STA6543 Project
March 26, 2020

The Predictability of Cardiovascular Disease in Patients

Allison Beatty(yxn127),
Joshua Ching (mrf145),
Ruyi Liang (hua075),
Ian Scarff (iie728)

Abstract

There are many risk factors, other than cholesterol and blood pressure, that play an important role in determining whether or not a patient is developing cardiovascular disease. The dataset contains information describing different aspects of patient characteristics, collected at the time of medical examination. The goal of the study is to build a good model for predicting whether a patient has cardiovascular disease, with a high degree of predictive accuracy. The dataset contains 70,000 observations and 12 variables (5 continuous and 6 categorical) describing characteristics for each patient.

Background

Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels such as: heart attack, stroke, heart failure, arrhythmia, and other heart diseases. CVDs are the leading cause of death and a major cause of disability worldwide. "An estimated 17.9 million people died from CVDs in 2016, representing 31% of all global deaths. Of these deaths, 85% are due to heart attack and stroke."[5] Cardiovascular diseases are also the leading cause of death for men, women, and people of most racial and ethnic groups in the United States. "More than 859,000 Americans die of heart disease, stroke, or other cardiovascular diseases every year—that's one-third of all US deaths."[4]

In terms of the national economy, cardiovascular diseases take a financial toll, "costing $213.8 billion a year to the healthcare system and causing $137.4 billion in lost productivity from premature death alone."[4] The U.S. government, primarily the Centers for Disease Control and Prevention (CDC), have responded to such national health concerns by initiating various health studies, branches, and prevention programs in the support of public health efforts that address cardiovascular diseases. For example, "the CDC's Division for Heart Disease and Stroke Prevention (DHDSP) works with partners across government, public health, health care, and private sectors to improve prevention, detection, and control of heart disease and stroke risk factors, with a focus on high blood pressure and high cholesterol."[4]

"The aging population, obesity epidemic, underuse of prevention strategies, and suboptimal control of risk factors could exacerbate the future CVD burden."[2] With such a pressing national health issue that is ongoing today, it is not surprising that business will arise in predicting the significant risk factors associated with those diseases, for the needs in further understanding of the underlying relationship. This would assist physicians in the early detection and management of cardiovascular diseases.

Variable Introduction and Definitions

This dataset contains 70,000 records of patient data and contains 11 predictive features, plus the target variable "Disease," with no missing values. The dataset can be found at https://www.kaggle.com/sulianova/cardiovascular-disease-dataset. Below is a list of the variables as found in the dataset with their descriptions:

Variable Name	Description	Data Type - Unit/Levels
Age	Patient's age.	Continuous - # of days.
Height	Patient's height.	Continuous - centimeters.
Weight	Patient's weight.	Continuous - kilograms.
Gender	Patient's gender.	2 level Categorical: 1 = woman, 2 = male.
Systolic	Patient's systolic blood pressure.	Continuous - millimeters of mercury (mmHg).
Diastolic	Patient's diastolic blood pressure.	Continuous - millimeters of mercury (mmHg).
Cholesterol	Patient's cholesterol level.	3 level Categorical: 1 = normal, 2 = above normal, 3 = well above normal.
Glucose	Patient's glucose level.	3 level Categorical: 1 = normal, 2 = above normal, 3 = well above normal.
Smoking	Indicator for whether a patient smokes.	2 level Categorical: 0 = No, 1 = Yes.
Alcohol	Indicator for whether a patient consumes alcohol.	2 level Categorical: 0 = No, 1 = Yes.
Physical	Indicator for whether a patient performs physical activity.	2 level Categorical: 0 = No, 1 = Yes.
Disease	Indicator for whether a patient has a CVD.	2 level Categorical: 0 = No, 1 = Yes.

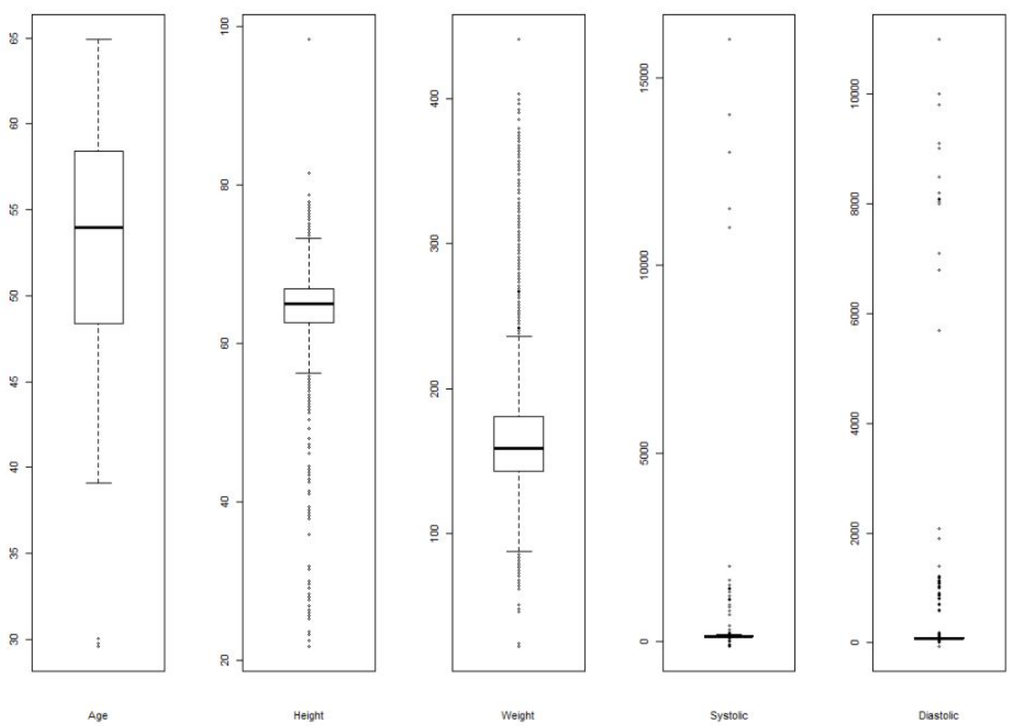
The following will analyze the relationship between the predictor variables and the response variable, “Disease”. The data will be preprocessed, which includes tasks such as transformation of continuous variables, as well as removal of highly correlated predictors. Both linear models and nonlinear models will be used to see which models perform well with the training data as well as examine the predictability for the testing data.

Data Preprocessing, Exploration, and Transformations

Many statistical models require data to be preprocessed prior to modeling. Preprocessing the data not only helps us appropriately tune the model and improve the predictive performance of the model, but it also decreases the computational time and complexity of the model. The first step we took was to check our data for any missing values. We found that our dataset did not contain any missing values.

We then adjusted the units and levels used in the original data. For “Age,” we converted from the number of days to the number of years (accounting for leap years by dividing by 365.25). We next converted “Weight” and “Height” from metric to imperial. Lastly, we adjusted all categorical levels that do not start at zero to start at zero, so that we can establish a baseline.

The next step we took was to visualize the data to see if there were any other problems. First, we examined the continuous variables (Age, Height, Weight, Systolic, and Diastolic). Below are boxplots of the continuous variables. Looking at these plots, we noticed that there are many illogical/unnatural values.



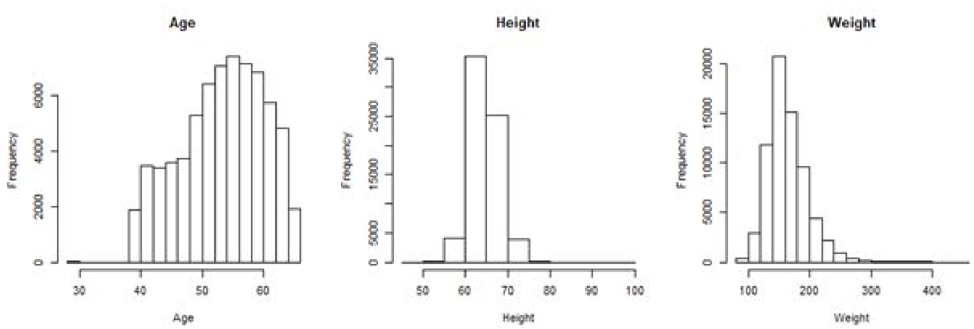
According to the figure displayed above, there are some blood pressures that are higher and lower than what is physically possible or that are not already a medical emergency. In addition, we found that there were some "Diastolic" values that were greater than "Systolic" values for a given observation. This is physically impossible. We can also see that "Weight" has some very low illogical values, given that the lowest value for "Age" is around 29. This same argument can also be applied to very below-average values for "Height."

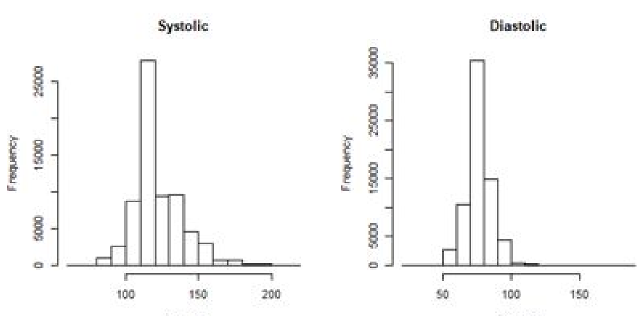
There were many assumptions we could have made to try to adjust these illogical values back to what is considered logical, but we decided to focus our study on observations that were already logical. To do this, we filtered the data based on the following restrictions:

1. Blood pressure can range from hypotension to hypertension. These include "Systolic" values ranging from 50 mmHg to 220 mmHg and "Diastolic" values ranging from 20 mmHg to 190 mmHg.
2. "Diastolic" values must be less than "Systolic" values.
3. The lowest "Weight" value is 80 pounds.
4. The lowest "Height" value is 48 inches.

Even with these restrictions, only approximately 2.05% of the original data was removed and 68,559 observations remain. This restricted data was then used for the remainder of the project.

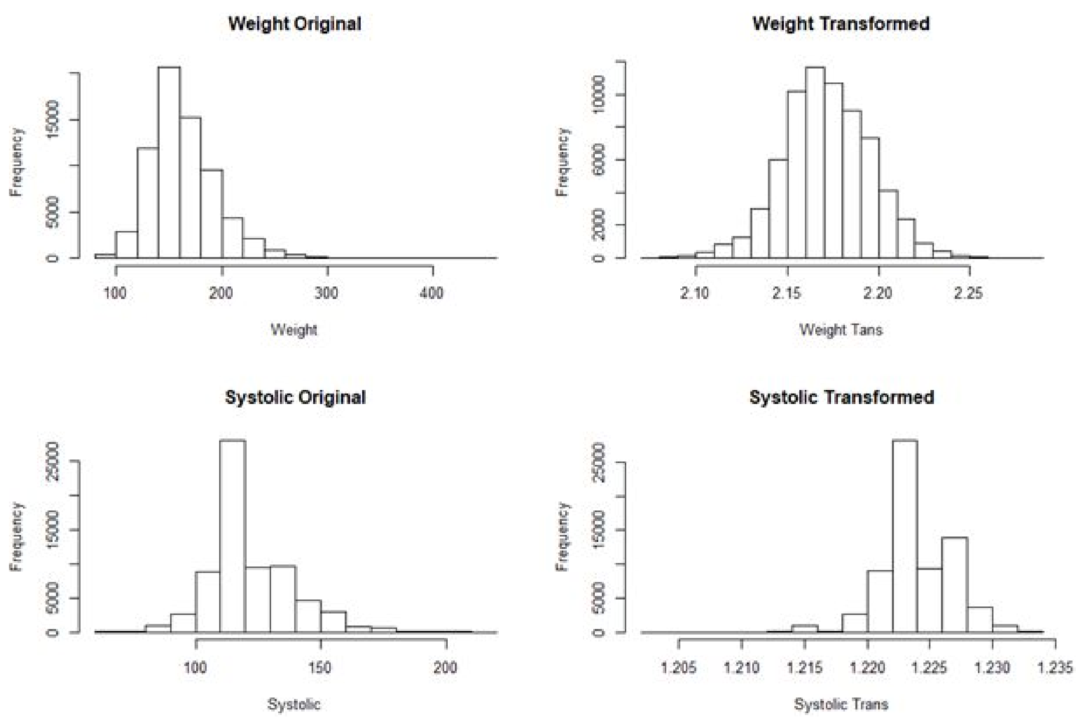
With this new data, we next examined the distributions of the continuous variables and possible transformations. The figures displayed below are histograms of each of the five continuous variables (Age, Height, Weight, Systolic, and Diastolic) and their measures of skewness.





Variable	Skewness
Age	-0.30564
Height	0.123961
Weight	1.001017
Systolic	0.874036
Diastolic	0.275779

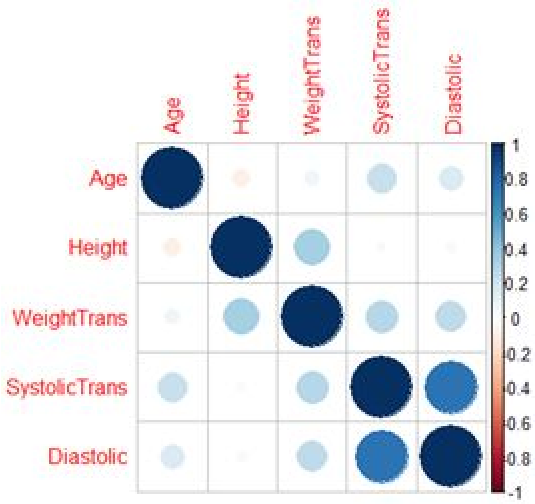
From these plots and tables, we can see that each variable has different levels of skewness. For this project, we considered predictors that have a skewness value between -0.5 and 0.5 to be nearly symmetric, moderately skewed if the absolute value is between 0.5 and 1, and heavily skewed otherwise. Based on this, we examined possible transformations for “Systolic” and “Weight.” Using the Box-Cox transformation method, optimal lambda values for “Systolic” and “Weight” were approximately -0.5 and -1, respectively. The plots displayed below compare the histograms between the original variables and the transformed variables. In addition, the table below compares the original skewness value to the new skewness value for each variable.

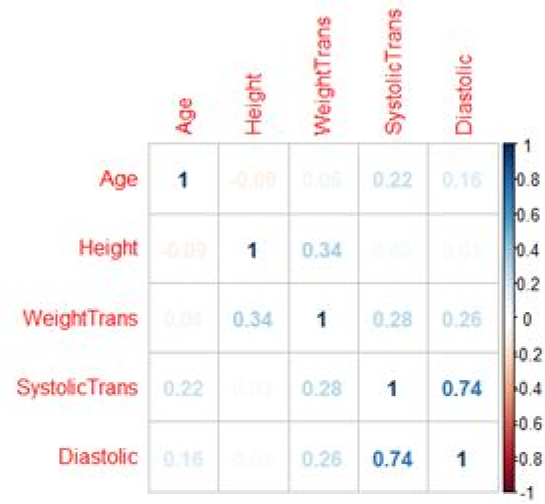


Variable	Weight	WeightTrans	Systolic	SystolicTans
Skewness	1.001017	-0.00207732	0.874036	-0.0429108

According to the histograms and table displayed above, we can see that these transformations significantly reduced the skewness in each of these variables. In addition to Box-Cox, we examined other data transformation methods, such as scaling, centering, and combinations of all three methods, but found that using Box-Cox alone provided the best skewness values.

Finally, we examined the correlation between continuous predictors. Below are the correlation plots of the continuous predictors. For the correlation matrix plot: dark blue colors indicate strong positive correlations, dark red is used for strong negative correlations, and white implies no empirical relationship between the predictors.

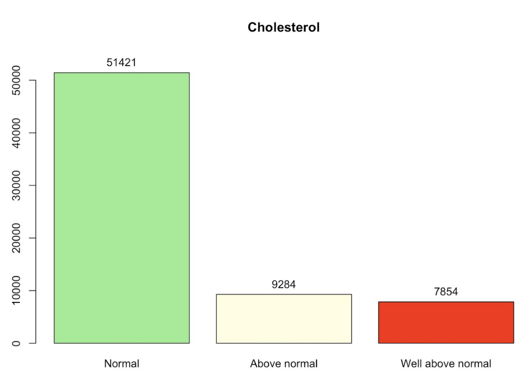


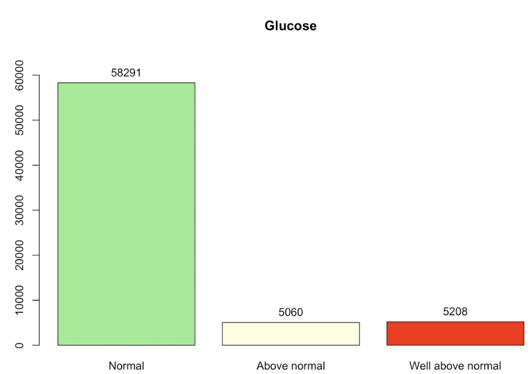


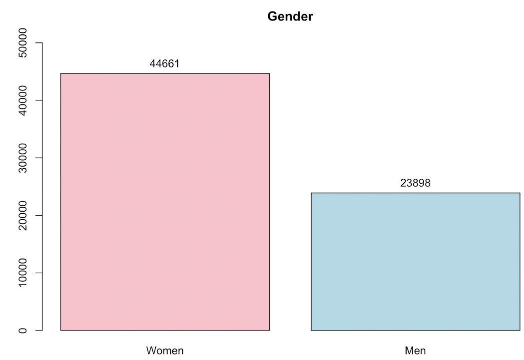
According to the correlation plots displayed above, we see that only "Diastolic" and the transformed "Systolic" variable have a high correlation. This is due to both measures being used to measure blood pressure. To resolve this, we decided to remove "Diastolic" from the analysis. This ensures no multicollinearity and in reality, systolic blood pressure is more important than diastolic blood pressure.

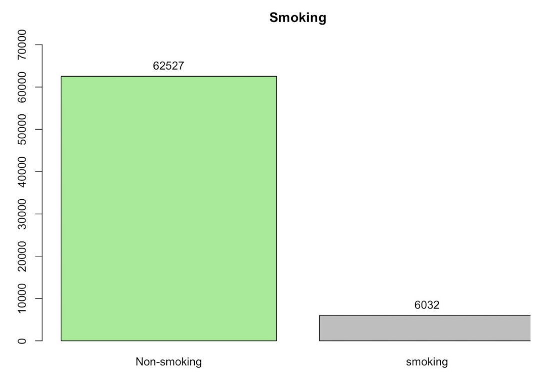
Data Splitting

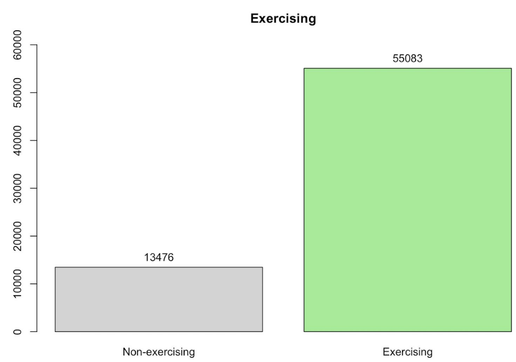
Prior to model building, we first needed to determine if the dataset was balanced or unbalanced. When examining the bar charts for the categorical variables (Cholesterol, Glucose, Gender, Smoking, Exercising, Alcohol, and Disease) below, we found that the target variable, "Disease," is pretty balanced.

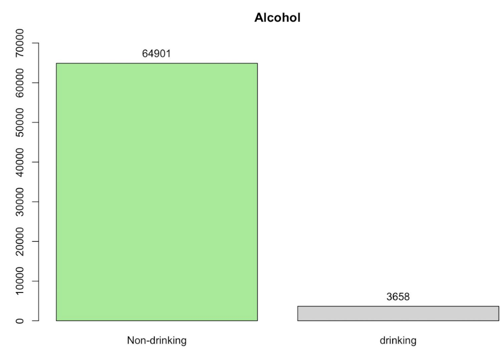


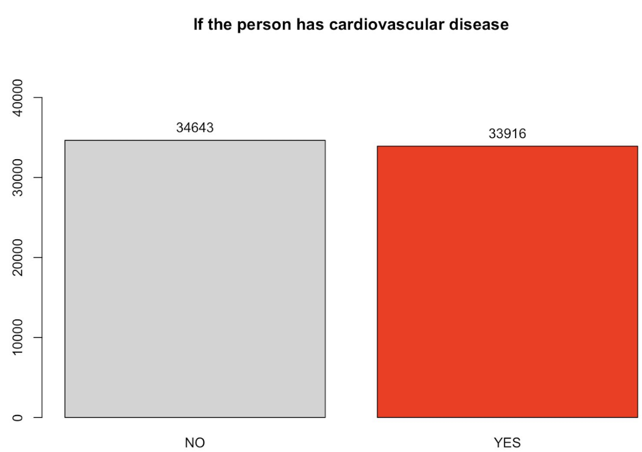












Based on this, we decided to use the createDataPartition function to split the data into training and testing datasets. The training dataset is often used to build the models while the testing dataset is used solely for validating the performance of the final models. We adopted a simple random sampling technique to split the data into 80% training and 20% in the testing dataset.

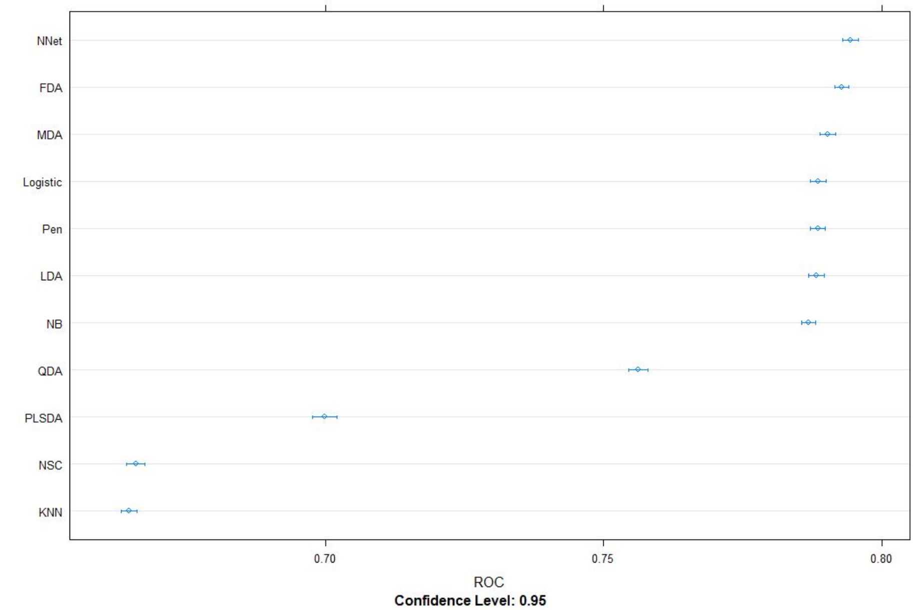
Model Building

In this section, we will discuss several models that include both linear and nonlinear classification methods in order to determine the best model in terms of the best predictive ability for the testing data. For this analysis, the following models were built: Logistic Regression, Linear Discriminant Analysis (LDA), Partial Least Squares Discriminant Analysis (PLSDA), Penalized models, Nearest Shrunken Centroids (NSC), Quadratic Discriminant Analysis (QDA), Mixture Discriminant Analysis (MDA), Flexible Discriminant Analysis (FDA), Naive Bayes, K-Nearest Neighbors (KNN), and Neural Network. We were unable to build any Support Vector Machine models for this data due to the following: computation time and computation errors. To determine the best model for predicting CVDs with our predictors, we compared each model using their test ROC curves, the area under these curves, and their test error rates. All models were built using the *train* function from the "caret" package and the same random seed (210).

Using these methods, we first determined the optimal tuning parameters for each model. These parameters are summarized in the table below. If the tuning parameter is marked as NA, this means that the model does not have a tuning parameter.

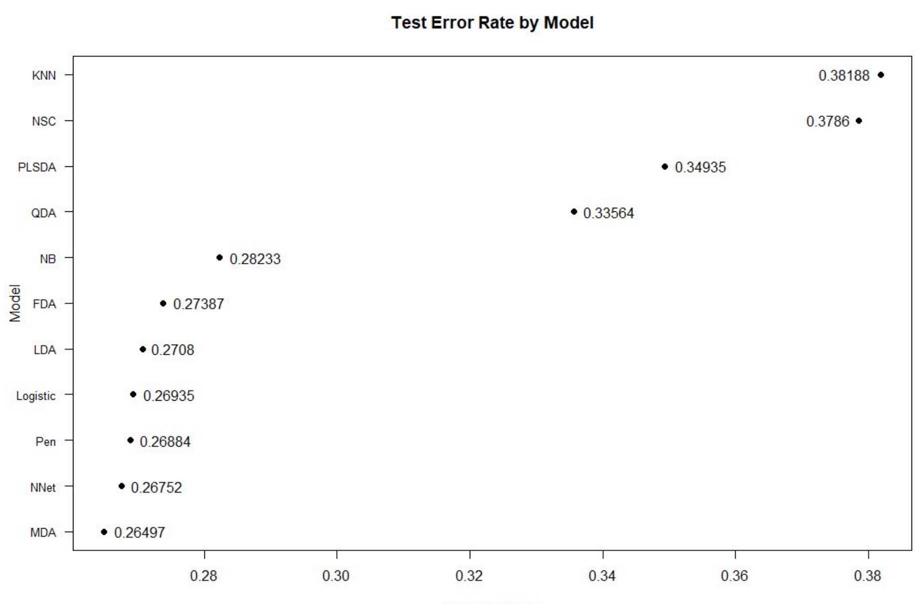
	Tuning Parameters
Logistic	NA
LDA	NA
PLSDA	# Components = 11
Penalized	α = 0.2, λ = 0.01
NSC	Threshold = 0
QDA	NA
MDA	Subclasses = 2
FDA	Degree = 1, nrpune = 11
Naïve Bayes	fL = 0, useKernal = True, adjust = 1
KNN	K = 90
NNET	Size = 8, Decay = 0.01

Using these tuning parameters, each model was applied to the test dataset. Confusion matrixes, variable importance, and other summary statistics for each model can be found in Appendix 1. We then compared these eleven models based on their cross-validation statistics which were implemented by using the resamples function with models that shared a common set of resampled data sets. The resamples function would collect the resampling results of these models into a single object. This could then be used for visualization and/or making some formal comparisons among these models. To further visualize the results, we created the following figure of the cross-validated ROC with a 95% confidence interval across the different models under consideration.

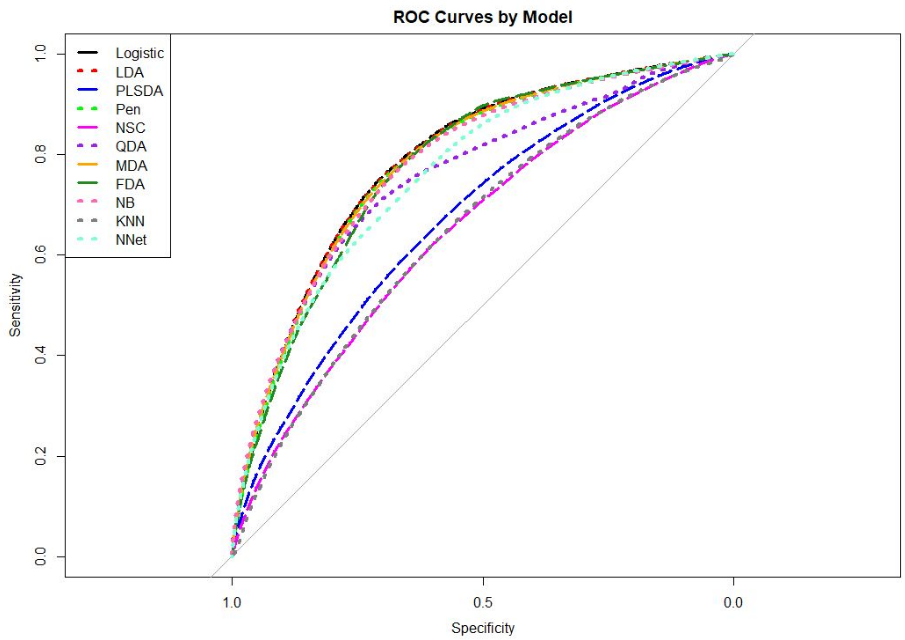


We observe from this figure that the KNN performs the worst and the NNet seems to perform the best in terms of the ROC. In addition, we also observe that other models behave similarly. To further assess possible differences among these 11 models, we compare the test error rates.

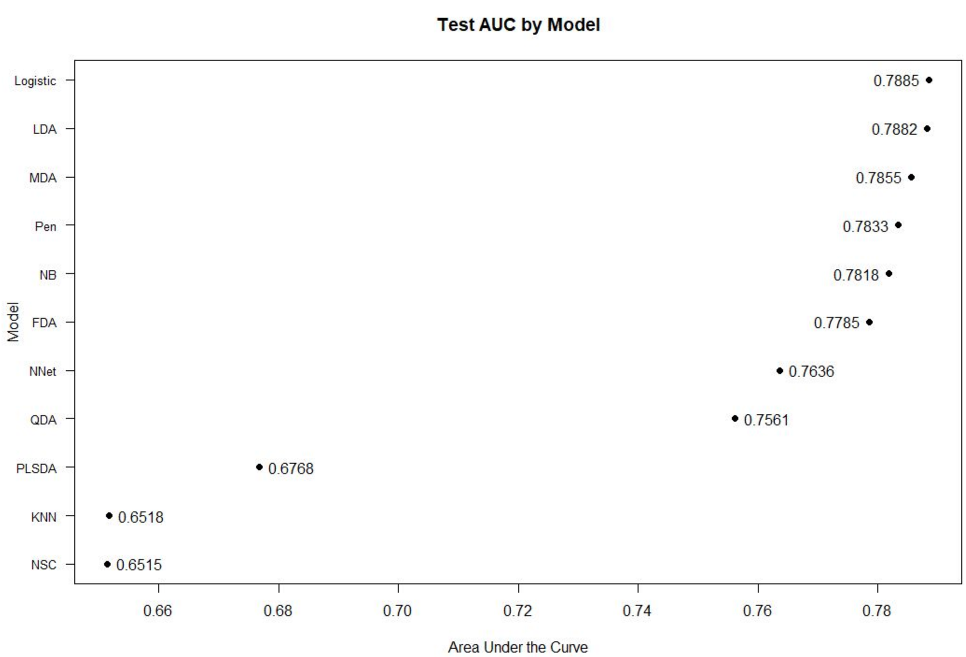
The plot below compares the test error rates for each model. From this plot, we can see that the worst-performing model in regards to the test error rate is KNN and the best is MDA. However, other models are close to MDA, we can see that there is only a less than 1% difference between these models.



The next plot compares the ROC curves between these models. Again, we can see that the models that had a less than 1% difference in the previous plot have very similar ROC curves, with the Logistic Regression model being slightly above the rest.



The final plot below compares the area under the ROC curves for each model. From this plot, we can see the KNN and NSC models have the lowest area while the Logistic Regression model has the highest area, closely followed by other models by a less than 1% difference.



Conclusion

In this project, we analyzed the Cardiovascular Disease dataset provided by kaggle.com, which contains information describing different aspects of patient characteristics, collected at the time of medical examination. Based on our statistical analysis for the data after preprocessing and splitting, we may conclude that the best model would be the Logistic Regression model. While other models may have had slightly better fit statistics, these differences were less than 1%. Therefore, we chose the simpler and more interpretable model as the best choice for this data. Based on Appendix 1, the result of the Logistic Regression model indicates that SystolicTrans and Age are the two most important variables. Also, people who have well-above normal cholesterol level are more likely to have cardiovascular disease.

However, the test error rate for the Logistic Regression model is 26.9%. This error rate may be considered a bit high when being used for medical purposes. Our goal was to build a model with high predictive accuracy, but was unable to do so. This does not mean that this model isn't useful. Our model can be used as a tool by doctors to assist in making a decision on whether or not to run more medical tests to determine whether a patient has a CVD.

We would recommend further investigation for predicting the presence of a cardiovascular disease in a patient. Running further tests could help to improve the high error rate, which in turn would improve the predictability of the models.

Works Cited

[1] American Heart Association. What is Cardiovascular Disease? 31 May 2017. <https://www.heart.org/en/health-topics/consumer-healthcare/what-is-cardiovascular-disease>.

[2] Mensah, George A. and David W. Brown. "An Overview Of Cardiovascular Disease Burden In The United States." Health Affairs 26.1 (2007): 38-48. <https://www.healthaffairs.org/doi/10.1377/hlthaff.26.1.38>.

[3] U.S. Department of Health & Human Services. Heart Disease. 2 December 2019. <https://www.cdc.gov/heartdisease/facts.htm>.

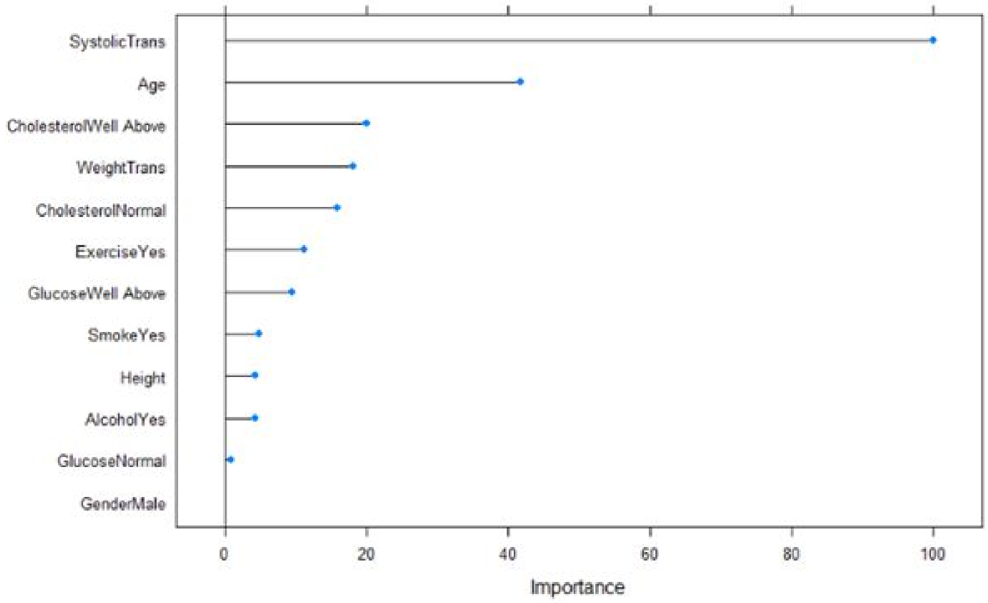
[4] —. Heart Disease and Stroke. 21 March 2019. <https://www.cdc.gov/chronicdisease/resources/publications/factsheets/heart-disease-stroke.htm>.

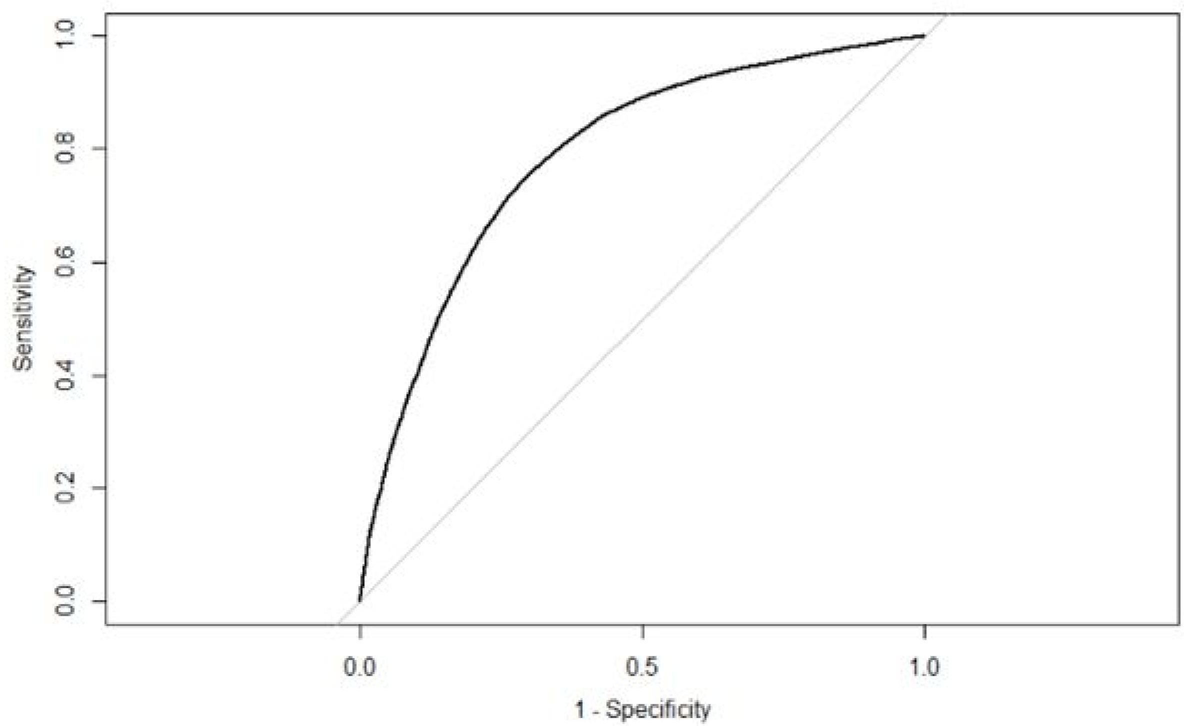
[5] World Health Organization. Cardiovascular diseases (CVDs). 17 May 2017. <https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)>.

Appendix 1: Supplementary Material for Models

I. Logistic Regression

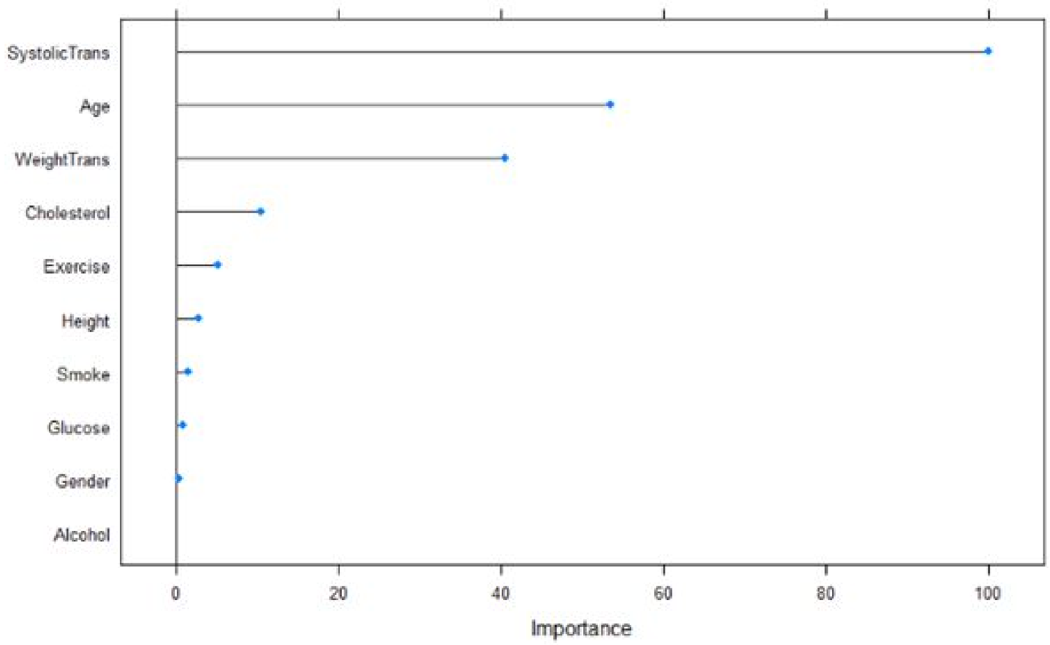
```
Generalized Linear Model

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results:

  ROC        Sens       Spec
  0.7885347  0.7744169  0.678567

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0741  -0.9339  -0.2358   0.9231   3.0430

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)            -4.773e+02  5.885e+00 -81.111  < 2e-16 ***
Age                     4.935e-02  1.517e-03  32.540  < 2e-16 ***
GenderMale             -1.409e-02  2.480e-02  -0.568 0.569876
Height                 -1.531e-02  3.926e-03  -3.899 9.66e-05 ***
WeightTrans             6.670e+00  4.607e-01  14.478  < 2e-16 ***
SystolicTrans           3.773e+02  4.887e+00  77.199  < 2e-16 ***
CholesterolNormal      -3.888e-01  3.053e-02 -12.735  < 2e-16 ***
CholesterolWell Above   7.481e-01  4.721e-02  15.846  < 2e-16 ***
GlucoseNormal          -4.713e-02  4.056e-02  -1.162 0.245215
GlucoseWell Above      -4.483e-01  5.764e-02  -7.778 7.37e-15 ***
SmokeYes               -1.674e-01  3.905e-02  -4.286 1.82e-05 ***
AlcoholYes             -1.831e-01  4.723e-02  -3.876 0.000106 ***
ExerciseYes            -2.254e-01  2.453e-02  -9.186  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 76029  on 54847  degrees of freedom
Residual deviance: 61527  on 54835  degrees of freedom
AIC: 61553

Number of Fisher Scoring iterations: 4

logicPred   No  Yes
      No  5437 2202
      Yes 1491 4581

glm variable importance

                      Overall
SystolicTrans        100.0000
Age                   41.7213
CholesterolWell Above 19.9365
WeightTrans           18.1511
CholesterolNormal     15.8773
ExerciseYes           11.2457
GlucoseWell Above      9.4084
SmokeYes               4.8521
Height                 4.3466
AlcoholYes             4.3169
GlucoseNormal          0.7749
GenderMale             0.0000
```

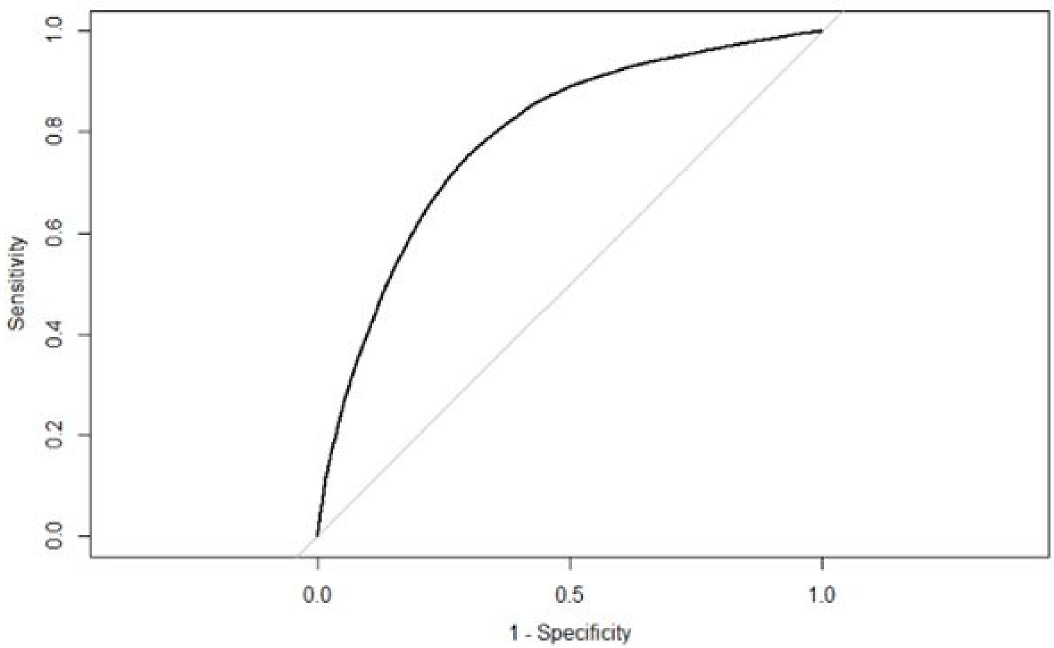




II. Linear Discriminant Analysis

```
Linear Discriminant Analysis

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results:

  ROC        Sens       Spec
  0.7882096  0.7754388  0.6763379

ldaPred   No  Yes
    No  5442 2227
    Yes 1486 4556

               Importance
SystolicTrans    100.0000
Age               53.4291
WeightTrans       40.5428
Cholesterol       10.4023
Exercise           5.0007
Height             2.6500
Smoke              1.4287
Glucose            0.7124
Gender             0.2334
Alcohol            0.0000
```



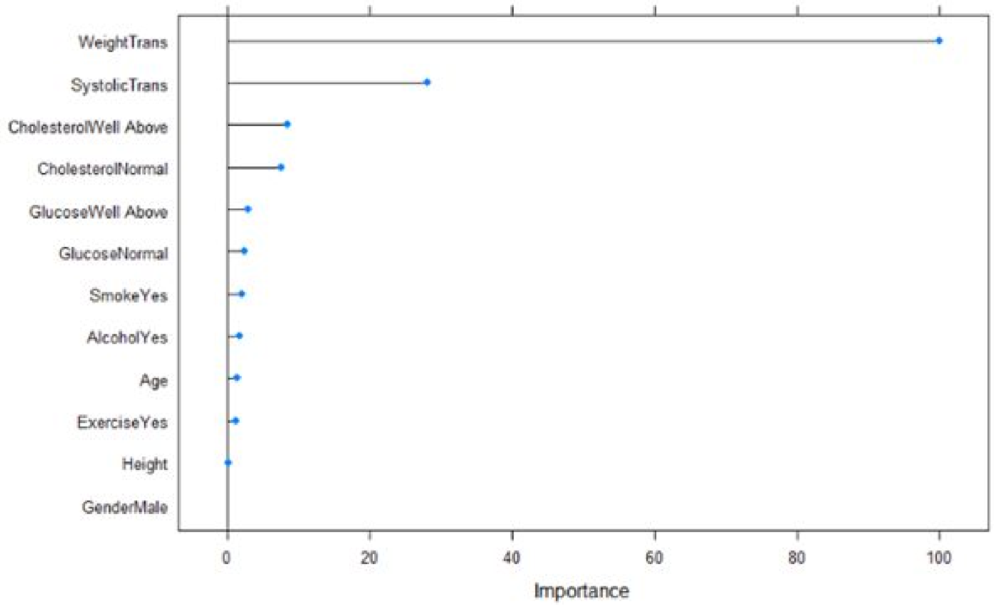


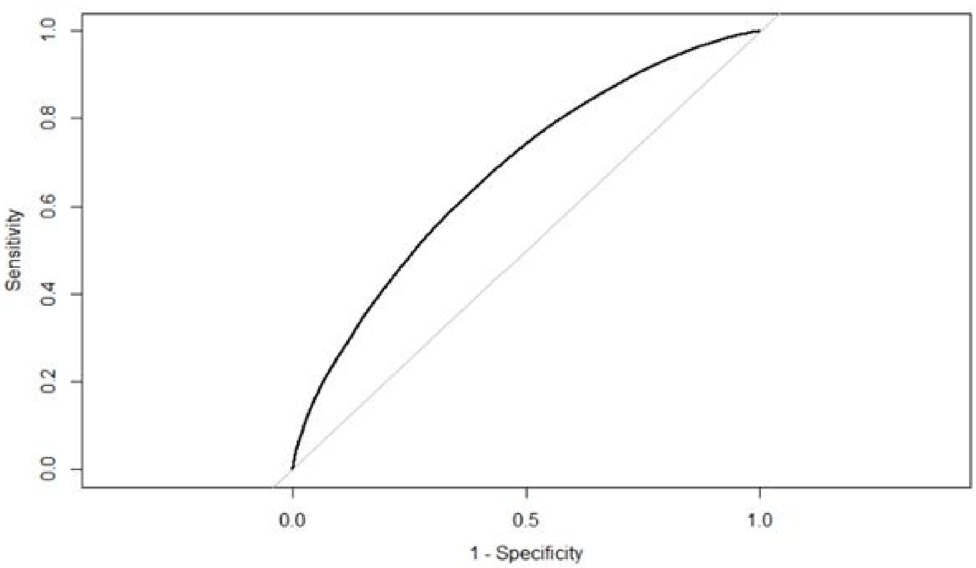
III. Partial Least Squares Discriminant Analysis

```
Partial Least Squares

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  ncomp  ROC        Sens       Spec
   1     0.6355115  0.5630831  0.6308418
   2     0.6487896  0.5954099  0.6150376
   3     0.6733142  0.6580427  0.5899808
   4     0.6750871  0.6674827  0.5819844
   5     0.6757603  0.6646594  0.5864957
   6     0.6769890  0.6695670  0.5841899
   7     0.6783642  0.6692436  0.5824797
   8     0.6810335  0.6729042  0.5849683
   9     0.6958609  0.6745958  0.6066696
  10     0.6992300  0.6781813  0.6053192
  11     0.6998670  0.6779734  0.6068052

ROC was used to select the optimal model using the largest value.
The final value used for the model was ncomp = 11.
```

```
plsPred   No  Yes
     No 4812 2674
    Yes 2116 4109
```

```
                        Overall
WeightTrans           100.00000
SystolicTrans          28.00463
CholesterolWell Above   8.38729
CholesterolNormal       7.49798
GlucoseWell Above       2.90622
GlucoseNormal           2.38350
SmokeYes                2.09274
AlcoholYes              1.69459
Age                     1.32596
ExerciseYes             1.11429
Height                  0.06204
GenderMale              0.00000
```


WeightTrans
SystolicTrans
CholesterolWell Above
CholesterolNormal
GlucoseWell Above
GlucoseNormal
SmokeYes
AlcoholYes
Age
ExerciseYes
Height
GenderMale
0
20
40
60
80
100
Importance

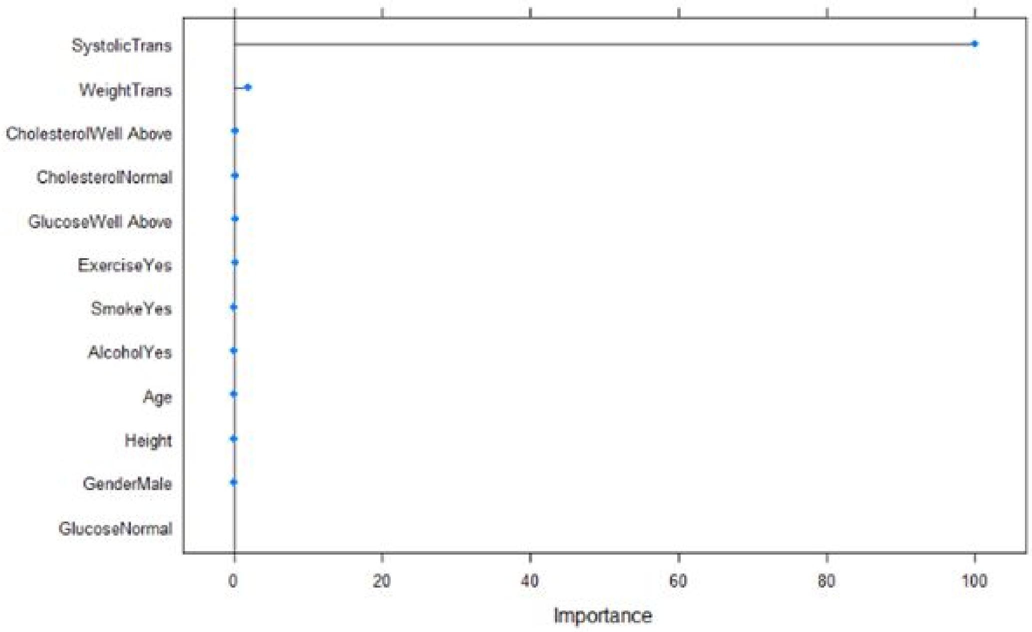

Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1 - Specificity

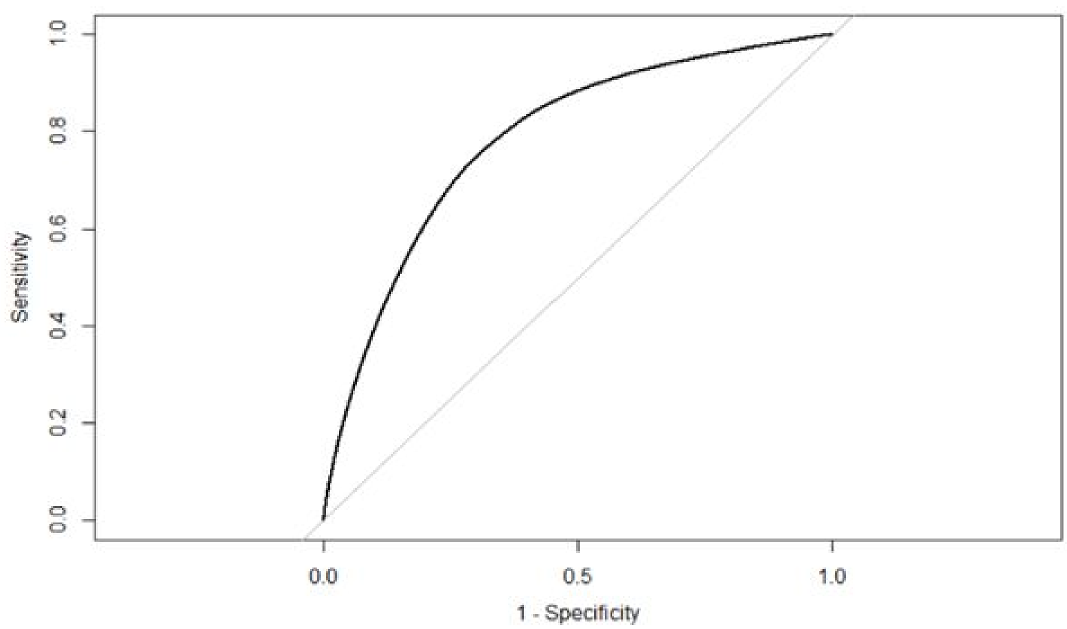
IV. Penalized Model

```
glmnet

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  alpha  lambda      ROC        Sens       Spec
  0.0    0.01000000  0.7882030  0.7726039  0.6790624
  0.0    0.02357143  0.7881643  0.7724654  0.6790034
  0.0    0.03714286  0.7878861  0.7720843  0.6791626
  0.0    0.05071429  0.7875939  0.7714203  0.6788972
  0.0    0.06428571  0.7872977  0.7711028  0.6784903
  0.0    0.07785714  0.7870018  0.7709700  0.6778888
  0.0    0.09142857  0.7867104  0.7707333  0.6771340
  0.0    0.10500000  0.7864231  0.7706236  0.6760902
  0.0    0.11857143  0.7861461  0.7706755  0.6753295
  0.0    0.13214286  0.7858769  0.7707737  0.6743978
  0.0    0.14571429  0.7856144  0.7708314  0.6736017
  0.0    0.15928571  0.7853601  0.7708776  0.6728645
  0.0    0.17285714  0.7851133  0.7711432  0.6720035
  0.0    0.18642857  0.7848717  0.7713799  0.6713254
  0.0    0.20000000  0.7846380  0.7717206  0.6703524
  0.1    0.01000000  0.7884570  0.7737471  0.6783075
  0.1    0.02357143  0.7882553  0.7731582  0.6772873
  0.1    0.03714286  0.7879644  0.7728522  0.6764676
  0.1    0.05071429  0.7876048  0.7727598  0.6754415
  0.1    0.06428571  0.7871934  0.7727945  0.6745924
  0.1    0.07785714  0.7868470  0.7729619  0.6739142
  0.1    0.09142857  0.7865387  0.7730600  0.6730591
  0.1    0.10500000  0.7862442  0.7732621  0.6723810
  0.1    0.11857143  0.7859635  0.7732044  0.6719092
  0.1    0.13214286  0.7857468  0.7735508  0.6711662
  0.1    0.14571429  0.7855760  0.7740820  0.6702875

  0.1    0.15928571  0.7854317  0.7745497  0.6691552
  0.1    0.17285714  0.7853318  0.7750058  0.6682412
  0.1    0.18642857  0.7852818  0.7755774  0.6674864
  0.1    0.20000000  0.7852642  0.7762125  0.6667905
  0.2    0.01000000  0.7884757  0.7741455  0.6776824
  0.2    0.02357143  0.7881513  0.7738915  0.6759074
  0.2    0.03714286  0.7875656  0.7743187  0.6739791
  0.2    0.05071429  0.7870476  0.7748499  0.6728173
  0.2    0.06428571  0.7866450  0.7753176  0.6716379
  0.2    0.07785714  0.7863617  0.7760046  0.6699455
  0.2    0.09142857  0.7862180  0.7767148  0.6686304
  0.2    0.10500000  0.7862071  0.7777309  0.6674628
  0.2    0.11857143  0.7862318  0.7785508  0.6664662
  0.2    0.13214286  0.7862605  0.7795958  0.6653634
  0.2    0.14571429  0.7862889  0.7805196  0.6643196
  0.2    0.15928571  0.7863119  0.7819053  0.6635058
  0.2    0.17285714  0.7863283  0.7832737  0.6624738
  0.2    0.18642857  0.7863302  0.7844284  0.6613593
  0.2    0.20000000  0.7863141  0.7854965  0.6601504
  0.4    0.01000000  0.7883970  0.7747864  0.6760725
  0.4    0.02357143  0.7873840  0.7758891  0.6725343
  0.4    0.03714286  0.7866489  0.7775058  0.6696447
  0.4    0.05071429  0.7863399  0.7792841  0.6670441
  0.4    0.06428571  0.7862272  0.7812875  0.6645968
  0.4    0.07785714  0.7860328  0.7836836  0.6612944
  0.4    0.09142857  0.7857126  0.7863799  0.6572018
  0.4    0.10500000  0.7852068  0.7889607  0.6531800
  0.4    0.11857143  0.7844617  0.7916744  0.6495651
  0.4    0.13214286  0.7834096  0.7949018  0.6460150
  0.4    0.14571429  0.7823942  0.7965993  0.6440277
  0.4    0.15928571  0.7813212  0.8015185  0.6381837
  0.4    0.17285714  0.7801439  0.8056351  0.6299455
  0.4    0.18642857  0.7788565  0.8091109  0.6196609
  0.4    0.20000000  0.7774928  0.8067206  0.6197081

ROC was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0.2 and lambda = 0.01.
```

```
penPred   No  Yes
    No  5445 2203
    Yes 1483 4580
```

```
                         Overall
SystolicTrans          1.000e+02
WeightTrans            1.787e+00
CholesterolWell Above  1.816e-01
CholesterolNormal      1.111e-01
GlucoseWell Above      7.717e-02
ExerciseYes            5.382e-02
SmokeYes               3.993e-02
AlcoholYes             3.870e-02
Age                    1.337e-02
Height                 3.560e-03
GenderMale             5.445e-04
GlucoseNormal          0.000e+00
```


SystolicTrans
WeightTrans
CholesterolWell Above
CholesterolNormal
GlucoseWell Above
ExerciseYes
SmokeYes
AlcoholYes
Age
Height
GenderMale
GlucoseNormal
0
20
40
60
80
100
Importance

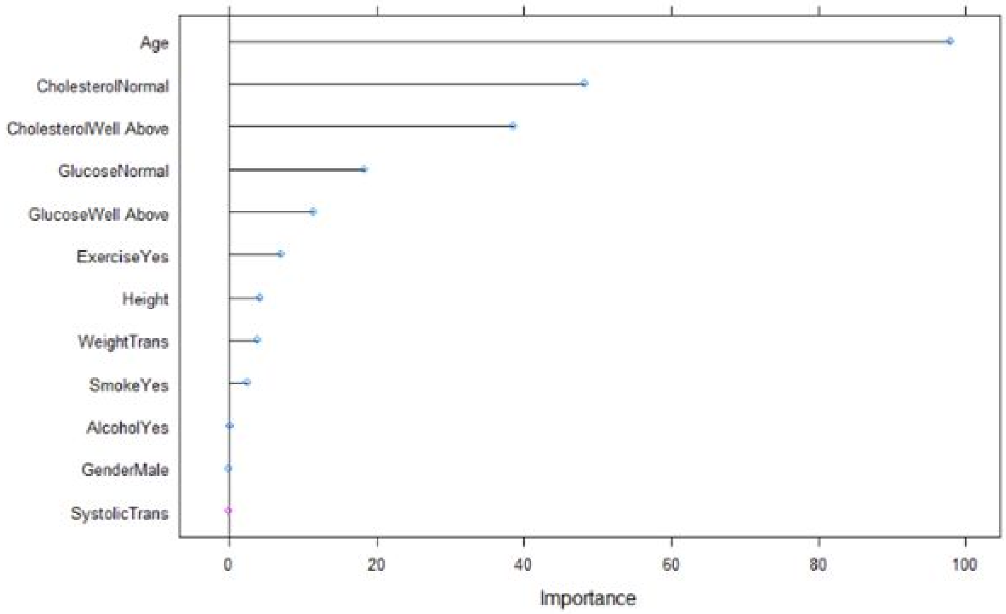

Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1 - Specificity

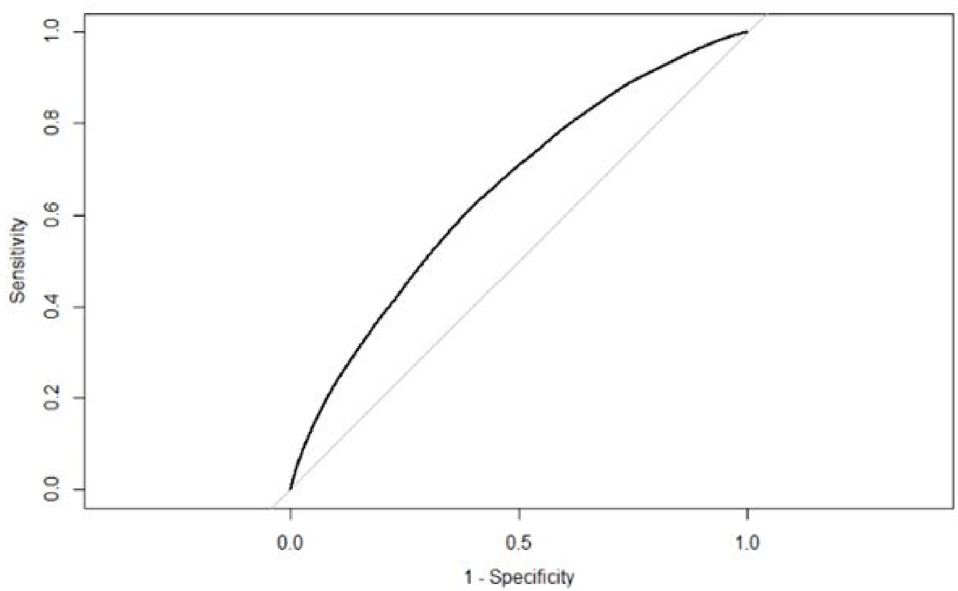
V. Nearest Shrunken Centroids

```
Nearest Shrunken Centroids

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  threshold  ROC        Sens       Spec
   0         0.6659480  0.6240127  0.6136282
   1         0.6655163  0.6186085  0.6178682
   2         0.6649961  0.6131062  0.6222969
   3         0.6642235  0.6090647  0.6255403
   4         0.6633550  0.6068072  0.6267197
   5         0.6628701  0.6071882  0.6260062
   6         0.6621705  0.6087182  0.6243255
   7         0.6614166  0.6095381  0.6225800
   8         0.6606440  0.6104965  0.6203745
   9         0.6595232  0.6110450  0.6189828
  10         0.6582969  0.6115878  0.6177503
  11         0.6573506  0.6122229  0.6168362
  12         0.6566522  0.6134238  0.6137815
  13         0.6557926  0.6148037  0.6098423
  14         0.6542360  0.6156178  0.6055315
  15         0.6522387  0.6158256  0.6035029
  16         0.6506552  0.6151790  0.6033908
  17         0.6491252  0.6142436  0.6012207
  18         0.6463103  0.6122806  0.5986024
  19         0.6432750  0.6078233  0.5994575
  20         0.6421701  0.6035508  0.6025004
  21         0.6412549  0.5992321  0.6052012
  22         0.6397251  0.5966975  0.6046410
  23         0.6371071  0.5953580  0.6015627
  24         0.6348986  0.5963395  0.5970338
  25         0.6348838  0.5991051  0.5937314

ROC was used to select the optimal model using the largest value.
The final value used for the model was threshold = 0.
```

```
nscPred   No  Yes
    No  4386 2649
    Yes 2542 4134
```

```
                      Importance
Age                    97.869807
CholesterolNormal      48.316269
CholesterolWell Above  38.712981
GlucoseNormal          18.393890
GlucoseWell Above      11.502461
ExerciseYes             7.089080
Height                  4.169910
WeightTrans             3.789643
SmokeYes                2.523441
AlcoholYes              0.213218
GenderMale              0.005682
SystolicTrans           0.000000
```

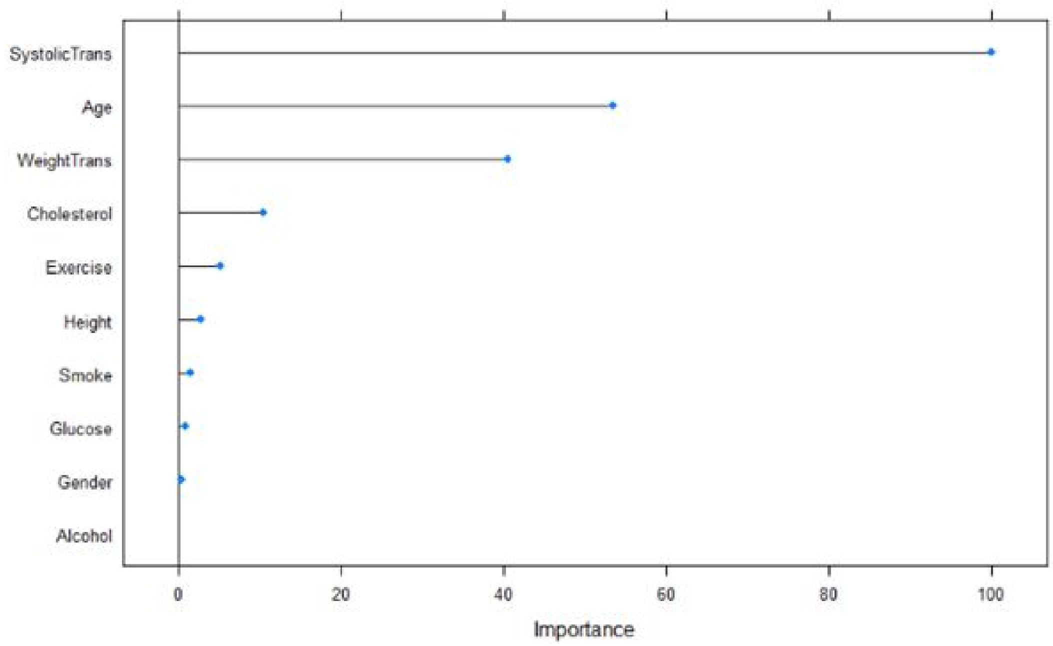


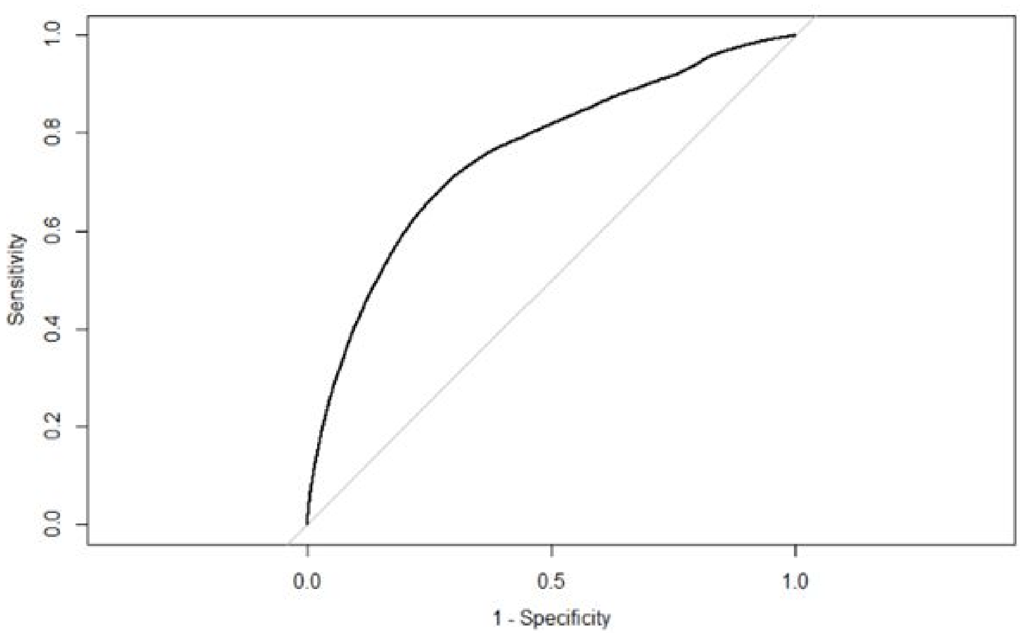


VI. Quadratic Discriminant Analysis

```
Quadratic Discriminant Analysis

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results:

  ROC        Sens       Spec
  0.7562116  0.8057217  0.5288751

qdaPred   No  Yes
    No  5589 3263
    Yes 1339 3520
```

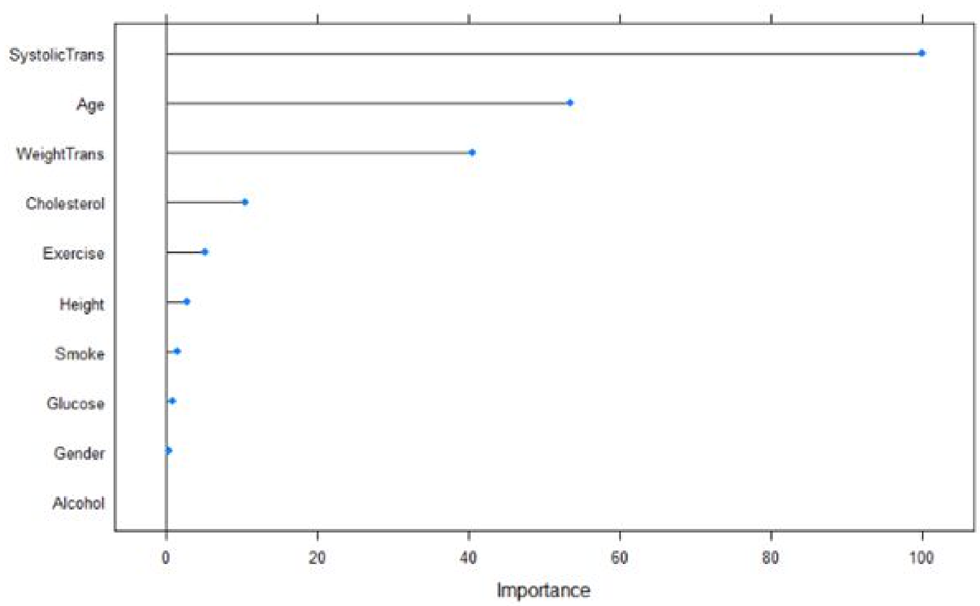
```
              Importance
SystolicTrans   100.0000
Age              53.4291
WeightTrans      40.5428
Cholesterol      10.4023
Exercise          5.0007
Height            2.6500
Smoke             1.4287
Glucose           0.7124
Gender            0.2334
Alcohol           0.0000
```

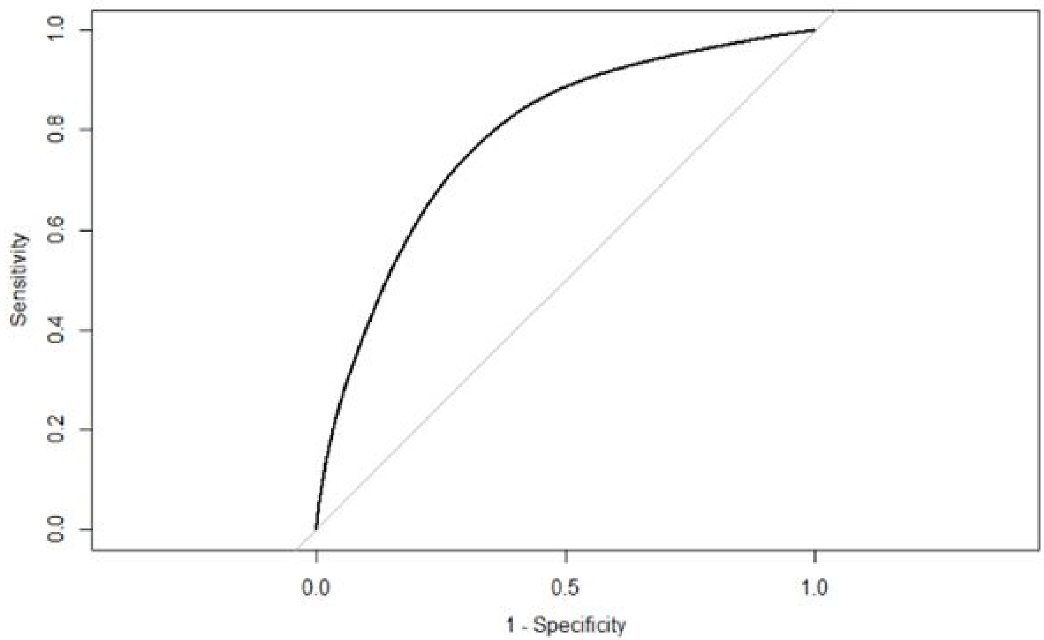




VII. Mixture Discriminant Analysis

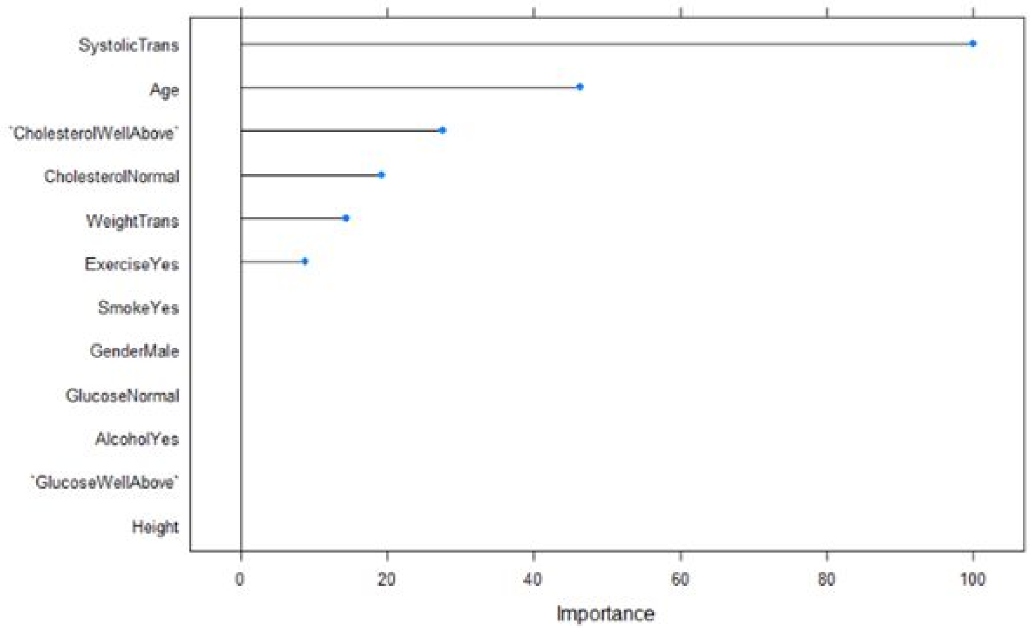
```
Mixture Discriminant Analysis

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  subclasses  ROC        Sens       Spec
   1          0.7882096  0.7754388  0.6763497
   2          0.7902966  0.7661547  0.6888103
   3          0.7902874  0.7654388  0.6885213
   4          0.7884634  0.7671363  0.6851953
   5          0.7855521  0.7618129  0.6839805
   6          0.7859941  0.7639145  0.6848533
   7          0.7830827  0.7601501  0.6822173
   8          0.7833445  0.7566224  0.6857143
   9          0.7810281  0.7465185  0.6907209
  10          0.7797384  0.7430774  0.6906089

ROC was used to select the optimal model using the largest value.
The final value used for the model was subclasses = 2.

mdaPred   No  Yes
    No  5412 2117
    Yes 1516 4666

              Importance
SystolicTrans   100.0000
Age              53.4291
WeightTrans      40.5428
Cholesterol      10.4023
Exercise          5.0007
Height            2.6500
Smoke             1.4287
Glucose           0.7124
Gender            0.2334
Alcohol           0.0000
```

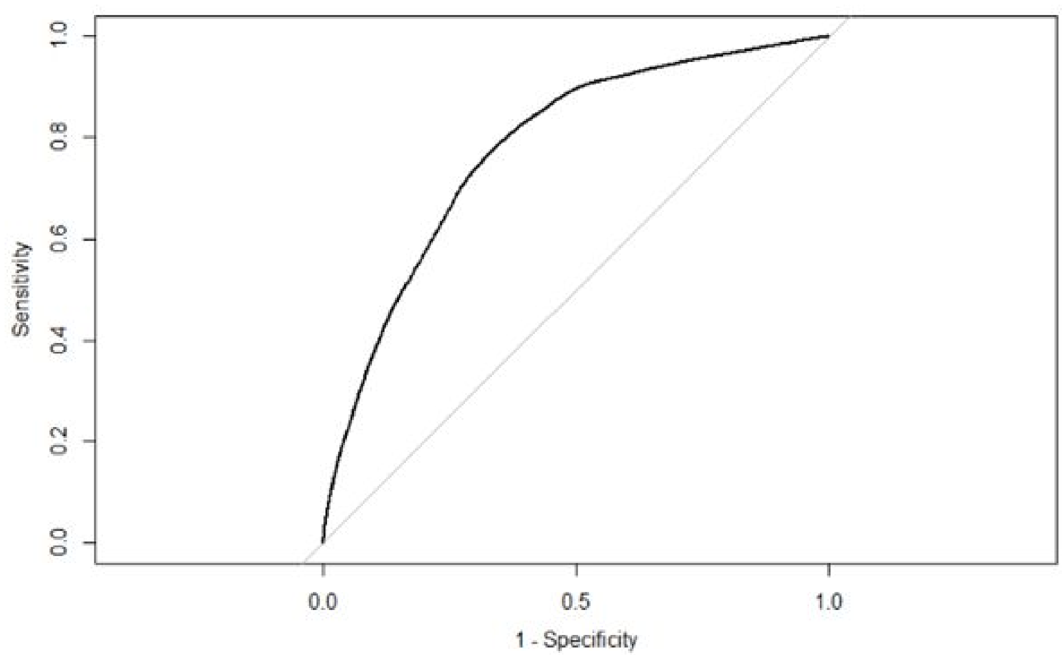




VIII. Flexible Discriminant Analysis

```
Flexible Discriminant Analysis

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  nprune  ROC        Sens       Spec
   2      0.7500078  0.8047864  0.6202211
   6      0.7860634  0.7981755  0.6425947
  11      0.7927760  0.8060450  0.6406664

Tuning parameter 'degree' was held constant at a value of 1
ROC was used to select the optimal model using the largest value.
The final values used for the model were degree = 1 and nprune = 11.

fdaPred   No  Yes
    No  5621 2448
    Yes 1307 4335

                        Overall
SystolicTrans           100.000
Age                      46.280
`CholesterolWellAbove`   27.641
CholesterolNormal        19.262
WeightTrans              14.408
ExerciseYes               8.858
GenderMale                0.000
`GlucoseWellAbove`        0.000
SmokeYes                  0.000
GlucoseNormal             0.000
Height                    0.000
AlcoholYes                0.000
```


SystolicTrans
Age
`CholesterolWellAbove`
CholesterolNormal
WeightTrans
ExerciseYes
SmokeYes
GenderMale
GlucoseNormal
AlcoholYes
`GlucoseWellAbove`
Height
0
20
40
60
80
100
Importance

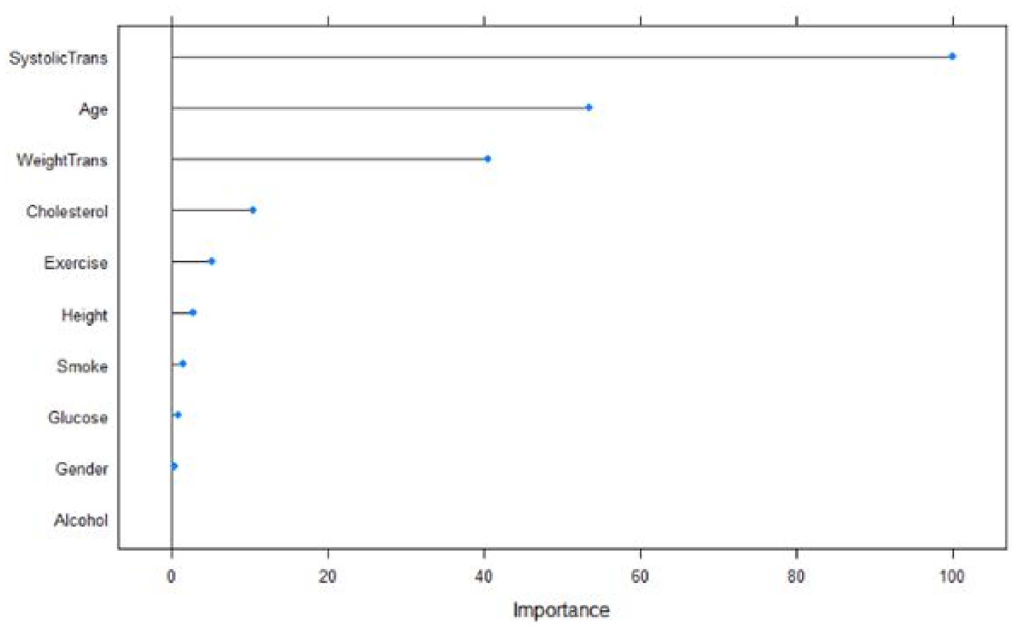

Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1 - Specificity

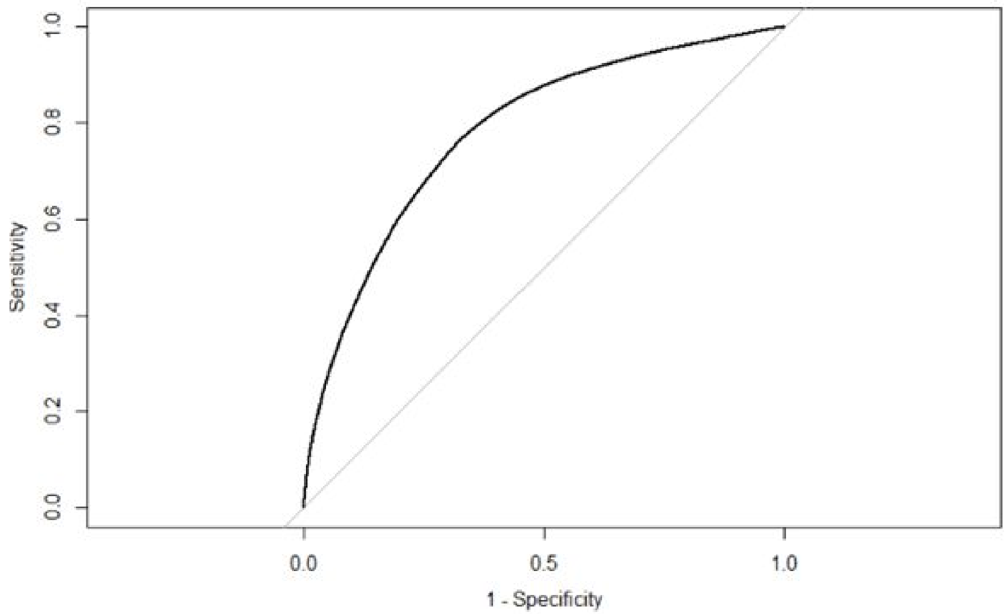
IX. Naive Bayes

```
Naive Bayes

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  usekernel  ROC        Sens       Spec
  FALSE      0.7824492  0.7672748  0.6674805
   TRUE      0.7868560  0.8285855  0.6033731

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter
 'adjust' was held constant at a value of 1
ROC was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

```
nbPred   No  Yes
   No  5771 2714
   Yes 1157 4069
```

```
               Importance
SystolicTrans    100.0000
Age               53.4291
WeightTrans       40.5428
Cholesterol       10.4023
Exercise           5.0007
Height             2.6500
Smoke              1.4287
Glucose            0.7124
Gender             0.2334
Alcohol            0.0000
```



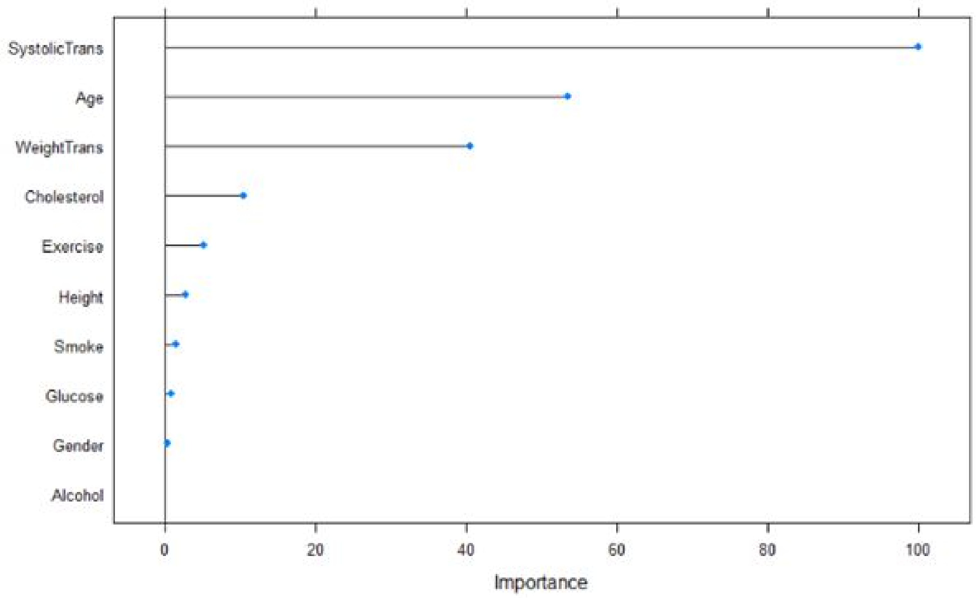


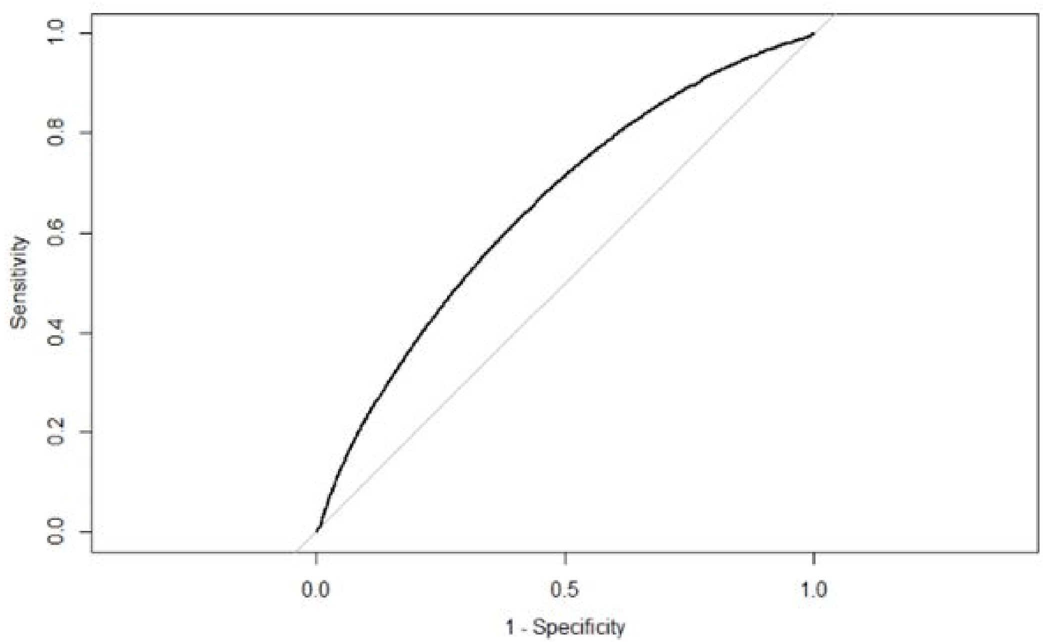
X. K-Nearest Neighbors

```
k-Nearest Neighbors

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   1  0.5531709  0.5581178  0.5482353
   2  0.5766450  0.5616282  0.5444612
   3  0.5915298  0.5819111  0.5608374
   4  0.6022008  0.5823961  0.5608905
   5  0.6107851  0.5971420  0.5681144
   6  0.6173465  0.5973961  0.5673655
   7  0.6230811  0.6067263  0.5720655
   8  0.6274421  0.6070439  0.5710512
   9  0.6308207  0.6146132  0.5755919
  10  0.6337767  0.6142379  0.5756627
  11  0.6366822  0.6199076  0.5779095
  12  0.6390669  0.6197402  0.5770662
  13  0.6407544  0.6237760  0.5783518
  14  0.6422975  0.6238106  0.5763467
  15  0.6437673  0.6282737  0.5762229
  16  0.6450278  0.6290069  0.5761286
  17  0.6465473  0.6326501  0.5754681
  18  0.6478803  0.6338222  0.5745658
  19  0.6490043  0.6378522  0.5746307
  20  0.6499920  0.6391339  0.5737343
  21  0.6509069  0.6409296  0.5735515
  22  0.6516753  0.6424423  0.5721480
  23  0.6524167  0.6454330  0.5729146
  24  0.6529511  0.6451328  0.5722070
  25  0.6534494  0.6465012  0.5712811
  26  0.6541318  0.6475289  0.5709627
  27  0.6547561  0.6501039  0.5708625
  28  0.6551820  0.6503637  0.5702727
```

```
29  0.6555215  0.6524018  0.5693823
30  0.6561211  0.6529619  0.5692113
31  0.6568358  0.6545958  0.5685154
32  0.6573352  0.6546247  0.5675247
33  0.6577442  0.6563395  0.5673714
34  0.6581379  0.6560450  0.5676544
35  0.6585870  0.6577829  0.5666460
36  0.6588742  0.6577367  0.5662273
37  0.6591750  0.6594053  0.5664868
38  0.6595728  0.6595035  0.5665635
39  0.6598742  0.6611721  0.5656140
40  0.6601957  0.6617783  0.5648651
41  0.6605810  0.6626674  0.5652779
42  0.6607394  0.6633256  0.5647943
43  0.6609734  0.6646074  0.5644287
44  0.6613057  0.6646074  0.5636031
45  0.6615033  0.6658314  0.5629368
46  0.6618434  0.6660912  0.5630134
47  0.6619942  0.6674365  0.5625534
48  0.6620865  0.6673383  0.5623942
49  0.6621940  0.6691166  0.5619991
50  0.6623479  0.6685508  0.5615922
51  0.6624864  0.6699423  0.5614035
52  0.6626081  0.6700693  0.5611381
53  0.6627521  0.6709122  0.5609789
54  0.6628362  0.6715069  0.5601297
55  0.6629119  0.6724192  0.5598939
56  0.6630207  0.6721536  0.5595518
57  0.6631615  0.6731351  0.5599351
58  0.6632705  0.6735046  0.5595223
59  0.6633263  0.6738857  0.5589149
60  0.6634235  0.6744977  0.5578888
61  0.6635179  0.6751559  0.5581660
62  0.6637002  0.6756871  0.5573994
63  0.6638416  0.6764896  0.5570809
64  0.6639803  0.6769053  0.5574230
65  0.6641498  0.6779677  0.5566033
66  0.6642961  0.6785855  0.5563438
67  0.6643954  0.6788453  0.5559959
68  0.6644556  0.6788972  0.5555713
69  0.6644882  0.6796536  0.5557718

 70  0.6644764  0.6800635  0.5552410
 71  0.6644738  0.6804273  0.5543388
 72  0.6644292  0.6803522  0.5536370
 73  0.6644647  0.6811085  0.5536960
 74  0.6644741  0.6812125  0.5528291
 75  0.6644047  0.6816628  0.5521333
 76  0.6644047  0.6815878  0.5515436
 77  0.6643363  0.6826443  0.5505823
 78  0.6643100  0.6829965  0.5504231
 79  0.6643266  0.6835739  0.5498039
 80  0.6643931  0.6842725  0.5497567
 81  0.6644572  0.6845381  0.5490314
 82  0.6645029  0.6853811  0.5489842
 83  0.6645222  0.6856120  0.5481350
 84  0.6645514  0.6856640  0.5476633
 85  0.6645801  0.6862298  0.5470441
 86  0.6646461  0.6867206  0.5467669
 87  0.6646042  0.6871247  0.5456347
 88  0.6646831  0.6872460  0.5458824
 89  0.6646797  0.6875520  0.5456995
 90  0.6647307  0.6874827  0.5455285
 91  0.6647263  0.6881524  0.5446617
 92  0.6647002  0.6885508  0.5446970
 93  0.6647141  0.6883891  0.5448327
 94  0.6646666  0.6887009  0.5439304
 95  0.6646077  0.6891686  0.5437063
 96  0.6645802  0.6895266  0.5436356
 97  0.6645058  0.6895670  0.5436120
 98  0.6645113  0.6896998  0.5432051
 99  0.6645242  0.6903522  0.5427746
100  0.6644779  0.6901617  0.5420610

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 90.

knnPred   No  Yes
    No  4732 3040
    Yes 2196 3743
```

	Importance
SystolicTrans	100.0000
Age	53.4291
WeightTrans	40.5428
Cholesterol	10.4023
Exercise	5.0007
Height	2.6500
Smoke	1.4287
Glucose	0.7124
Gender	0.2334
Alcohol	0.0000



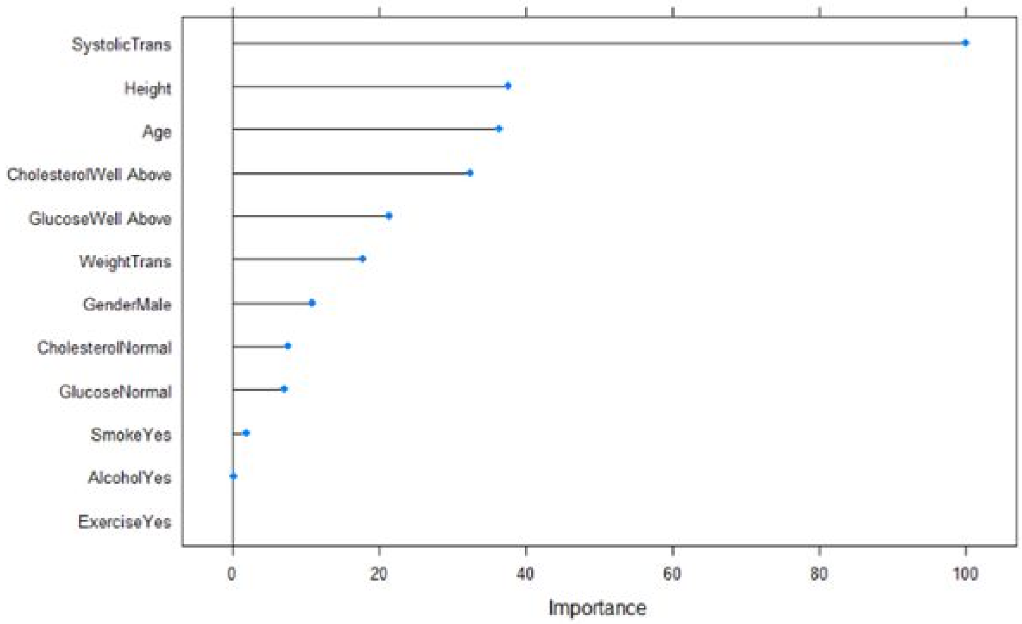


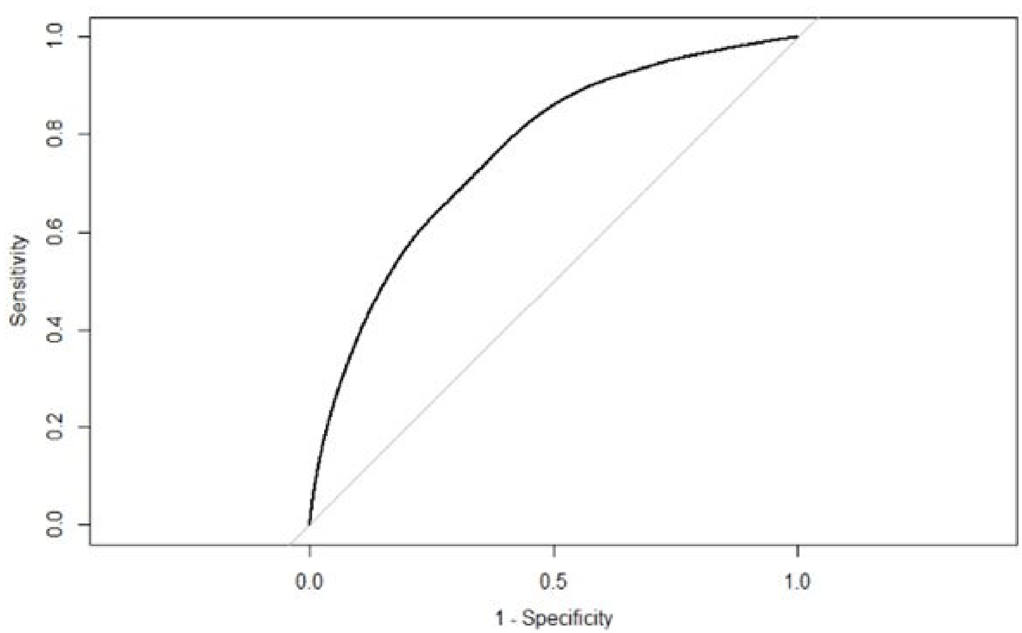
XI. Neural Network

```
Neural Network

54848 samples
   10 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 41137, 41137, 41137, 41137, 41137, 41137, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.00   0.5643346  0.9382564  0.1676367
   1    0.01   0.6647351  0.8248961  0.4378358
   1    0.10   0.7570364  0.7750635  0.6238420
   1    0.50   0.7102799  0.7933545  0.5429102
   2    0.00   0.5554200  0.9263164  0.1687336
   2    0.01   0.7501860  0.7915069  0.5859590
   2    0.10   0.7371549  0.7891975  0.5795312
   2    0.50   0.7788922  0.7379619  0.6884682
   3    0.00   0.5821396  0.9155831  0.2233820
   3    0.01   0.7679598  0.7846478  0.6151732
   3    0.10   0.7824676  0.7668476  0.6683238
   3    0.50   0.7630939  0.7475404  0.6630871
   4    0.00   0.6087686  0.9015820  0.2761787
   4    0.01   0.7720788  0.7694400  0.6484269
   4    0.10   0.7906050  0.7537644  0.7002565
   4    0.50   0.7729096  0.7494804  0.6631107
   5    0.00   0.6414549  0.8745092  0.3558131
   5    0.01   0.7937158  0.7477425  0.7104349
   5    0.10   0.7910820  0.7543187  0.7000737
   5    0.50   0.7811261  0.7402252  0.6908801
   6    0.00   0.6968653  0.8227945  0.4914168
   6    0.01   0.7844264  0.7580312  0.6818694
   6    0.10   0.7913389  0.7538626  0.7008816
   6    0.50   0.7805268  0.7387413  0.6911632
   7    0.00   0.7099556  0.8166570  0.5227893
   7    0.01   0.7833756  0.7534700  0.6849005
   7    0.10   0.7914762  0.7524654  0.7020257
   7    0.50   0.7743754  0.7497113  0.6642430
   8    0.00   0.7436450  0.8144284  0.5661448
   8    0.01   0.7943617  0.7462413  0.7127466
   8    0.10   0.7821807  0.7626386  0.6736488
   8    0.50   0.7811289  0.7400000  0.6914522
   9    0.00   0.7383971  0.8021189  0.5738110
   9    0.01   0.7727820  0.7669630  0.6551732
   9    0.10   0.7909299  0.7528811  0.7011706
   9    0.50   0.7811614  0.7404273  0.6910865
  10    0.00   0.7465481  0.7814723  0.6114699
  10    0.01   0.7941097  0.7460566  0.7129707
  10    0.10   0.7909446  0.7520843  0.7017544
  10    0.50   0.7802166  0.7382333  0.6915465

ROC was used to select the optimal model using the largest value.
The final values used for the model were size = 8 and decay = 0.01.
```

```
nnetPred   No  Yes
     No  5303 2043
     Yes 1625 4740
```

```
                      Overall
SystolicTrans         100.000
Height                 37.546
Age                    36.327
Cholesterolwell Above  32.433
Glucosewell Above      21.317
WeightTrans            17.670
GenderMale             10.784
CholesterolNormal       7.579
GlucoseNormal           6.972
SmokeYes                1.917
AlcoholYes              0.183
ExerciseYes             0.000
```


SystolicTrans
Height
Age
CholesterolWell Above
GlucoseWell Above
WeightTrans
GenderMale
CholesterolNormal
GlucoseNormal
SmokeYes
AlcoholYes
ExerciseYes
0
20
40
60
80
100
Importance


1.0
0.8
0.6
0.4
0.2
0.0
Sensitivity
0.0
0.5
1.0
1 - Specificity

R Code

```
### Data Algorithms II
### Project Code


################## Data Cleaning ########################

### Import data
data <- read.csv("cardio.csv")
View(data)

### The purpose of this code is to clean the data and apply unit conversions from metri to
imperial

### ID column can be removed
data <- data[,-1]

### Create better column names
varNames <- c("Age", "Gender", "Height", "Weight", "Systolic", "Diastolic",
        "Cholesterol", "Glucose", "Smoke", "Alcohol", "Exercise", "Disease")
colnames(data) <- varNames

### Age is in days, Height is in cm, Weight is in kg.
### Change these to years, inches, and pounds, respectively.

data$Age <- round(data$Age / 365.25, 2) ### This accounts for leap years
data$Height <- round(data$Height / 2.54, 2)
data$Weight <- round(data$Weight * 2.20462, 2)


### Gender: 1 = women, 2 = men.
### Cholesterol: 1 = normal, 2 = above normal, 3 = well above normal
### Glucose: 1 = normal, 2 = above normal, 3 = well above normal

### Subtract 1 from everyone of these variables to have a base level of zero
data$Gender <- data$Gender - 1
data$Cholesterol <- data$Cholesterol - 1
data$Glucose <- data$Glucose - 1


### Now convert all categorical variables to factor type
data[,c(2,7:12)] <- lapply(data[,c(2,7:12)], factor)


### Output new csv
```

```
write.csv(data, "cardio2.csv")


rm(list = ls())


data <- read.csv("cardio2.csv")
data <- data[,-1]
data[,c(2,7:12)] <- lapply(data[,c(2,7:12)], factor)

### Grab continuous variables
contin <- data[,c(1,3,4,5,6)]

### Look at boxplots

par(mfrow=c(1,5))
for (i in 1:5){
  boxplot(contin[,i], xlab = colnames(contin)[i])
}


### We can see that systolic and diastolic have massive outliers that make no physical sense.
Why is this?
dev.off()
plot(contin$Systolic, contin$Diastolic)

### There are plenty of assumptions we could make to fix the data to our needs
### However, there are just too many that are needed for our analysis to still be valid
### Instead, we will focus on the BP range from hypotension to hypertension

data2 <- data[which(data$Systolic >= 50 & data$Systolic < 220),]
data2 <- data2[which(data2$Diastolic >= 20 & data2$Diastolic < 190),]

plot(data2$Systolic, data2$Diastolic)

### Remove cases where Diastolic is greater than systolic

data2 <- data2[-which(data2$Diastolic > data2$Systolic),]
plot(data2$Systolic, data2$Diastolic)


### Now lets look at height
quantile(data2$Height)

### Someone who suffers from Dwarfism has a height below 4ft 10in.
```

```
### For this study, we will look at subject's who are greater than 4ft
data2 <- data2[which(data2$Height >= 48),]

### Weight has some illogical values for adults.
### Use values of weight above 80lb
data2 <- data2[which(data2$Weight >= 80),]


### Look at boxplots again
par(mfrow=c(1,5))
for (i in 1:5){
  boxplot(contin2[,i], xlab = colnames(contin2)[i])
}

### How much was removed?
1 - nrow(data2)/nrow(data)

### Only 2.05%. This is good.

### Output new csv
write.csv(data2, "cardio3.csv")


rm(list = ls())



##################### Data Preprocessing/ Exploration ###############


### Import Data
data <- read.csv("cardio3.csv")
data <- data[,-1]
data[,c(2,7:12)] <- lapply(data[,c(2,7:12)], factor)

library(caret)
library(corrplot)
library(e1071)


################## Continuous Variable ##################



### Scatter plots
```

```
contin <- data[,c(1,3,4,5,6)]
plot(contin, col = as.numeric(data$Disease)+1,
     cex = 0.5, pch = 16)

### Corr Plots
par(mfrow=c(1,2))
corrplot(cor(contin))
corrplot(cor(contin), method = "number")

### Histograms
par(mfrow = c(2,3))
for(i in 1:5){
  hist(contin[,i], xlab = colnames(contin)[i], main = colnames(contin)[i])
}


### Box Plots
par(mfrow=c(1,5))
for(i in 1:5){
  boxplot(contin[,i], xlab = colnames(contin)[i])
}

### Skewness

apply(contin, 2, skewness)
for(i in 1:5){
  print(abs(max(contin[i])/min(contin[i])))
}

### Possible transformations

apply(contin, 2, BoxCoxTrans)

### Quadratic transformation for Age
### Square-root transformation for Height
### Inverse square-root transformation for Weight
### Inverse transformation for Systolic
### No transformation for Daistolic

continTrans <- data.frame(contin$Age ^ 2, contin$Height ^ 0.5,
                          contin$Weight ^ -0.5, contin$Systolic ^ -1,
                          contin$Diastolic)


dev.off()
```

```
plot(continTrans, col = as.numeric(data$Disease)+1,
     cex = 0.5, pch = 16)

par(mfrow = c(2,3))
for(i in 1:5){
  hist(continTrans[,i], xlab = colnames(continTrans)[i])
}
apply(continTrans, 2, skewness)

corrplot(cor(continTrans))
corrplot(cor(continTrans), method = "number")

### Not much changed.
### Try center and scaling

continPP <- preProcess(contin, method = c("scale","center"))
contin2 <- predict(continPP, contin)


dev.off()
plot(contin2, col = as.numeric(data$Disease)+1,
     cex = 0.5, pch = 16)


### Didn't change much.
### Try BoxCox, center and scaling

continPP <- preProcess(contin, method = c("scale","center","BoxCox"))
contin3 <- predict(continPP, contin)

plot(contin3, col = as.numeric(data$Disease)+1,
     cex = 0.5, pch = 16)

### Has increased the seperation of the target variable

par(mfrow = c(2,3))
for(i in 1:5){
  hist(contin3[,i], xlab = colnames(contin3)[i])
}
apply(contin3, 2, skewness)

### This looks good now

par(mfrow=c(1,2))
corrplot(cor(contin3))
```

```
corrplot(cor(contin3), method = "number")

### Correlations haven't really change.
### How much can one pair of correlated variable mess with classification?

### Use BoxCox only


################## Categorical Variable ##################

##Gender
Bgender <- barplot((table(data$Gender)),
                names.arg=c("Women","Men"),
                col = c("Pink","lightblue"),
                ylim = c(0, 50000),
                main = "Gender")
text(x=Bgender, y= table(data$Gender),
    labels=as.character(table(data$Gender)),
    pos = 3,
    col = "Black")


##Cholesterol
BCholesterol <- barplot((table(data$Cholesterol)),
                    names.arg=c("Normal", "Above normal", "Well above normal"),
                    col = c("lightgreen", "lightyellow", "red"),
                    ylim = c(0, 55000),
                    main = "Cholesterol")
text(x=BCholesterol, y= table(data$Cholesterol),
    labels=as.character(table(data$Cholesterol)),
    pos = 3,
    col = "Black")


##Glucose
BGlucose <- barplot((table(data$Glucose)),
                names.arg=c("Normal", "Above normal", "Well above normal"),
                col = c("lightgreen", "lightyellow", "red"),
                ylim = c(0, 65000),
                main = "Glucose")
text(x=BGlucose, y= table(data$Glucose),
    labels=as.character(table(data$Glucose)),
    pos = 3,
    col = "Black")
```

```
##Smoke
BSmoke <- barplot((table(data$Smoke)),
                  names.arg=c("Non-smoking", "smoking"),
                  col = c("lightgreen","gray"),
                  ylim = c(0, 70000),
                  main = "Smoking")
text(x=BSmoke, y= table(data$Smoke),
     labels=as.character(table(data$Smoke)),
     pos = 3,
     col = "Black")


##Alcohol
BAlcohol <- barplot((table(data$Alcohol)),
                    names.arg=c("Non-drinking", "drinking"),
                    col = c("lightgreen","lightgray"),
                    ylim = c(0, 70000),
                    main = "Alcohol")
text(x=BAlcohol , y= table(data$Alcohol ),
     labels=as.character(table(data$Alcohol)),
     pos = 3,
     col = "Black")


##Exercise
BExercise <- barplot((table(data$Exercise)),
                     names.arg=c("Non-exercising", "Exercising"),
                     col = c("lightgray","lightgreen"),
                     ylim = c(0, 60000),
                     main = "Exercising")
text(x=BExercise , y= table(data$Exercise),
     labels=as.character(table(data$Exercise)),
     pos = 3,
     col = "Black")


##Disease
BDisease <- barplot((table(data$Disease)),
                    names.arg=c("NO", "YES"),
                    col = c("lightgray","red"),
                    ylim = c(0, 45000),
                    main = "If the person has cardiovascular disease")
text(x=BDisease , y= table(data$Disease),
     labels=as.character(table(data$Disease)),
```

```
    pos = 3,
    col = "Black")


rm(list = ls())


################################## MODELING ######################

### Import Data
data <- read.csv("cardio3.csv")
data <- data[,-1]
data[,c(2,7:12)] <- lapply(data[,c(2,7:12)], factor)
#View(data)
#str(data)

library(caret)
library(corrplot)
library(e1071)


### Data Pre-Processing

### Remove Diastolic
data <- data[,-6]


par(mfrow=c(2,2))
hist(data$Weight, xlab = "Weight", main = "Weight Original")

### Apply transformations to Weight and Systolic
weightPP <- BoxCoxTrans(data$Weight)
WeightTrans <- predict(weightPP, data$Weight)


hist(WeightTrans, xlab = "Weight Tans", main = "Weight Transformed")

hist(data$Systolic, xlab = "Systolic", main = "Systolic Original")

systolicPP <- BoxCoxTrans(data$Systolic)
SystolicTrans <- predict(systolicPP, data$Systolic)

hist(SystolicTrans, xlab = "Systolic Trans", main = "Systolic Transformed")

### Put transformations in data
```

```
data$Weight <- WeightTrans
data$Systolic <- SystolicTrans

### Update column names
colnames(data)[c(4,5)] <- c("WeightTrans", "SystolicTrans")




par(mfrow=c(1,2))
corrplot(cor(data[,c(1,3:6)]))
corrplot(cor(data[,c(1,3:6)]), method = "number")



### Make a copy of the data.
### One will hold numeric factors
### One will hold string factors

data2 <- data




### Make catigorical variables strings
data2$Gender <- ifelse(data$Gender == 0, "Female", "Male")
data2$Alcohol <- ifelse(data$Alcohol == 0, "No", "Yes")
data2$Smoke <- ifelse(data$Smoke == 0, "No", "Yes")
data2$Exercise <- ifelse(data$Exercise == 0, "No", "Yes")
data2$Disease <- ifelse(data$Disease == 0, "No", "Yes")

data2$Cholesterol <- ifelse(data$Cholesterol == 0, "Normal",
                      ifelse(data$Cholesterol == 1, "Above", "Well Above"))
data2$Glucose <- ifelse(data$Glucose == 0, "Normal",
                  ifelse(data$Glucose == 1, "Above", "Well Above"))

data2[,c(2,6:11)] <- lapply(data2[,c(2,6:11)], factor)

### Create control function
set.seed(210)
ctrl <- trainControl(method = "LGOCV",
                 summaryFunction = twoClassSummary,
                 classProbs = TRUE,
                 savePredictions = TRUE)

### Create training and testing data
```

```
set.seed(210)
inTrain <- createDataPartition(data$Disease, p = 0.8)[[1]]
Train <- data[inTrain,]
Test <- data[-inTrain,]

Train2 <- data2[inTrain,]
Test2 <- data2[-inTrain,]



### Logistic Regression
set.seed(210)
logicTune  <- train(x = Train2[,c(1:10)], y = Train2$Disease,
                    method = "glm",
                    metric = "ROC",
                    trControl = ctrl)
logicTune

summary(logicTune)

logicPred <- predict(logicTune, Test2)
table(logicPred, Test2$Disease)

### Test error rate
mean(logicPred != Test2$Disease)

### Importance
varImp(logicTune)
plot(varImp(logicTune))

### Test ROC
library(pROC)

logicROC <- roc(response = logicTune$pred$obs,
                predictor = logicTune$pred$Yes,
                levels = rev(levels(logicTune$pred$obs)))
plot(logicROC, legacy.axes = TRUE)
auc(logicROC)

### Save results

Test_Error <- c(mean(logicPred != Test2$Disease))
Test_AUC <- c(0.7885)
Models <- c("Logistic")
```

```
### LDA
set.seed(210)
ldaTune  <- train(form = Disease~., data = Train2,
                method = "lda",
                metric = "ROC",
                trControl = ctrl)
ldaTune

ldaPred <- predict(ldaTune, Test2)
table(ldaPred, Test2$Disease)

### Test error rate
mean(ldaPred != Test2$Disease)

### Importance
varImp(ldaTune)
plot(varImp(ldaTune))

### Test ROC
library(pROC)

ldaROC <- roc(response = ldaTune$pred$obs,
             predictor = ldaTune$pred$Yes,
             levels = rev(levels(ldaTune$pred$obs)))
plot(ldaROC, legacy.axes = TRUE)
auc(ldaROC)

### Save results

Test_Error <- append(mean(ldaPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7882, Test_AUC)
Models <- append("LDA", Models)




### PLS DA
set.seed(210)
plsTune  <- train(form = Disease~., data = Train2,
                method = "pls",
                metric = "ROC",
                tuneGrid = expand.grid(.ncomp = 1:11),
                trControl = ctrl)
plsTune
```

```
plsPred <- predict(plsTune, Test2)
table(plsPred, Test2$Disease)

### Test error rate
mean(plsPred != Test2$Disease)

### Importance
varImp(plsTune)
plot(varImp(plsTune))

### Test ROC
library(pROC)

plsROC <- roc(response = plsTune$pred$obs,
            predictor = plsTune$pred$Yes,
            levels = rev(levels(plsTune$pred$obs)))
plot(plsROC, legacy.axes = TRUE)
auc(plsROC)

### Save results

Test_Error <- append(mean(plsPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.6768, Test_AUC)
Models <- append("PLSDA", Models)



### Penalized Model
memory.limit(size = 20000)

set.seed(210)
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4),
                  .lambda = seq(.01, .2, length = 15))

penTune  <- train(form = Disease~., data = Train2,
             method = "glmnet",
             tuneGrid = glmnGrid,
             metric = "ROC",
             trControl = ctrl)
penTune

penPred <- predict(penTune, Test2)
table(penPred, Test2$Disease)
```

```
### Test error rate
mean(penPred != Test2$Disease)

### Importance
varImp(penTune)
plot(varImp(penTune))

### Test ROC
library(pROC)

penROC <- roc(response = penTune$pred$obs,
          predictor = penTune$pred$Yes,
          levels = rev(levels(penTune$pred$obs)))
plot(penROC, legacy.axes = TRUE)
auc(penROC)

### Save results

Test_Error <- append(mean(penPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7833, Test_AUC)
Models <- append("Pen", Models)



### Nearest Shrunken Centroids
nscGrid <- data.frame(.threshold = 0:25)
nscTune <- train(form = Disease~., data = Train2,
            method = "pam",
            tuneGrid = nscGrid,
            metric = "ROC",
            trControl = ctrl)
nscTune

nscPred <- predict(nscTune, Test2)
table(nscPred, Test2$Disease)

### Test error rate
mean(nscPred != Test2$Disease)

### Importance
varImp(nscTune)
plot(varImp(nscTune))

### Test ROC
library(pROC)
```

```
nscROC <- roc(response = nscTune$pred$obs,
          predictor = nscTune$pred$Yes,
          levels = rev(levels(nscTune$pred$obs)))
plot(nscROC, legacy.axes = TRUE)
auc(nscROC)

### Save results

Test_Error <- append(mean(nscPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.6515, Test_AUC)
Models <- append("NSC", Models)



### QDA
set.seed(210)
qdaTune <- train(form = Disease~., data = Train2,
            method = "qda",
            metric = "ROC",
            trControl = ctrl)
qdaTune


qdaPred <- predict(qdaTune, Test2)
table(qdaPred, Test2$Disease)

### Test error rate
mean(qdaPred != Test2$Disease)

### Importance
varImp(qdaTune)
plot(varImp(qdaTune))

### Test ROC
library(pROC)

qdaROC <- roc(response = qdaTune$pred$obs,
          predictor = qdaTune$pred$Yes,
          levels = rev(levels(qdaTune$pred$obs)))
plot(qdaROC, legacy.axes = TRUE)
auc(qdaROC)

### Save results
```

```
Test_Error <- append(mean(qdaPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7561, Test_AUC)
Models <- append("QDA", Models)


### MDA
set.seed(210)
mdaTune <- train(form = Disease~., data = Train2,
            method = "mda",
            metric = "ROC",
            tuneGrid = expand.grid(.subclasses = 1:10),
            trControl = ctrl)
mdaTune



mdaPred <- predict(mdaTune, Test2)
table(mdaPred, Test2$Disease)

### Test error rate
mean(mdaPred != Test2$Disease)

### Importance
varImp(mdaTune)
plot(varImp(mdaTune))

### Test ROC
library(pROC)

mdaROC <- roc(response = mdaTune$pred$obs,
          predictor = mdaTune$pred$Yes,
          levels = rev(levels(mdaTune$pred$obs)))
plot(mdaROC, legacy.axes = TRUE)
auc(mdaROC)

### Save results

Test_Error <- append(mean(mdaPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7855, Test_AUC)
Models <- append("MDA", Models)



### FDA
set.seed(210)
```

```
fdaTune <- train(form = Disease~., data = Train2,
                 method = "fda",
                 metric = "ROC",
                 trControl = ctrl)
fdaTune

fdaPred <- predict(fdaTune, Test2)
table(fdaPred, Test2$Disease)

### Test error rate
mean(fdaPred != Test2$Disease)

### Importance
varImp(fdaTune)
plot(varImp(fdaTune))

### Test ROC
library(pROC)

fdaROC <- roc(response = fdaTune$pred$obs,
              predictor = fdaTune$pred$Yes,
              levels = rev(levels(fdaTune$pred$obs)))
plot(fdaROC, legacy.axes = TRUE)
auc(fdaROC)

### Save results

Test_Error <- append(mean(fdaPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7785, Test_AUC)
Models <- append("FDA", Models)


### Naive Bayes
set.seed(210)
nbTune <- train(x = Train2[,c(1:10)], y = Train2$Disease,
                method = "nb",
                metric = "ROC",
                trControl = ctrl)
nbTune

nbPred <- predict(nbTune, Test2)
table(nbPred, Test2$Disease)

### Test error rate
mean(nbPred != Test2$Disease)
```

```
### Importance
varImp(nbTune)
plot(varImp(nbTune))

### Test ROC
library(pROC)

nbROC <- roc(response = nbTune$pred$obs,
            predictor = nbTune$pred$Yes,
            levels = rev(levels(nbTune$pred$obs)))
plot(nbROC, legacy.axes = TRUE)
auc(nbROC)

### Save results

Test_Error <- append(mean(nbPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7818, Test_AUC)
Models <- append("NB", Models)


### KNN
set.seed(210)
knnTune <- train(form = Disease~., data = Train2,
                method = "knn",
                metric = "ROC",
                tuneGrid = data.frame(.k = c(1:100)),
                trControl = ctrl)
knnTune

knnPred <- predict(knnTune, Test2)
table(knnPred, Test2$Disease)

### Test error rate
mean(knnPred != Test2$Disease)

### Importance
varImp(knnTune)
plot(varImp(knnTune))

### Test ROC
library(pROC)

knnROC <- roc(response = knnTune$pred$obs,
             predictor = knnTune$pred$Yes,
```

```
              levels = rev(levels(knnTune$pred$obs)))
plot(knnROC, legacy.axes = TRUE)
auc(knnROC)

### Save results

Test_Error <- append(mean(knnPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.6518, Test_AUC)
Models <- append("KNN", Models)


### Neural Network
set.seed(210)
nnetGrid <- expand.grid(.size = 1:10,
                    .decay = c(0, .01, .1, 0.5))
maxSize <- max(nnetGrid$.size)
numWts <-200
set.seed(210)
nnetTune <- train(form = Disease~., data = Train2,
               method = "nnet",
               metric = "ROC",
               tuneGrid = nnetGrid,
               trace = FALSE,
               maxit = 1000,
               MaxNWts = numWts,
               ## ctrl was defined in the previous chapter
               trControl = ctrl)


nnetTune

nnetPred <- predict(nnetTune, Test2)
table(nnetPred, Test2$Disease)

### Test error rate
mean(nnetPred != Test2$Disease)

### Importance
varImp(nnetTune)
plot(varImp(nnetTune))

### Test ROC
library(pROC)

nnetROC <- roc(response = nnetTune$pred$obs,
```

```
            predictor = nnetTune$pred$Yes,
            levels = rev(levels(nnetTune$pred$obs)))
plot(nnetROC, legacy.axes = TRUE)
auc(nnetROC)

### Save results

Test_Error <- append(mean(nnetPred != Test2$Disease), Test_Error)
Test_AUC <- append(0.7636, Test_AUC)
Models <- append("NNet", Models)



############## Plots ###################



### Make plots comparing fit statistics

### Combine into one data frame
testResults <- data.frame(Test_Error,Test_AUC,Models)



### Test error rate, sorted
OrderErr <- testResults[order(testResults$Test_Error),]


plot(OrderErr$Test_Err, c(1:11), yaxt = "n",
    main = "Test Error Rate by Model",
    xlab = "Test Error Rate",
    ylab = "Model",
    pch = 16)
axis(2, at = c(1:11), labels = as.character(OrderErr$Models), las = 2,
    cex.axis = 0.8)
text(OrderErr$Test_Err[1:9], c(1:9),
    labels = as.character(round(OrderErr$Test_Err[1:9], 5)),
    adj = -0.2)
text(OrderErr$Test_Err[10:11], c(10:11),
    labels = as.character(round(OrderErr$Test_Err[10:11], 5)),
    adj = 1.2)
```

```
### AUC, sorted
OrderAUC <- testResults[order(testResults$Test_AUC),]


plot(OrderAUC$Test_AUC, c(1:11), yaxt = "n",
    main = "Test AUC by Model",
    xlab = "Area Under the Curve",
    ylab = "Model",
    pch = 16)
axis(2, at = c(1:11), labels = as.character(OrderAUC$Models), las = 2,
    cex.axis = 0.8)
text(OrderAUC$Test_AUC[1:5], c(1:5),
    labels = as.character(round(OrderAUC$Test_AUC[1:5], 5)),
    adj = -0.2)
text(OrderAUC$Test_AUC[6:11], c(6:11),
    labels = as.character(round(OrderAUC$Test_AUC[6:11], 5)),
    adj = 1.2)



### Make plot of ROC curves
plot(logicROC, legacy.axis = TRUE, lty = 5, lwd = 3.5, main = "ROC Curves by Model")
lines(ldaROC, col = "red", lty = 3, lwd = 4)
lines(plsROC, col = "blue", lty = 5, lwd = 3.5)
lines(penROC, col = "green", lty = 3, lwd = 4)
lines(nscROC, col = "magenta", lty = 5, lwd = 3.5)
lines(qdaROC, col = "purple", lty = 3, lwd = 4)
lines(mdaROC, col = "orange", lty = 5, lwd = 3.5)
lines(fdaROC, col = "forestgreen", lty = 5, lwd = 3.5)
lines(nbROC, col = "hotpink", lty = 3, lwd = 4)
lines(knnROC, col = "grey50", lty = 3, lwd = 4)
lines(nnetROC, col = "aquamarine", lty = 3, lwd = 4)

legend("topleft",
      legend = as.character(rev(testResults$Models)),
      lty = c(5,3,5,3,5,3,5,5,3,3,3),
      lwd = c(3.5,4,3.5,4,3.5,4,3.5,3.5,4,4,4),
      col = c("black","red","blue","green","magenta","purple",
            "orange","forestgreen","hotpink","grey50","aquamarine"))



library(lattice)
resamp = resamples(list(Logistic = logicTune, LDA = ldaTune, PLSDA = plsTune,
                  Pen = penTune, NSC = nscTune, QDA = qdaTune,
```

```
                    MDA = mdaTune, FDA = fdaTune, NB = nbTune,
                    KNN = knnTune, NNet = nnetTune))

dotplot(resamp, metric = "ROC")

ModelDiff <- diff(resamp)
ModelDiff$statistics$ROC
```