WGU D212 PA 2v2

Andrew Shrestha

Part I: Research Question

A1. Question: Are we able to identify and quantify the risk at which customers will churn on their service provider through the identification of relationships with regards to specific variables? This will be answered through the use of the principal component analysis whereby we use the significant variable features found through this process relating to churn decisions by customers in order to aid stakeholder decision making.

A2. Objectives and Goals: The goal of this research analysis is to add a greater insight into stakeholder decision making with the aim in increasing customer loyalty to the company. With out results we are able to provide the specific characteristics and features customers have that ultimately influence their decision on whether to churn or not. This knowledge will thus allow stakeholders to capitalize and implement decisions and policies to address these specific features in order to reduce customer incentives to churn.

Part II: Method Justification

B1. Explanation of PCA Method

The Principle Component Analysis is a reduction method that is used to reduce the dimensionality of large data sets and transforming them to smaller data sets that still retains the majority of significant information from the original. In our case, it will eliminate all of the unnecessary or insignificant customer features in our data set and leave us with those that are of significance in terms of customer churn (**Zakaria Jaadi 2021).**

The PCA process follows these 5 steps:

1) Standardize the range of continuous initial variables

2) Compute the covariance matrix to identify correlations

3) Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

4) Create a feature vector to decide which components to keep

5) Recast the data along the principal component's axes

B2. Summary of One Assumption for PCA

One assumption to acknowledge when using the PCA is the fact that we are assuming the all the principle components with high variance must be significant, while the lower variance are insignificant and can be considered as “noise” in the data **(Pavan Vadapalli 2020).**

Part III: Data Preparation

C1. Identification of Continuous Variables

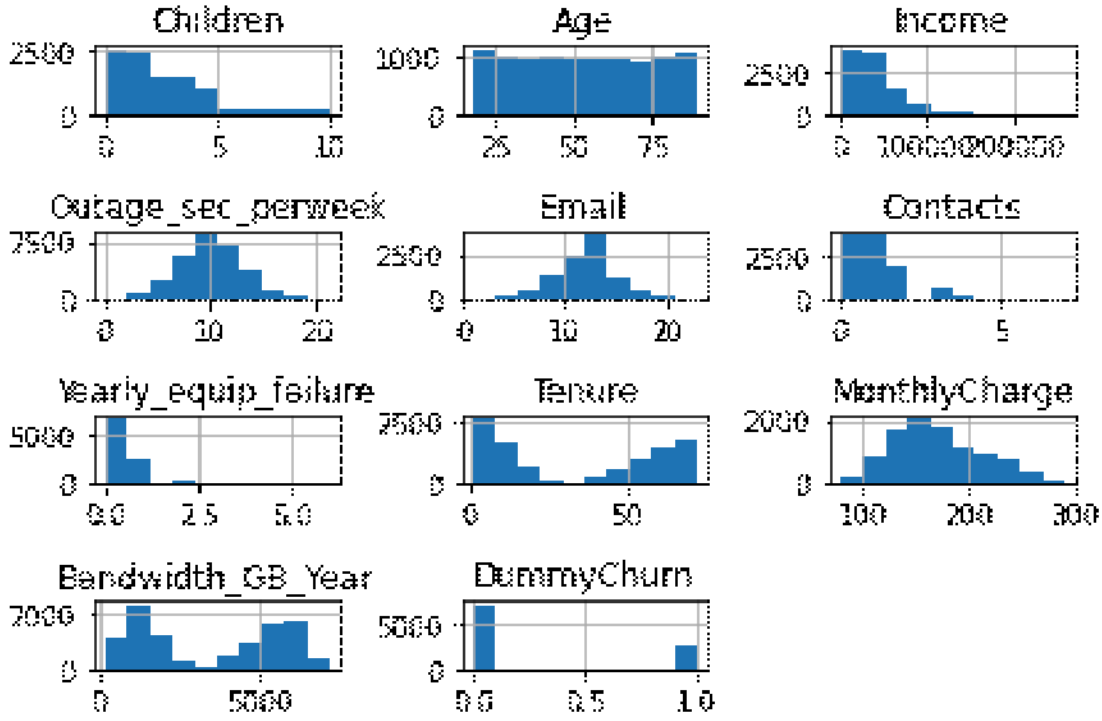
The Variables listed below are the ones that are needed to answer our PCA question. These are continuous variables that we are taking into consideration due to their influence on our target result of Churn. Since the Churn variable is a binary categorical variable, along with cleaning/prepping the data set, we are in need to transforming these continuous variables with dummy variables that take on binary 1 or 0 results as well.

I. Children
II. Age
III. Income
IV. Outage_Sec_Perweek
V. Email
VI. Contacts
VII. Yearly_Equip_Failure
VIII. Tenure
IX. MonthlyCharge
X. Bandwidth_GB_Year

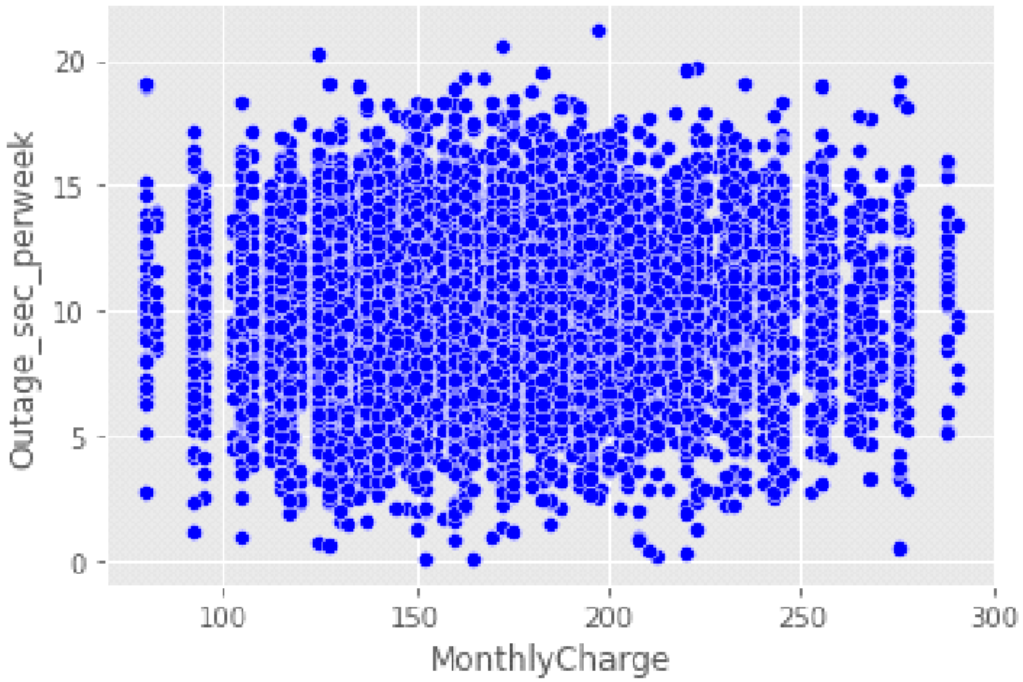
```
#Importing the standard Python libraries for Analysis and Visualizations
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

#Visuals
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
%matplotlib inline

# Scikit-Learn
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import IncrementalPCA
from sklearn.cluster import KMeans
from sklearn import metrics

# PCA Application
from sklearn.decomposition import PCA

# Feature Scaling
import scipy
from scipy.cluster.vq import whiten


#Load the CSV data set file into our Pandas Dataframe
churn_clean_df =
pd.read_csv(r"C:\Users\andre\OneDrive\Desktop\churn_clean.csv")


#now we want to get a surface level understanding of our data that we just
imported via basic statistical analysis
churn_clean_df.shape


(10000, 50)


churn_clean_df.columns


Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'J
ob',
       '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',
```

```
churn_clean_df.head()


churn_clean_df.info


churn_clean_df.describe()


churn_clean_df.dtypes


# Assign binary categorical variable with Dummy Values where Churn = 1
churn_clean_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in
churn_clean_df['Churn']]


# Drop categorical variable "churn" from the data
churn_clean_df = churn_clean_df.drop(columns=['Churn'])


# Remove less significant non-numerical categorical variables thus leaving
dataset with only numerical variables
churn_clean_df = churn_clean_df.drop(columns=['CaseOrder', 'Customer_id',
'Interaction', 'UID', 'City', 'State',
                                    'County', 'Zip', 'Lat', 'Lng', 'Area',
'TimeZone',
                                    'Job', 'Marital', 'PaymentMethod',
'Gender', 'Techie',
                                    'Contract', 'Port_modem', 'Tablet',
                                    'InternetService', 'Phone', 'Multiple',
'OnlineSecurity',
                                    'OnlineBackup', 'DeviceProtection',
'TechSupport',
                                    'StreamingTV', 'StreamingMovies',
'PaperlessBilling',
                                    'Item2', 'Item3', 'Item4', 'Item5',
'Item6', 'Item7',
                                    'Item8'])


# Set Dummychurn as target
churn_clean_df = churn_clean_df[['Children', 'Age', 'Income',
'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
'Bandwidth_GB_Year', 'DummyChurn']]


churn_clean_df.head()


churn_clean_df.columns
```

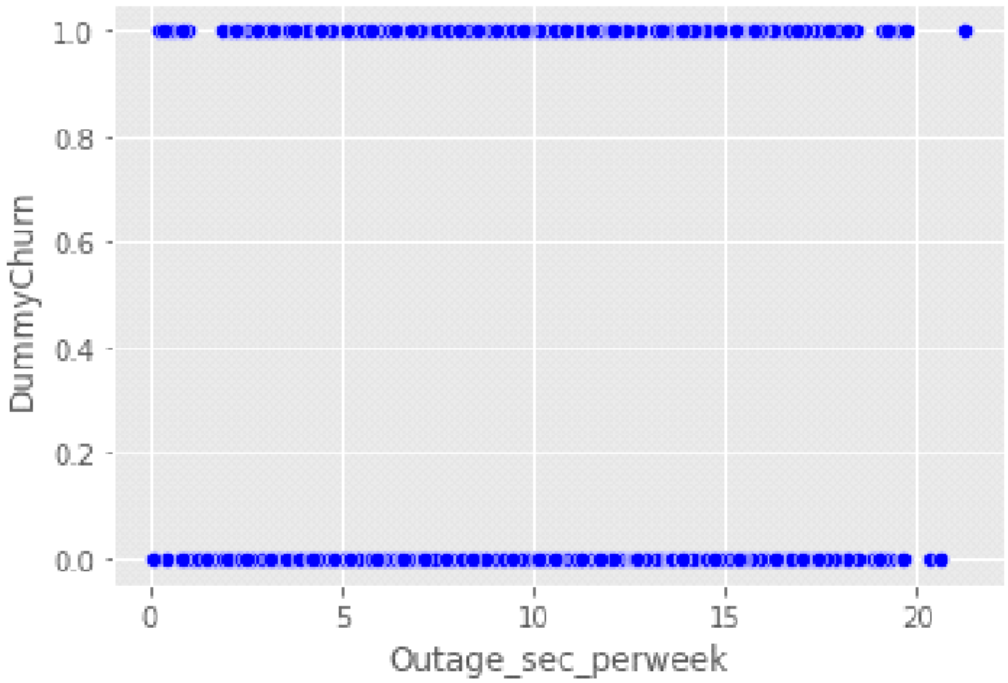
```
# Visuals of contiuous variables & categorical variables
churn_clean_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
          'Bandwidth_GB_Year', 'DummyChurn']].hist()
plt.tight_layout()
```



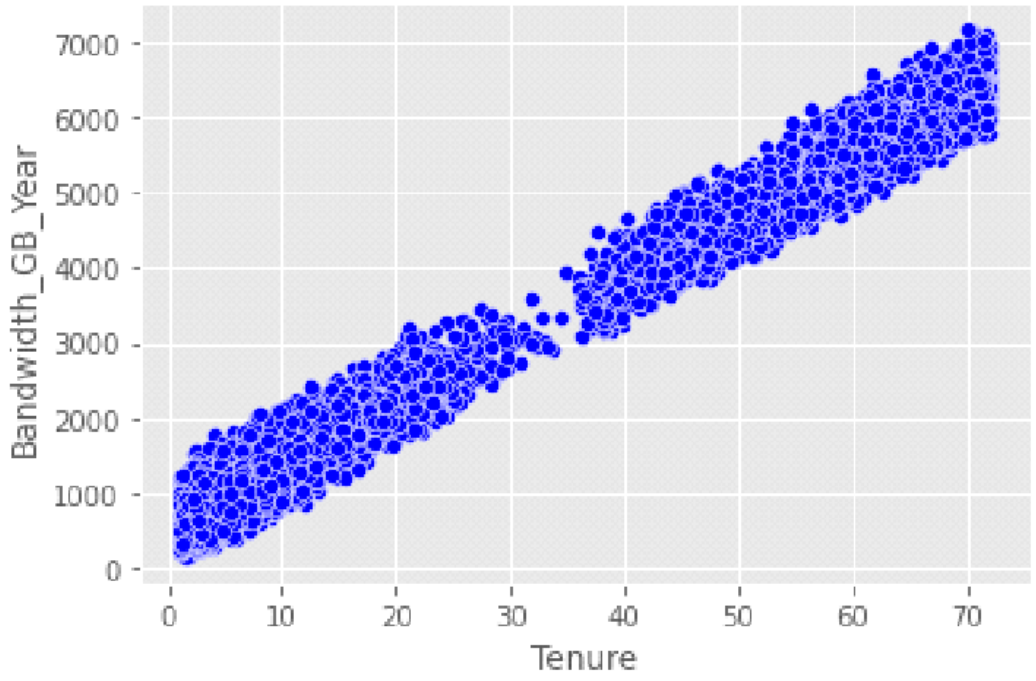

```
#Create scatterplot visuals to view if there are correlations between
hypothesized relationship of choosen variables
plt.style.use('ggplot')
sns.scatterplot(x=churn_clean_df['MonthlyCharge'],
y=churn_clean_df['Outage_sec_perweek'], color='blue')
plt.show();
```




```
sns.scatterplot(x=churn_clean_df['Outage_sec_perweek'],
y=churn_clean_df['DummyChurn'], color='blue')
plt.show();
```



```
sns.scatterplot(x=churn_clean_df['Tenure'],
y=churn_clean_df['Bandwidth_GB_Year'], color='blue')
plt.show();
```




```
# Checking the data set for any missing values or anomolies
data_nulls = churn_clean_df.isnull().sum()
print(data_nulls)
```

```
Children                0
Age                     0
Income                  0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
DummyChurn              0
dtype: int64
```

```
# Showing a list of features we will be using for our analysis
features = (list(churn_clean_df.columns[:-1]))
print(features)
```

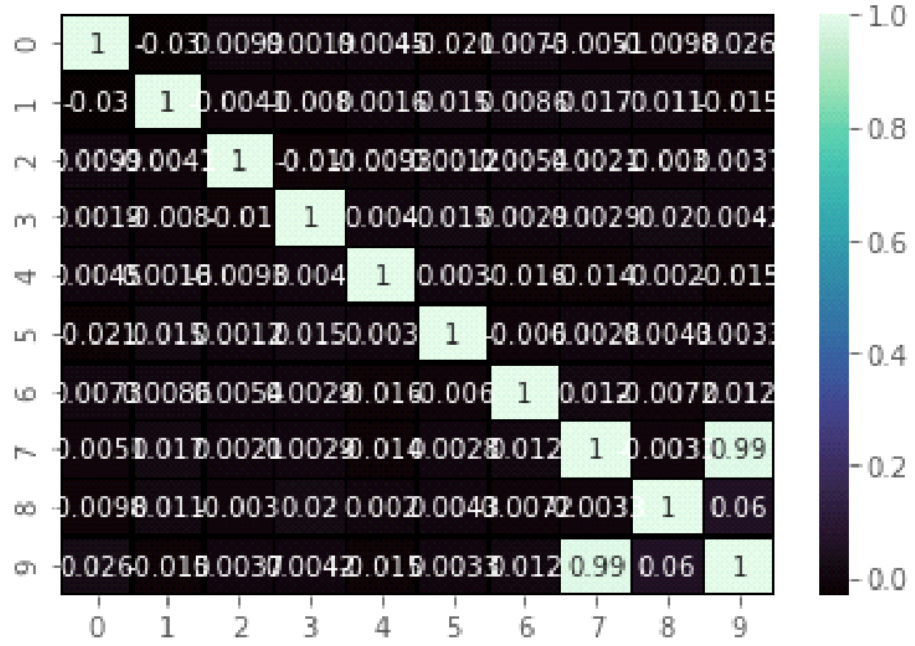
```
['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', '
Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year']
```

```
# Extract our Cleaned dataset
churn_clean_df.to_csv('data_clean_churn_pca.csv')
```

C2. Standardize the Continuous Data Set Variables

```
# Load our extracted cleanded dataset
churn_df = pd.read_csv('data_clean_churn_pca.csv')


#We then define our feature matrix by X
X = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
'Contacts', 'Yearly_equip_failure', 'Tenure',
'MonthlyCharge','Bandwidth_GB_Year']]


# Import the StandardScaler
from sklearn.preprocessing import StandardScaler

# Standardize the feature data
X_standardized = StandardScaler().fit_transform(X)


# Create our covariance matrix
va = np.mean(X_standardized, axis=0)
covariance_matrix = (X_standardized - va).T.dot((X_standardized - va)) /
(X_standardized.shape[0] - 1)

# Print covariance matrix
print('Covariance matrix: \n%s' %covariance_matrix)


Covariance matrix:
[[ 1.00010001 -0.02973451  0.00994335  0.00188944  0.00447925 -0.02077811
   0.00732132 -0.00509183 -0.00978238  0.02558738]
 [-0.02973451  1.00010001 -0.00409101 -0.00804752  0.00158808  0.01506913
   0.00857821  0.01698097  0.01072958 -0.01472512]
 [ 0.00994335 -0.00409101  1.00010001 -0.01001155 -0.00926842  0.00123332
   0.00542382  0.00211458 -0.00301427  0.00367392]
 [ 0.00188944 -0.00804752 -0.01001155  1.00010001  0.00399413  0.01509319
   0.00290902  0.00293225  0.02049812  0.00417608]
 [ 0.00447925  0.00158808 -0.00926842  0.00399413  1.00010001  0.00304067
  -0.01635598 -0.01446932  0.00199675 -0.01458061]
 [-0.02077811  0.01506913  0.00123332  0.01509319  0.00304067  1.00010001
  -0.00603285  0.00282037  0.00425907  0.00329905]
 [ 0.00732132  0.00857821  0.00542382  0.00290902 -0.01635598 -0.00603285
   1.00010001  0.01243615 -0.00717299  0.0120349 ]
 [-0.00509183  0.01698097  0.00211458  0.00293225 -0.01446932  0.00282037
   0.01243615  1.00010001 -0.00333714  0.99159435]
 [-0.00978238  0.01072958 -0.00301427  0.02049812  0.00199675  0.00425907
  -0.00717299 -0.00333714  1.00010001  0.06041247]
 [ 0.02558738 -0.01472512  0.00367392  0.00417608 -0.01458061  0.00329905
   0.0120349   0.99159435  0.06041247  1.00010001]]
```

```
# Visualize our features scaling via heatmap
sns.heatmap(covariance_matrix, annot=True, cmap="mako", linecolor='black',
linewidths=0.5)
plt.show()
```




```
# Eigendecomposition on covariance matrix
covariance_matrix = np.cov(X_standardized.T)
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)

# Print Eigenvectors and Eigenvalues
print('Eigenvectors: \n%s' %eigen_vectors)
print('Eigenvalues: \n%s' %eigen_values)
```

```
Eigenvectors:
[[ 2.15854596e-02  1.41347924e-02 -5.59467157e-01 -2.82399326e-01
  -6.46748662e-01 -2.85318727e-01  1.41418217e-01 -2.87326245e-01
   5.77211536e-02  3.16792374e-02]
 [-2.23657297e-02  1.70801624e-03  4.79835590e-01 -5.78528649e-01
  -2.07964687e-01  4.21944284e-01 -8.98051752e-02 -4.05096045e-01
  -1.25005511e-01 -1.59620872e-01]
 [ 9.35369421e-04  4.35978315e-03 -2.23932319e-01 -9.07206677e-02
   3.02723086e-01  2.67257143e-01  1.66467676e-01 -2.94875246e-01
  -2.10454046e-01  7.87135785e-01]
 [-2.80743720e-04  5.88358241e-03  2.12259615e-01 -4.42194433e-01
   3.67329262e-01 -4.79537437e-01  5.78437841e-01  1.69773973e-03
   2.43383022e-01 -2.56863653e-02]
 [-2.46034405e-04 -2.07788587e-02  1.07066510e-01  2.05475213e-01
   2.29615135e-01 -4.38464782e-01 -4.54311812e-01 -6.86127907e-01
   1.53996990e-01 -4.96007074e-03]
 [ 9.42747188e-04  4.17502587e-03  4.58770120e-01  2.54312989e-01
  -4.38267152e-01  1.38442926e-02  1.04530277e-01  4.31843019e-02
   5.50932285e-01  4.65025757e-01]
 [ 9.52581748e-05  1.75653215e-02 -1.43554702e-01  4.08175882e-01
   7.89968226e-02  3.95130635e-01  5.30963217e-01 -4.24544209e-01
   2.27787102e-01 -3.68863854e-01]
 [ 7.05262361e-01  7.05422257e-01  1.85082253e-03 -2.22443617e-02
   2.97190659e-02  2.10784846e-02 -4.17351931e-02  4.47130889e-03
   3.70435709e-02 -4.96324517e-03]
 [ 4.57545853e-02  4.04234226e-02  3.44887052e-01  3.28189805e-01
  -2.44887367e-01 -2.99619131e-01  3.29363949e-01 -1.16153637e-01
  -7.04988074e-01  2.99153688e-02]
 [-7.06783878e-01  7.06916770e-01 -7.92224048e-03  9.11019719e-03
   2.31786341e-04 -1.96605152e-02 -1.28030621e-02  8.34585697e-04
  -2.61919415e-03  4.62684898e-03]]
Eigenvalues:
[0.0054677  1.99433311 1.05333463 0.96035059 0.96476747 1.02755391
 1.01255858 0.98905858 0.99377999 0.99979555]
```

Part IV: Analysis

D1. Matrix of all Principal Components

```
# List descending sorted Eigenvalues
eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i]) for i in
range(len(eigen_values))]
eigen_pairs.sort(key=lambda x: x[0], reverse=True)

# Print list of Eigenvalues, descending
print('Eigenvalues:')
for i in eigen_pairs:
    print(i[0])
```

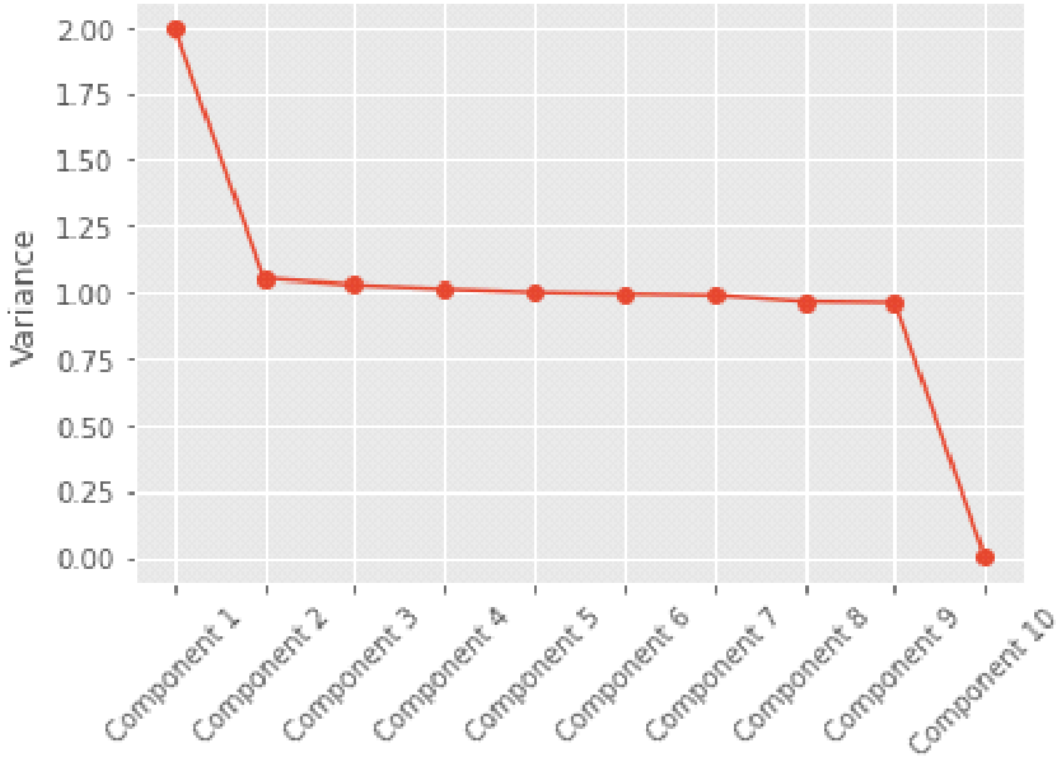
```
Eigenvalues:
1.9943331101364756
1.0533346256283929
1.0275539108057472
1.0125585769497125
0.9997955481739443
0.9937799880491925
0.9890585843412322
0.9647674706143334
0.9603505880457052
0.005467697265331806
```

```
# Fit standardized features to PCA
pca = PCA().fit(X_standardized)

# Print explained variance ratio
print(pca.explained_variance_ratio_)
```

```
[0.19941337 0.10532293 0.10274512 0.10124573 0.09996956 0.09936806
 0.09889597 0.0964671  0.09602546 0.00054672]
```

D2. Identification of Total Number of Components

```
# scree plot
def screeplot(pca, standardized_values):
    y = np.std(pca.transform(standardized_values), axis=0)**2
    x = np.arange(len(y)) + 1
    plt.plot(x, y, "o-")
    plt.xticks(x, ['Component ' + str(i) for i in x], rotation=45)
    plt.ylabel('Variance')
    plt.show()

# Visualize scree plot
screeplot(pca, X_standardized)
```



D3. Variance of each Principal Component

```
# List importance of components
def pca_summary(pca, standardized_data, out=True):
    names = ['PC ' + str(i) for i in range(1,
len(pca.explained_variance_ratio_) + 1)]
    a = list(np.std(pca.transform(standardized_data), axis = 0))
    b = list(pca.explained_variance_ratio_)
    c = [np.sum(pca.explained_variance_ratio_[:i]) for i in range(1,
len(pca.explained_variance_ratio_) + 1)]
    columns = pd.MultiIndex.from_tuples([('standard_deviation', 'Standard
Deviation'),
                                         ('proportion_of_variation',
'Proportion of Variation'),
                                         ('cumulative_proportion',
'Cumulative Proportion')])
    summary = pd.DataFrame(zip(a, b, c), index=names, columns=columns)
    if out:
        print('Component importance:')
    return summary

# Display summary
summary = pca_summary(pca, X_standardized)
summary.standard_deviation**2
```

Component importance:

Out[39]:

	Standard Deviation
PC 1	1.994134
PC 2	1.053229
PC 3	1.027451
PC 4	1.012457
PC 5	0.999696
PC 6	0.993681
PC 7	0.988960
PC 8	0.964671

D4. Total Variance Captured by Principal Components

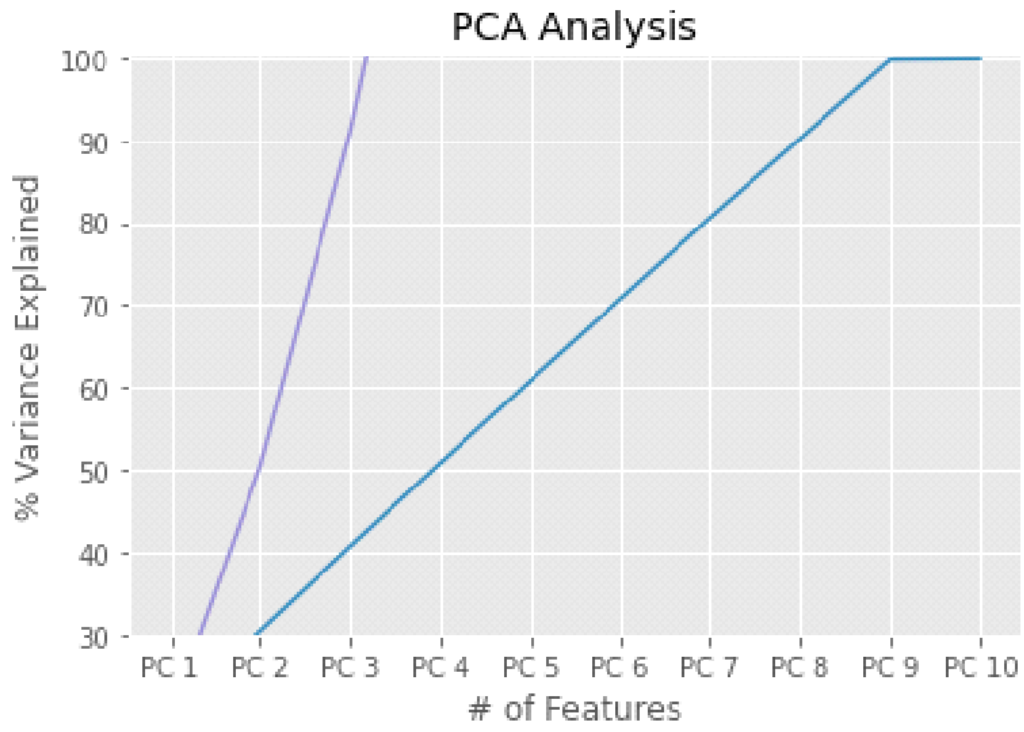
```
# Identify total variance captured by the principal components
np.sum(summary.standard_deviation**2)
```

```
Standard Deviation    10.0
dtype: float64
```

```
# Visualize total variance captured by components
var = np.cumsum(np.round(summary, decimals=3)*100)
```

```
plt.ylabel('% Variance Explained')
plt.xlabel('# of Features')
plt.title('PCA Analysis')
plt.ylim(30,100.5)
# plt.style.context('seaborn-whitegrid')
```

```
plt.plot(var)
```



D5. Results Summary of Data Analysis

As we see in our total variance of components graph above, 4 principal components are enough to explain about 50% of our variances. Linking this finding to our scree plot created earlier in this analysis, we can also see that these 4 components also display a variance of atleast 1.0 or greater.

When the number of components hit 9, we can explain 100% of the variance. Likewise, linking this finding to our scree plot, we can see that the components 5-9 display a leveled variance of 1.0 before taking a huge dip when component 10 in incorporated. This agrees with the statement that we only need 9 components to explain 100% of the variance.

The purpose of using our PCA analysis is to reduce or restrict the data we are working with to only significant variables or features that relate to customer churn.

Ultimately, we have found that there are 4 components that have a variance greater than 1.0. This is important as now the telecom company can utilize further analysis and extract more meaningful conclusions from these distinguished components in a bid to reduce customer churn. This is done by observing the commonalities among the features listed in the identified components of those customers that have chosen to churn and allocating resources to these factors in order to reduce churn.

Part V: Attachments

Panapto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6b5b3e44-36d4-4026-ad2a-ae9c005709e5

E. Third Party Codes

Michael Galarnyk(2017). PCA Using Python(scikit-learn).
https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

Statology(2021). How to Create a Scree Plot in Python (Step-by-Step).
https://www.statology.org/scree-plot-python/

F. Source References

Zakaria Jaadi(2021). A Step-by-Step Explanation of Principal Component Analysis (PCA).
https://builtin.com/data-science/step-step-explanation-principal-component-analysis

Pavan Vadapalli(2020). PCA in Machine Learning : Assumptions, Steps to Apply & Applications.
https://www.upgrad.com/blog/pca-in-machine-learning/#Assumptions_in_PCA