: Principles, Problems and Solutions. Elsevier Health Sciences. p. 354. ISBN 978-0-7020-4916-3.