**D212 Performance Assessment Task 2**

**DIMENSIONAL REDUCTIONALITY FOR MEDICAL DATA**

Fahim A. Akbar Student ID 001434895 Masters Data Analytics (January 1, 2021) Program Mentor: Mandy Rasmuson (385) 428-5957x5957 [fakbar3@wgu.edu](mailto:fakbar3@wgu.edu)

**Part I: Research Question**

**A. Data Mining Report Justification**

**A1. Research Question:** Which variables can tell us the most about our patient population in the medical records database?

**A2. Research Goal:** One goal of this data analysis is to find the principal variables that can help us better understand the patient data. Once that is done, the data dimensions will be reduced to those main principal component variables for further analysis.

**Part II: Method Justification**

**B. Principal Component Analysis Justification**

**B1. Justification of using PCA and what the expected outcomes are**

Datasets with high dimensionality can often cause issues whenever it comes to performing an analysis. The high dimensionality can lead to overfitting any models created from the dataset. This in turn makes it harder to fit any other data except for the training set on the model. Furthermore, this also requires machines that have a lot of memory and processing power. One great solution to dealing with such datasets is principal component analysis, or PCA. This is a dimensionality reduction technique used to help reduce high dimensional data with multiple features to fewer variables. These fewer variables are called principal components, and they essentially combine the most information about the data. PCA analyzes the selected data set by transforming the large set of data values into much smaller ones that still have most of the relevant information in the large set, thus reducing the amount of processing power needed for the final model. (Bigabid, 2023). The expected outcomes for the PCA include reducing the number of variables currently present, reducing the dimensionality of the data, and decrease the accuracy but retain most of the information. (Lee, 2022).

**B2.  One assumption of the clustering technique.**

One assumption of PCA is that linearity exists in the data set and the variables in the dataset are correlated to each other. This is because the variables themselves need to form a linear relationship in order for the PCA to work. In this case, demographic data helps the hospital gauge what kind of patients they’re dealing with and how they can prepare in advance. (Ye, 2020).

**Part III: Data Preparation**

**C. Data Preparation of the dataset**

**C1. Variables used for the analysis**

The following continuous dataset variables will be used in order to answer the PCA question proposed in part A1.

* Age: Age of patient as reported
* Income: Annual income of the patient
* VitD\_levels: The patient’s vitamin D levels
* Initial\_days: Number of days a patient stayed in the hospital during the initial visit.
* TotalCharge: The total amount that the patient was charged
* Additional\_charges: The average amount charged to the patient on top of their expense

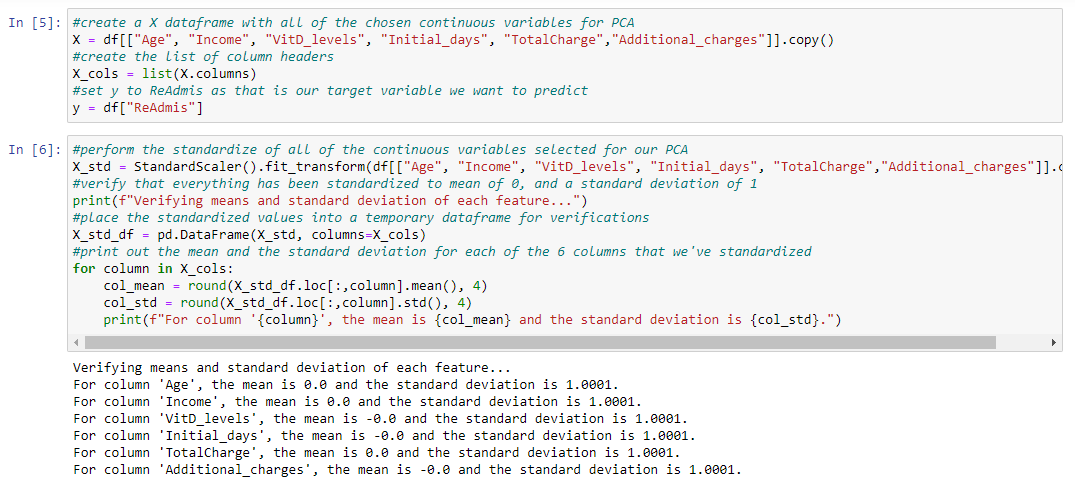
These variables were selected to answer our proposed research question as they pertain to individual circumstances and conditions of each patient that could in turn be related to their likelihood of readmission. Our goal with the PCA is to see the linear relationship between these chosen variables and our target, readmissions. It’s possible that some of these variables may not have a significant impact at all, and we’ll use the process of PCA to omit features accordingly.

**C2.  Standardization of the continuous dataset variables**

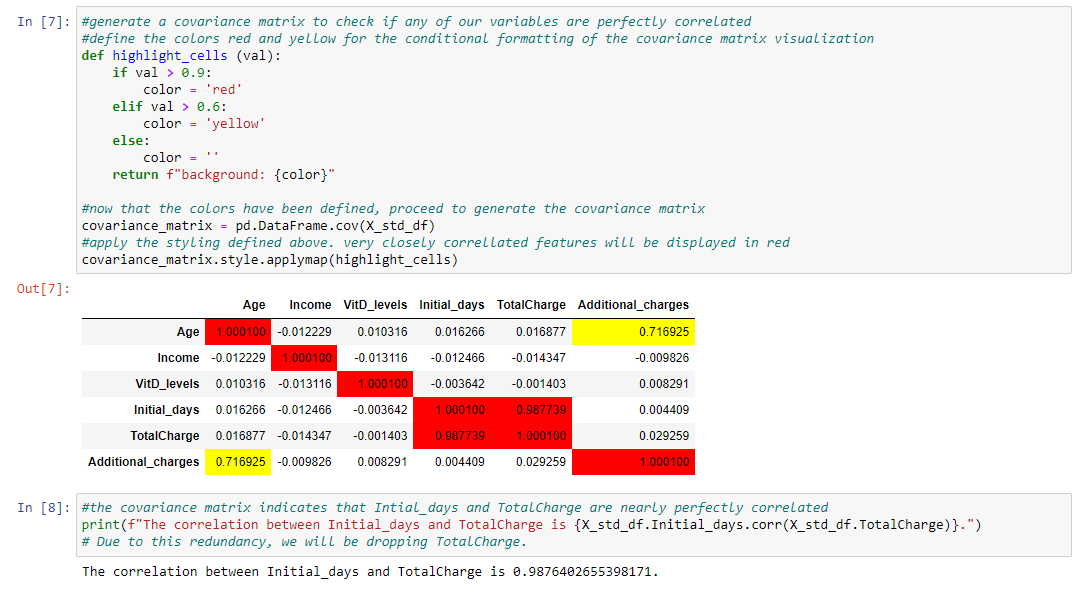
The variables chosen were standardized to a mean of 0 and a standard deviation of 0. A covariance matrix was also generated, showing that two features, Intial\_days and TotalCharge, were nearly perfectly correlated. In response, one of the features, TotalCharge, was removed.

The standardization of the continuous dataset variables was performed in a Python using a Jupyter notebook environment. The Jupyter notebook file is attached to the task submission. A pdf copy of the notebook and a txt. file of code used is provided with the task submission as well. A copy of the cleaned dataset is also provided with the task submission. Lastly, the entire code used is also provided at the end of the document.

The code used for standardizing the variables is showcased below:



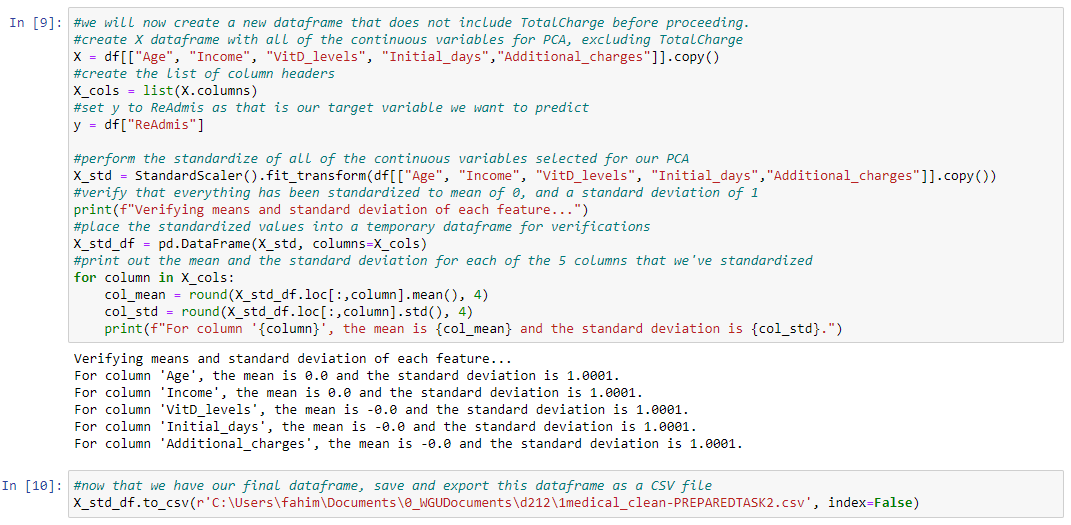
After the variables were standardized, the following covariance matrix was generated to check if any of the variables were perfectly correlated.



As shown above, the correlation between Initial\_days and TotalCharge is 0.9876402655398171.

This shows us that these two features are perfectly correlated and give us almost the same information. Keeping both variables would therefore be redundant, so one of the variables will be dropped. For this PCA we chose to drop TotalCharge.

Once TotalCharge was dropped, the standardization process was repeated and a new data frame for our PCA was created. The data frame was exported as a csv file named “1medical\_clean-PREPAREDTASK2.csv” and is available as an attachment in the task 2 submission.

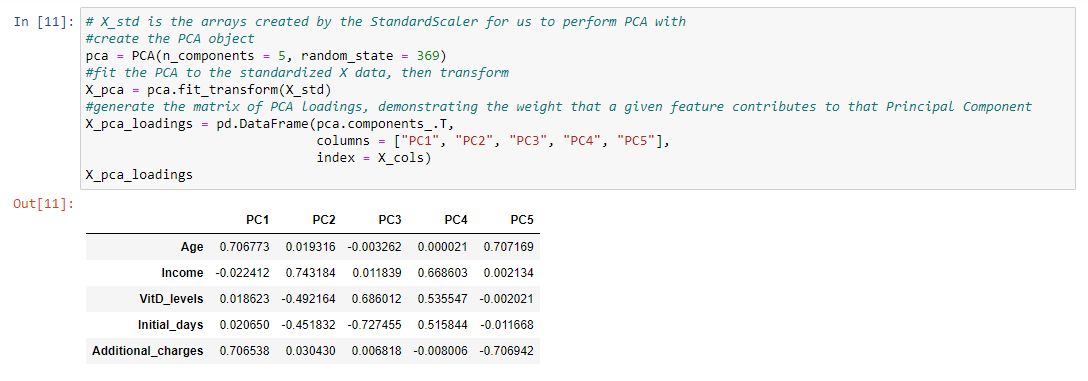


**Part IV: Analysis**

**D. Principal Component Analysis Performance**

**D1. Matrix of all the principal components**

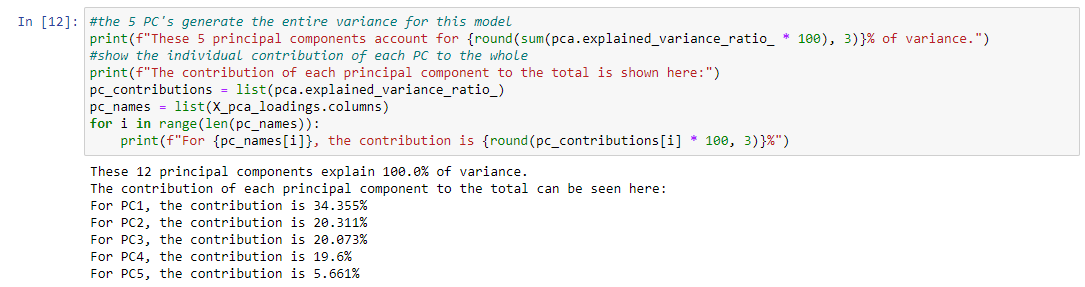
Now that our data was standardized, the PCA can be performed to reduce the dimensionality of our dataset.



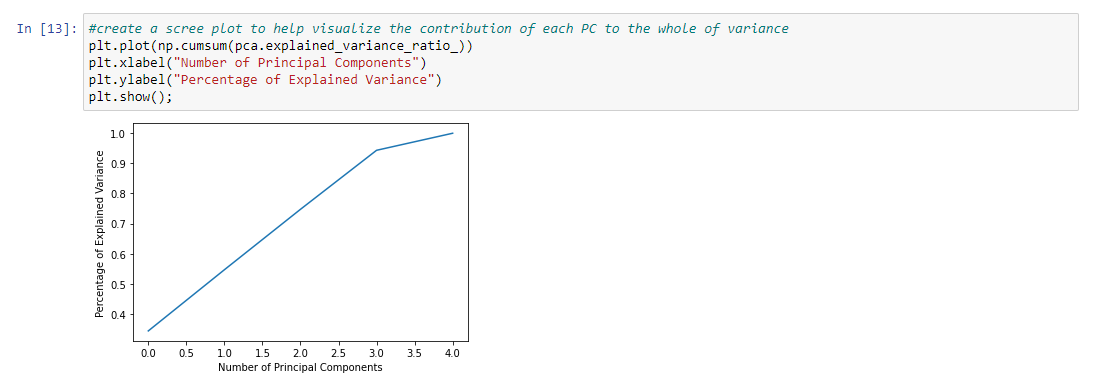
The PCA loadings generated can show us the weight that each variable contributes to each of the 5 principal components. For instance, the PCA loadings show us that PC1 is most influenced by the patient's age and additional charges, while the other variables have a much smaller weight. PC2 is mainly influenced by Income. PC3 is influenced most by VitD\_Levels. PC4 is influenced most by Income, VitD\_levels, and Initial\_days. PC5 is influenced most by Age.

**D2. Total number of principal components**

For this analysis, the total number of principal components turned out to be five. We can now determine how much variance in the data each PC is responsible for. The PCs with the least amount of influence will be removed from our final model.



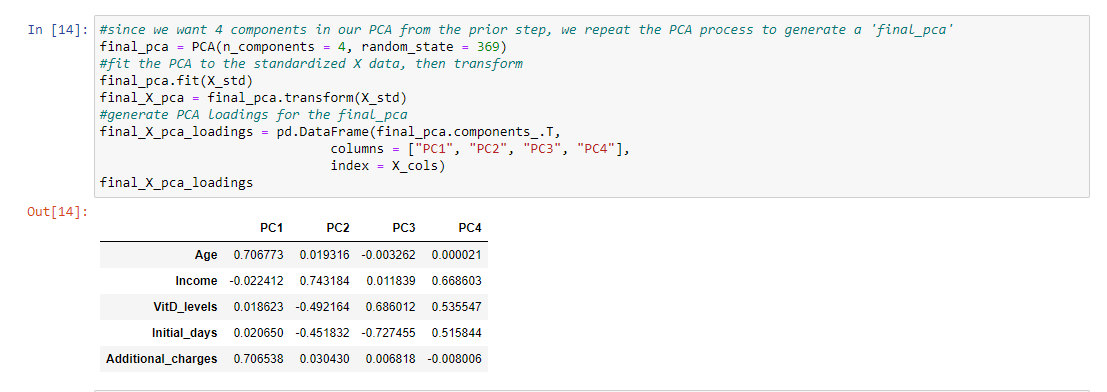
The following scree plot was then generated



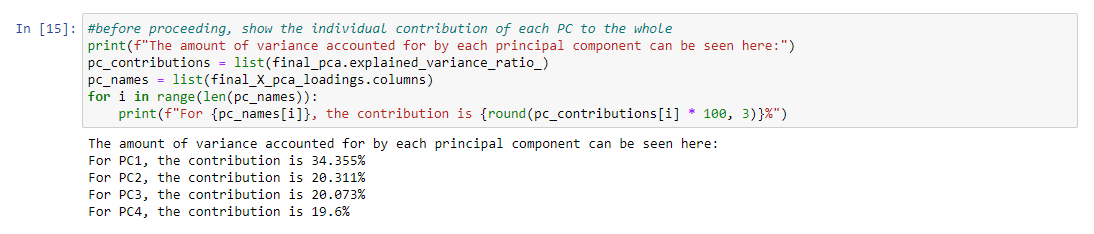
For the model to be successful, we would want to see the first few components account for the majority of variance. Our goal is to have as few features as possible in order to generate a more accurate model for PCA (CITE). Based on our results, the first principal component provided the most amount of variance at over 34%. Components 2 and 3 provided over 20% variance and component 4 provided over 19% variance. Component 5 provided the least amount of variance at just under 6%. For further model refinement, PC5 will be removed due to the low contribution towards variance.

**D3. Variance of Principal Components**

Now that we decided to omit PC5 and reduce the amount of components to 4, the PCA process is repeated once more to account for this change. This is demonstrated in the code showcased below.

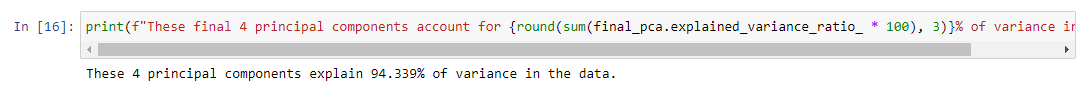


The variance of each principal component identified goes as follows:



These values are consistent with what we found in section D2, so we can proceed with the model.

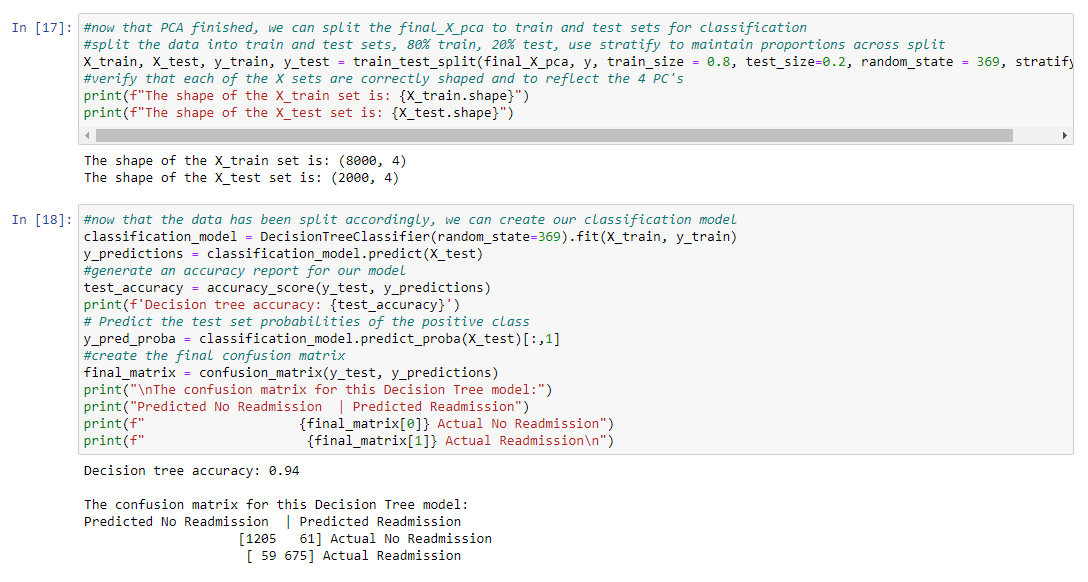
**D4. Total variance captured by the principal components**



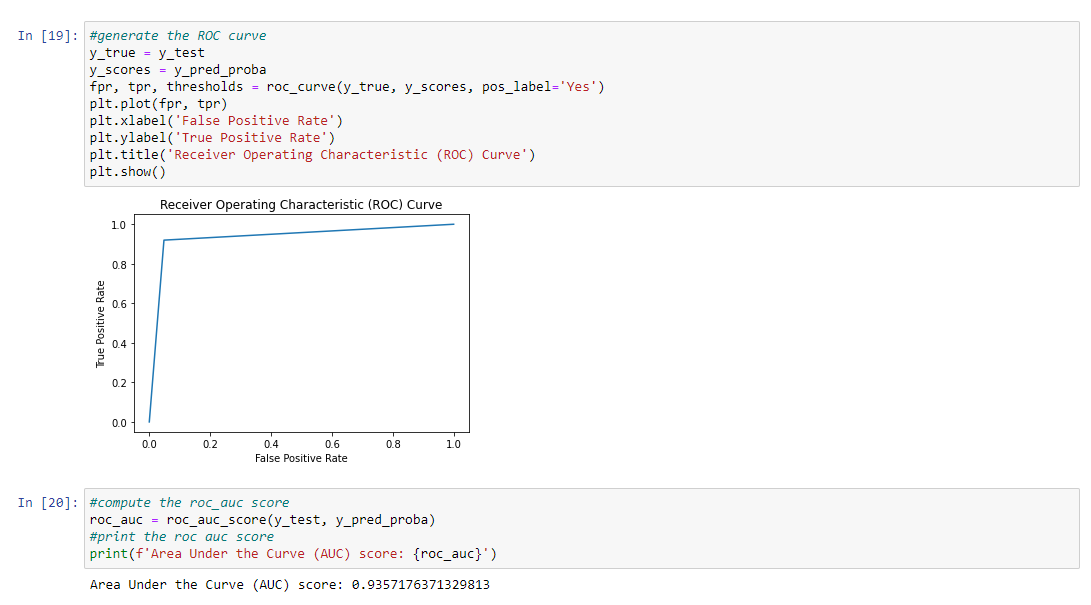
These final 4 principal components account for 94.339% of variance in the data. This is expected as principal component 5 was removed and it initially accounted for almost 6% of variance.

**D5. Data Analysis Results**

The goal of this PCA was to reduce the dimensionality of the dataset and help predict the chance of a patient’s readmission. Since this is a classification problem, PCA was deployed and was successful in reducing the dimensionality of the dataset to 4 components. To predict the likelihood of patient readmission, a decision tree is then employed to classify patients using these remaining four principal components. This was done by splitting the final PCA data. The DecisionTreeClassifier function in python was used to fit the training data to make predictions for the test data. This is demonstrated in the code showcased below.



The classification model based on these 4 principal components did an excellent job predicting the likelihood of patient readmission, with an accuracy score of 94%. Out of 2000 patients in the test set, our model could predict readmission and no readmissions for 1880 patients (1205 + 675) correctly, and only got 120 patients (59 + 61) incorrect. The success of the model was also demonstrated by creating an ROC curve for visualization as well as calculating an AUC score. This is showcased below:



The model had an AUC score of 0.9357176371329813, or 94%. This high score indicates that the model is very strong at predicting correct outcomes of a patient’s chance of readmission. Even though the final model was reduced to four variables, it appears that these are strong predictors for a patient’s chance of readmission in the future. Age is expected to impact a patient’s likelihood of readmission, as older people tend to need more medical attention and repeat visits. The initial days spent in a hospital as well as the Vitamin D levels are interesting as it's not as apparent why these could impact the chance of readmission. The accuracy score of 94% is exceptional, but the model can always be made even more precise. Furthermore, additional variables can be tested and accounted for, such as the number of children a patient has. Parenthood does add additional stress to an individual’s life and could lead to slower recovery which in turn could lead to readmission. We can also expand this investigation by focusing on the kind of treatment patients received instead. From there we could use PCA to see which treatments may have actually reduced or increased the likelihood of readmission.

**Part V: Attachments**

**E. Web sources used for third-party code:**

<https://www.kaggle.com/code/sid321axn/principal-component-analysis-pca/notebook>

<https://www.kaggle.com/code/akbarhuseynov23/roc-and-auc>

<https://www.kaggle.com/code/lilyelizabethjohn/standardization-using-standardscaler>

<https://towardsdatascience.com/visualising-the-classification-power-of-data-54f5273f640>

<https://towardsdatascience.com/a-quick-and-easy-guide-to-conditional-formatting-in-pandas-8783035071ee>

<https://datatofish.com/covariance-matrix-python/>

<https://datascienceplus.com/principal-component-analysis-pca-with-python/>

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

<https://builtin.com/machine-learning/pca-in-python>

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

**F. Additional Sources**

Bigabid. (2023, February 8). *What is Principal Component Analysis (PCA) & How to use it?.* Retrieved June 6, 2023, from <https://www.bigabid.com/what-is-pca-and-how-can-i-use-it/#:~:text=PCA%20can%20help%20us%20improve,uncorrelated%20features%20of%20the%20data>.

Ye, Andre (2020, August 3). *Beyond Ordinary PCA: Nonlinear Principal Component Analysis.* Medium. Retrieved June 6, 2023, from <https://medium.com/dataseries/beyond-ordinary-pca-nonlinear-principal-component-analysis-54a93915a702>

Lee, W. (2022, January 30). *Using Principal Component Analysis (PCA) for Machine Learning.* Towards Data Science. Retrieved June 6, 2023, from <https://towardsdatascience.com/using-principal-component-analysis-pca-for-machine-learning-b6e803f5bf1e>

**Full code used for the project:**

**DATA CLEANING AND PREPARATION CODE**

#import packages and clean data before running the principal component analysis

import numpy as np

import pandas as pd

from sklearn import linear\_model

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

pd.set\_option('display.max\_columns', None)

import pylab

from pylab import rcParams

import statsmodels.api as sm

import statistics

from scipy import stats

import sklearn

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.metrics import classification\_report

from scipy.stats import chisquare

from scipy.stats import chi2\_contingency

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

from sklearn.metrics import accuracy\_score

df = pd.read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d208\1medical\_clean.csv')

df.head()

df.info()

#check if there are any missing data entries - if there are none then the output should be False

df.isna().any()

#identify the continuous variables

df.dtypes

#identify the continuous variables

cont = df.select\_dtypes("number")

cont.head()

**CODE USED FOR CREATING THE FIRST DATAFRAME**

#create a X dataframe with all of the chosen continuous variables for PCA

X = df[["Age", "Income", "VitD\_levels", "Initial\_days", "TotalCharge","Additional\_charges"]].copy()

#create the list of column headers

X\_cols = list(X.columns)

#set y to ReAdmis as that is our target variable we want to predict

y = df["ReAdmis"]

**CODE USED FOR STANDARDIZATION OF THE VARIABLES FROM OUR FIRST DATAFRAME**

#perform the standardize of all of the continuous variables selected for our PCA

X\_std = StandardScaler().fit\_transform(df[["Age", "Income", "VitD\_levels", "Initial\_days", "TotalCharge","Additional\_charges"]].copy())

#verify that everything has been standardized to mean of 0, and a standard deviation of 1

print(f"Verifying means and standard deviation of each feature...")

#place the standardized values into a temporary dataframe for verifications

X\_std\_df = pd.DataFrame(X\_std, columns=X\_cols)

#print out the mean and the standard deviation for each of the 6 columns that we've standardized

for column in X\_cols:

col\_mean = round(X\_std\_df.loc[:,column].mean(), 4)

col\_std = round(X\_std\_df.loc[:,column].std(), 4)

print(f"For column '{column}', the mean is {col\_mean} and the standard deviation is {col\_std}.")

**CODE USED FOR CREATING THE COVARIANCE MATRIX**

#generate a covariance matrix to check if any of our variables are perfectly correlated

#define the colors red and yellow for the conditional formatting of the covariance matrix visualization

def highlight\_cells (val):

if val > 0.9:

color = 'red'

elif val > 0.6:

color = 'yellow'

else:

color = ''

return f"background: {color}"

#now that the colors have been defined, proceed to generate the covariance matrix

covariance\_matrix = pd.DataFrame.cov(X\_std\_df)

#apply the styling defined above. very closely correllated features will be displayed in red

covariance\_matrix.style.applymap(highlight\_cells)

#the covariance matrix indicates that Intial\_days and TotalCharge are nearly perfectly correlated

print(f"The correlation between Initial\_days and TotalCharge is {X\_std\_df.Initial\_days.corr(X\_std\_df.TotalCharge)}.")

# Due to this redundancy, we will be dropping TotalCharge.

**CODE USED FOR CREATING THE NEW DATAFRAME**

#we will now create a new dataframe that does not include TotalCharge before proceeding.

#create X dataframe with all of the continuous variables for PCA, excluding TotalCharge

X = df[["Age", "Income", "VitD\_levels", "Initial\_days","Additional\_charges"]].copy()

#create the list of column headers

X\_cols = list(X.columns)

#set y to ReAdmis as that is our target variable we want to predict

y = df["ReAdmis"]

**CODE USED FOR STANDARDIZATION OF THE VARIABLES FOR OUR NEW DATAFRAME**

#perform the standardize of all of the continuous variables selected for our PCA

X\_std = StandardScaler().fit\_transform(df[["Age", "Income", "VitD\_levels", "Initial\_days","Additional\_charges"]].copy())

#verify that everything has been standardized to mean of 0, and a standard deviation of 1

print(f"Verifying means and standard deviation of each feature...")

#place the standardized values into a temporary dataframe for verifications

X\_std\_df = pd.DataFrame(X\_std, columns=X\_cols)

#print out the mean and the standard deviation for each of the 5 columns that we've standardized

for column in X\_cols:

col\_mean = round(X\_std\_df.loc[:,column].mean(), 4)

col\_std = round(X\_std\_df.loc[:,column].std(), 4)

print(f"For column '{column}', the mean is {col\_mean} and the standard deviation is {col\_std}.")

#now that we have our final dataframe, save and export this dataframe as a CSV file

X\_std\_df.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d212\1medical\_clean-PREPAREDTASK2.csv', index=False)

**CODE USED TO PERFORM PCA**

# X\_std is the arrays created by the StandardScaler for us to perform PCA with

#create the PCA object

pca = PCA(n\_components = 5, random\_state = 369)

#fit the PCA to the standardized X data, then transform

X\_pca = pca.fit\_transform(X\_std)

#generate the matrix of PCA loadings, demonstrating the weight that a given feature contributes to that Principal Component

X\_pca\_loadings = pd.DataFrame(pca.components\_.T,

columns = ["PC1", "PC2", "PC3", "PC4", "PC5"],

index = X\_cols)

X\_pca\_loadings

#the 5 PC's generate the entire variance for this model

print(f"These 5 principal components account for {round(sum(pca.explained\_variance\_ratio\_ \* 100), 3)}% of variance.")

#show the individual contribution of each PC to the whole

print(f"The contribution of each principal component to the total is shown here:")

pc\_contributions = list(pca.explained\_variance\_ratio\_)

pc\_names = list(X\_pca\_loadings.columns)

for i in range(len(pc\_names)):

print(f"For {pc\_names[i]}, the contribution is {round(pc\_contributions[i] \* 100, 3)}%")

**CODE USED TO CREATE SCREE PLOT**

#create a scree plot to help visualize the contribution of each PC to the whole of variance

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

plt.xlabel("Number of Principal Components")

plt.ylabel("Percentage of Explained Variance")

plt.show();

**CODE USED FOR FINAL PCA**

#since we want 4 components in our PCA from the prior step, we repeat the PCA process to generate a 'final\_pca'

final\_pca = PCA(n\_components = 4, random\_state = 369)

#fit the PCA to the standardized X data, then transform

final\_pca.fit(X\_std)

final\_X\_pca = final\_pca.transform(X\_std)

#generate PCA loadings for the final\_pca

final\_X\_pca\_loadings = pd.DataFrame(final\_pca.components\_.T,

columns = ["PC1", "PC2", "PC3", "PC4"],

index = X\_cols)

final\_X\_pca\_loadings

#before proceeding, show the individual contribution of each PC to the whole

print(f"The amount of variance accounted for by each principal component can be seen here:")

pc\_contributions = list(final\_pca.explained\_variance\_ratio\_)

pc\_names = list(final\_X\_pca\_loadings.columns)

for i in range(len(pc\_names)):

print(f"For {pc\_names[i]}, the contribution is {round(pc\_contributions[i] \* 100, 3)}%")

print(f"These final 4 principal components account for {round(sum(final\_pca.explained\_variance\_ratio\_ \* 100), 3)}% of variance in the data.")

**CODE USED FOR SPLITTING THE final\_X\_pca TO TRAINING AND TESTING SETS**

#now that PCA finished, we can split the final\_X\_pca to train and test sets for classification

#split the data into train and test sets, 80% train, 20% test, use stratify to maintain proportions across split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(final\_X\_pca, y, train\_size = 0.8, test\_size=0.2, random\_state = 369, stratify = y)

#verify that each of the X sets are correctly shaped and to reflect the 4 PC's

print(f"The shape of the X\_train set is: {X\_train.shape}")

print(f"The shape of the X\_test set is: {X\_test.shape}")

**CODE USED FOR CREATING OUR CLASSIFICATION MODEL**

#now that the data has been split accordingly, we can create our classification model

classification\_model = DecisionTreeClassifier(random\_state=369).fit(X\_train, y\_train)

y\_predictions = classification\_model.predict(X\_test)

#generate an accuracy report for our model

test\_accuracy = accuracy\_score(y\_test, y\_predictions)

print(f'Decision tree accuracy: {test\_accuracy}')

# Predict the test set probabilities of the positive class

y\_pred\_proba = classification\_model.predict\_proba(X\_test)[:,1]

#create the final confusion matrix

final\_matrix = confusion\_matrix(y\_test, y\_predictions)

print("\nThe confusion matrix for this Decision Tree model:")

print("Predicted No Readmission | Predicted Readmission")

print(f" {final\_matrix[0]} Actual No Readmission")

print(f" {final\_matrix[1]} Actual Readmission\n")

**CODE USED FOR ROC CURVE AND AUC SCORE**

#generate the ROC curve

y\_true = y\_test

y\_scores = y\_pred\_proba

fpr, tpr, thresholds = roc\_curve(y\_true, y\_scores, pos\_label='Yes')

plt.plot(fpr, tpr)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.show()

#compute the roc\_auc score

roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba)

#print the roc auc score

print(f'Area Under the Curve (AUC) score: {roc\_auc}')