D212 Task 2 Dimensionality Reduction Methods

Western Governs University

Table of Contents

Part I: Research Question

A1. Proposal Question

Can we identify the variables that account for the most variance of the selected dataset so that the dimensionality of the dataset can be reduced?

A2. Defined Goal

By applying PCA to telecommunication data you can identify the principal components that capture the most important patterns of variation in the data and reduce dimensionally of the dataset. PCA works by identifying the principal components of the dataset, which are the directions in which the data varies the most. The principal components represent the linear combinations of the original variables that explain the maximum amount of variance in the data.

Part II: Method Justification

B1. Explanation of PCA

Principal component analysis (PCA) is a statistical technique used to analyze the structure of a dataset by identifying patterns in the data (Mirshra et al. 2016). The goal of PCA is to identify a new set of uncorrelated variables that can explain variability in the data. PCA can only be completed with quantitative variables after missing data has been treated. Expected outcomes are reduced dimensionality, identification of key factors, improved accuracy, ad visualization of relationships between variables.

B2. PCA Assumption

PCA assumes a linear relationship between features (*A Guide to Principal Component Analysis (PCA) for Machine Learning*, n.d.). PCA captures the linear relationships and identifies the direction of maximum variance in the data.

Part III. Data Preparation

C1. Continuous Dataset Variables

The following variables were used to preform k-means clustering:

Variable	**Data Type**
Children	Continuous
Age	Continuous
Income	Continuous
Outage_sec_perweek	Continuous

Email	Continuous
Contacts	Continuous
Yearly_equip_failure	Continuous
Tenure	Continuous
MonthlyCharge	Continuous
Bandwidth_GB_Year	Continuous

C2. Standardization of dataset variables

The following method was used to standardize the data:

1. Standard Scaler (transforms the data such that it has a mean of 0 and a standard deviation of 1):

```
sc = StandardScaler()
sc.fit(df_numeric)
scaled_data_array = sc.transform(df_numeric)
scaled_data = pd.DataFrame(scaled_data_array, columns = df_numeric.columns)
scaled_data.head()
```

	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	Bandwidth_GB_Year
0	-0.497975	-1.267929	-0.661753	0.577519	-0.007198	-1.045438	1.072633	-1.258019	1.624861	-1.180036
1	1.088855	-0.153217	-1.143209	0.254153	-1.003385	-1.045438	1.072633	-0.706000	-0.298598	-0.606271
2	-0.497975	-0.250148	-0.772394	1.675978	0.988989	1.174939	-0.642890	-0.655588	-1.228882	-0.555988
3	-1.026918	1.446153	0.069452	-0.636170	1.321052	1.174939	1.072633	-1.238570	-0.531205	-1.422357
4	0.559911	1.446153	-0.623721	-0.542683	0.988989	2.285128	1.072633	-1.037010	0.284363	-1.070944

2. Robust Scaler (designed to handle outliers):

```
robust_scaler = RobustScaler()
scaled_data_robust = robust_scaler.fit_transform(scaled_data)
scaled_data_robust_df = pd.DataFrame(scaled_data_robust, columns=scaled_data.columns)
scaled_data_robust_df.head()
```

	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	Bandwidth_GB_Year
0	0.000000	-0.722222	-0.333151	0.428237	0.00	-0.5	1.0	-0.534913	1.203057	-0.533094
1	1.000000	-0.083333	-0.697033	0.187467	-0.75	-0.5	1.0	-0.262048	-0.120663	-0.244968
2	0.000000	-0.138889	-0.416773	1.246120	0.75	0.5	0.0	-0.237130	-0.760883	-0.219717
3	-0.333333	0.833333	0.219491	-0.475444	1.00	0.5	1.0	-0.525299	-0.280744	-0.654780
4	0.666667	0.833333	-0.304407	-0.405835	0.75	1.0	1.0	-0.425667	0.280529	-0.478312

Cleaned data file is attached.

df_numeric.to_csv('D212_prepared_task2.csv')

Part IV. Analysis

D1. Principial Components

The 10 principal components are represented as a 10 x10 matrix where each row represents a principal component and each column represents a variable in the original dataset.

Matrix of all the principle components (Brownlee, 2019):

```
[[ 1.14989059e-02 -3.27899047e-03  9.08263711e-01 -2.25150966e-01
  -3.44908254e-01 -6.42299839e-04  1.15247937e-02  2.12211556e-02
  -6.54384995e-02  1.98399819e-02]
 [-6.49819490e-03  2.19267837e-03  3.04771637e-01 -1.96955442e-01
   9.23098291e-01  3.00506767e-04 -2.05889445e-02 -8.68474268e-02
  -1.75908316e-02 -8.88223252e-02]
 [ 5.36567775e-02 -2.62352796e-02  2.83421309e-01  9.41186931e-01
   1.17669304e-01  1.55247560e-02  1.56759371e-02  3.52544045e-02
   1.13217024e-01  4.33404770e-02]
 [ 2.17703474e-02  1.10905972e-02  1.07070629e-02 -1.22933509e-01
   9.73179412e-02 -5.40706467e-04  1.33136828e-02  6.04948940e-01
   4.52999067e-01  6.35103187e-01]
 [-1.26612140e-02  4.73297807e-02  3.19514101e-02 -7.72172657e-02
  -7.24467690e-02  6.16312197e-04 -1.50593118e-02 -3.42120413e-01
   8.79954048e-01 -3.06227017e-01]
 [ 9.87360135e-01 -9.68143101e-02 -2.47673127e-02 -5.09254412e-02
   2.31991840e-03 -4.38527741e-02  9.63298738e-02 -3.47599173e-02
   3.26968138e-04 -1.11302075e-02]
 [-9.65740953e-02  1.58190663e-02 -5.92188815e-03 -1.07959274e-02
   1.86311100e-02  6.27006577e-03  9.94726091e-01 -1.24154177e-02
   5.59495405e-03 -1.48224738e-02]
 [ 1.00704135e-01  9.93262591e-01  5.76855432e-03  2.44789604e-02
   2.18113188e-03  1.52191660e-02 -5.42727502e-03  2.13707522e-02
  -4.36063968e-02 -5.61649361e-03]
 [ 4.16103773e-02 -1.90936391e-02 -5.06552260e-03 -1.72751297e-02
  -2.27946531e-03  9.98781034e-01 -2.14599157e-03 -2.09919769e-03
  -1.46745184e-03 -6.05491148e-05]
 [-1.52612589e-02  1.93557861e-02 -8.26719919e-04  2.92413463e-05
   8.10450762e-05 -4.99496172e-04 -2.57702819e-05 -7.11288413e-01
  -3.35843375e-02  7.01664016e-01]]
```

```
           PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    -1.636362  0.945825  1.279319 -0.548498  0.474996 -0.499239  0.168230
1    -0.891948  1.606271  0.114526 -0.116317  1.204080 -1.325216  0.324575
2    -0.928973 -0.654153  1.009774  0.378834 -1.920302 -0.808427  0.606658
3    -1.939959 -1.719185 -0.133993  0.627797  0.341616  0.806677  1.913070
4    -1.490911 -1.274104  1.036840  0.669387  0.646400  0.725839  1.756721
...        ...       ...       ...       ...       ...       ...       ...
8945  0.853645 -0.578123 -1.594023 -0.268192 -0.863053  0.642198  0.188212
8946  1.906935  0.429743 -0.238099  0.665133 -1.613112  0.330036 -0.437068
8947  0.590843 -0.073424 -1.470899 -0.553060  0.283267 -0.183729 -0.813355
8948  2.034713 -0.473441  1.691762 -1.155315 -0.549907 -0.168077 -0.371037
8949  1.572977 -0.310424  1.368555 -1.504578 -1.498001 -0.288392  0.527828

           PC8       PC9      PC10
0    -2.002846 -0.677345 -0.039746
1     0.248946  0.077112  0.060546
2     1.148405 -0.486594  0.131189
3     0.314777 -0.095498 -0.055620
4     0.768883  1.658037 -0.017636
...        ...       ...       ...
8945 -0.315314  0.264517  0.127097
8946 -0.225180  1.121735  0.081403
8947 -0.681747 -0.159596 -0.087195
8948 -1.069592 -0.019026 -0.069603
8949 -0.920415 -0.062012 -0.034041
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Children	0.009316	0.648334	0.115351	-0.150357	0.111512	0.189706	-0.036604	0.458496	0.532266	-0.019331
Age	-0.004663	-0.478585	0.023297	-0.052707	0.605775	0.208335	0.042425	0.580435	-0.134336	0.022284
Income	0.001571	0.131226	-0.261266	0.395283	-0.098951	0.776154	-0.304766	-0.050473	-0.225013	-0.001258
Outage_sec_perweek	0.007887	0.195154	0.678742	0.204553	-0.219139	-0.127292	-0.154145	0.317601	-0.520062	0.000047
Email	-0.024338	-0.115414	0.145073	-0.464208	-0.419422	0.456016	0.591527	0.062222	-0.109137	0.000125
Contacts	0.001430	-0.421418	0.295088	0.548945	-0.289100	0.069631	0.113133	0.090862	0.569653	-0.000452
Yearly_equip_failure	0.012286	0.307542	0.111529	0.428672	0.438422	0.032569	0.660990	-0.256120	-0.115280	-0.000030
Tenure	0.705357	-0.021716	-0.040365	0.004880	-0.020868	-0.011823	0.025125	0.035981	-0.022062	-0.705236
MonthlyCharge	0.042143	-0.096557	0.578800	-0.275137	0.332284	0.296532	-0.282775	-0.520022	0.165229	-0.046486
Bandwidth_GB_Year	0.706949	0.004787	-0.000129	-0.014536	-0.015270	0.007681	0.003597	-0.004135	0.007659	0.706830

D2. Identification of Total Number of Components

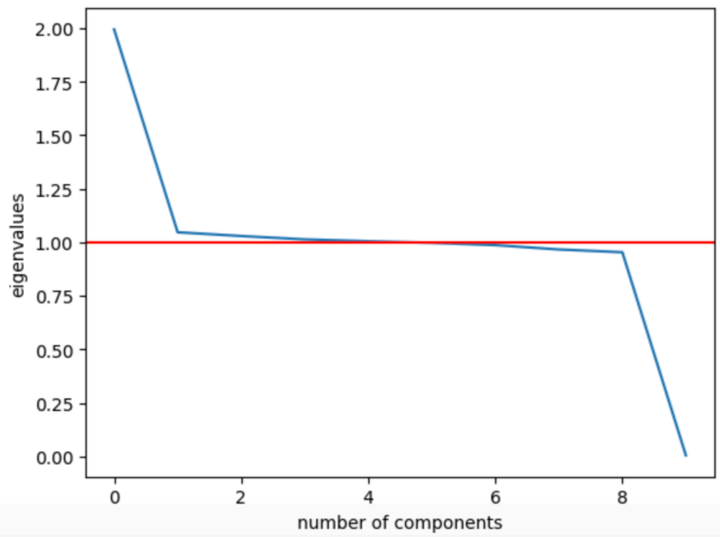
The kaiser rule was used to identify the total number of principal components by following these steps:

- Perform PCA on your dataset and obtain the eigenvalues of the covariance matrix.

```
]: cov_matrix = np.dot(test_pca_normalized.T, test_pca_normalized) / test_pca.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector in pca.components_]

# Display eigenvalues
plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalues')
plt.axhline(y=1, color="red")
plt.show()

# Display eigenvectors
for i, eigenvector in enumerate(pca.components_):
    print(f"Eigenvector {i+1}: {eigenvector}")
```




```
Eigenvector 1: [ 0.00931598 -0.0046634   0.00157121  0.00788728 -0.02433839  0.0014298
  0.01228583  0.70535729  0.04214329  0.70694871]
Eigenvector 2: [ 0.64833449 -0.4785846   0.13122628  0.19515395 -0.11541433 -0.42141835
  0.30754205 -0.0217156  -0.09655709  0.00478748]
Eigenvector 3: [ 1.15350759e-01  2.32971286e-02 -2.61265823e-01  6.78742187e-01
  1.45073226e-01  2.95087743e-01  1.11528669e-01 -4.03645344e-02
  5.78800212e-01 -1.29154945e-04]
Eigenvector 4: [-0.1503574  -0.05270686  0.39528306  0.20455266 -0.46420787  0.54894455
  0.42867169  0.00487989 -0.27513737 -0.01453559]
Eigenvector 5: [ 0.11151206  0.60577503 -0.09895086 -0.21913869 -0.41942153 -0.2891002
  0.43842194 -0.02086817  0.33228428 -0.01526998]
Eigenvector 6: [ 0.18970555  0.20833491  0.77615389 -0.12729183  0.45601631  0.06963065
  0.03256854 -0.01182279  0.29653249  0.00768117]
Eigenvector 7: [-0.03660351  0.04242548 -0.30476646 -0.15414451  0.59152712  0.1131329
  0.66098992  0.02512519 -0.2827753   0.00359654]
Eigenvector 8: [ 0.45849594  0.58043489 -0.05047326  0.31760068  0.0622216   0.09086206
 -0.25611969  0.03598119 -0.52002172 -0.0041351 ]
Eigenvector 9: [ 0.53226617 -0.13433637 -0.22501291 -0.52006181 -0.10913675  0.5696535
 -0.11527971 -0.0220624   0.16522895  0.00765905]
Eigenvector 10: [-1.93312361e-02  2.22843747e-02 -1.25779407e-03  4.68856751e-05
  1.24739519e-04 -4.52001698e-04 -3.01321144e-05 -7.05236477e-01
 -4.64863999e-02  7.06829846e-01]
```

```
eigenvalues
```

```
[1.9939555160702047,
 1.0468251058908893,
 1.0293024460169544,
 1.0136490886231624,
 1.004916823168951,
 0.9975841763327815,
 0.9869594518662925,
 0.9665862155796137,
 0.9536203473940985,
 0.0054835106212907]
```

- Sort the eigenvalues in descending order.

- Examine the eigenvalues and count the number of eigenvalues that are greater than 1.

- The total number of principal components to retain is equal to the number of eigenvalues greater than 1.

```
# Sort eigenvalues in descending order
eigenvalues_sorted = np.sort(eigenvalues)[::-1]

# Calculate the number of eigenvalues greater than 1
num_components = np.sum(eigenvalues_sorted > 1)

print(f"Number of Principal Components to Retain: {num_components}")
```

```
Number of Principal Components to Retain: 5
```

D3. Total Variance of Components

The kaiser criterion revealed PC1 through PC5 was most significant. Below is the total variance of each principal component.

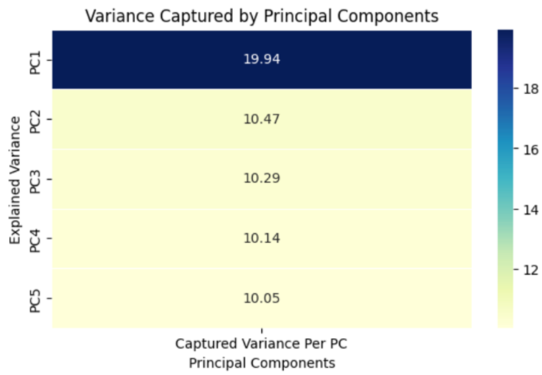
```
pc5 = PCA(n_components=5, random_state=2020)
pc5.fit(scaled_data)
var_pca = pc5.transform(scaled_data)

pca_5 = pc5.explained_variance_ratio_ * 100
var_df1 = pd.DataFrame(pca_5, columns=['Captured Variance Per PC'],
                       index=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
var_df1 = var_df1.round(2)

var_df1
```

	Captured Variance Per PC
PC1	19.94
PC2	10.47
PC3	10.29
PC4	10.14
PC5	10.05

```
plt.figure(figsize=(7, 4))
sns.heatmap(var_df1, annot=True, fmt='.2f', linewidths=0.5, cmap='YlGnBu')
plt.title('Variance Captured by Principal Components')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance')
plt.show()
```



D4. Total Variance Captured by Components

To identify the total variance captured by the principal components, the sum of the explained variance ratios of all the selected principal components was computed.

```
total_variance = np.sum(pc5.explained_variance_ratio_)
print("Total Variance Captured by Principal Components: {:.2f}%".format(total_variance * 100))
```

```
Total Variance Captured by Principal Components: 60.89%
```

D5. Summary of Data Analysis

The results of the PCA analysis provided information about the relationships between the variables in the churn dataset how they contribute to the variation in the data set. The PCA produced 5 principal components from the churn data set. PC1 captured 19.94% of the explained variance ratio. PC1 had the highest variance indicating it explains the most significant portion of the variability in the data (*A Guide to Principal Component Analysis (PCA) for Machine Learning*, n.d.-b). The other PCs also contribute to the overall variance. The cumulative total variance captured by the 5 components was 60.89%. PC1-PC5 explained 60.89% variance in the data. Thus, either of these components can be used to pass through machine learning algorithms to find patterns in customer data.

Part V. Attachments

E. Sourced for Third-Party Code

Brownlee, J. (2019). How to Calculate Principal Component Analysis (PCA) from Scratch in Python. *MachineLearningMastery.com*. https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/

F. Sources

A Guide to Principal Component Analysis (PCA) for Machine Learning. (n.d.). https://www.keboola.com/blog/pca-machine-learning

Cheplyaka, R. (2017). Explained variance in PCA. *ro-che.info*. https://ro-che.info/articles/2017-12-11-pca-explained-variance#:~:text=The%20total%20variance%20is%20the,divide%20by%20the%20total%20variance.

Mishra, S. P., Sarkar, U. K., Taraphder, S., Datta, S. K., Swain, D. P., Saikhom, R., Panda, S., & Laishram, M. (2016). Multivariate Statistical Data Analysis- Principal Component Analysis (PCA) -. *International Journal of Livestock Research*, *7*(5), 60–78. https://www.bibliomed.org/?mno=261590