William Stults - Dimensionality Reduction Methods (D212 Task 2)

February 1, 2023

---

1 Part I: Research Question

1.1 Research Question

My data set for this data mining exercise includes data on a telco company's current and former subscribers, with an emphasis on customer churn (whether customers are maintaining or discontinuing their subscription to service). Data analysis performed on the dataset will be aimed with this research question in mind: how many principal components does the data set contain when using the continuous numerical data in the data set as input? Continuous numerical data will include numerical data which includes a measurable variable, rather than numerical data used as a label.

1.2 Objectives and Goals

Conclusions gleaned from the analysis of this data can benefit stakeholders by revealing information on how far the dimensionality of this data set might be reduced. Such information may be used to both reduce the feature set and potentially identify variables that exhibit covariance, indicating they may be related. My goal will be to determine how many principal compenents the continuous data contains, as well as the explained variance of each principal component.

---

2 Part II: Method Justification

2.1 Principal Component Analysis

Principal Component Analysis, or "PCA", is an unsupervised learning technique. It utilizes only continuous data and does not take into consideration any target variables. It is primarily a dimensionality reduction technique. It uses a covariance matrix to identify highly correlated features and represent those features as a smaller number of uncorrelated features. The algorithm continues this correlation reduction in an attempt to identify directions of maximum variance in the original data and projecting them onto a reduced dimensional dimensional space. The resulting components are called "principal components" (Pramoditha, 2020).

PCA assumes that there exists a correlation between the features in a data set. PCA will not be able to determine any principal components in a data set within which no correlation between features is present (Keboola, 2022).

The expected outcome will be a low number of principal components (which satisfy the Kaiser criterion) to which this original group of continuous variables can be reduced while still maintaining the correlation and variance characteristics of the original data.

---

3 Part III: Data Preparation

3.1 Data Preparation Goals and Data Manipulations

I would like my data to include only variables relevant to my research question, and to be clean and free of missing values and duplicate rows. PCA can only operate on continuous variables, so my first goal in data preparation is to make sure the data I will be working with contains no categorical data.

A list of the variables I will be using for my analysis is included below, along with their variable types and a brief description of each.

- Population - **continuous** - *Population within a mile radius of customer*
- Children - **continuous** - *Number of children in customer's household*
- Age - **continuous** - *Age of customer*
- Income - **continuous** - *Annual income of customer*
- Outage_sec_perweek - **continuous** - *Average number of seconds per week of system outages in the customer's neighborhood*
- Email - **continuous** - *Number of emails sent to the customer in the last year*
- Contacts - **continuous** - *Number of times customer contacted technical support*
- Yearly_equip_failure - **continuous** - *The number of times customer's equipment failed and had to be reset/replaced in the past year*
- Tenure - **continuous** - *Number of months the customer has stayed with the provider*
- MonthlyCharge - **continuous** - *The amount charged to the customer monthly*
- Bandwidth_GB_Year - **continuous** - *The average amount of data used, in GB, in a year by the customer*
- Item1: Timely response - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item2: Timely fixes - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item3: Timely replacements - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item4: Reliability - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item5: Options - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item6: Respectful response - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item7: Courteous exchange - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*
- Item8: Evidence of active listening - **continuous** - *survey response - scale of 1 to 8 (1 = most important, 8 = least important)*

My first steps will be to import the complete data set and execute functions that will give me information on its size and the data types of its variables. I will then narrow the data set to a new dataframe containing only the variables I am concerned with, and then utilize functions to determine if any null values or duplicate rows exist. By using the index_col parameter in my import I utilize CaseOrder, the data set's natural index column, as the index column in my pandas dataframe.

[1]:
```
# Imports and housekeeping
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[2]:
```
# Import the main dataset
df = pd.read_csv('churn_clean.csv', dtype={'locationid':np.int64},␣
↪index_col=[0])
```

[3]:
```
# Display data frame info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 49 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Customer_id           10000 non-null  object
 1   Interaction           10000 non-null  object
 2   UID                   10000 non-null  object
 3   City                  10000 non-null  object
 4   State                 10000 non-null  object
 5   County                10000 non-null  object
 6   Zip                   10000 non-null  int64
 7   Lat                   10000 non-null  float64
 8   Lng                   10000 non-null  float64
 9   Population            10000 non-null  int64
 10  Area                  10000 non-null  object
 11  TimeZone              10000 non-null  object
 12  Job                   10000 non-null  object
 13  Children              10000 non-null  int64
 14  Age                   10000 non-null  int64
 15  Income                10000 non-null  float64
 16  Marital               10000 non-null  object
 17  Gender                10000 non-null  object
 18  Churn                 10000 non-null  object
 19  Outage_sec_perweek    10000 non-null  float64
 20  Email                 10000 non-null  int64
 21  Contacts              10000 non-null  int64
```

```
 22  Yearly_equip_failure  10000 non-null  int64
 23  Techie                10000 non-null  object
 24  Contract              10000 non-null  object
 25  Port_modem            10000 non-null  object
 26  Tablet                10000 non-null  object
 27  InternetService       10000 non-null  object
 28  Phone                 10000 non-null  object
 29  Multiple              10000 non-null  object
 30  OnlineSecurity        10000 non-null  object
 31  OnlineBackup          10000 non-null  object
 32  DeviceProtection      10000 non-null  object
 33  TechSupport           10000 non-null  object
 34  StreamingTV           10000 non-null  object
 35  StreamingMovies       10000 non-null  object
 36  PaperlessBilling      10000 non-null  object
 37  PaymentMethod         10000 non-null  object
 38  Tenure                10000 non-null  float64
 39  MonthlyCharge         10000 non-null  float64
 40  Bandwidth_GB_Year     10000 non-null  float64
 41  Item1                 10000 non-null  int64
 42  Item2                 10000 non-null  int64
 43  Item3                 10000 non-null  int64
 44  Item4                 10000 non-null  int64
 45  Item5                 10000 non-null  int64
 46  Item6                 10000 non-null  int64
 47  Item7                 10000 non-null  int64
 48  Item8                 10000 non-null  int64
dtypes: float64(7), int64(15), object(27)
memory usage: 3.8+ MB
```

[4]:
```
# Display data frame top 5 rows
df.head()
```

[4]:
```
          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

                                        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

                          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  ...

          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

          Item6 Item7  Item8
CaseOrder
1             4     3      4
2             3     4      4
3             3     3      3
4             4     3      3
5             4     4      5

[5 rows x 49 columns]
```

[5]:
```python
# Trim data frame to variables relevant to research question
columns = ['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']
df_data = pd.DataFrame(df[columns])
# Store the data frame in variable 'X'
X = df_data
```

[6]:
```python
# Check data for null or missing values
df_data.isna().any()
```

[6]:
```
Population              False
Children                False
Age                     False
Income                  False
Outage_sec_perweek      False
Email                   False
```

```
Contacts                False
Yearly_equip_failure    False
Tenure                  False
MonthlyCharge           False
Bandwidth_GB_Year       False
Item1                   False
Item2                   False
Item3                   False
Item4                   False
Item5                   False
Item6                   False
Item7                   False
Item8                   False
dtype: bool
```

[7]:
```
# Check data for duplicated rows
df_data.duplicated().sum()
```

[7]:
```
0
```

[8]:
```
# Display new data frame top 5 rows
df_data.head()
```

[8]:
```
           Population  Children  Age    Income  Outage_sec_perweek  Email  \
CaseOrder
1                  38         0   68  28561.99            7.978323     10
2               10446         1   27  21704.77           11.699080     12
3                3735         4   50   9609.57           10.752800      9
4               13863         1   48  18925.23           14.913540     15
5               11352         0   83  40074.19            8.147417     16

           Contacts  Yearly_equip_failure     Tenure  MonthlyCharge  \
CaseOrder
1                 0                     1   6.795513     172.455519
2                 0                     1   1.156681     242.632554
3                 0                     1  15.754144     159.947583
4                 2                     0  17.087227     119.956840
5                 2                     1   1.670972     149.948316

           Bandwidth_GB_Year  Item1  Item2  Item3  Item4  Item5  Item6  Item7  \
CaseOrder
1                 904.536110      5      5      5      3      4      4      3
2                 800.982766      3      4      3      3      4      3      4
3                2054.706961      4      4      2      4      4      3      3
4                2164.579412      4      4      4      2      5      4      3
5                 271.493436      4      4      4      3      4      4      4
```

```
           Item8
CaseOrder
1              4
2              4
3              3
4              3
5              5
```

3.2 Summary Statistics

I can use the describe() function to display the summary statistics for the entire dataframe, as well as each variable I'll be evaluating for inclusion in the PCA exercise.

[9]:
```
# Display summary statistics for entire data frame
df_data.describe()
```

[9]:
```
         Population    Children            Age         Income  \
count  10000.000000  10000.0000  10000.000000   10000.000000
mean    9756.562400      2.0877     53.078400   39806.926771
std    14432.698671      2.1472     20.698882   28199.916702
min        0.000000      0.0000     18.000000     348.670000
25%      738.000000      0.0000     35.000000   19224.717500
50%     2910.500000      1.0000     53.000000   33170.605000
75%    13168.000000      3.0000     71.000000   53246.170000
max   111850.000000     10.0000     89.000000  258900.700000

       Outage_sec_perweek         Email      Contacts  Yearly_equip_failure  \
count        10000.000000  10000.000000  10000.000000          10000.000000
mean            10.001848     12.016000      0.994200              0.398000
std              2.976019      3.025898      0.988466              0.635953
min              0.099747      1.000000      0.000000              0.000000
25%              8.018214     10.000000      0.000000              0.000000
50%             10.018560     12.000000      1.000000              0.000000
75%             11.969485     14.000000      2.000000              1.000000
max             21.207230     23.000000      7.000000              6.000000

             Tenure  MonthlyCharge  Bandwidth_GB_Year         Item1  \
count  10000.000000   10000.000000       10000.000000  10000.000000
mean      34.526188     172.624816        3392.341550      3.490800
std       26.443063      42.943094        2185.294852      1.037797
min        1.000259      79.978860         155.506715      1.000000
25%        7.917694     139.979239        1236.470827      3.000000
50%       35.430507     167.484700        3279.536903      3.000000
75%       61.479795     200.734725        5586.141370      4.000000
max       71.999280     290.160419        7158.981530      7.000000
```

```
              Item2          Item3          Item4          Item5          Item6  \
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.505100      3.487000      3.497500      3.492900      3.497300
std        1.034641      1.027977      1.025816      1.024819      1.033586
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        4.000000      3.000000      3.000000      3.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        7.000000      8.000000      7.000000      7.000000      8.000000

              Item7          Item8
count  10000.000000  10000.000000
mean       3.509500      3.495600
std        1.028502      1.028633
min        1.000000      1.000000
25%        3.000000      3.000000
50%        4.000000      3.000000
75%        4.000000      4.000000
max        7.000000      8.000000
```

3.3 Further Preparation Steps

I will use the StandardScaler function to scale my variables for more accurate feature weighting. StandardScaler transforms each variable value to have a mean of 0 and a variance of 1. Once done, every variable value will fall between -1 and 1, and the data set values can be considered "standardized". The standardized data set is then assigned to variable "X_scaled".

[10]:
```
# Scaling continuous variables with StandardScaler
scaler = StandardScaler()
scaler.fit(X)
StandardScaler(copy=True, with_mean=True, with_std=True)
X_scaled = scaler.transform(X)
```

3.4 Copy of Prepared Data Set

Below is the code used to export the prepared data set to CSV format.

[11]:
```
df_prepared = pd.DataFrame(X_scaled, columns=df_data.columns)
# Export prepared dataframe to csv
df_prepared.to_csv(r'C:\Users\wstul\d212\churn_clean_prepared.csv')
```

---

4 Part IV: Analysis

4.1 Matrix of All Principal Components

To begin performing my PCA analysis of the data I instantiated a PCA model using the number of features in the original data set. The model is then fitted to the scaled data. The scaled data is then transformed using the PCA model and rendered as numbered principal components (PC1, PC2, etc.). A loadings matrix is then generated, displaying a weight value for each data set feature in each principal component.

[12]:
```
pca = PCA(n_components = X.shape[1])
pca.fit(X_scaled)
```

[12]:
```
PCA(n_components=19)
```

[13]:
```
df_matrix = pd.DataFrame(pca.transform(X_scaled), columns = ['PC1', 'PC2',␣
↪'PC3', 'PC4', 'PC5', 'PC6',
                                                            'PC7', 'PC8',␣
↪'PC9', 'PC10', 'PC11', 'PC12',
                                                            'PC13', 'PC14',␣
↪'PC15', 'PC16', 'PC17',
                                                            'PC18', 'PC19'])
```

[14]:
```
loadings = pd.DataFrame(pca.components_.T, columns = ['PC1', 'PC2', 'PC3',␣
↪'PC4', 'PC5', 'PC6',
                                                            'PC7', 'PC8',␣
↪'PC9', 'PC10', 'PC11', 'PC12',
                                                            'PC13', 'PC14',␣
↪'PC15', 'PC16', 'PC17',
                                                            'PC18', 'PC19'],␣
↪index = df_prepared.columns)
loadings
```

[14]:
```
                          PC1       PC2       PC3       PC4       PC5  \
Population          -0.002109 -0.005463  0.014732 -0.292151  0.264958
Children             0.004072  0.015862  0.028393  0.510569  0.345310
Age                  0.006459  0.000294 -0.029319 -0.455297 -0.417933
Income               0.001038  0.006035  0.025865  0.252065 -0.285030
Outage_sec_perweek  -0.017516  0.003927 -0.014363 -0.220115  0.339482
Email                0.008744 -0.020609 -0.003459 -0.190450  0.519450
Contacts            -0.008761  0.003318 -0.011853 -0.420731 -0.124577
Yearly_equip_failure -0.007688  0.017604  0.008199  0.167516 -0.373155
Tenure              -0.016320  0.702323 -0.063085 -0.005355 -0.007568
MonthlyCharge        0.000930  0.039858 -0.009499 -0.298690  0.113921
Bandwidth_GB_Year   -0.016845  0.703831 -0.062132  0.005068  0.022808
Item1                0.458670  0.031340  0.280974 -0.010952 -0.002667
Item2                0.433847  0.038755  0.282381 -0.020122 -0.002284
Item3                0.400488  0.035504  0.280527 -0.004108  0.013294
```

```
Item4                 0.145802 -0.039380 -0.568452  0.015142  0.001544
Item5                -0.175698  0.056278  0.587090 -0.038899 -0.023275
Item6                 0.405080 -0.006648 -0.183525 -0.000910  0.002245
Item7                 0.358246  0.002051 -0.181337  0.031194 -0.022614
Item8                 0.308733 -0.013634 -0.131655 -0.028435 -0.001573

                           PC6       PC7       PC8       PC9      PC10  \
Population            0.402355  0.355864  0.329128  0.161654  0.580378
Children             -0.089376  0.119069  0.226847  0.155912 -0.175953
Age                   0.183902  0.152752 -0.024113  0.346066 -0.180481
Income               -0.084983 -0.429611  0.581477  0.449649  0.219833
Outage_sec_perweek   -0.591284  0.273527  0.262607 -0.149557  0.125521
Email                 0.319498 -0.103117  0.170129  0.290785 -0.592268
Contacts             -0.146366 -0.275202  0.508824 -0.434373 -0.248703
Yearly_equip_failure -0.147092  0.686465  0.241921  0.114547 -0.334113
Tenure                0.048576  0.000016  0.007554 -0.028780 -0.001590
MonthlyCharge        -0.537631 -0.112559 -0.284655  0.562547  0.029519
Bandwidth_GB_Year     0.005063 -0.009132 -0.001688  0.001654  0.001271
Item1                -0.016377  0.018684 -0.013618 -0.012424 -0.012899
Item2                -0.022966  0.000550 -0.001653 -0.014345 -0.006536
Item3                 0.022177  0.008048 -0.032817 -0.017965 -0.011038
Item4                 0.013909  0.010808 -0.024875 -0.020863 -0.000102
Item5                 0.008693 -0.010701 -0.011610 -0.013280  0.001989
Item6                 0.002666  0.000918  0.025175  0.009562 -0.007608
Item7                 0.010414 -0.059311  0.048238 -0.001604 -0.022373
Item8                -0.034953  0.044129  0.010774  0.014507  0.097526

                          PC11      PC12      PC13      PC14      PC15  \
Population            0.167295  0.229022 -0.057130  0.019142 -0.016176
Children              0.655599 -0.241974  0.017020 -0.012250 -0.014011
Age                   0.234748 -0.590829 -0.045336  0.002513 -0.002495
Income               -0.252659 -0.057674 -0.020484 -0.079018 -0.007573
Outage_sec_perweek   -0.319263 -0.439536 -0.089844  0.016926 -0.008716
Email                -0.328652  0.061145  0.061158 -0.017175 -0.016342
Contacts              0.371468  0.241548  0.044032 -0.035285 -0.003279
Yearly_equip_failure -0.146136  0.365394  0.020739  0.006446 -0.015853
Tenure               -0.028600 -0.027147  0.005940 -0.003507  0.006548
MonthlyCharge         0.228176  0.375187 -0.005920  0.014551 -0.016508
Bandwidth_GB_Year    -0.000743  0.008874  0.010520 -0.003326  0.005612
Item1                 0.007946  0.022275 -0.069228 -0.116810 -0.046422
Item2                 0.009542 -0.013865 -0.111545 -0.170010 -0.066139
Item3                -0.020962  0.000952 -0.176045 -0.249291 -0.147591
Item4                 0.008042  0.023763 -0.173905 -0.480655 -0.442505
Item5                -0.008550 -0.014829  0.137294  0.057896 -0.206302
Item6                 0.001311  0.017187 -0.060350  0.062041  0.759347
Item7                -0.005070  0.011386 -0.170669  0.804890 -0.377415
Item8                -0.018232 -0.065755  0.921694 -0.018911 -0.113107
```

```
                         PC16      PC17      PC18      PC19
Population            0.001210 -0.005661 -0.002356 -0.000322
Children              0.014490  0.020915 -0.000948 -0.021615
Age                  -0.009405  0.005784  0.013696  0.022421
Income               -0.002561  0.005301  0.013466 -0.000910
Outage_sec_perweek    0.013529  0.018262  0.013516  0.000361
Email                 0.006449 -0.017253  0.000961  0.000226
Contacts             -0.026498  0.020255 -0.000813 -0.000948
Yearly_equip_failure -0.001308  0.007488 -0.021448 -0.000145
Tenure               -0.007773 -0.004625  0.007519 -0.705251
MonthlyCharge        -0.000068  0.021494 -0.012007 -0.045778
Bandwidth_GB_Year    -0.006119 -0.002188  0.001815  0.706780
Item1                 0.025057 -0.240545  0.792965  0.002979
Item2                 0.073917 -0.590696 -0.573547 -0.001144
Item3                -0.395875  0.673812 -0.176863  0.000077
Item4                 0.431933  0.088483  0.018686  0.000105
Item5                 0.694089  0.264886 -0.041819 -0.000811
Item6                 0.400452  0.229253 -0.063809 -0.000593
Item7                 0.071323  0.066812 -0.041324  0.000486
Item8                -0.045658  0.045412 -0.043235 -0.001994
```

4.2 Kaiser Criterion

I will use the Kaiser rule to determine which principal components are most important. The Kaiser rule works by calculating an eigenvalue for each principal component. An eigenvalue of 1.0 indicates that a principal component is as releveant as an individual variable from the data set, so principal components that rise above the eigenvalue of 1.0 are considered better.

These eigenvalues can be visualized using a scree plot, making it easy to determine visually how many key principal components were discovered by PCA.

[15]:
```
cov_matrix = np.dot(df_prepared.T, df_prepared) / X.shape[0]
```

[16]:
```
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for␣
 ↪eigenvector in pca.components_]
eigenvalues
```

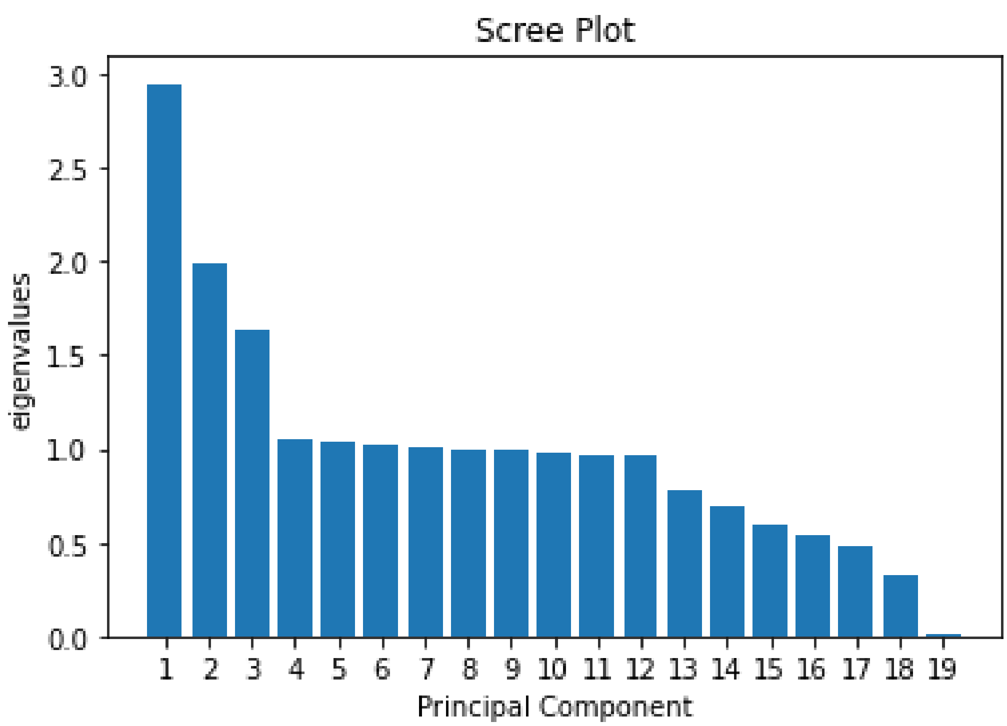
[16]:
```
[2.9485673172848346,
 1.996911994408803,
 1.6365536031172965,
 1.0569148738277545,
 1.029893736468177,
 1.0192367682908854,
 1.004692279595719,
 0.9991972786161447,
```

```
0.9915298714872935,
0.9806403932632919,
0.9644011257661943,
0.9608777750015292,
0.7786582754423637,
0.6901350220375765,
0.592279454452642,
0.5377461931170225,
0.4817273164763822,
0.3245773347065548,
0.005459386639422779]
```

[25]:

```
#The following code constructs the Scree plot
labels = [str(x) for x in range(1, len(eigenvalues)+1)]

plt.bar(x=range(1,len(eigenvalues)+1), height=eigenvalues, tick_label=labels)
plt.ylabel('eigenvalues')
plt.xlabel('Principal Component')
plt.title('Scree Plot')
plt.show()
```



4.3 Individual and Cumulative Variance

Based on the scree plot, the first three principal components are the most important. Using the code below I have printed the explained variance of each of these principal components, as well as their total explained variance. Explained variance is defined as a statistical measure of how much variation in a dataset can be attributed to each of the principal components generated by PCA (Kumar, 2022).

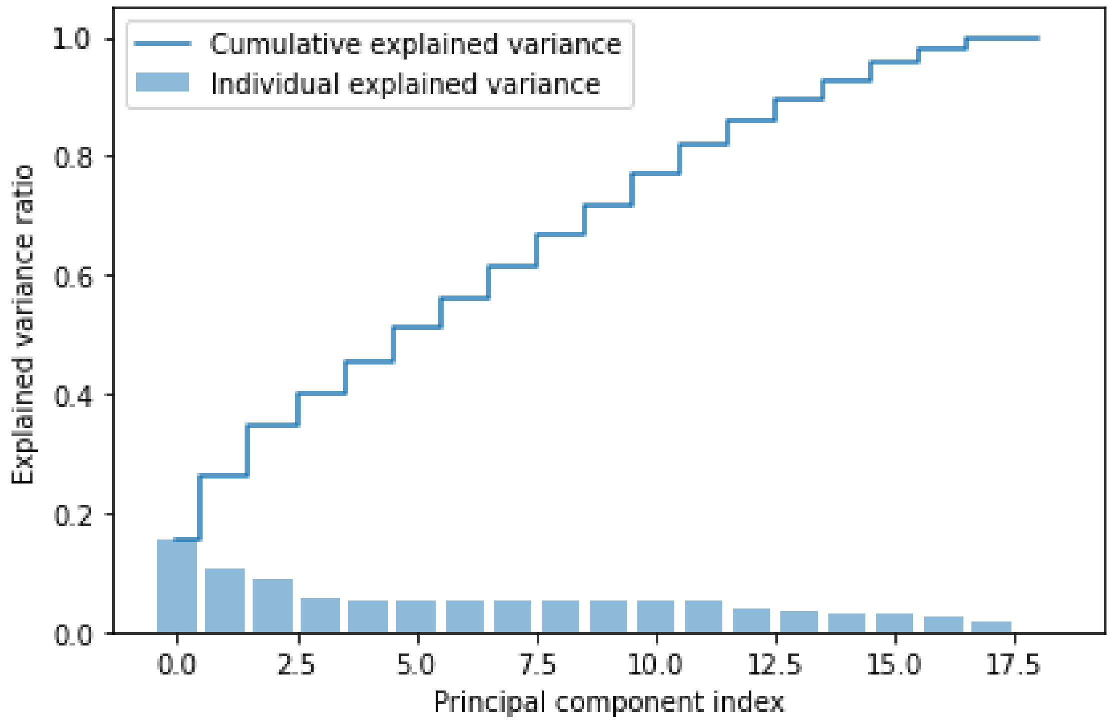
[18]:
```
total_eigenvalues = sum(eigenvalues)
var_exp = [(i/total_eigenvalues) for i in sorted(eigenvalues, reverse=True)]
print("Explained Variance for PC1: " + str(var_exp[0]))
print("Explained Variance for PC2: " + str(var_exp[1]))
print("Explained Variance for PC3: " + str(var_exp[2]))
```

```
Explained Variance for PC1: 0.15518775354130804
Explained Variance for PC2: 0.10510063128467449
Explained Variance for PC3: 0.08613440016406877
```

[19]:
```
cum_sum_exp = np.cumsum(var_exp)
print("Total Explained Variance for PC1, PC2 and PC3: " + str(cum_sum_exp[2]))
```

```
Total Explained Variance for PC1, PC2 and PC3: 0.3464227849900513
```

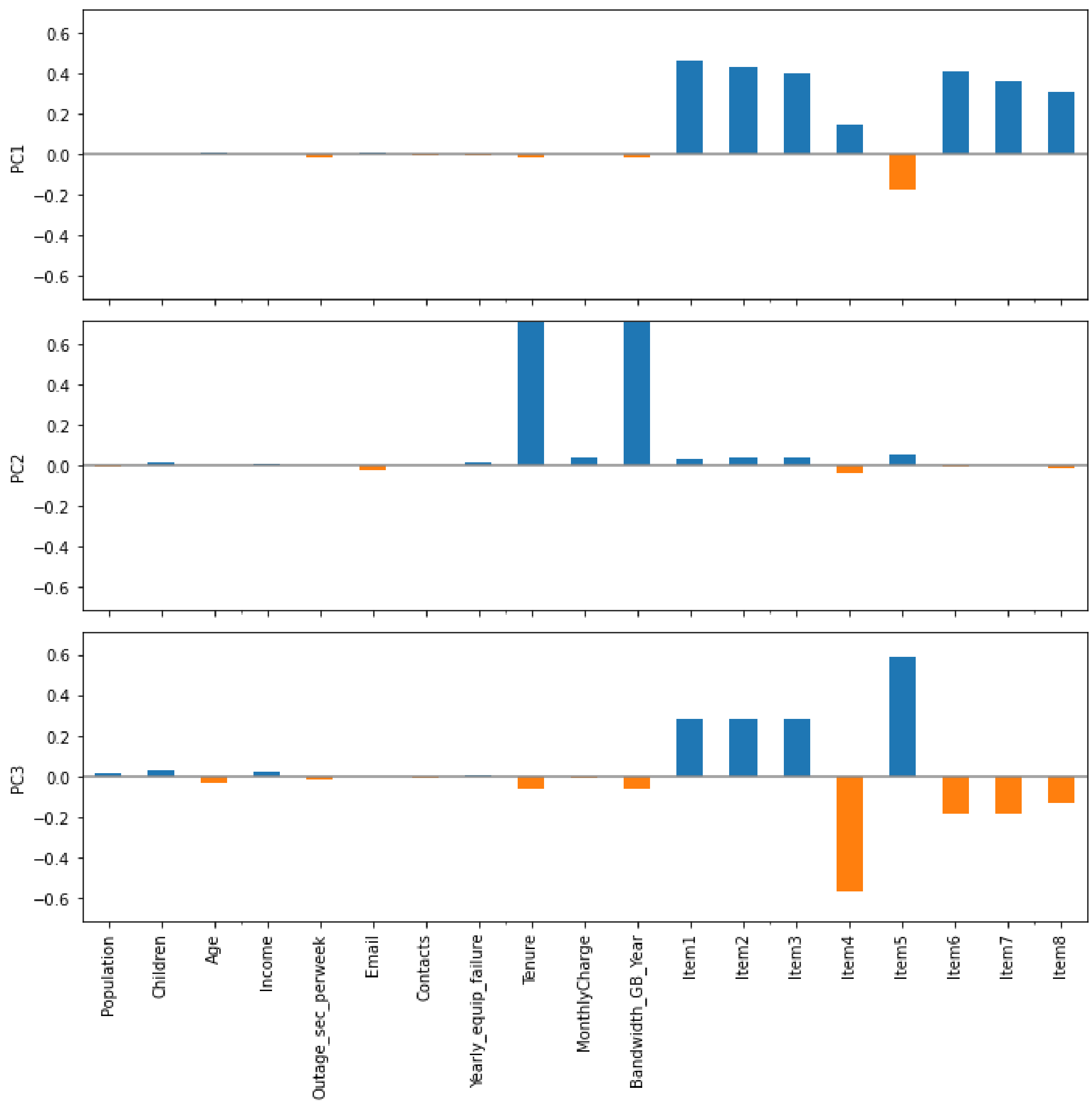
[20]:
```
# Plot the explained variance against cumulative explained variance
#
cum_sum_exp = np.cumsum(var_exp)
plt.bar(range(0,len(var_exp)), var_exp, alpha=0.5, align='center',␣
 ↪label='Individual explained variance')
plt.step(range(0,len(cum_sum_exp)), cum_sum_exp, where='mid',label='Cumulative␣
 ↪explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal component index')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



4.4 Results of Data Analysis

PCA determined there are 3 principal components when the 19 continuous variables of the data set were evaluated, answering the original research question "how many principal components does the data set contain when using the continuous numerical data in the data set as input?". A graphical representation of the variables influencing each of these principal components is included below.

[21]:
```
loadings_df = pd.DataFrame(pca.components_[0:3, :],
                        columns=df_data.columns)
maxPC = 1.01 * np.max(np.max(np.abs(loadings_df.loc[0:3, :])))
f, axes = plt.subplots(3, 1, figsize=(10, 10), sharex=True)
for i, ax in enumerate(axes):
    pc_loadings = loadings_df.loc[i, :]
    colors = ['C0' if l > 0 else 'C1' for l in pc_loadings]
    ax.axhline(color='#888888')
    pc_loadings.plot.bar(ax=ax, color=colors)
    ax.set_ylabel(f'PC{i+1}')
    ax.set_ylim(-maxPC, maxPC)
plt.tight_layout()
plt.show()
```



---

5 Web Sources

https://github.com/StatQuest/pca_demo/blob/master/pca_demo.py

https://vitalflux.com/pca-explained-variance-concept-python-example/

https://medium.com/analytics-vidhya/pca-and-how-to-interpret-it-with-python-8aa664f7a69a

---

6 References

Keboola. (2022, April 2). *A Guide to Principal Component Analysis (PCA) for Machine Learning.* https://www.keboola.com/blog/pca-machine-learning

Pramoditha, R. (2020, August 3). *Principal Component Analysis (PCA) with Scikit-learn.* Towards Data Science. https://towardsdatascience.com/principal-component-analysis-pca-with-scikit-learn-1e84a0c731b0

Kumar, A. (2022, August 11). *PCA Explained Variance Concepts with Python Example.* Data Analytics. https://vitalflux.com/pca-explained-variance-concept-python-example/