Analysis and Prediction of Heart Disease

By: Kevin Calloway, Rosario Fabian, Pramathesh Shukla, and Cassandra Steffey

College of Computing and Digital Media, DePaul University

DSC 510: Health Data Science

Stephanie Besser

June 11, 2021

Abstract

The incidence of Heart Disease has been steadily increasing over the years. Medical practitioners are able to use medical technology such as cardiac catheterization and electrocardiograms for the diagnosis of heart conditions. However; the ever expanding amount of data collected by medical and healthcare practitioners has provided opportunities to improve the outcomes of patients diagnosed with or at risk for heart disease. This study uses machine learning methods to predict coronary heart disease/heart attack and identify the relationship between coronary heart/heart attack and health indicators. The source of the data for the study was the Heart Disease Data Set from the UCI Machine Learning(ML) Repository. The dataset consists of 303 instances, a subset of 14 out of the 75 attributes were examined. The attributes consist of categorical, binary and numeric values. Subjects in the study experienced a vascular event (i.e., myocardial infarction (MI) or syncopal event). Exploratory analysis was performed on the data which showed that there is a relationship between gender and diabetes, gender and diabetes and 'chol'(Cholesterol) and 'trtbps'(Resting BP). Several data-mining algorithms (such as Logistic Regression, k-Nearest Neighbors, Random Forest, Gradient Boosting, Naive Bayes and Support Vector Machine) were used to develop classifiers and determine their accuracy. The study showed the Naive Bayes model performed the best in predicting MI with an overall accuracy of 0.85, a sensitivity of 0.84 and a specificity of 0.87.

Introduction

Medical researchers have explored the use of ML as a predictive tool for MI. This study investigates the performance of data mining ML methods and identifies which model has the best performance for predicting MI. The main research question to be presented in the project is “Using clinical data, can we build a predictive model to accurately predict the outcome of Myocardial Infarction”. Given the dataset, the goal was to utilize different statistical methods to determine which

performed more accurately in the prediction of MI. While exploring the dataset, we wanted to look at a few other aspects of interest. These secondary questions include the relationships of gender with disease and resting blood pressure, age with disease and resting ecg results, resting ECG results with cholesterol, and exercise induced angina with resting blood pressure.

Literature Review

The incidence of Congenital Heart Disease (CHD) has been steadily increasing each year. Hoffman et.al defines incidence as "the number of new affected persons per unit of time or population". In 2013, heart attack was the leading cause of death (Chitra, 2013). The increase in CHD was explored to better understand the influences behind this disease. It was found that the development and common use of the electrocardiogram in nurseries has allowed for the reporting and diagnosis of mild conditions (Hoffman, 2002). In other words, the increase in incidence was influenced more by the diagnosis of these mild conditions than the increase in moderate or severe CHD (Hoffman, 2002). Even with the use of electrocardiograms, the use of Machine Learning techniques to improve the prediction of cardiovascular disease (CVD) can aid medical practitioners in their decision making (Mohan, 2019).

There have been many studies performed that utilize machine learning for the prediction of heart disease using clinical data. Naive Bayes, Decision Trees, Random Forest, K-Nearest Neighbors, Support Vector Machines, and Artificial Neural Networks were the most commonly run methods in previous studies (Chitra, 2013, Garate-Escamila 2020, Mohan, 2019). In addition, Melillo used AdaboostM1 to predict myocardial infarction and stroke in hypertensive patients weeks/months before the event (Melillo, 2014). It was also concluded that neural networks are good for disease prediction in the early stage with accuracy improvement with reduction in features (Chitra, 2013). Logistic Regression and Regression Trees were compared and it was found that the Logistic Regression model had higher accuracy (Austin, 2010). Some novel approaches include a Hybrid Random Forest with a Linear Model

method proposed by Mohan et.al. The combination of these two methods allow for a higher accuracy than the approaches individually (Mohan 2019). In the literature by Chaurasia et. al, one article explored the use of J48 Decision Trees and Bagging algorithms for classification and the other utilized a Classification and Regression Tree (CART) method that performed the best for the provided heart disease clinical data (Chaurasia 2013, Chaurasia, 2014).

Methods

In this study, we reviewed previous literature on topics related to machine learning in heart disease prediction. Following a similar process to these previous studies, we explored the data by visualizing distributions, determining relationships among features, and performing normalizations and data reduction. After finalizing the data, we explored many classification methods by utilizing parameter optimization and cross validation processes to find the best performing model among each method. The best models were compared using statistical metrics to determine the best prediction models for myocardial infarction.

The first stage of the project looked at data preprocessing and exploratory analysis. During this phase, there was confusion about the “Thall” variable and the way it was recorded by the researcher. Due to this, the thall variable was excluded from the rest of the analysis. The five continuous variables which were scaled in different units were normalized using min-max normalization. Looking at the correlation plot, we found a few highly correlated variables and removed them accordingly.

As part of the exploratory data analysis, we analyzed some of the variables in SAS to get a better grasp of the relationships between features. This secondary research utilized different SAS procedures such as the proc univariate, sgplots, and Npar1way functions. There were three normality tests to determine significance used for these research questions; the Mann-Whitney (rank) test, the Chi-

Squared test, and the Mann-Whitney U (Wilcoxon) - nonparametric test based on their respective variable types.

Based on the distribution of the output variable, it is ideal to have approximately fifteen features left in the dataset. Principal Component Analysis (PCA) was used to reduce the dimensionality of the data to be within three features. Common factor analysis, also called principal factor analysis (PFA) or principal axis factoring (PAF) was used to identify the fewest factors which can account for the common variance (correlation) of a set of variables. This was followed by confirmatory analysis. The factorability of the PCA and PFA components were tested by performing Kaiser-Meyer-Olkin factor adequacy Test, Bartlett's Test of Sphericity, and Reliability Analysis using Cronbach's Alpha.

For the primary research question, the output of MI was modeled using Logistic Regression (LR), K-Nearest Neighbor (KNN) classification, Random Forest (RF) classification, Gradient Boost (GB) classification, Radial Basis Function Kernel Support Vector Machine (RBF SVM) classifier, and Naive Bayes (NB) classification. Hyperparameter tuning was performed to improve accuracy but the best performing models are the ones included in this article. These models have first been performed using all features in the normalized dataset and then again with the principal components found during PCA. Models were evaluated using the following metrics; accuracy, specificity, sensitivity, positive predictive value, negative predictive value, and area under the receiver operating characteristic curve.

Results

Overall the analysis of the secondary research questions demonstrated that many of the features are non-normal, as expected of categorical data, and the relationships mostly showed non-significance between many of the features. However, the variable sex shows there was a difference with different variables. After performing the Kaiser-Meyer-Olkin factor adequacy Test Overall MSA = 0.72.

Bartlett's Test of Sphericity shows p-value < 2.22e-16 which is very small that shows we had enough variance in the data so we can perform factor analysis.

By performing PCA, we see that there are five components that could be used to lower the dimensionality of the data but maintain enough variance.

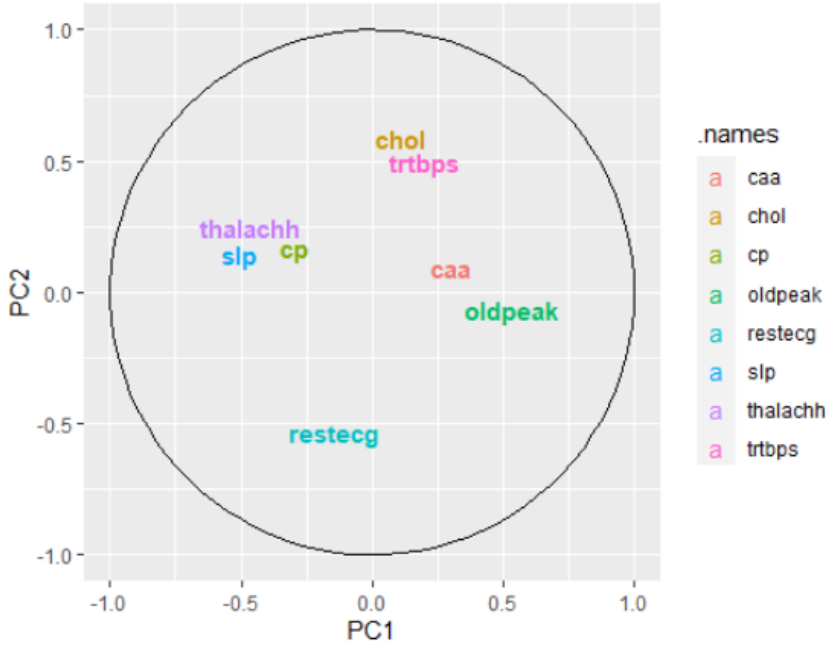


Figure 1: Plot of the contributions of the dataset attributes to the three principal components. The original variables for 4 distinct groupings after VARIMAX rotation are shown.

However, based on the percent variance and simplicity, only three components were used when evaluating our models. The first three components account for 56.12% of the variance in the data, which was determined to be adequate.

The first model performed was Logistic Regression. The first results with normalized dataset show that the best model gives the accuracy of 72%, it is a parsimonious model where the most important variables are cholesterol, sex and age. It is important to mention that when we tried to add more medical variables these create correlation with cholesterol. Then, in the second model we worked with the PCA dataset which consists of three main components. The model gives the curve of ROC of 87% and the three components are significant in the model of prediction of heart disease.

The Logistic Regression model results show that the accuracy is 0.76, the sensitivity is 0.75, the specificity is 0.77, the positive predictive value is 0.73 and the negative predictive value is 0.79. These values show that 22% of the patients with illness could be categorized with error type 1. This results in the ROC curves similar to the result of the article (Austin, Tu & Le, 2010). It shows that we are getting similar performance compared with other studies.

The KNN model was found to be the most accurate when working with the three principal components compared to the normalized data. The model with the best performance using this data was a model with a K value of 9 and a weight parameter set to 'distance'. These parameters were determined using the grid search method with cross validation. The accuracy for this model on the training data was 1.0 and 0.72 on the testing data which suggests that the model overfits the training data slightly. This model showed eight instances of false negatives and nine cases of false positives. The other metrics are shown in Table 1.

The Random Forest model was run with a split value of 4 and gave a total of 500 trees. The model gives different age, sex, and resting ecg as the important variables. The Random Forest Model shows the accuracy of the training set is 78% and for the testing set 52.61%.

The Gradient Boost classifier model produced an accuracy of 77 percent after hyperparameter tuning on the validation set. The RBF SVM classifier model produced an accuracy of 72 percent after hyperparameter tuning. Five values had a Type II error and 4 had a Type I error. The accuracy of the default RBF SVM model was identical to the parameter tuned model. Parameter tuning prevents overfitting and reduces the predictive accuracy on the dataset. Through the GridSearch method, the optimal parameter values used in the default model were identified. .

Naive Bayes was performed on the normalized data and the principal components where they performed equally well. The accuracy on the training dataset was 0.82 and the accuracy on the testing

data was 0.85 which does not suggest overfitting. This model showed five instances of false negatives and four cases of false positives. The other metrics are shown in Table 1.

Model Comparison Table							
Model	**Parameters**	**Accuracy**	**Sensitivity**	**Specificity**	**PPV**	**NPV**	**AUC**
LR	Three PCA components	0.76	0.75	0.77	0.73	0.79	0.76
KNN	N_neighbors = 7 Weights = 'uniform'	0.72	0.74	0.70	0.72	0.72	0.72
RF	Split value = 4 Number of trees = 500	0.72	0.75	0.74	0.70	0.73	0.75
GB	Learning_rate = 0.25 N_estimators = 50 Max_depth = 3	0.77	0.80	0.75	0.69	0.87	0.85
RBF Kernel SVM	C = 5.090	0.71	0.87	0.56	0.56	0.87	0.81
NB	defaults	0.85	0.84	0.87	0.87	0.84	0.85

Table 1: Table of the six methods performed during this analysis and the resulting metrics. The models from top to bottom are Logistic Regression (LR), K-Nearest Neighbors (KNN), Random Forest (RF), Gradient Boost (GB), Radial Basis Function Kernel Support Vector Machine (RBF Kernel SVM), and Naive Bayes (NB). The metrics from right to left are the accuracy score on the test data, the sensitivity, the specificity, the positive predictive value (PPV), the negative predictive value (NPV), and the aurea under the receiver operating characteristic curve (AUC).

Discussion

It was found that the Naive Bayes method performed the best based on the metrics used for model comparisons. The key metrics to mention are the overall accuracy of the model which was 0.85, the sensitivity which was 0.84 and the specificity of 0.87. Looking at these values, we can say that those with heart disease that are predicted as having heart disease is 84% and those that do not have heart disease that are predicted as such is 87%.

This model performs well but it is probable that the use of feature selection or hybrid models could better the prediction of heart disease. Some hybrid models, such as the ones researched in the literature, could be a good starting point to work from in future studies. In addition, one of the largest limitations for this dataset is the number of observations. Due to the smaller number of observations, it may be beneficial to look at sampling techniques to get more accurate results from the models. The distribution of observations with disease and without disease were fairly proportional; however, the use of Synthetic Minority Over-Sampling Technique to make the distribution more even. The use of a different sampling method may give different results for each of the models.

References

Austin, P. C., Tu, J. V., & Lee, D. S. (2010). Logistic regression had superior performance compared with regression trees for predicting in-hospital mortality in patients hospitalized with heart failure. Journal of clinical epidemiology, 63(10), 1145-1155.

Chaurasia, V., & Pal, S. (2014). Data mining approach to detect heart diseases. International Journal of Advanced Computer Science and Information Technology (IJACSIT) Vol, 2, 56-66.

Chaurasia, V., & Pal, S. (2013). Early prediction of heart diseases using data mining techniques. Caribbean Journal of Science and Technology, 1, 208-217.

Chitra, R., & Seenivasagam, V. (2013). Review of heart disease prediction systems using data mining and hybrid intelligent techniques. ICTACT journal on soft computing, 3(04), 605-09.

Gárate-Escamila, A. K., El Hassani, A. H., & Andrès, E. (2020). Classification models for heart disease prediction using feature selection and PCA. Informatics in Medicine Unlocked, 19, 100330.

Hoffman, J. I., & Kaplan, S. (2002). The incidence of congenital heart disease. Journal of the American college of cardiology, 39(12), 1890-1900.

Melillo, P., Izzo, R., Orrico, A., Scala, P., Attanasio, M., Mirra, M., ... & Pecchia, L. (2015). Automatic prediction of cardiovascular and cerebrovascular events using heart rate variability analysis. PloS one, 10(3), e0118504.

Mohan, S., Thirumalai, C., & Srivastava, G. (2019). Effective heart disease prediction using hybrid machine learning techniques. IEEE Access, 7, 81542-81554.

Appendix

I. Data Cleaning and Analysis

Code:

```
### Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.preprocessing import MinMaxScaler, StandardScaler

### Load the Data
heart = pd.read_csv('heart_Kaggle.csv')

### Look at the data
heart.head()

<b>Data Description </b>

age (numeric) - the age of the patient in years

sex (categorical) - the sex of the patient (1 = male, 0 = female)

cp (numerical/categorical) - chest pain type (0 = typical angina, 1 = atypical angina, 2 = non-anginal pain, 3 = asymptomatic)

trtbps (numeric) - resting blood pressure in mmHg

chol (numeric) - serum cholesterol in mg/dl

fbs (binary) - fasting blood sugar > 120 mg/dl (1 = true, 0 = false)

restecg (numeric/discrete) - resting electrocardiographic results (0 = normal, 1 = ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)

thalach (numeric) - maximum heart rate achieved

exang (binary) - exercise induced angina (1 = yes, 0 = no)

oldpeak (numeric) - ST depression induced by exercise relative to rest

slp (numeric/categorical) - the slope of the peak exercise ST segment (0 = upsloping, 1 = flat, 2 = downsloping)

ca (numeric) - number of major vessels (0-3) colored by fluoroscopy
```

thal (numeric) - Thalium uptake rate (1 = fixed defect, 2 = normal, 3 = reversible defect)

Condition (binary) - disease condition (0 = no disease, 1 = disease)

```
#Copy data
heart_s = heart.copy()

### Change Categorical Data to Strings
heart_s['sex'].replace([0,1], ['Female', 'Male'], inplace=True)
heart_s['cp'].replace([0,1, 2, 3], ['Typical Angina', 'Atypical Angina', 'Non-Anginal',
'Asymptomatic'], inplace=True)
heart_s['fbs'].replace([0,1], ['False', 'True'], inplace=True)
heart_s['restecg'].replace([0,1, 2], ['Normal', 'STT Wave', 'Ventricular Hypertrophy'],
inplace=True)
heart_s['exng'].replace([0,1], ['No', 'Yes'], inplace=True)
heart_s['slp'].replace([0,1, 2], ['Upsloping', 'Flat', 'Downsloping'], inplace=True)
heart_s['output'].replace([0,1], ['No Disease', 'Disease'], inplace=True)

heartY = heart['output']
heartX = heart.drop('output', axis = 1)
heartX = heartX.drop('thall', axis = 1)
heartX_s = heart_s.drop('output', axis = 1)
heartX_s = heart_s.drop('thall', axis = 1)

### Correlations
heartX.corr(method='spearman')

### Show the correlation as a heatmap - pearson
corplot = sb.heatmap(heartX.corr(method='spearman'), cmap="YlGnBu")
plt.show()

### Correlations
heartX_s.corr(method='spearman')

### Correlations
heartX_s.corr(method='spearman')

### Show the correlation as a heatmap - pearson
corplot = sb.heatmap(heartX_s.corr(method='spearman'), cmap="YlGnBu")
plt.show()
#Look at the breakdown of features
heartX_s.describe(include=[np.number])

### Box Plot for Numeric Data
heartX_s.boxplot(column=['age'], return_type='axes')
plt.show()
```

```
heartX_s.boxplot(column=['trtbps'], return_type='axes')
plt.show()

heartX_s.boxplot(column=['chol'], return_type='axes')
plt.show()

heartX_s.boxplot(column=['thalachh'], return_type='axes')
plt.show()

heartX_s.boxplot(column=['oldpeak'], return_type='axes')
plt.show()

heartX_s.boxplot(column=['caa'], return_type='axes')
plt.show()

### Looking at the distribution of the categorical variables
heartX_s.describe(include=[np.object])

### Histograms for Categorical Data
plt.hist(heartX_s['sex'])
plt.title('Sex Distribution')
plt.show()

plt.hist(heartX_s['cp'])
plt.title('Cp Distribution')
plt.show()

plt.hist(heartX_s['fbs'])
plt.title('Fbs Distribution')
plt.show()

plt.hist(heartX_s['restecg'])
plt.title('Restecg Distribution')
plt.show()

plt.hist(heartX_s['exng'])
plt.title('Exng Distribution')
plt.show()

plt.hist(heartX_s['slp'])
plt.title('Slp Distribution')
plt.show()

plt.hist(heartX_s['output'])
plt.title('Output Distribution')
plt.show()

### Get the actual counts
```

```
heart['output'].value_counts()

#Separate the data by disease
noDisease_s = heart_s[heart_s.output == 'No Disease']
Disease_s = heart_s[heart_s.output == 'Disease']

### Boxplot to visualize distribution - age
plt.subplot(121)
noDisease_s.boxplot(column=['age'], return_type='axes')
plt.title('Age Distribution-No Disease')

plt.subplot(122)
Disease_s.boxplot(column=['age'], return_type='axes')
plt.title('Age Distribution-Disease')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - trtbps
plt.subplot(121)
noDisease_s.boxplot(column=['trtbps'], return_type='axes')
plt.title('Rest BP Distribution-No Disease')

plt.subplot(122)
Disease_s.boxplot(column=['trtbps'], return_type='axes')
plt.title('Rest BP Distribution-Disease')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - chol
plt.subplot(121)
noDisease_s.boxplot(column=['chol'], return_type='axes')
plt.title('Cholesterol Distribution-No Disease')

plt.subplot(122)
Disease_s.boxplot(column=['chol'], return_type='axes')
plt.title('Cholesterol Distribution-Disease')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - thalachh
plt.subplot(121)
noDisease_s.boxplot(column=['thalachh'], return_type='axes')
plt.title('Max HR Distribution-No Disease')

plt.subplot(122)
```

```
Disease_s.boxplot(column=['thalachh'], return_type='axes')
plt.title('Max HR Distribution-Disease')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - oldpeak
plt.subplot(121)
noDisease_s.boxplot(column=['oldpeak'], return_type='axes')
plt.title('ST Distribution-No Disease')

plt.subplot(122)
Disease_s.boxplot(column=['oldpeak'], return_type='axes')
plt.title('ST Distribution-Disease')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - oldpeak
plt.subplot(121)
noDisease_s.boxplot(column=['caa'], return_type='axes')
plt.title('Vessels Distribution-No Disease')

plt.subplot(122)
Disease_s.boxplot(column=['caa'], return_type='axes')
plt.title('Vessels Distribution-Disease')

plt.tight_layout()
plt.show()

#Crosstab Histograms for Categorical data
#Crosstab - sex/output
out_sex = pd.crosstab(heartX_s['sex'], heartX_s['output'])
print(out_sex)

#plot
out_sex.plot(kind='bar')
plt.title('Sex Distribution')
plt.show()

#Crosstab - cp/output
out_cp = pd.crosstab(heartX_s['cp'], heartX_s['output'])
print(out_cp)

#plot
out_cp.plot(kind='bar')
plt.title('Chest Pain Distribution')
plt.show()
```

```
#Crosstab - fbs/output
out_fbs = pd.crosstab(heartX_s['fbs'], heartX_s['output'])
print(out_fbs)

#plot
out_fbs.plot(kind='bar')
plt.title('Fasting Blood Sugar Distribution')
plt.show()

#Crosstab - restecg/output
out_restecg = pd.crosstab(heartX_s['restecg'], heartX_s['output'])
print(out_restecg)

#plot
out_restecg.plot(kind='bar')
plt.title('Resting ECG Distribution')
plt.show()

#Crosstab - exng/output
out_exng = pd.crosstab(heartX_s['exng'], heartX_s['output'])
print(out_exng)

#plot
out_exng.plot(kind='bar')
plt.title('Exercise Induced Angina Distribution')
plt.show()

#Crosstab - slp/output
out_slp = pd.crosstab(heartX_s['slp'], heartX_s['output'])
print(out_slp)

#plot
out_slp.plot(kind='bar')
plt.title('Slope ST Peak Distribution')
plt.show()

#Normalizaion - Zscore
scaler = StandardScaler()
scaler.fit(heartX)
x_Scal = scaler.transform(heartX)

#Normaliztaion - min max
scaler2 = MinMaxScaler()
scaler2.fit(heartX)
x_Scal2 = scaler2.transform(heartX)

#Create df
```

```
x_Stand = pd.DataFrame(x_Scal)
x_Stand.columns = heartX.columns

x_MinMax = pd.DataFrame(x_Scal2)
x_MinMax.columns = heartX.columns

### Boxplot to visualize distribution - age
plt.subplot(121)
x_Stand.boxplot(column=['age'], return_type='axes')
plt.title('Age Distribution-Standard')

plt.subplot(122)
x_MinMax.boxplot(column=['age'], return_type='axes')
plt.title('Age Distribution-Min Max')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - trtbps
plt.subplot(121)
x_Stand.boxplot(column=['trtbps'], return_type='axes')
plt.title('Rest BP Distribution-Standard')

plt.subplot(122)
x_MinMax.boxplot(column=['trtbps'], return_type='axes')
plt.title('Rest BP Distribution-Min Max')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - chol
plt.subplot(121)
x_Stand.boxplot(column=['chol'], return_type='axes')
plt.title('Cholesterol Distribution-Standard')

plt.subplot(122)
x_MinMax.boxplot(column=['chol'], return_type='axes')
plt.title('Cholesterol Distribution-Min Max')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - thalachh
plt.subplot(121)
x_Stand.boxplot(column=['thalachh'], return_type='axes')
plt.title('Max HR Distribution-Standard')

plt.subplot(122)
```

```
x_MinMax.boxplot(column=['thalachh'], return_type='axes')
plt.title('Max HR Distribution-Min Max')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - oldpeak
plt.subplot(121)
x_Stand.boxplot(column=['oldpeak'], return_type='axes')
plt.title('ST Distribution-Standard')

plt.subplot(122)
x_MinMax.boxplot(column=['oldpeak'], return_type='axes')
plt.title('ST Distribution-Min Max')

plt.tight_layout()
plt.show()

### Boxplot to visualize distribution - oldpeak
plt.subplot(121)
x_Stand.boxplot(column=['caa'], return_type='axes')
plt.title('Vessels Distribution-Standard')

plt.subplot(122)
x_MinMax.boxplot(column=['caa'], return_type='axes')
plt.title('Vessels Distribution-Min Max')

plt.tight_layout()
plt.show()
for c in heartX.columns:
   plt.subplot(121)
   plt.hist(x_Stand[c])
   plt.title(str(c) + ' Distribution-Standard')

   plt.subplot(122)
   plt.hist(x_MinMax[c])
   plt.title(str(c) + ' Distribution-Min Max')

   plt.tight_layout()
   plt.show()

#Correlations
### Show the correlation as a heatmap - spearman
corplot = sb.heatmap(x_Stand.corr(method='spearman'), cmap="YlGnBu")
plt.show()

### Show the correlation as a heatmap - spearman
corplot = sb.heatmap(x_MinMax.corr(method='spearman'), cmap="YlGnBu")
```

```
plt.show()
```

Outputs:

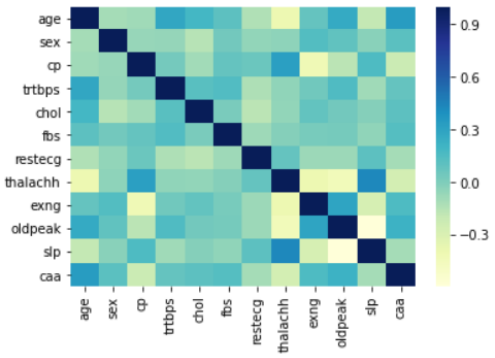


Figure 2: Correlation matrix for original data features

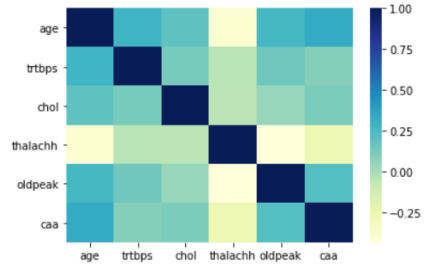


Figure 3: Correlation matrix for numeric data features from the original data

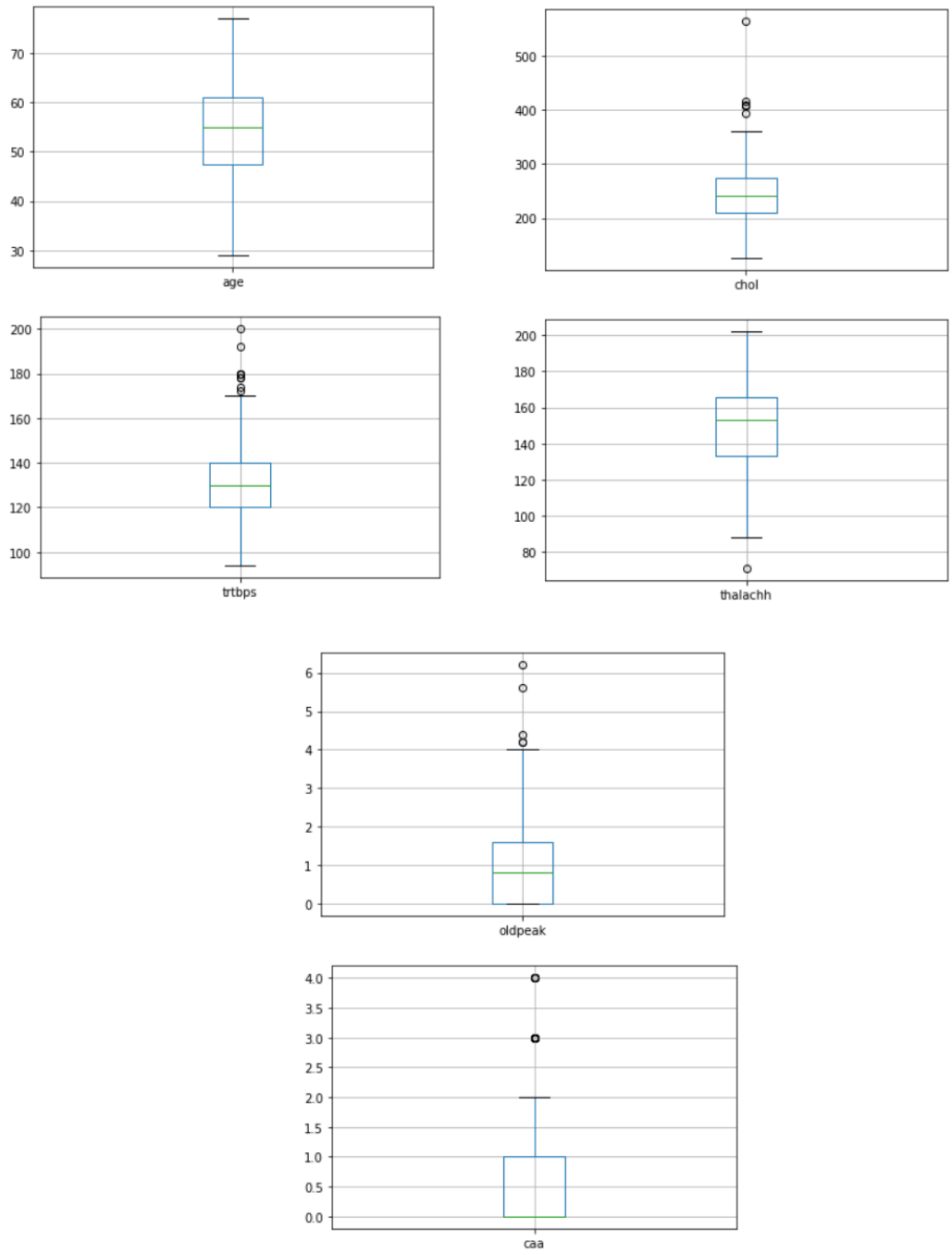


Figure 4: Boxplots of numeric features in the original data

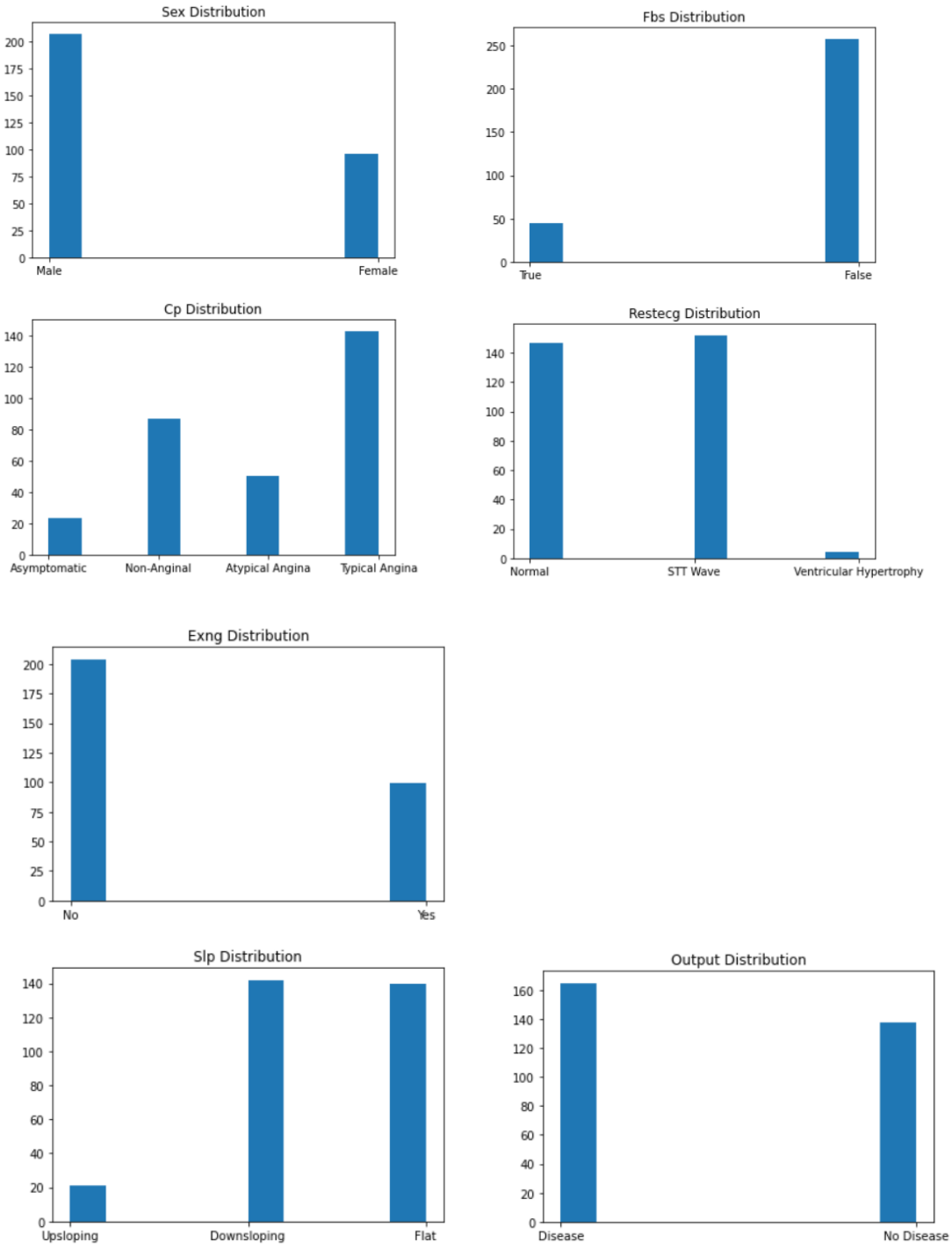


Figure 5: Histograms of categorical features in the original data

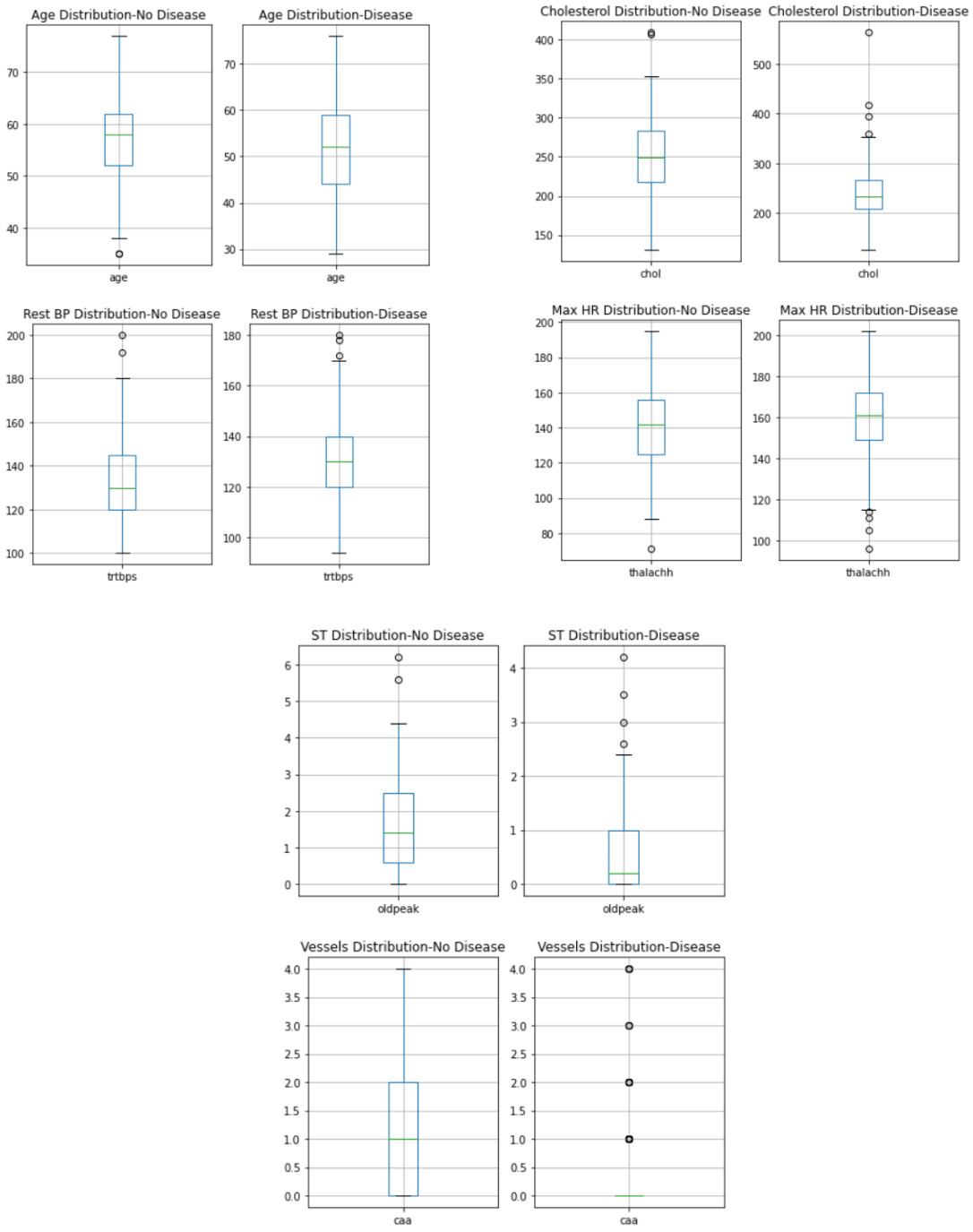


Figure 6: Boxplots of numerical features in the original data by output class

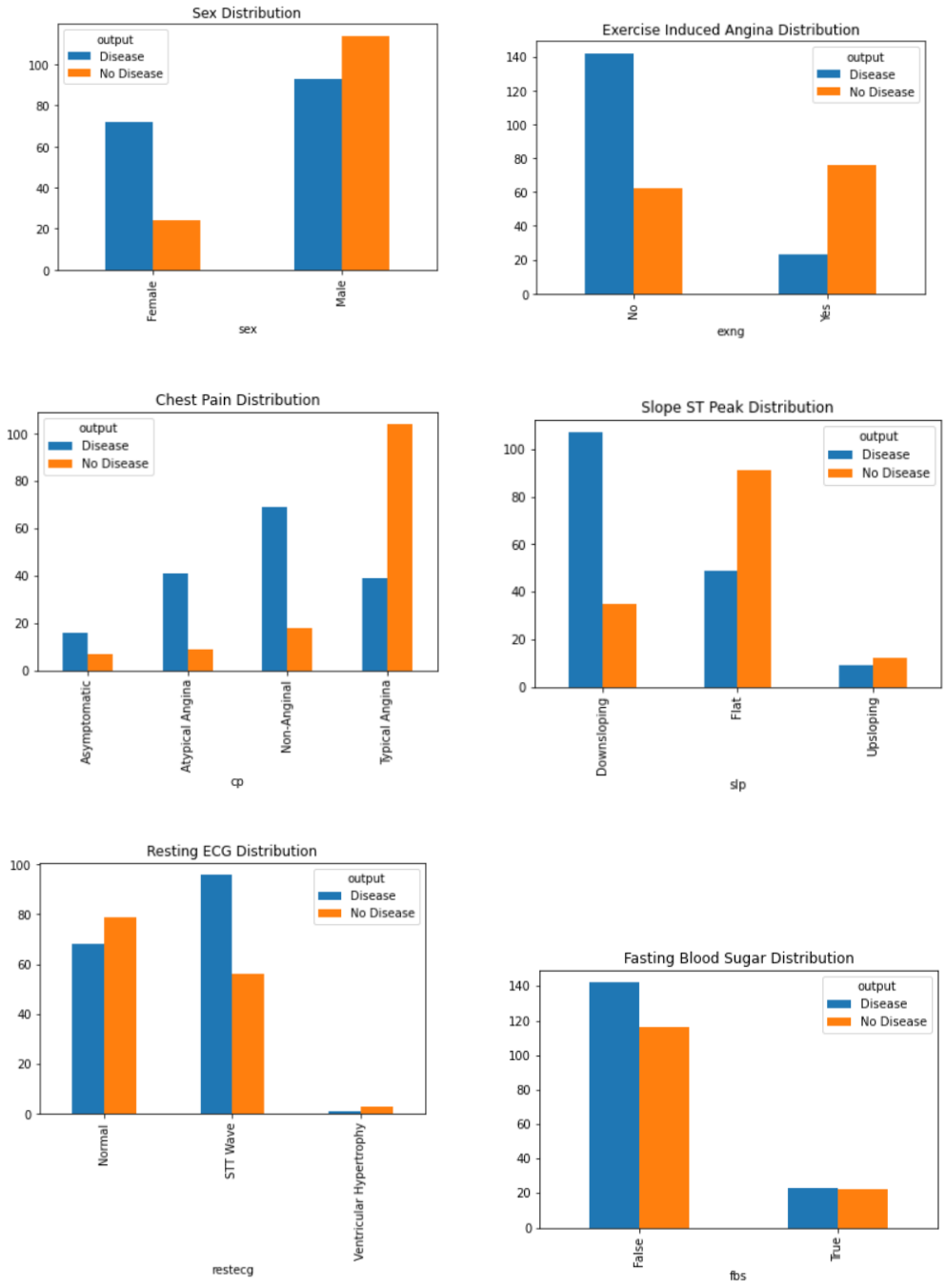


Figure 7: Histograms of categorical features in the original data by output class and their crosstab

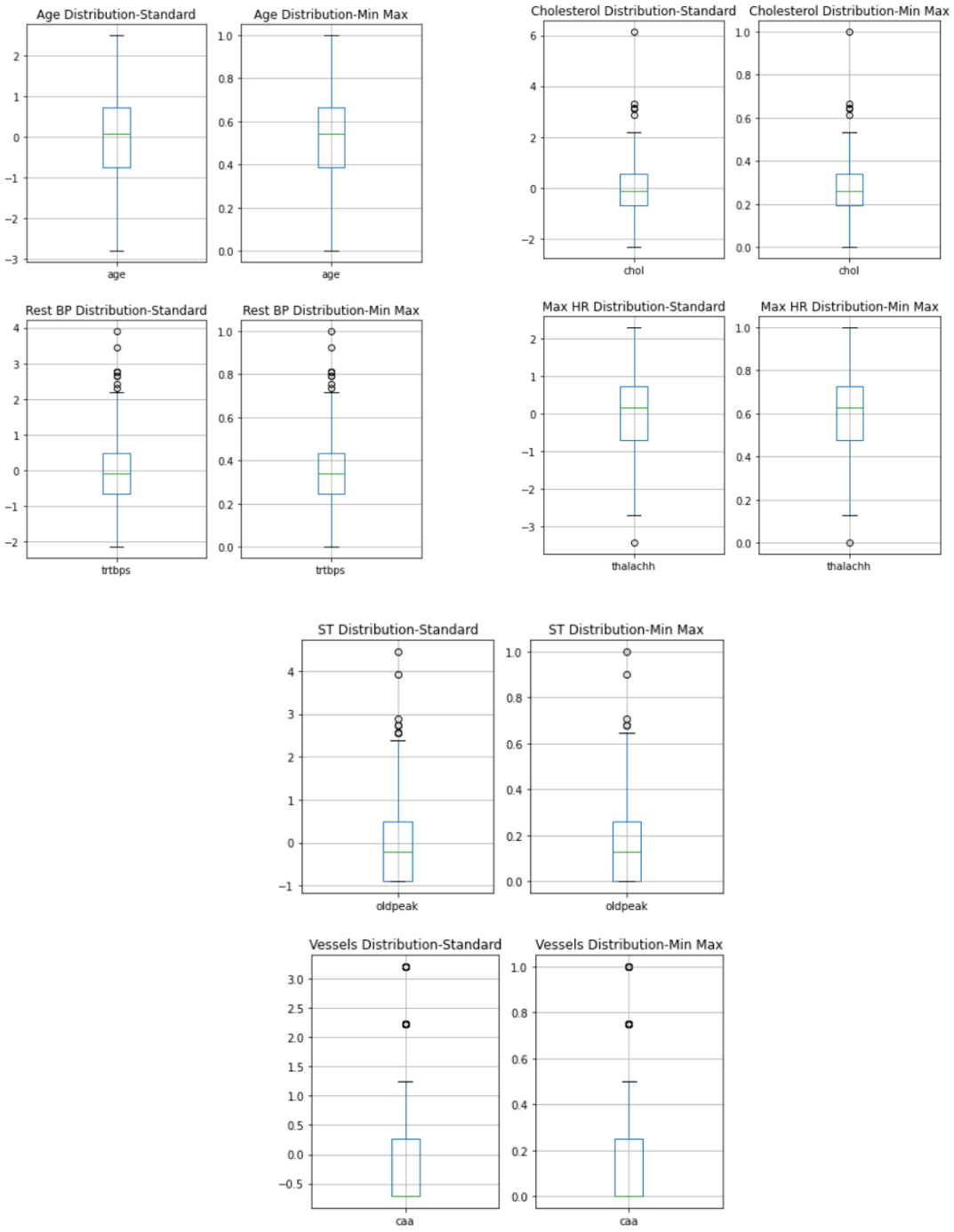
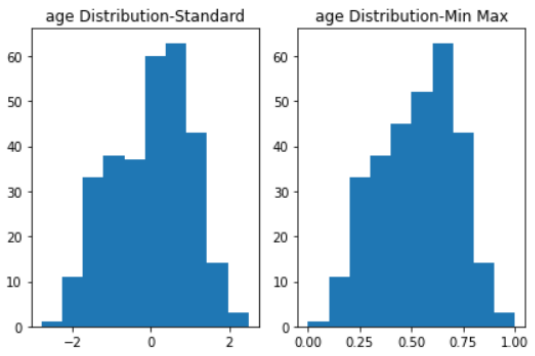
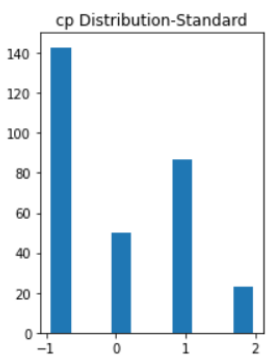
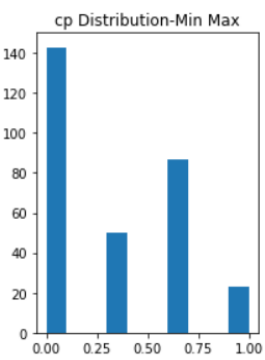


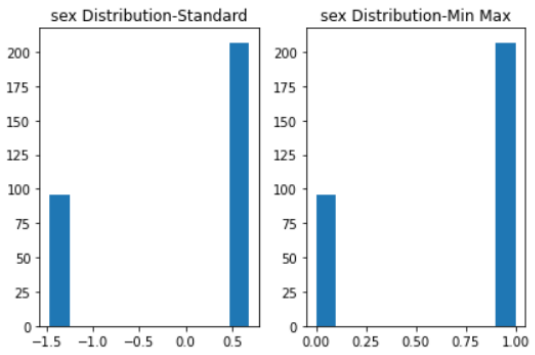
Figure 8: Boxplots of numeric features normalized using standard scaler and min max scaler normalization


age Distribution-Standard

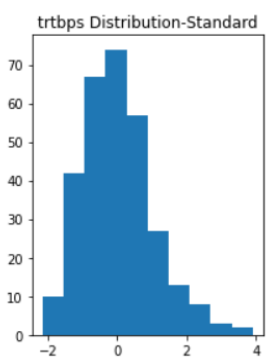
age Distribution-Min Max

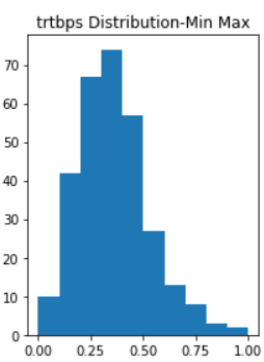

cp Distribution-Standard

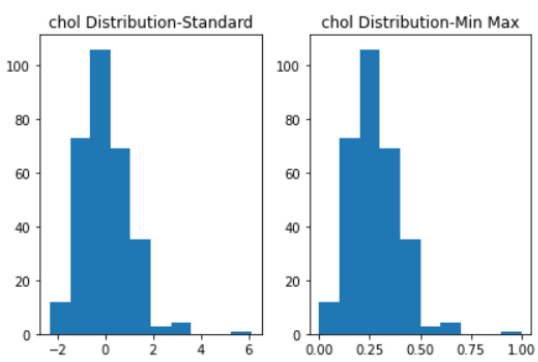

cp Distribution-Min Max


sex Distribution-Standard

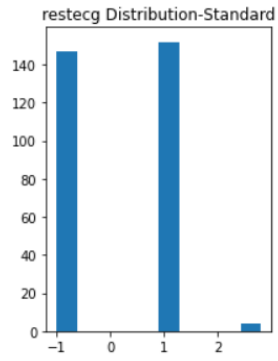
sex Distribution-Min Max

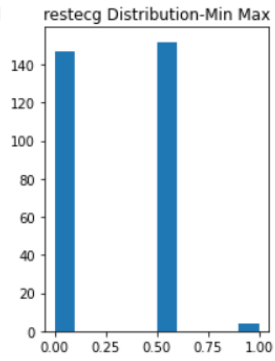

trtbps Distribution-Standard

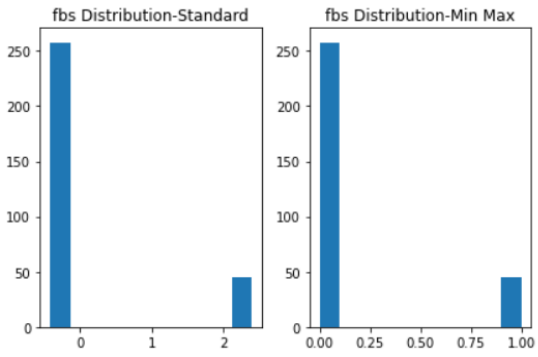

trtbps Distribution-Min Max


chol Distribution-Standard

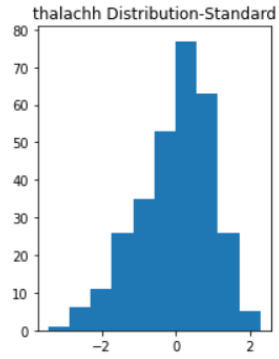
chol Distribution-Min Max

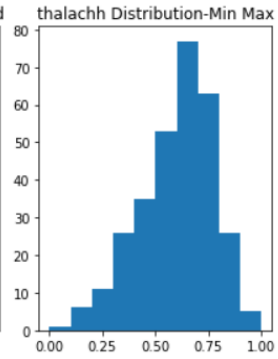

restecg Distribution-Standard


restecg Distribution-Min Max


fbs Distribution-Standard

fbs Distribution-Min Max


thalachh Distribution-Standard


thalachh Distribution-Min Max

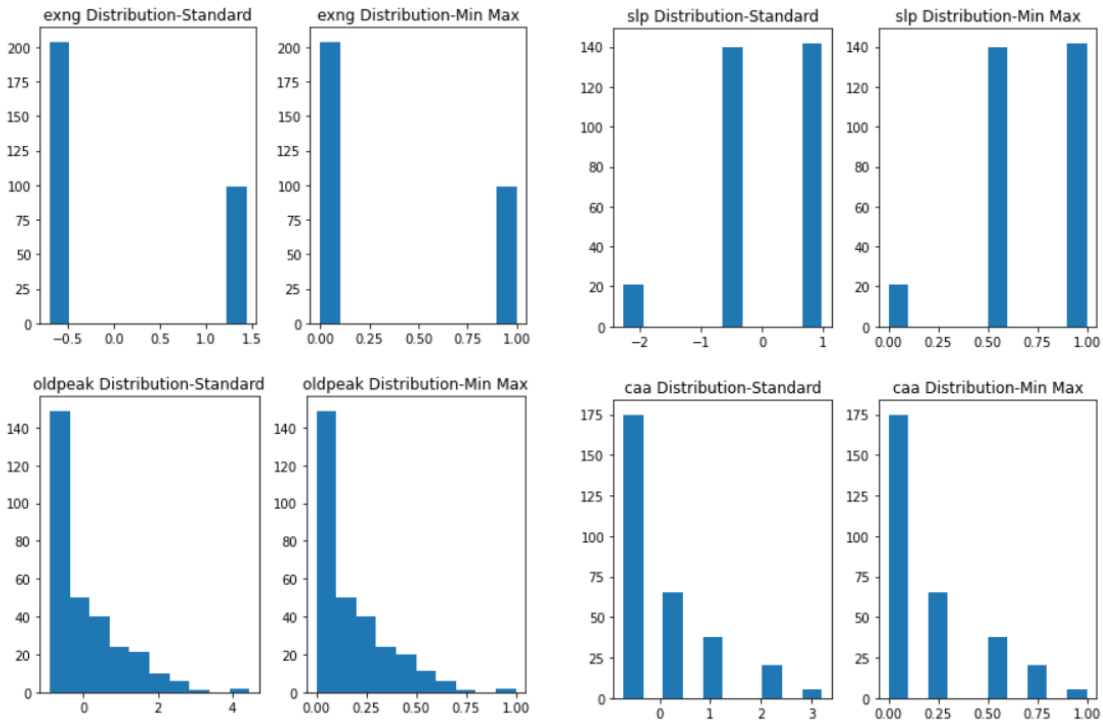


Figure 9: Histograms of numeric features normalized using standard scaler and min max scaler normalization

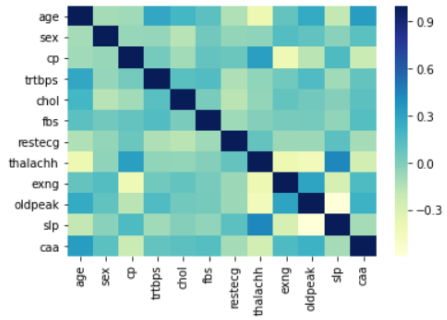
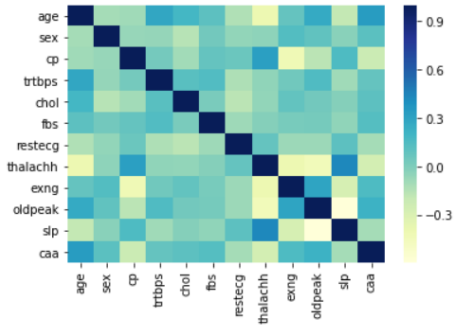


Figure 10: Correlation matrix for the standard scaler data



No

Figure 11: Correlation matrix for the Min Max scaler data

II. Principal Component Analysis

Code:

```
setwd('C:/Users/PBS/Desktop/CSC 510')

dataset <- read.csv(file="heart_cleaned_MinMaxNormalization_thall_removed.csv", header=TRUE, sep=",")

library(corrplot)

##############################################################################

# PCA_Plot functions

##############################################################################

PCA_Plot = function(pcaData)

{

  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)

  circle = data.frame(x = cos(theta), y = sin(theta))

  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))

  p + geom_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names, colour = .names, fontface="bold")) +

          coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")

}

PCA_Plot_Secondary = function(pcaData)

{

  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)

  circle = data.frame(x = cos(theta), y = sin(theta))

  p = ggplot(circle,aes(x,y)) + geom_path()
```

```
  loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))

  p + geom_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names, colour = .names,
fontface="bold")) +

        coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4")

}

PCA_Plot_Psyc = function(pcaData)

{

  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)

  circle = data.frame(x = cos(theta), y = sin(theta))

  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = as.data.frame(unclass(pcaData$loadings))

  s = rep(0, ncol(loadings))

  for (i in 1:ncol(loadings))

  {

        s[i] = 0

        for (j in 1:nrow(loadings))

        s[i] = s[i] + loadings[j, i]^2

        s[i] = sqrt(s[i])

  }

  for (i in 1:ncol(loadings))

        loadings[, i] = loadings[, i] / s[i]

  loadings$.names = row.names(loadings)

  p + geom_text(data=loadings, mapping=aes(x = RC1, y = RC2, label = .names, colour = .names,
fontface="bold")) +

        coord_fixed(ratio=1) + labs(x = "RC1", y = "RC2")
```

```
}

PCA_Plot_Psyc_Secondary = function(pcaData)

{

 library(ggplot2)

 theta = seq(0,2*pi,length.out = 100)

 circle = data.frame(x = cos(theta), y = sin(theta))

 p = ggplot(circle,aes(x,y)) + geom_path()

 loadings = as.data.frame(unclass(pcaData$loadings))

 s = rep(0, ncol(loadings))

 for (i in 1:ncol(loadings))

 {

        s[i] = 0

  for (j in 1:nrow(loadings))

        s[i] = s[i] + loadings[j, i]^2

        s[i] = sqrt(s[i])

 }

 for (i in 1:ncol(loadings))

        loadings[, i] = loadings[, i] / s[i]

 loadings$.names = row.names(loadings)

 print(loadings)

 p + geom_text(data=loadings, mapping=aes(x = RC3, y = RC4, label = .names, colour = .names,
fontface="bold")) +

        coord_fixed(ratio=1) + labs(x = "RC3", y = "RC4")

}

############################################################

# Data Preparation
```

```
#########################################################

heart = read.csv("heart_cleaned_MinMaxNormalization_thall_removed.csv") #PC6(93.111%) Scree (3PCs:60.16%)

head(heart)

output <- heart["output"]

# head(output)

#########################################################

colnames(heart)

# Create table with "cp", "trtbps", "chol", "restecg","thalachh", slp", "caa"

heart_PCA1 = heart[, c(3:5, 7:8, 10:12)]

head(heart_PCA1)

heartpc1 = prcomp(heart_PCA1, scale=T)

summary(heartpc1)

round(heartpc1$rotation, 4)

round(head(heartpc1$x), 4)

plot(heartpc1)

abline(1, 0, col="red")  # Put in a line at var=1

head(heartpc1$x)

heart_mm <- heartpc1$x[, c(1:3)]

head(heart_mm)

# Add 'output' attribute column to file

# heart_mm[,output] <- output

heart_mm2 <- cbind(heart_mm, output)

head(heart_mm2)
```

Output:

```
> head(heart)
  age sex cp trtbps   chol fbs restecg thalachh exng oldpeak slp caa output
1  63   1  3 0.4811 0.2443   1       0   0.6031    0  0.3710   0   0      1
2  37   1  2 0.3396 0.2831   0       1   0.8855    0  0.5645   0   0      1
3  41   0  1 0.3396 0.1781   0       0   0.7710    0  0.2258   2   0      1
4  56   1  1 0.2453 0.2511   0       1   0.8168    0  0.1290   2   0      1
5  57   0  0 0.2453 0.5205   0       1   0.7023    1  0.0968   2   0      1
6  57   1  0 0.4340 0.1507   0       1   0.5878    0  0.0645   1   0      1
> output <- heart["output"]
> ####################################################
> colnames(heart)
 [1] "age"      "sex"      "cp"       "trtbps"   "chol"     "fbs"
 [7] "restecg"  "thalachh" "exng"     "oldpeak"  "slp"      "caa"
[13] "output"
> # Create table with "cp", "trtbps", "chol", "restecg","thalachh", slp", "caa"
> heart_PCA1 = heart[, c(3:5, 7:8, 10:12)]
> head(heart_PCA1)
  cp trtbps   chol restecg thalachh oldpeak slp caa
1  3 0.4811 0.2443       0   0.6031  0.3710   0   0
2  2 0.3396 0.2831       1   0.8855  0.5645   0   0
3  1 0.3396 0.1781       0   0.7710  0.2258   2   0
4  1 0.2453 0.2511       1   0.8168  0.1290   2   0
5  0 0.2453 0.5205       1   0.7023  0.0968   2   0
6  0 0.4340 0.1507       1   0.5878  0.0645   1   0
> heartpc1 = prcomp(heart_PCA1, scale=T)
> summary(heartpc1)
Importance of components:
                          PC1   PC2   PC3   PC4   PC5    PC6    PC7    PC8
Standard deviation     1.1.47 1.104 1.053 0.956 0.921 0.8630 0.7837 0.6249
Proportion of Variance 0.27 0.152 0.139 0.114 0.106 0.0931 0.0768 0.0488
Cumulative Proportion  0.27 0.423 0.561 0.675 0.781 0.8744 0.9512 1.0000
> round(heartpc1$rotation, 4)
             PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
cp       -0.2904  0.1703 -0.6293  0.1074  0.1201 -0.5250 -0.4316 -0.0534
trtbps    0.1998  0.4988 -0.3641  0.4494 -0.0307  0.6074 -0.0504  0.0660
chol      0.1083  0.5867  0.2779 -0.1362 -0.6685 -0.2968 -0.1099  0.0347
restecg  -0.1482 -0.5314 -0.1025  0.5283 -0.6324 -0.0079 -0.0048  0.0771
thalachh -0.4679  0.2479 -0.1462  0.0786  0.0052 -0.1452  0.8045  0.1539
oldpeak   0.5293 -0.0618 -0.2913  0.0160 -0.0931 -0.2117  0.3625 -0.6679
slp      -0.5027  0.1447  0.3652  0.2600  0.0689  0.1405 -0.1383 -0.6941
caa       0.3005  0.0964  0.3796  0.6441  0.3527 -0.4262  0.0365  0.1848
```

Figure 11

```
> round(head(heartpc1$x), 4)
        PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
[1,]  1.1960  0.6387 -2.9039 -0.9725  0.5104 -0.7344 -0.1613  0.5805
[2,]  0.8513 -0.4346 -2.6238 -0.3575 -1.0894 -1.3109  1.9286  0.2921
[3,] -0.9639  0.3096 -0.2578 -0.5820  0.9785  0.4104  0.8221 -0.9790
[4,] -1.6890 -0.5265  0.0384  0.0948 -0.5695 -0.0626  0.7972 -0.4631
[5,] -0.9458  0.4925  1.4268 -0.3736 -2.1952 -0.0980  0.3759 -0.3184
[6,] -0.0253 -1.1587 -0.3037  0.0885 -0.2403  1.4256  0.2971  0.7894
> plot(heartpc1)
> abline(1, 0, col="red")  # Put in a line at var=1
> head(heartpc1$x)
          PC1      PC2       PC3       PC4      PC5       PC6      PC7
[1,]  1.196019  0.63868 -2.903942 -0.972528  0.51041 -0.734418 -0.16133
[2,]  0.851346 -0.43456 -2.623793 -0.357473 -1.08939 -1.310945  1.92857
[3,] -0.963895  0.30958 -0.257824 -0.582011  0.97847  0.410359  0.82206
[4,] -1.689012 -0.52648  0.038425  0.094849 -0.56954 -0.062579  0.79718
[5,] -0.945770  0.49251  1.426751 -0.373572 -2.19516 -0.098012  0.37595
[6,] -0.025327 -1.15869 -0.303727  0.088502 -0.24028  1.425645  0.29712
         PC8
[1,]  0.58049
[2,]  0.29207
[3,] -0.97900
[4,] -0.46311
[5,] -0.31839
[6,]  0.78937
> heart_mm <- heartpc1$x[, c(1:3)]
> head(heart_mm)
          PC1      PC2       PC3
[1,]  1.196019  0.63868 -2.903942
[2,]  0.851346 -0.43456 -2.623793
[3,] -0.963895  0.30958 -0.257824
[4,] -1.689012 -0.52648  0.038425
[5,] -0.945770  0.49251  1.426751
[6,] -0.025327 -1.15869 -0.303727
```

Figure 12

```
> # Add 'output' attribute column to file
> # heart_mm[,output] <- output
> heart_mm2 <- cbind(heart_mm, output)
> head(heart_mm2)
        PC1      PC2       PC3 output
1  1.196019  0.63868 -2.903942      1
2  0.851346 -0.43456 -2.623793      1
3 -0.963895  0.30958 -0.257824      1
4 -1.689012 -0.52648  0.038425      1
5 -0.945770  0.49251  1.426751      1
6 -0.025327 -1.15869 -0.303727      1
>
```

Figure 13

III. **Principal Factor Analysis**

Output:

```
        Bartlett test of homogeneity of variances

data:  heart_PCA1
Bartlett's K-squared = 2612.4, df = 7, p-value < 2.2e-16


Kaiser-Meyer-Olkin factor adequacy
Call: psych::KMO(r = heart_PCA1)
Overall MSA =  0.64
MSA for each item =
      cp   trtbps     chol  restecg thalachh  oldpeak      slp      caa
    0.65     0.65     0.57     0.61     0.73     0.63     0.60     0.66
```

Figure 14: The Bartlett test shows that there is significant correlation in the variables. The KMO test reveals that the data is suited for factor analysis.

```
Principal Components Analysis
Call: psych::principal(r = heart_PCA1, nfactors = 3, rotate = "varimax",
    scores = TRUE)
Standardized loadings (pattern matrix) based upon correlation matrix
           RC1   RC3   RC2   h2   u2 com
cp        0.04  0.81  0.01 0.66 0.34 1.0
trtbps   -0.36  0.30  0.57 0.54 0.46 2.3
chol      0.12 -0.19  0.69 0.53 0.47 1.2
restecg   0.03  0.07 -0.63 0.40 0.60 1.0
thalachh  0.54  0.52  0.08 0.57 0.43 2.0
oldpeak  -0.82 -0.13  0.10 0.70 0.30 1.1
slp       0.85  0.07  0.01 0.72 0.28 1.0
caa      -0.14 -0.53  0.25 0.37 0.63 1.6

                       RC1  RC3  RC2
SS loadings           1.85 1.36 1.28
Proportion Var        0.23 0.17 0.16
Cumulative Var        0.23 0.40 0.56
Proportion Explained  0.41 0.30 0.28
Cumulative Proportion 0.41 0.72 1.00

Mean item complexity =  1.4
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is  0.13
 with the empirical chi square  296.22  with prob <  3.9e-60

Fit based upon off diagonal values = 0.52
```

Figure 15: Principal Component Analysis was performed on the “transformed” dataset. Three components are sufficient for hypothesis testing. The Chi square test p =3.9e-60 < alpha=0.05 therefore we fail to reject the null hypothesis Ho: therefore, the loadings are correlated.

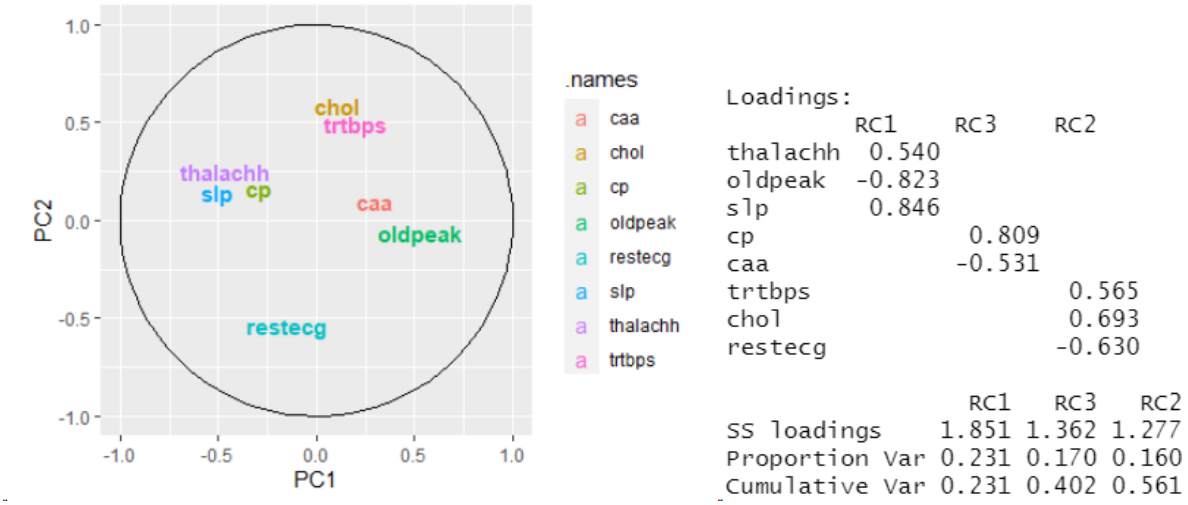


Figure 16: Plot of the contributions of the dataset attributes to the three principal components. The original variables for 4 distinct groupings after VARIMAX rotation are shown. The table below shows loadings of the dataset. Adjustment of the cutoff parameter enabled the cleanup of the loading which provides clear separation of the dataset variable groups.

```
> summary(p2)

Factor analysis with Call: psych::principal(r = heart_PCA1, nfactors = 3, rotate = "varimax",
    scores = TRUE)

Test of the hypothesis that 3 factors are sufficient.
The degrees of freedom for the model is 7  and the objective function was  0.78
The number of observations was  303  with Chi Square =  232.65  with prob <  1.4e-46

The root mean square of the residuals (RMSA) is  0.13
```

Figure 17: The summary of the Factor Analysis provides key details about the RMSE as a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

IV. Confirmatory Factor Analysis

Output:

```
lavaan 0.6-8 ended normally after 109 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        19

  Number of observations                           303

Model Test User Model:

  Test statistic                                42.489
  Degrees of freedom                                17
  P-value (Chi-square)                           0.001

Model Test Baseline Model:

  Test statistic                               274.616
  Degrees of freedom                                28
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.897
  Tucker-Lewis Index (TLI)                       0.830

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)                -766.591
  Loglikelihood unrestricted model (H1)        -745.346

  Akaike (AIC)                                 1571.182
  Bayesian (BIC)                               1641.743
  Sample-size adjusted Bayesian (BIC)          1581.485

Root Mean Square Error of Approximation:

  RMSEA                                          0.070
  90 Percent confidence interval - lower         0.044
  90 Percent confidence interval - upper         0.097
  P-value RMSEA <= 0.05                          0.095
```

```
Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Good_Health =~
    thalachh          1.000
    oldpeak          -1.644    0.236   -6.954    0.000
    slp               5.186    0.738    7.030    0.000
  Poor_Health =~
    trtbps            1.000
    chol              0.585    0.279    2.095    0.036
    restecg          -2.769    1.312   -2.111    0.035
  Borderline_Health =~
    cp                1.000
    caa              -1.153    0.420   -2.745    0.006

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  Good_Health ~~
    Poor_Health      -0.002    0.001   -2.215    0.027
    Borderlin_Hlth    0.019    0.006    2.981    0.003
  Poor_Health ~~
    Borderlin_Hlth   -0.009    0.006   -1.606    0.108

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .thalachh          0.023    0.002   10.913    0.000
   .oldpeak           0.014    0.003    5.472    0.000
   .slp               0.173    0.027    6.347    0.000
   .trtbps            0.023    0.003    7.879    0.000
   .chol              0.012    0.001    9.537    0.000
   .restecg           0.243    0.027    9.089    0.000
   .cp                0.896    0.100    8.996    0.000
   .caa               0.823    0.112    7.317    0.000
    Good_Health       0.008    0.002    4.015    0.000
    Poor_Health       0.004    0.003    1.700    0.089
    Borderlin_Hlth    0.165    0.082    2.009    0.045
```

Figure 18: The table above shows the summary of the Confirmatory Factor Analysis (CFA). To create this analysis, The loading from PFA were examined to create labels for the variable grouping column of factors. A label was assigned to each RC group that describes what connects the variables. The variables grouped with RC1 were labeled “Good_Health”, RC2 (“Borderline_Health”) and RC3 (Poor_Health). The variables grouped with each RC group are associated with these labels. The right table shows details on how these groups perform under the CFA model. The Chi-square for the “Model User Test Model” is < alpha=0.05; therefore can reject the null hypothesis, Ho. The “Model User Test Model '' is a good representative of the data and the data has a RMSE of 0.70 indicating the data is concentrated around the line of best fit.

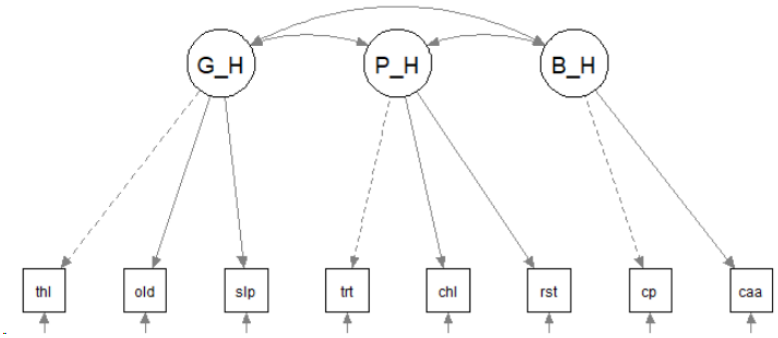


Figure 19: Confirmatory Factor Analysis plots the relationship between the original variables and their label groupings. The primary factors associated with poor health (risk of heart attack/myocardial infarction are 'cholesterol'(chl) and 'resting electrocardiographic results'(rst) and the secondary factor is 'resting blood pressure' (trtbps)

V. Logistic Regression

Code:

```
data = read.csv("heart_data_final.csv")
summary(data)
library(corrplot)
library(gmodels)
corrplot(cor(data, method = "spearman"))
data2 <- data[, c(2,4,6,12,15,16,17,25,26)]
data3 <- filter(data2, HEARTRTE > 0)
summary(data3)

#reg_dataset4 <- data3[, c(2,4,5,6,8,9,12,13,15,16,17,25,26)]
reg_dataset4 <- data

##### Model
table(reg_dataset4$output)
logistic <- glm(output ~ ., data = reg_dataset4, family = "binomial")
summary(logistic)
summary(reg_dataset4)
#Coefficients in exponential form
install.packages("dplyr")
library(dplyr)
logistic %>%
 gtsummary::tbl_regression(exp = TRUE)

#### Machine Learning
install.packages("caret")
install.packages("vip")

#Libraries
library(dplyr)   # data wrangling
library(ggplot2)   # plotting
library(rsample)   # training and testing splitting
library(caret)   # for logistic regression modeling and prediction outputs
library(vip)      # variable importance

# Create training (70%) and test (30%) sets
set.seed(123)  # use a set seed point for reproducibility
split <- initial_split(reg_dataset4, prop = .7, strata = "output")
train <- training(split)
test  <- testing(split)
summary(train)
```

```
#Logistic Regression
#For explaining dependent variable
reg_dataset4$output <- as.factor(reg_dataset4$output)

log_reg <- glm(
output ~ scores_1 + scores_2 + scores_3,
family = "binomial",
data = reg_dataset4
)
summary(log_reg) #Coefficients Not in exponential form
tidy(log_reg) #Coefficients Not in exponential form
#Coefficients in exponential form
log_reg %>%
 gtsummary::tbl_regression(exp = TRUE)
train$output<- as.factor(train$output)

#For Predicting dependent variable
log_reg = train(
 form = output ~ scores_1 + scores_2 + scores_3,
data = train,
method = "glm",
family = "binomial"
)

#Confusion Matrix
confusionMatrix(predict(log_reg, test), as.factor(test$output))

#Variables of Importance
vip(log_reg, num_features = 10)
/* Programming SAS
PROC IMPORT DATAFILE="C:\Users\RFABIAN1\Documents\SAS\PCA\heart_data_final.csv"
 OUT=project.base
 DBMS=csv
 REPLACE;
 GETNAMES=YES;
 RUN;
PROC CONTENTS DATA=PROJECT.base NODS;
RUN;
PROC PRINT DATA=project.base; RUN;

/* Logistic Model for PCA */
PROC LOGISTIC DATA=project.base DESCENDING;
MODEL output = scores_1 scores_2 scores_3 /
SELECTION = FORWARD
CTABLE PPROB=(0 to 1 by .1)
```

```
LACKFIT
RISKLIMITS;
RUN;
/* ROC Curves -  Determining Optimal Cutoff Point */
ODS GRAPHIC ON;
PROC LOGISTIC DATA = project.base;
 MODEL OUTPUT (EVENT='1')=scores_1 scores_2 scores_3/OUTROC=ROCDATA;
 ROC; ROCCONTRAST;
 RUN;
ODS GRAPHIC OFF;
```

Output:

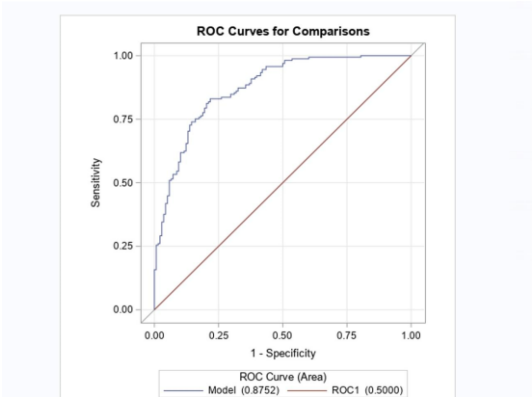


Figure 20: After the PCA analysis, The ROC curve of the Logistic Model is 0.87. This value is much higher than the previous models without PCA.

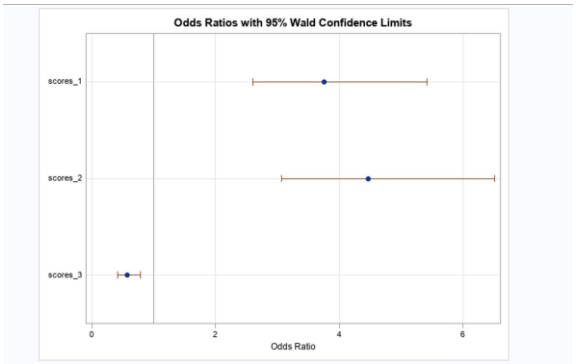


Figure 21: The Odds Ratio show that the 3 factors are significant

Characteristic	OR[1]	95% CI[1]	p-value
scores_1	3.75	2.65, 5.52	<0.001
scores_2	4.47	3.12, 6.65	<0.001
scores_3	0.57	0.41, 0.78	<0.001

[1] OR = Odds Ratio, CI = Confidence Interval

Figure 22: The odds ratio of scores 1 and score 2 are above 1. It means that these factors increase the risk of heart illness while the score 3 is negative. It means the score 3 reduces the risk of getting heart illness.

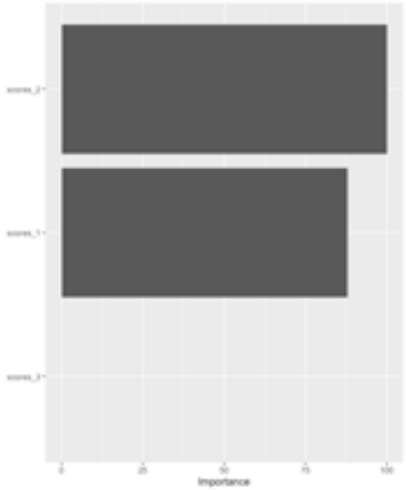


Figure 23: The score 2 is the variable more important, then the score 1 and the score 3 is less important.

```
          Reference
Prediction  0  1
         0 31 11
         1 10 38

               Accuracy : 0.7667
                 95% CI : (0.6657, 0.8494)
    No Information Rate : 0.5444
    P-Value [Acc > NIR] : 1.061e-05

                  Kappa : 0.5306

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7561
            Specificity : 0.7755
         Pos Pred Value : 0.7381
         Neg Pred Value : 0.7917
             Prevalence : 0.4556
         Detection Rate : 0.3444
   Detection Prevalence : 0.4667
      Balanced Accuracy : 0.7658

       'Positive' Class : 0
```

Figure 24: Confusion Matrix and Sensitivity Analysis of Logistic Model

VI. K-Nearest Neighbors

Code:

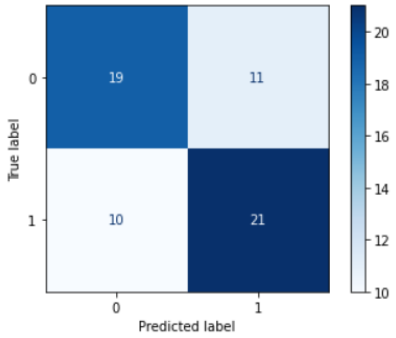
```
### Imports
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

from sklearn import neighbors
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, auc, roc_curve,
plot_confusion_matrix

#ignore warnings
import warnings
warnings.filterwarnings('ignore')

### Load the Data
heart = pd.read_csv('heart_data_Final.csv')

heart

heartY = heart['output']
heartX = heart.drop('output', axis = 1)

print(heartX.shape)
print(heartY.shape)

heartX.head()
heartY.head()

trainX, testX, trainY, testY = train_test_split(heartX, heartY, test_size = 0.2, random_state = 123)

#Look at the shapes of each dataset
print('TrainX is: ', trainX.shape)
print('TrainY is: ', trainY.shape)
print('TestX is: ', testX.shape)
print('TestY is: ', testY.shape)

### Change the Data into Numpy Arrays
trainX_np = np.array(trainX)
trainY_np = np.array(trainY)
testX_np = np.array(testX)
testY_np = np.array(testY)

### Running KNN with Grid Search Method
```

```
# Set Options for KNN
grid_params = {"n_neighbors": [3, 5, 7, 9], "weights": ['uniform', 'distance']}

# Run Iterations for KNN
gsKNN = GridSearchCV(neighbors.KNeighborsClassifier(), grid_params, cv = 10)
gsKNN_results = gsKNN.fit(trainX, trainY)

# Best Parameters
gsKNN_results.best_params_

### Evaluate Model with Best Parameters and Fit to Training Data
knn = neighbors.KNeighborsClassifier(n_neighbors = 9, weights = 'distance')
knn.fit(trainX_np, trainY_np)

## Prediction on Testing Data
knn_pred = knn.predict(testX_np)

### Metrics
def metrics(x, y, p):
   'returns the accuracy, sensitivity, specificity, positive predictive value, negative predictive
value, and auc'

   #accuracy
   a = accuracy_score(y, p)

   #confusion matrix
   tn, fp, fn, tp = confusion_matrix(y, p).ravel()

   sens = (tp/(tp+fn)) #sensitivity
   spec = (tn/(tn+fp)) #specificity
   ppv = (tp/(tp+fp)) #positive predictive value
   npv = (tn/(tn+fn)) #negative predictive value

   #auc
   fpr, tpr, thresholds = roc_curve(y, p)
   area = auc(fpr, tpr)

   print('Metrics')
   print('accuracy score: ', a)
   print('sensitivity: ', sens)
   print('specificity: ', spec)
   print('positive predictive value: ', ppv)
   print('negative predictive value: ', npv)
   print('area under curve: ', area)


metrics(testX_np, testY_np, knn_pred)
```

```
## Average Accuracy Scores For both Test and Training Data
print('Accuracy Score for Training: ', knn.score(trainX_np, trainY_np))
print('Accuracy Score for Testing ', knn.score(testX_np, testY_np))

plot_confusion_matrix(knn, testX_np, testY_np, cmap=plt.cm.Blues)

### Load the Data
heart = pd.read_csv('heart_data_Final.csv')

heart

heartY = heart['output']
heartX = heart.drop('output', axis = 1)

print(heartX.shape)
print(heartY.shape)

heartX.head()
heartY.head()

trainX, testX, trainY, testY = train_test_split(heartX, heartY, test_size = 0.2, random_state = 123)

#Look at the shapes of each dataset
print('TrainX is: ', trainX.shape)
print('TrainY is: ', trainY.shape)
print('TestX is: ', testX.shape)
print('TestY is: ', testY.shape)

### Change the Data into Numpy Arrays
trainX_np = np.array(trainX)
trainY_np = np.array(trainY)
testX_np = np.array(testX)
testY_np = np.array(testY)

### Running KNN with Grid Search Method

# Set Options for KNN
grid_params = {"n_neighbors": [3, 5, 7, 9], "weights": ['uniform', 'distance']}

# Run Iterations for KNN
gsKNN = GridSearchCV(neighbors.KNeighborsClassifier(), grid_params, cv = 10)
gsKNN_results = gsKNN.fit(trainX, trainY)

# Best Parameters
gsKNN_results.best_params_

### Evaluate Model with Best Parameters and Fit to Training Data
```

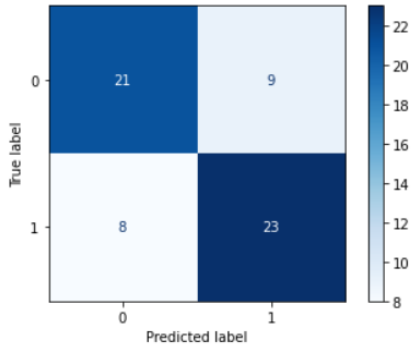
```
knn = neighbors.KNeighborsClassifier(n_neighbors = 9, weights = 'distance')
knn.fit(trainX_np, trainY_np)

## Prediction on Testing Data
knn_pred = knn.predict(testX_np)

### Metrics
def metrics(x, y, p):
   'returns the accuracy, sensitivity, specificity, positive predictive value, negative predictive
value, and auc'

   #accuracy
   a = accuracy_score(y, p)

   #confusion matrix
   tn, fp, fn, tp = confusion_matrix(y, p).ravel()

   sens = (tp/(tp+fn)) #sensitivity
   spec = (tn/(tn+fp)) #specificity
   ppv = (tp/(tp+fp)) #positive predictive value
   npv = (tn/(tn+fn)) #negative predictive value

   #auc
   fpr, tpr, thresholds = roc_curve(y, p)
   area = auc(fpr, tpr)

   print('Metrics')
   print('accuracy score: ', a)
   print('sensitivity: ', sens)
   print('specificity: ', spec)
   print('positive predictive value: ', ppv)
   print('negative predictive value: ', npv)
   print('area under curve: ', area)


metrics(testX_np, testY_np, knn_pred)

## Average Accuracy Scores For both Test and Training Data
print('Accuracy Score for Training: ', knn.score(trainX_np, trainY_np))
print('Accuracy Score for Testing ', knn.score(testX_np, testY_np))

plot_confusion_matrix(knn, testX_np, testY_np, cmap=plt.cm.Blues)
```

Outputs:



Metrics
accuracy score: 0.6557377049180327
sensitivity: 0.6774193548387096
specificity: 0.6333333333333333
positive predictive value: 0.65625
negative predictive value: 0.6551724137931034
area under curve: 0.6553763440860214

Figure 25: The figure on the left is the confusion matrix performing K Nearest Neighbors on the normalized data. On the right are the corresponding metrics from the model.



Metrics
accuracy score: 0.7213114754098361
sensitivity: 0.7419354838709677
specificity: 0.7
positive predictive value: 0.71875
negative predictive value: 0.7241379310344828
area under curve: 0.7209677419354839

Figure 26: The figure on the left is the confusion matrix performing K Nearest Neighbors on the principal components data. On the right are the corresponding metrics from the model.

VII. Random Forest

Code:

```
library(Hmisc) #Describe Function
library(psych) #Multiple Functions for Statistics and Multivariate Analysis
library(GGally) #ggpairs Function
library(ggplot2) #ggplot2 Functions
library(vioplot) #Violin Plot Function
library(corrplot) #Plot Correlations
library(REdaS) #Bartlett's Test of Sphericity
library(psych) #PCA/FA functions
library(factoextra) #PCA Visualizations
library("FactoMineR") #PCA functions
library(ade4) #PCA Visualizations

#Set Working Directory
setwd('C:/Users/PBS/Desktop/CSC 510')
dataset <- read.csv(file="projectData.csv", header=TRUE, sep=",")
dim(dataset)
sum(is.na(dataset))

# NO missing values hence no need for preprocessing.
head(dataset)
library(randomForest)

#trtbps + chol + thalachh + age + oldpeak + caa + sex + cp + fbs + restecg + exng + slp
randomForest <- randomForest(output~ scores_1 + scores_2 + scores_3,   data=dataset)
print(randomForest) # view results
importance(randomForest) # importance of each predictor
install.packages("vip")
library(vip)
vip(randomForest, num_features = 10)
```

Output:

```
> print(randomForest) # view results

Call:
 randomForest(formula = output ~ trtbps + chol + thalachh + age +      oldpeak + c
aa + sex + cp + fbs + restecg + exng + slp, data = dataset)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 0.13167
                    % Var explained: 46.91
```

Figure 27

```
> importance(randomForest) # importance of each predictor
         IncNodePurity
trtbps          5.4339
chol            5.4119
thalachh        8.6792
age             5.8694
oldpeak         8.6019
caa             9.4249
sex             2.9072
cp             11.8291
fbs             0.5116
restecg         1.0488
exng            4.8240
slp             3.9537
```

Figure 28

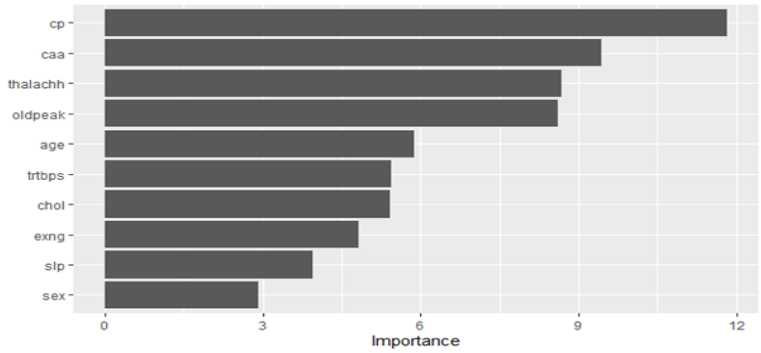


Figure 29

VIII. Gradient Boost

Output:

```
Gradient Boosting Classifier with Parameter Optimization (Train)
--------------------------------
Accuracy Score:          0.8263
Recall:                  0.8774
Specificity:             0.7619
Precision:               0.8230
Balanced Accuracy:       0.8196
F1 Score:                0.8493
```

```
Gradient Boosting Classifier with Default Parameters (Test)
--------------------------------
Accuracy Score:          0.7097
Recall:                  0.8000
Specificity:             0.6250
Precision:               0.6667
Balanced Accuracy:       0.7125
F1 Score:                0.7273
```

Figure 30: The two charts show the metrics obtained from the Gradient Boosting Classifier model. Hyperparameter optimization was performed through the GridSearch Cross Validation method. The best parameters used to test this model were 'learning_rate': 0.25, 'max_depth': 1 and 'n_estimators': 10. The metrics in the top table were obtained from the training dataset. The bottom table are the results from the test set.

Model (Test dataset)	Confusion Matrix (table) TP, FP/FN, TN	Accuracy(%)	Misclassification Accuracy(%)	Precision, % (TP/TP+FP)	Sensitivity, % (TP/TP+FN)	Specificity, % (TN/TN+FP)	Positive Predictive Value (%)	Negative Predictive Value (%)
Gradient Boosting	11, 5/ 2, 13	77.42	22.58	72.22	68.75	86.67	0.69	0.87
RBF Kernel SVM	9, 7/ 2, 13	70.97	29.03	65.00	86.67	56.25	0.56	0.87

Table 2: Model Properties - Confusion Matrices, Model Accuracy, and Misclassification Error

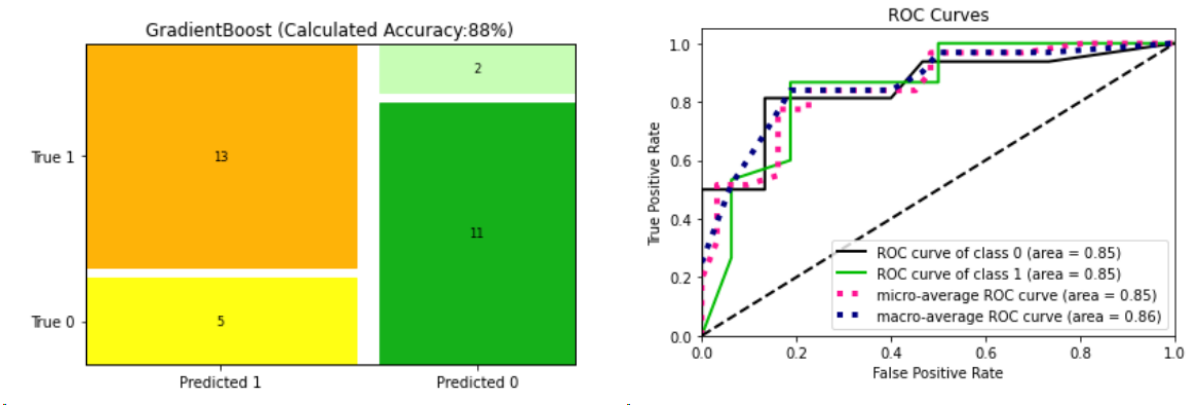


Figure 31: Left Chart is a mosaic Confusion Matrix. GB has 2 false negatives and 5 false positives.

IX. Radial Basis Function Kernel Support Vector Machine

Output:

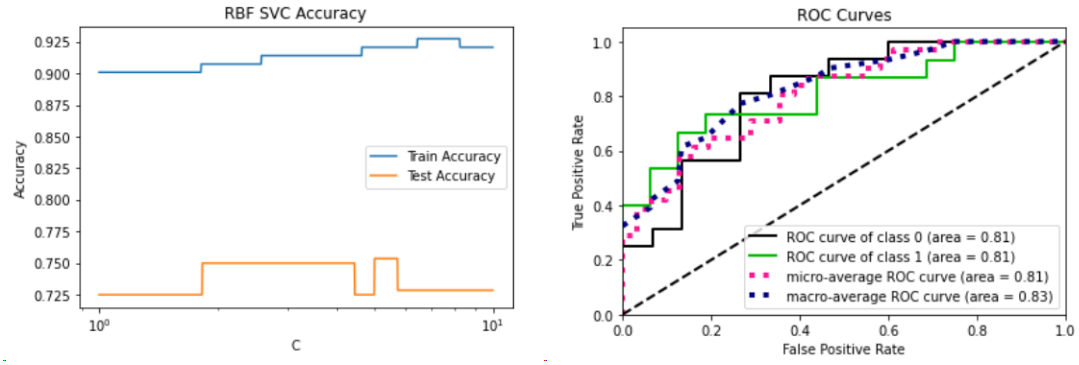


Figure 32: On the right is a plot of the 'C' parameter to test Accuracy on the RBF Kernel SVC model. On the left is a plot of the ROC curves for RBF Kernel SVC model

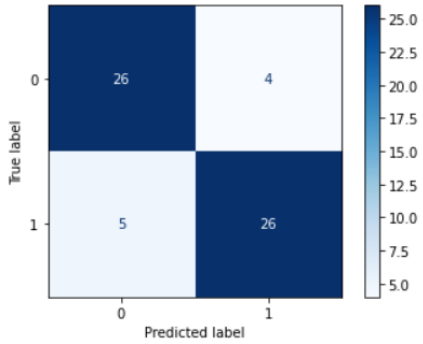
X. Naive Bayes

Code:

```
### Imports
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, auc, roc_curve,
plot_confusion_matrix

#ignore warnings
import warningsJJ
warnings.filterwarnings('ignore')

### Metrics
def metrics(x, y, p):
   'returns the accuracy, sensitivity, specificity, positive predictive value, negative predictive
value, and auc'

   #accuracy
   a = accuracy_score(y, p)

   #confusion matrix
   tn, fp, fn, tp = confusion_matrix(y, p).ravel()

   sens = (tp/(tp+fn)) #sensitivity
   spec = (tn/(tn+fp)) #specificity
   ppv = (tp/(tp+fp)) #positive predictive value
   npv = (tn/(tn+fn)) #negative predictive value

   #auc
   fpr, tpr, thresholds = roc_curve(y, p)
   area = auc(fpr, tpr)

   print('Metrics')
   print('accuracy score: ', round(a, 2))
   print('sensitivity: ', round(sens, 2))
   print('specificity: ', round(spec, 2))
   print('positive predictive value: ', round(ppv, 2))
   print('negative predictive value: ', round(npv, 2))
   print('area under curve: ', round(area, 2))
```

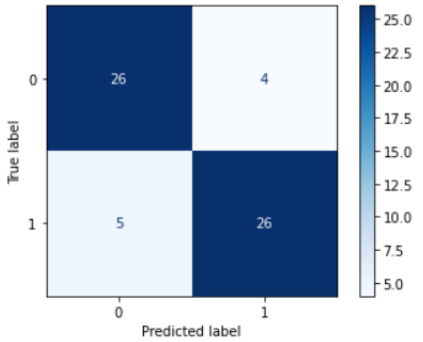
```
### Load the Data
heartPCA = pd.read_csv('heart_data_Final.csv')
heart = pd.read_csv('heart_Norm.csv')

heartPCA_y = heart['output']
heartPCA_x = heart.drop('output', axis = 1)

heart_y = heart['output']
heart_x = heart.drop('output', axis = 1)

trainPCA_x, testPCA_x, trainPCA_y, testPCA_y = train_test_split(heartPCA_x, heartPCA_y,
test_size = 0.2, random_state = 123)
train_x, test_x, train_y, test_y = train_test_split(heart_x, heart_y, test_size = 0.2, random_state
= 123)


#model
gnb = GaussianNB()

#Fit to data
y_pred = gnb.fit(train_x, train_y).predict(test_x)
yPCA_pred = gnb.fit(trainPCA_x, trainPCA_y).predict(testPCA_x)

#Normalizied

metrics(test_x, test_y, y_pred)

## Average Accuracy Scores For both Test and Training Data
print('Accuracy Score for Training: ', gnb.score(train_x, train_y))
print('Accuracy Score for Testing ', gnb.score(test_x, test_y))

plot_confusion_matrix(gnb, test_x, test_y, cmap=plt.cm.Blues)

##PCA

metrics(testPCA_x, testPCA_y, yPCA_pred)

## Average Accuracy Scores For both Test and Training Data
print('Accuracy Score for Training: ', gnb.score(trainPCA_x, trainPCA_y))
print('Accuracy Score for Testing ', gnb.score(testPCA_x, testPCA_y))

plot_confusion_matrix(gnb, testPCA_x, testPCA_y, cmap=plt.cm.Blues)
```

Output:



Metrics
accuracy score: 0.85
sensitivity: 0.8387096774193549
specificity: 0.8666666666666667
positive predictive value: 0.8666666666666667
negative predictive value: 0.8387096774193549
area under curve: 0.8526881720430108

Figure 33: The figure on the left is the confusion matrix performing Naive Bayes on the normalized data. On the right are the corresponding metrics from the model.



Metrics
accuracy score: 0.85
sensitivity: 0.84
specificity: 0.87
positive predictive value: 0.87
negative predictive value: 0.84
area under curve: 0.85

Figure 34: The figure on the left is the confusion matrix performing Naive Bayes on the principal component data. On the right are the corresponding metrics from the model.

XI. Research Questions

1. Is there a relationship between 'sex' and 'fbs'(DIABETES)?

   Code:

```
/* Dependent variable: Diabetes*/
/* Data Type: Binary    NOTE: Anytime you deal with binary or categorical data, it is a non parametric test. */
/* Independent variable: sex*/
/* 'Not paired' data)no before and after */

/* Null Ho: There is no relationship between 'sex' and 'fbs'(DIABETES) */
/* Alternate Ha: There is a relationship between 'sex' and 'fbs'(DIABETES) */

ODS pdf file='C:\Users\KCALLOW1\OneDrive\Documents\My SAS Files\9.4\DSC 510 - Group Project\OSD PDF\heart_data_prob1a.pdf';
PROC FREQ DATA=HEART.heart_data;
        TABLES sex*fbs / CHISQ; /* Use the Chi-Squared test(CHISQ) if you expected counts >= 5, Use Fischer(exact)*/
RUN;                /* test if you expect  counts < 5 */
ODS pdf close;

/* Null Hypothesis, Ho: “There is not a relationship between gender and fbs(diabetes).*/
/* Alt. Hypothesis, Ha: There is a relationship between gender and fbs(diabetes). */
```

Output:

The FREQ Procedure

Frequency Percent Row Pct Col Pct	Table of sex by fbs		
	fbs		
sex	0	1	Total
0	83 27.57 87.37 32.30	12 3.99 12.63 27.27	95 31.56
1	174 57.81 84.47 67.70	32 10.63 15.53 72.73	206 68.44
Total	257 85.38	44 14.62	301 100.00

Table 3: Table of the frequency of ‘fbs’(Diabetes) to gender created in SAS

Statistics for Table of sex by fbs

Statistic	DF	Value	Prob
Chi-Square	1	0.4388	0.5077
Likelihood Ratio Chi-Square	1	0.4486	0.5030
Continuity Adj. Chi-Square	1	0.2371	0.6263
Mantel-Haenszel Chi-Square	1	0.4374	0.5084
Phi Coefficient		0.0382	
Contingency Coefficient		0.0382	
Cramer's V		0.0382	

Fisher's Exact Test	
Cell (1,1) Frequency (F)	83
Left-sided Pr <= F	0.7975
Right-sided Pr >= F	0.3176
Table Probability (P)	0.1152
Two-sided Pr <= P	0.5999

Sample Size = 301

Table 4: SAS table of the statistics showing the probability of a relationship between gender and 'fbs'

Results:
/* Chi-Square: p=0.5077 > alpha=0.05 therefore cannot reject the null hypothesis Ho: There is no a relationship between 'sex' and 'fbs'(DIABETES)*/

2. Is there a relationship between 'sex'(GENDER) and 'output'(Heart Disease)?

Code:

/*The '*' in sex*output represents the interaction between the two variables. The '/' indicates any options or tests to be performed(ex. Chi-Squared test(CHISQ)The 'Mantel-Haenszel Chi-Square' table values gives a before and after view of the data set. */

/* Null Ho: There is no relationship between 'sex'(GENDER) and 'output'(Heart Disease)
Alternate : There is a relationship between 'sex'(GENDER) and 'output'(Heart Disease) */

```
ODS pdf file='C:\Users\KCALLOW1\OneDrive\Documents\My SAS Files\9.4\DSC 510 - Group
Project\OSD PDF\heart_data_prob1b.pdf';

PROC FREQ DATA=HEART.heart_data;
        TABLES sex*output / CHISQ;
RUN;
RUN;
ODS pdf close;
```

Output:

Statistics for Table of sex by output

Statistic	DF	Value	Prob
Chi-Square	1	22.9572	<.0001
Likelihood Ratio Chi-Square	1	23.8281	<.0001
Continuity Adj. Chi-Square	1	21.7794	<.0001
Mantel-Haenszel Chi-Square	1	22.8809	<.0001
Phi Coefficient		-0.2762	
Contingency Coefficient		0.2662	
Cramer's V		-0.2762	

Fisher's Exact Test	
Cell (1,1) Frequency (F)	24
Left-sided Pr <= F	<.0001
Right-sided Pr >= F	1.0000
Table Probability (P)	<.0001
Two-sided Pr <= P	<.0001

Sample Size = 301

Table 5: SAS table of the statistics showing the probability of a relationship between gender and 'output'(Heart Disease)

Results:

```
/* Chi-Square: p<.0001 < alpha=0.05 therefore can reject the null hypothesis Ho: */
/* There is no relationship between 'sex' and 'output'(Heart Disease).*/
```

3. Is there a relationship between 'chol'(Choleterol) and 'trtbps'(Resting BP)?

Code:

```
/*The '*' in chol*trtbps represents the interaction between the two variables. The '/' indicates
any options or tests to be performed(ex. Chi-Squared test(CHISQ)The 'Mantel-Haenszel Chi-
Square' table values gives a before and after view of the data set.

Null Ho: There is not relationship between 'chol'(Choleterol) and 'trtbps'(Resting BP)
Alternate Ha: There is a relationship between 'chol'(Choleterol) and 'trtbps'(Resting BP)*/

ODS pdf file='C:\Users\KCALLOW1\OneDrive\Documents\My SAS Files\9.4\DSC 510 - Group
Project\OSD PDF\heart_data_prob2a.pdf';
PROC FREQ DATA=HEART.heart_data;
        TABLES chol*trtbps / CHISQ;
RUN;
RUN;
ODS pdf close;
```

Output:

DSC 510 Group Project - Heart Attack Analysis and Prediction

The FREQ Procedure

Statistics for Table of chol by trtbps

Statistic	DF	Value	Prob
Chi-Square	7248	6900.5407	0.9983
Likelihood Ratio Chi-Square	7248	1512.3550	1.0000
Mantel-Haenszel Chi-Square	1	4.4972	0.0340
Phi Coefficient		4.7880	
Contingency Coefficient		0.9789	
Cramer's V		0.6911	

WARNING: 100% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 301

Table 6: SAS table of the statistics showing the relationship between Cholesterol and Resting Blood Pressure

Results:

```
/* Chi-Square: p=0.9983 > alpha=0.05 therefore cannot reject the null hypothesis Ho:
There is no relationship between 'chol'(Choleterol) and 'trtbps'(Resting BP).*/
```

4. Is there a difference between Gender and Resting Blood?

Code:

```
/* Research Questions for Final Project */

/* Set Working Directory */
LIBNAME LAB 'C:\Users\CSTEFFEY\OneDrive - DePaul University\510';

/* View what is in the library */
PROC CONTENTS DATA=lab._ALL_ NODS;
RUN;

/* Read in Dataset */
PROC IMPORT DATAFILE="C:\Users\CSTEFFEY\OneDrive - DePaul
University\510\heartNonCat.csv"
   OUT=LAB.heart
   DBMS=csv
   REPLACE;
   GETNAMES=YES;
RUN;

/* Check that File was Read in Correctly */
PROC PRINT DATA=LAB.heart; RUN;


/******************/
/* Research Question 1: Is there a difference between Gender and Resting Blood Pressure? */
PROC SORT DATA=Lab.heart;
          BY sex;
RUN;

PROC UNIVARIATE DATA=LAB.heart NORMAL PLOT CIPCTLDF;
          BY sex;
          VAR trtbps;
          HISTOGRAM trtbps / NORMAL;
          QQPLOT / NORMAL (MU=est SIGMA=est);
RUN;

/* Create labeled boxplot */
PROC SGPLOT DATA=LAB.heart;
          TITLE "Boxplots of Resting Blood Pressure by Gender";
          VBOX trtbps / Category=sex;
RUN;

/* What do the normality tests tell us? */
/* The gender variable is not normal so the appropriate test would be the wilcoxin U */
/* Mann-Whitney U (Wilcoxon) test - Nonparametric T-Test */
PROC NPAR1WAY DATA=LAB.heart WILCOXON;
          CLASS sex;
          VAR trtbps;
RUN;
```

Output:

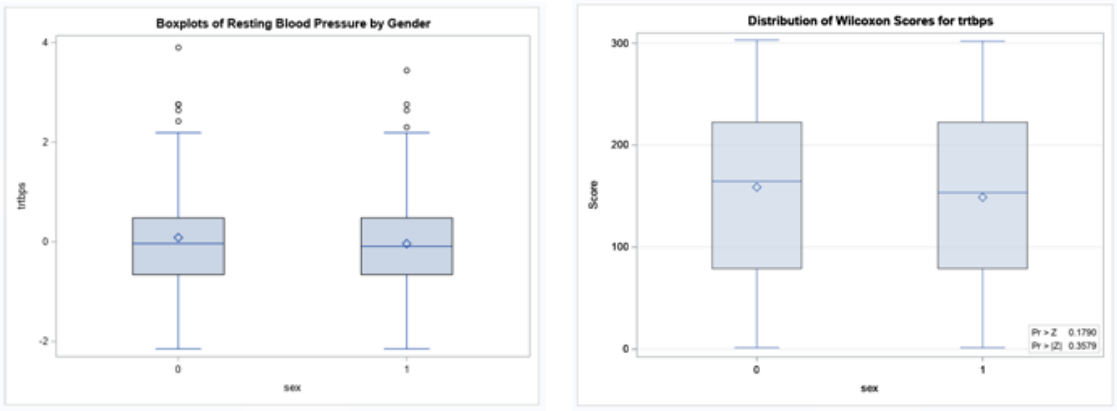


Figure 35: The figure on the left is the boxplot of the resting blood pressure by gender and the figure on the right is the boxplots for the Wilcoxon scores.

Boxplots of Resting Blood Pressure by Gender

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable trtbps Classified by Variable sex

sex	N	Sum of Scores	Expected Under H0	Std Dev Under H0	Mean Score
0	96	15243.0	14592.0	707.593434	158.781250
1	207	30813.0	31464.0	707.593434	148.855072

Average scores were used for ties.

Wilcoxon Two-Sample Test

Statistic	Z	Pr > Z	Pr > |Z|	t Approximation Pr > Z	t Approximation Pr > |Z|
15243.00	0.9193	0.1790	0.3579	0.1793	0.3587

Z includes a continuity correction of 0.5.

Kruskal-Wallis Test

Chi-Square	DF	Pr > ChiSq
0.8464	1	0.3576

Table 7: SAS table of the NPAR1WAY results showing the probability of there being a difference between Resting Blood Pressure and Gender

Results:

/* The Wilcoxon test shows probability, Pr > |Z| = 0.3587 > alpha=0.05; therefore, cannot reject the null hypothesis, Ho. Thus, there is no relationship between gender and resting blood pressure. */

5. Is there a difference between Exercise Induced Angina and Resting Blood Pressure?

Code:

```
/* Research Question 2: Is there a difference between Exercise Induced Angina and Resting
Blood Pressure? */
PROC SORT DATA=Lab.heart;
	BY exng;
RUN;

PROC UNIVARIATE DATA=LAB.heart NORMAL PLOT CIPCTLDF;
	BY exng;
	VAR trtbps;
	HISTOGRAM trtbps / NORMAL;
	QQPLOT / NORMAL (MU=est SIGMA=est);
RUN;

/* Create labeled boxplot */
PROC SGPLOT DATA=LAB.heart;
	TITLE "Boxplots of Resting Blood Pressure by Exercise Induced Angina";
	VBOX trtbps / Category=exng;
RUN;
/* What do normality tests tell us? */
/* These are not normal so we use the wilcoxin U test */


/* Mann-Whitney U (Wilcoxon) test - Nonparametric T-Test */

PROC NPAR1WAY DATA=LAB.heart WILCOXON;
	CLASS exng;
	VAR trtbps;
RUN;
```

Output:

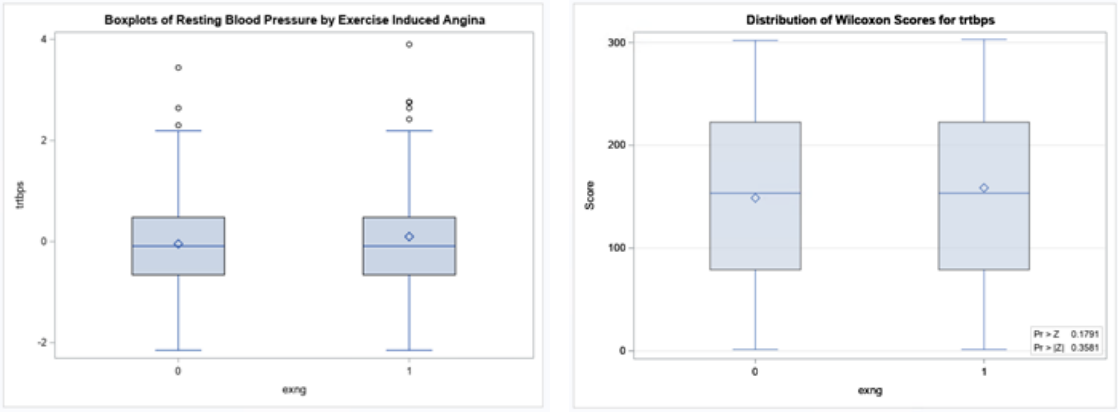


Figure 36: The figure on the left is the boxplot of the Resting Blood Pressure by Exercise Induced Angina and the figure on the right is the boxplots for the Wilcoxon scores.

Boxplots of Resting Blood Pressure by Exercise Induced Angina

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable trtbps Classified by Variable exng

exng	N	Sum of Scores	Expected Under H0	Std Dev Under H0	Mean Score
0	204	30352.0	31008.0	713.338536	148.784314
1	99	15704.0	15048.0	713.338536	158.626263

Average scores were used for ties.

Wilcoxon Two-Sample Test

				t Approximation	
Statistic	Z	Pr > Z	Pr > |Z|	Pr > Z	Pr > |Z|
15704.00	0.9189	0.1791	0.3581	0.1794	0.3589

Z includes a continuity correction of 0.5.

Kruskal-Wallis Test

Chi-Square	DF	Pr > ChiSq
0.8457	1	0.3578

Table 8: SAS table of the NPAR1WAY results showing the relationship between Resting Blood Pressure and Exercise Induced Angina

Results:

/* The Wilcoxon test shows probability, Pr > |Z| = 0.3589 > alpha=0.05; therefore, cannot reject the null hypothesis, Ho. Thus, there is no relationship between gender and resting blood pressure. */

6. Is there a relationship between 'chol'(Cholesterol) and 'restecg'(resting electrocardiographic results)?
   Null Ho: There is no relationship between 'chol'(Cholesterol) and 'restecg'(resting electrocardiographic results)

Alternate Ha: There is a relationship between 'chol'(Cholesterol) and 'restecg'(resting electrocardiographic results)

Code:

```
ODS pdf file='C:\Users\KCALLOW1\OneDrive\Documents\My SAS Files\9.4\DSC 510 - Group Project\OSD PDF\heart_data_prob3.pdf';

PROC FREQ DATA=HEART.heart_data;
        TABLES chol*restecg / CHISQ;
RUN;

RUN;
ODS pdf close;
```

Output:

The FREQ Procedure

Statistics for Table of chol by restecg

Statistic	DF	Value	Prob
Chi-Square	302	326.5487	0.1587
Likelihood Ratio Chi-Square	302	254.2367	0.9788
Mantel-Haenszel Chi-Square	1	6.8631	0.0088
Phi Coefficient		1.0416	
Contingency Coefficient		0.7214	
Cramer's V		0.7365	

WARNING: 100% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 301

Table 9: SAS table of FREQ results showing the relationship between Cholesterol and Resting Electrocardiographic Results

Results:

Chi-Square: p=0.0.1587 > alpha=0.05 therefore cannot reject the null hypothesis, Ho.
There is no relationship between 'chol'(Cholesterol) and 'restecg'(resting electrocardiographic results).