Final Project

Group 1

2/25/2022

Data Description

```
#Pima Indians Diabetes Dataset Found Inside Caret Function
data(PimaIndiansDiabetes)# There are two of them, versions
df <- PimaIndiansDiabetes
# df
str(df)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

```
#Summary Statistics
summary(df)
```

```
##     pregnant         glucose         pressure         triceps     
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00  
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00  
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54  
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00  
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
##     insulin           mass          pedigree           age        diabetes
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   neg:500 
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00   pos:268 
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00           
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24           
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00           
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00           
```

Data Preparation

- No near zero variance predictors. No action necessary.
- No NA values. No action necessary.
- There are a significant number of 0 Values

```r
#Confirmation of No Near Zero Variance for Predictor Variables
predictors <- PimaIndiansDiabetes[ , -(9)]
print(nearZeroVar(predictors))
```

```
## integer(0)
```

```r
#Check for missing values
#Confirmed No Missing Values
sapply(df, function(x) sum(is.na(x)))
```

```
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##        0        0        0        0        0        0        0        0
## diabetes
##        0
```

Process Zero values

Logic Behind 6 Zero Markers * pregnant - not all woman have a baby, likely 0 is a true value, will keep predictor variable * glucose - only 5 values are missing, will keep predictor variable, will fill zeros with bag Impute. * pressure - only 35 values are missing, will keep predictor variable, will fill zeros with bag Impute. * triceps - approximately 30% of the data contains 0 values. Initial predictions show that this predictor does not help the models. It will be dropped. * insulin - almost 50% of the data has 0 values, will keep predictor variable, will fill zeros with bag Impute. * mass - only 11 values are missing, will fill zeros with bag Impute.

```r
# drop triceps as this does not seem to improve the predictions
df <- df[,-4]
# replace zeros with NA
df[df == 0] <- NA
#Return Pregnant NA back to 0(zer0)
df$pregnant[is.na(df$pregnant)] <- 0
# Transform all feature to dummy variables.
dummy.vars <- dummyVars(~ ., data = df)
train.dummy <- predict(dummy.vars, df)
#impute
pre.process <- preProcess(train.dummy, method = "bagImpute")
imputed.data <- predict(pre.process, train.dummy)
#Replace zeros with imputed dummy variables
df$glucose <- imputed.data[,2]
df$pressure <- imputed.data[,3]
df$insulin <- imputed.data[,4]
df$mass <- imputed.data[,5]
#Check to make sure that it worked
zerobycolumn <-colSums(df==0)
summary(df)
```

```
##     pregnant        glucose        pressure        insulin
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 14.0
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.: 90.0
##  Median : 3.000   Median :117.0   Median : 72.00   Median :129.1
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.33   Mean   :154.3
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:190.4
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :846.0
##       mass          pedigree            age        diabetes
##  Min.   :18.20   Min.   :0.0780   Min.   :21.00   neg:500
##  1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00   pos:268
##  Median :32.30   Median :0.3725   Median :29.00
##  Mean   :32.46   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Skewness

Generally values between -1 and 1 are acceptable. Insulin, Age and Pedigree have skewness values beyond these thresholds. Using the log of these functions removes the skewness. *Note doesn't boxcox correct for this?

```r
#skewness
skewness(df$pregnant) #0.898
```

```
## [1] 0.8981549
```

```r
skewness(df$glucose) #0.529
```

```
## [1] 0.5294202
```

```r
skewness(df$pressure) #0.145
```

```
## [1] 0.1504179
```

```r
skewness(df$insulin) #2.026
```

```
## [1] 2.166369
```

```r
skewness(df$mass) #0.595
```

```
## [1] 0.5942641
```

```r
skewness(df$pedigree) #1.912
```

```
## [1] 1.912418
```

```r
skewness(df$age) #1.125
```

```
## [1] 1.125188
```

```r
skewness(log(df$age))
```

```
## [1] 0.5993976
```

```r
skewness(log(df$pedigree))
```

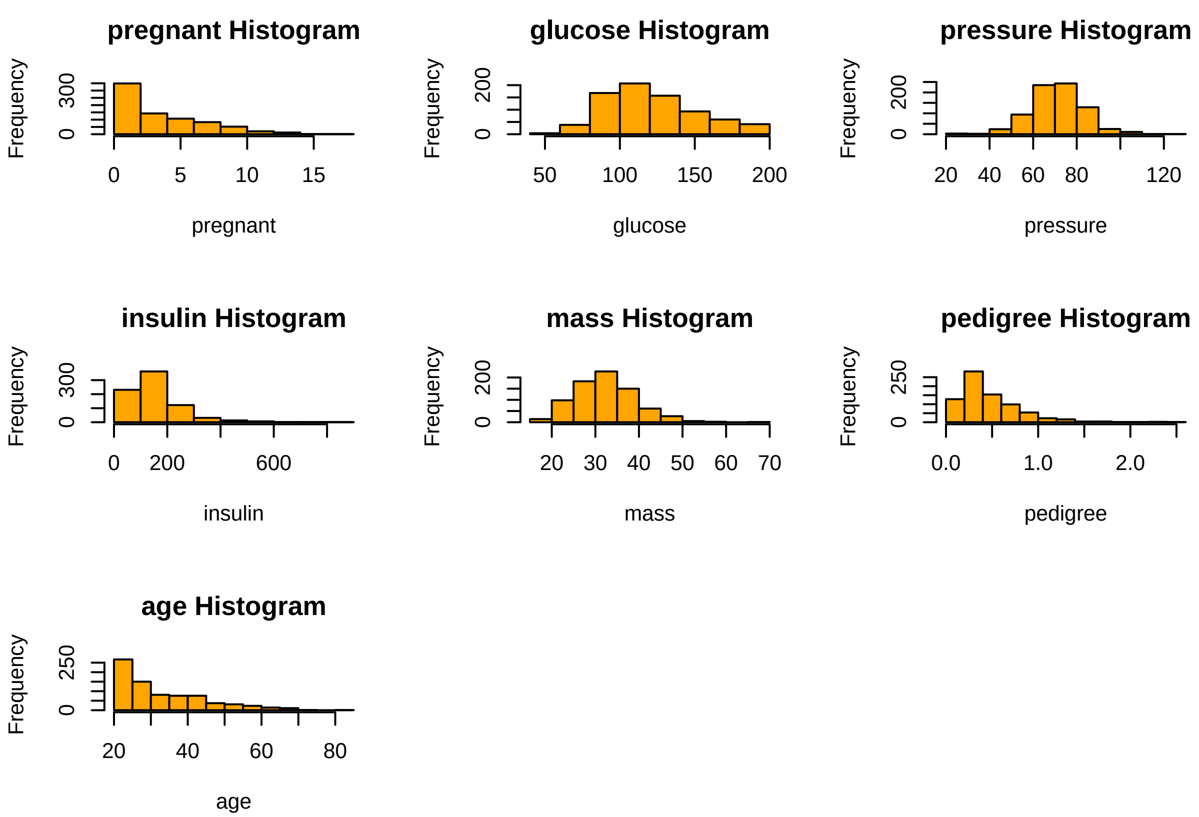
```
## [1] 0.1137321
```

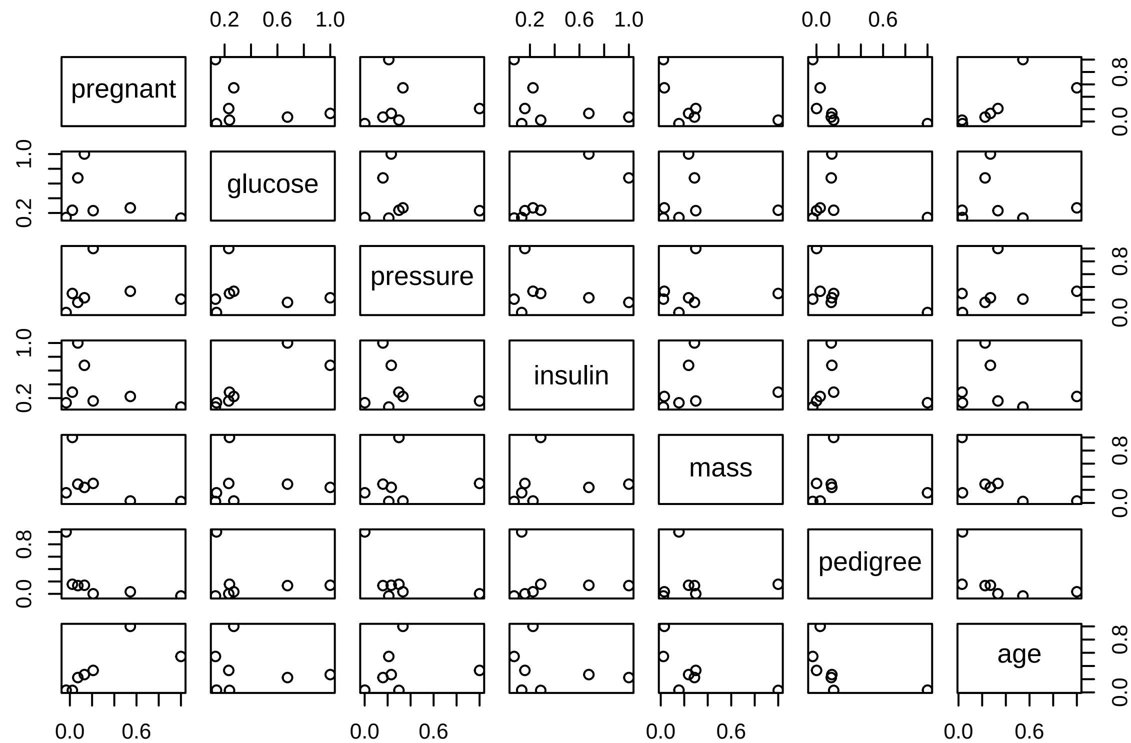
```r
skewness(log(df$insulin))
```

```
## [1] -0.2110642
```

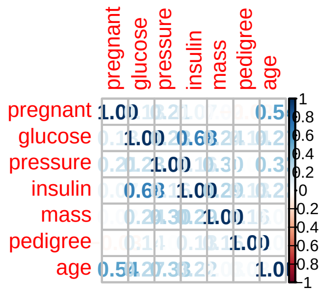
Graphical Review of data

```r
#Histograms of Diabetes: Predictor Variables
n <-df[,1:(ncol(df)-1)] #Predictors are variables 1-8
par(mfrow = c(3,3)) #Histograms will be 3x3
for (i in 1:ncol(n))
{hist(n[ ,i], xlab = names(n[i]), main = paste(names(n[i]), "Histogram"), col="orange")
}
#Correlation Plot of Diabetes: Predictor Variables
x <- cor(df[1:ncol(df)-1])
pairs(x)
```

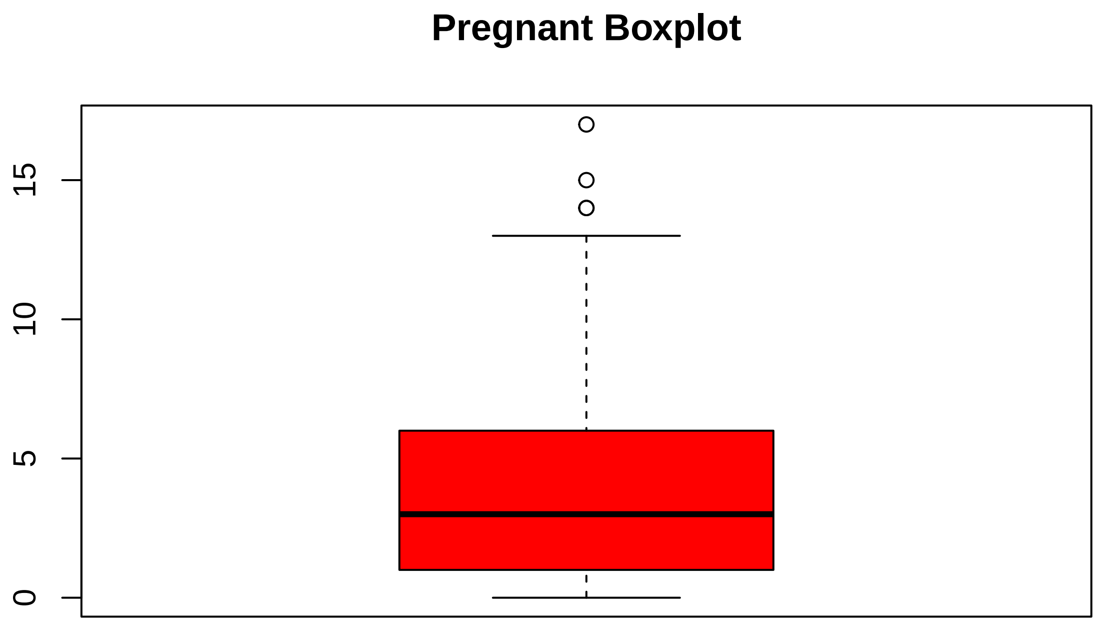




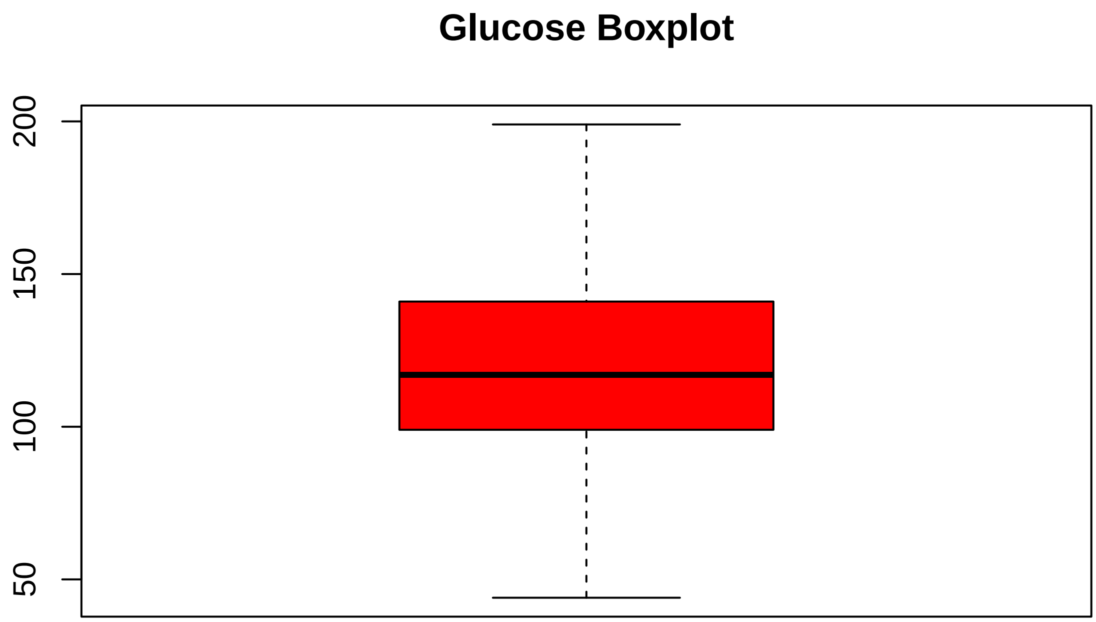

```
corrplot(x, method="number")
```



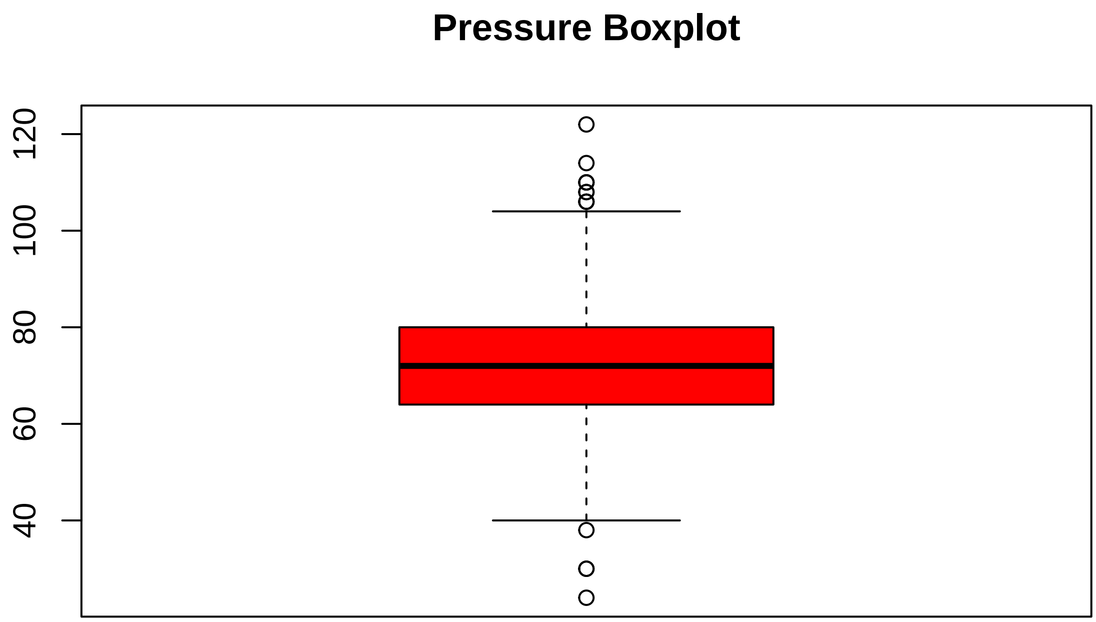

```
#Box Plots of Diabetes: Predictor Variables
boxplot(df$pregnant, main = "Pregnant Boxplot", col = "red")
```



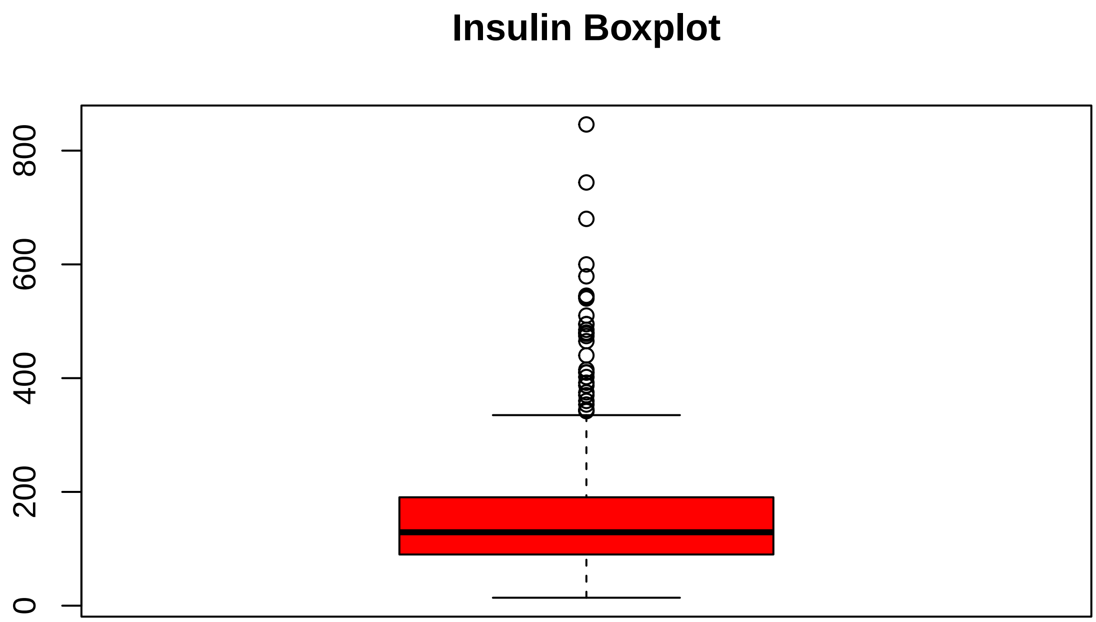

```
boxplot(df$glucose, main = "Glucose Boxplot", col = "red")
```



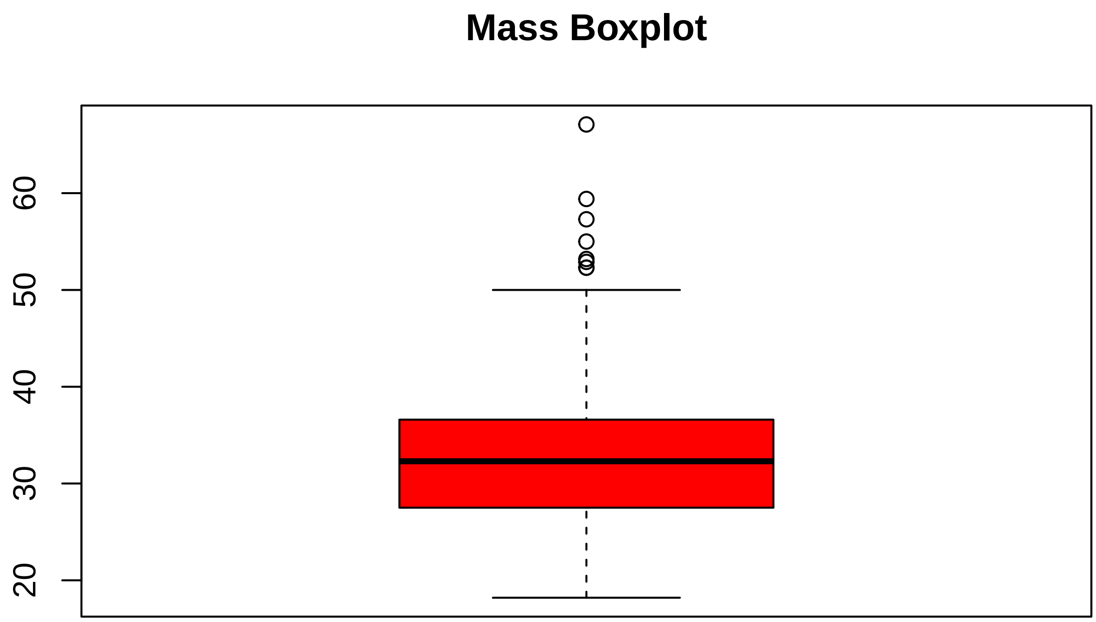

```
boxplot(df$pressure, main = "Pressure Boxplot", col = "red")
```



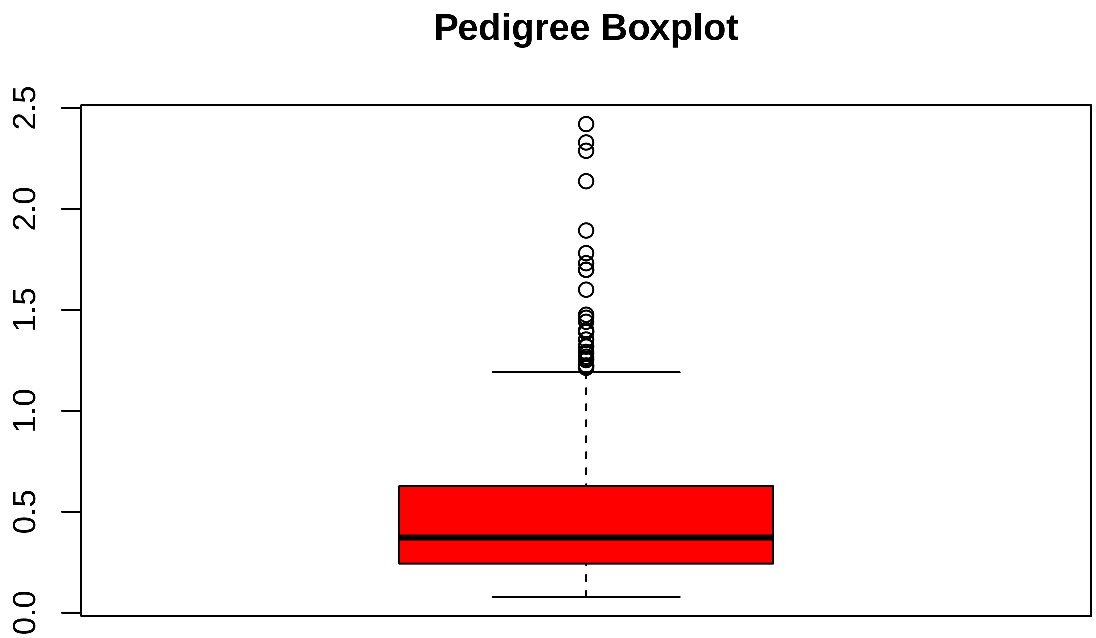

```r
#boxplot(df$triceps, main = "Triceps Boxplot", col = "red")
boxplot(df$insulin, main = "Insulin Boxplot", col = "red")
```



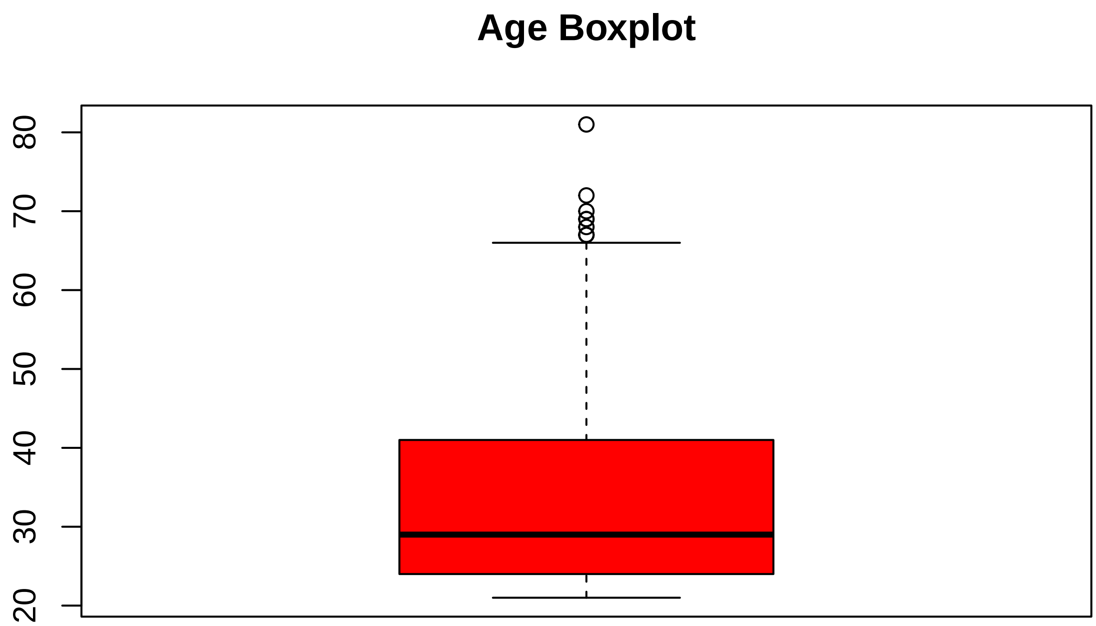

```
boxplot(df$mass, main = "Mass Boxplot", col = "red")
```




```r
boxplot(df$pedigree, main = "Pedigree Boxplot", col = "red")
```




```
boxplot(df$age, main = "Age Boxplot", col = "red")
```



Data Splitting

Data will be split 80%/20% train/testing.

```
#Split Training and Test Data, 80/20
set.seed(1)
split <- caret::createDataPartition(y = df$diabetes, times = 1, p = 0.8, list = FALSE)
#Train_data Split, 80%
train_data <- df[split,]
#Test_data Split, 20%
test_data <- df[-split,]
#Summary Statistics
summary(train_data)
```

```
##     pregnant         glucose       pressure         insulin
##  Min.   : 0.000   Min.   : 44   Min.   : 24.00   Min.   : 14.0
##  1st Qu.: 1.000   1st Qu.:100   1st Qu.: 64.00   1st Qu.: 90.0
##  Median : 3.000   Median :118   Median : 72.00   Median :129.1
##  Mean   : 3.839   Mean   :122   Mean   : 72.39   Mean   :153.8
##  3rd Qu.: 6.000   3rd Qu.:140   3rd Qu.: 80.00   3rd Qu.:188.8
##  Max.   :17.000   Max.   :199   Max.   :114.00   Max.   :846.0
##       mass          pedigree            age          diabetes
##  Min.   :18.20   Min.   :0.0780   Min.   :21.00   neg:400
##  1st Qu.:27.55   1st Qu.:0.2440   1st Qu.:24.00   pos:215
```

```
##  Median :32.40   Median :0.3700   Median :29.00
##  Mean   :32.54   Mean   :0.4705   Mean   :33.43
##  3rd Qu.:36.60   3rd Qu.:0.6250   3rd Qu.:41.00
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Model Training

The following models will be trained on the training data.
Logistic Regression

```
##################Training Models##########################
#Logistic Regression: Training Model
#No Tuning Parameters for Simple Logistic Regression
set.seed(1)
lr_train_data <- caret::train(diabetes ~., data = train_data,
                          method = "glm",
                          metric = "ROC",
                          tuneLength = 10,
                          trControl = trainControl(method = "cv", number = 10,
                                                   classProbs = T, summaryFunction = twoClassSummary),
                          preProcess = c("center","scale", "BoxCox"))
lr_train_data$preProcess
```

```
## Created from 615 samples and 7 variables
##
## Pre-processing:
##   - Box-Cox transformation (6)
##   - centered (7)
##   - ignored (0)
##   - scaled (7)
##
## Lambda estimates for Box-Cox transformation:
## 0.1, 1, 0.1, 0, -0.1, -1.1
```

```
lr_train_data
```

```
## Generalized Linear Model
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), Box-Cox transformation (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results:
##
##   ROC        Sens   Spec
##   0.8522294  0.865  0.6088745
```

```r
summary(lr_train_data)
```

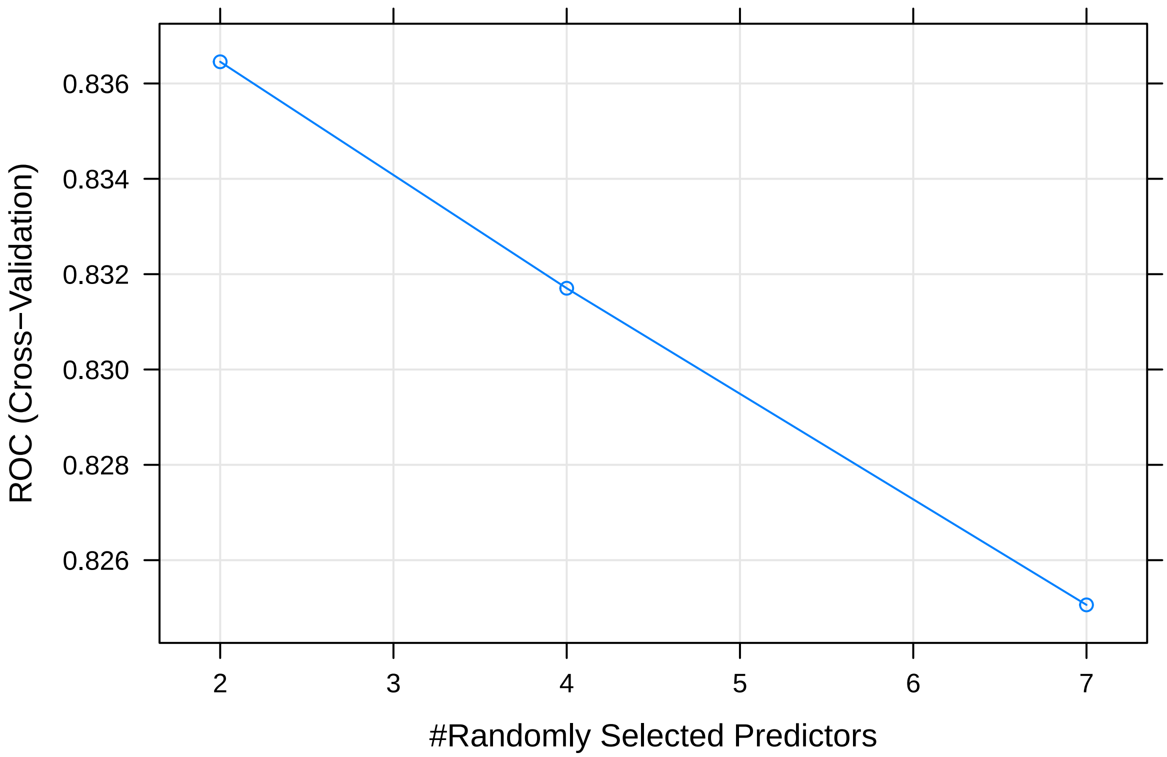
```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5121  -0.6642  -0.3248   0.6521   2.5776
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.00782    0.11901  -8.468  < 2e-16 ***
## pregnant     0.23848    0.12988   1.836  0.06634 .
## glucose      1.17190    0.16573   7.071 1.54e-12 ***
## pressure    -0.19035    0.11855  -1.606  0.10835
## insulin      0.02944    0.15848   0.186  0.85263
## mass         0.77956    0.13231   5.892 3.81e-09 ***
## pedigree     0.34706    0.11129   3.118  0.00182 **
## age          0.51377    0.14565   3.527  0.00042 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 796.05  on 614  degrees of freedom
## Residual deviance: 538.57  on 607  degrees of freedom
## AIC: 554.57
##
## Number of Fisher Scoring iterations: 5
```

```r
#Random Forest: Training Model
set.seed(1)
rf_train_data <- caret::train(diabetes ~., data = train_data,
                             method = "rf",
                             metric = "ROC",
                             trControl = trainControl(method = "cv", number = 10,
                                                      classProbs = T, summaryFunction = twoClassSummary)
                             preProcess = c("center","scale"))
rf_train_data
```

```
## Random Forest
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens    Spec
```

```
##   2     0.8364556  0.8475  0.6181818
##   4     0.8317045  0.8475  0.6415584
##   7     0.8250622  0.8475  0.6277056
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_train_data)
```

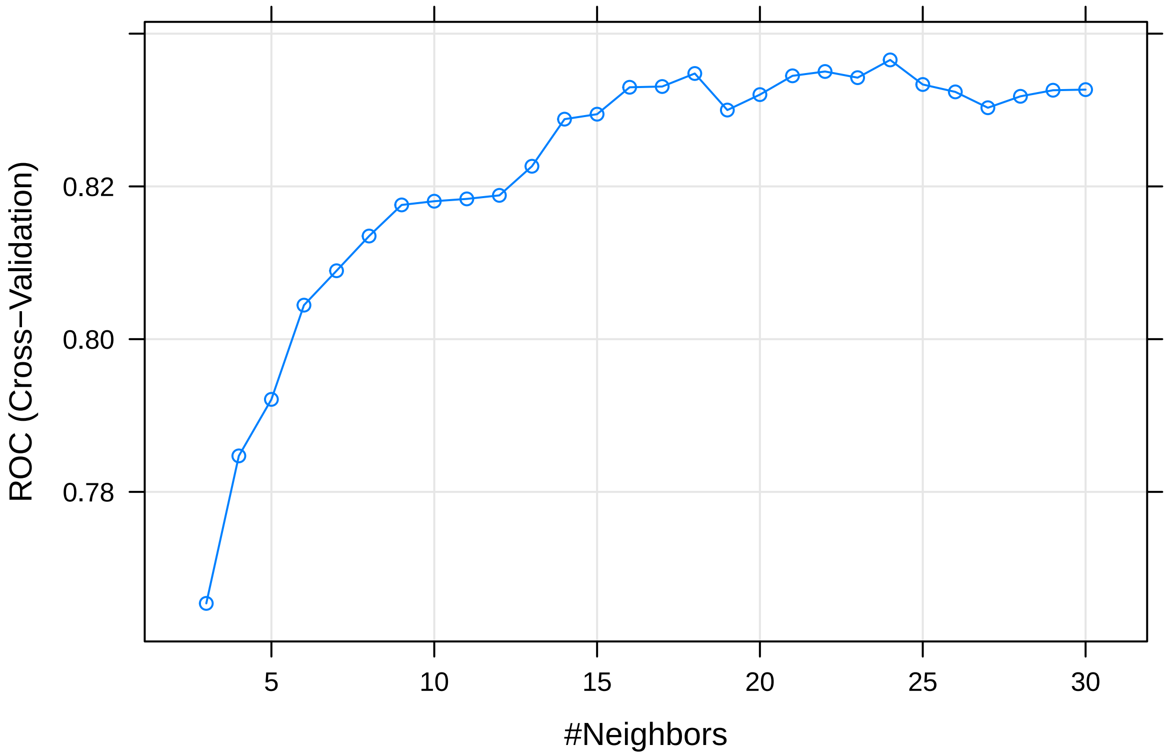



```
FinalTree = rf_train_data$finalModel$importance.mode
#K Nearest Neighbor: Training Model
set.seed(1)
knn_train_data <- caret::train(diabetes ~., data = train_data,
                          method = "knn",
                          metric = "ROC",
                          tuneGrid = expand.grid(.k = c(3:30)),
                          trControl = trainControl(method = "cv", number = 10,
                                                   classProbs = T, summaryFunction = twoClassSummary),
                          preProcess = c("center","scale"))
knn_train_data
```

```
## k-Nearest Neighbors
##
## 615 samples
##   7 predictor
```

```
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens    Spec
##    3  0.7654004  0.8025  0.6004329
##    4  0.7847159  0.8050  0.5722944
##    5  0.7921158  0.8275  0.5997835
##    6  0.8044508  0.8350  0.6138528
##    7  0.8089556  0.8225  0.6088745
##    8  0.8135011  0.8500  0.5997835
##    9  0.8175758  0.8500  0.5857143
##   10  0.8180601  0.8450  0.6086580
##   11  0.8183658  0.8500  0.5623377
##   12  0.8188258  0.8525  0.5813853
##   13  0.8226380  0.8500  0.5902597
##   14  0.8288068  0.8525  0.5720779
##   15  0.8294562  0.8500  0.5954545
##   16  0.8329816  0.8600  0.5950216
##   17  0.8330790  0.8675  0.5764069
##   18  0.8347835  0.8725  0.5948052
##   19  0.8299973  0.8700  0.5809524
##   20  0.8320184  0.8700  0.5759740
##   21  0.8344805  0.8750  0.5857143
##   22  0.8350460  0.8825  0.5898268
##   23  0.8342370  0.8775  0.5943723
##   24  0.8365639  0.8850  0.5852814
##   25  0.8333442  0.8775  0.5850649
##   26  0.8323755  0.8750  0.5850649
##   27  0.8303003  0.8725  0.5850649
##   28  0.8317965  0.8750  0.5616883
##   29  0.8325947  0.8775  0.5571429
##   30  0.8326732  0.8775  0.5614719
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 24.
```

```
plot(knn_train_data)
```




```r
#Classification and Regression Trees (CART): Training Model
set.seed(1)
cart_train_data <- caret::train(diabetes ~., data = train_data,
                                method = "rpart",
                                metric = "ROC",
                                tuneLength = 20,
                                trControl = trainControl(method = "cv", number = 10,
                                                         classProbs = TRUE, summaryFunction = twoClassSumma
                                preProcess = c("center","scale"))
cart_train_data
```

```
## CART
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens    Spec
##   0.00000000  0.8087500  0.8500  0.6140693
##   0.01640147  0.7892857  0.8400  0.6469697
```

```
##   0.03280294  0.7282197  0.8425  0.5621212
##   0.04920441  0.6888366  0.8925  0.4837662
##   0.06560588  0.6888366  0.8925  0.4837662
##   0.08200734  0.6888366  0.8925  0.4837662
##   0.09840881  0.6888366  0.8925  0.4837662
##   0.11481028  0.6888366  0.8925  0.4837662
##   0.13121175  0.6832413  0.8750  0.4885281
##   0.14761322  0.6832413  0.8750  0.4885281
##   0.16401469  0.6832413  0.8750  0.4885281
##   0.18041616  0.6813095  0.8650  0.4976190
##   0.19681763  0.6813095  0.8650  0.4976190
##   0.21321909  0.6813095  0.8650  0.4976190
##   0.22962056  0.6634686  0.8975  0.4294372
##   0.24602203  0.6634686  0.8975  0.4294372
##   0.26242350  0.6415043  0.9250  0.3580087
##   0.27882497  0.6415043  0.9250  0.3580087
##   0.29522644  0.6415043  0.9250  0.3580087
##   0.31162791  0.5747565  0.9625  0.1870130
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.
```

```
FinalTree = cart_train_data$finalModel
rpartTree = as.party(FinalTree)
dev.new()
plot(rpartTree)
#Neural Net
registerDoParallel(cores=7)
nnetGrid <- expand.grid(.decay = c(0, 0.01, 0.1),
                        .size = c(1:10),
                        .bag = FALSE
)
set.seed(1)
nnet_train_data <- caret::train(diabetes ~., data = train_data,
                                method = "avNNet",
                                tuneGrid = nnetGrid,
                                metric = "ROC",
                                trControl = trainControl(method = "cv", number = 10,
                                                         classProbs = TRUE, summaryFunction = twoClassSu
                                preProcess = c("center","scale"),
                                linout = TRUE,
                                trace = FALSE,
                                MaxNWts = 10 * (ncol(train_data) + 1) + 10 + 1,
                                maxit = 500)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```r
nnet_train_data
```

```
## Model Averaged Neural Network
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   decay  size  ROC        Sens    Spec
##   0.00    1    0.8447078  0.8750  0.5945887
##   0.00    2    0.8439989  0.8625  0.5950216
##   0.00    3    0.8370400  0.8550  0.6088745
##   0.00    4    0.8288636  0.8375  0.6138528
##   0.00    5    0.8272186  0.8500  0.6274892
##   0.00    6    0.8222998  0.8525  0.6038961
##   0.00    7    0.8101948  0.8375  0.6417749
##   0.00    8    0.8069913  0.8400  0.5997835
##   0.00    9    0.8094697  0.8150  0.5764069
##   0.00   10          NaN     NaN        NaN
##   0.01    1    0.8442478  0.8750  0.5991342
##   0.01    2    0.8465747  0.8750  0.5902597
##   0.01    3    0.8395400  0.8625  0.5621212
##   0.01    4    0.8347781  0.8475  0.6000000
##   0.01    5    0.8349405  0.8425  0.5906926
##   0.01    6    0.8246320  0.8425  0.6093074
##   0.01    7    0.8237554  0.8475  0.6138528
##   0.01    8    0.8231385  0.8350  0.6041126
##   0.01    9    0.8124188  0.8275  0.5627706
##   0.01   10          NaN     NaN        NaN
##   0.10    1    0.8444805  0.8725  0.6038961
##   0.10    2    0.8448106  0.8700  0.5811688
##   0.10    3    0.8438907  0.8450  0.5993506
##   0.10    4    0.8422240  0.8500  0.5764069
##   0.10    5    0.8326028  0.8525  0.5854978
##   0.10    6    0.8261255  0.8400  0.5720779
##   0.10    7    0.8294968  0.8350  0.5766234
##   0.10    8    0.8205303  0.8425  0.5857143
##   0.10    9    0.8091613  0.8200  0.5854978
##   0.10   10          NaN     NaN        NaN
##
## Tuning parameter 'bag' was held constant at a value of FALSE
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were size = 2, decay = 0.01 and bag = FALSE.
```

```r
plot(nnet_train_data)
################# Support Vector Machines ####################
set.seed(1)
svmFit <- train(diabetes ~., data = train_data,
```

```r
                method = "svmRadial",
                metric = "ROC",
                tuneLength = 14,
                preProcess = c("center","scale", "BoxCox"),
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
svmFit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), Box-Cox transformation (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   C        ROC        Sens    Spec
##      0.25  0.8431061  0.8725  0.5530303
##      0.50  0.8448052  0.8700  0.5673160
##      1.00  0.8432305  0.8650  0.5718615
##      2.00  0.8351353  0.8550  0.5716450
##      4.00  0.8219589  0.8575  0.5480519
##      8.00  0.7946537  0.8625  0.4971861
##     16.00  0.7742424  0.8600  0.4731602
##     32.00  0.7561039  0.8625  0.4554113
##     64.00  0.7352814  0.8575  0.3995671
##    128.00  0.7301786  0.8750  0.3205628
##    256.00  0.7323268  0.8825  0.3103896
##    512.00  0.7306872  0.8950  0.3106061
##   1024.00  0.7290584  0.8800  0.3346320
##   2048.00  0.7337338  0.8875  0.3577922
##
## Tuning parameter 'sigma' was held constant at a value of 0.1232676
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1232676 and C = 0.5.
```

```r
plot(svmFit)
################# Boosted #####################
gbmGrid <- expand.grid(.interaction.depth = seq(1, 7, by = 2),
                       .n.trees = seq(100, 1000, by = 50),
                       .shrinkage = c(0.01, 0.1),
                       .n.minobsinnode = 10)
set.seed(1)
gbmFit <- train(diabetes ~., data = train_data,
                method = "gbm",
                tuneGrid = gbmGrid,
                preProcess = c("center","scale"),
                verbose = FALSE,
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```r
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  ROC        Sens    Spec
##   0.01       1                   100     0.8273187  0.9400  0.3811688
##   0.01       1                   150     0.8333496  0.9225  0.4277056
##   0.01       1                   200     0.8388555  0.9150  0.4465368
##   0.01       1                   250     0.8421266  0.9050  0.5064935
##   0.01       1                   300     0.8443398  0.8975  0.5344156
##   0.01       1                   350     0.8464610  0.8925  0.5480519
##   0.01       1                   400     0.8470833  0.8900  0.5525974
##   0.01       1                   450     0.8477056  0.8875  0.5621212
##   0.01       1                   500     0.8486364  0.8875  0.5668831
##   0.01       1                   550     0.8487608  0.8825  0.5714286
##   0.01       1                   600     0.8490043  0.8750  0.5714286
##   0.01       1                   650     0.8502868  0.8700  0.5714286
##   0.01       1                   700     0.8501948  0.8700  0.5714286
##   0.01       1                   750     0.8512500  0.8725  0.5807359
##   0.01       1                   800     0.8511688  0.8650  0.5854978
##   0.01       1                   850     0.8513799  0.8625  0.5807359
##   0.01       1                   900     0.8512933  0.8625  0.5764069
##   0.01       1                   950     0.8509470  0.8575  0.5857143
##   0.01       1                  1000     0.8503788  0.8550  0.5720779
##   0.01       3                   100     0.8513068  0.9100  0.4926407
##   0.01       3                   150     0.8512500  0.8975  0.5203463
##   0.01       3                   200     0.8516613  0.8850  0.5484848
##   0.01       3                   250     0.8530465  0.8800  0.5530303
##   0.01       3                   300     0.8518939  0.8775  0.5764069
##   0.01       3                   350     0.8503030  0.8700  0.5952381
##   0.01       3                   400     0.8513528  0.8675  0.6093074
##   0.01       3                   450     0.8503734  0.8625  0.6093074
##   0.01       3                   500     0.8506548  0.8600  0.6093074
##   0.01       3                   550     0.8514827  0.8625  0.6045455
##   0.01       3                   600     0.8502165  0.8625  0.6136364
##   0.01       3                   650     0.8500162  0.8600  0.6188312
##   0.01       3                   700     0.8504762  0.8575  0.6186147
##   0.01       3                   750     0.8493182  0.8575  0.6138528
##   0.01       3                   800     0.8484740  0.8600  0.6186147
##   0.01       3                   850     0.8468615  0.8600  0.6140693
##   0.01       3                   900     0.8457900  0.8600  0.6186147
##   0.01       3                   950     0.8452435  0.8575  0.6183983
```

```
##   0.01     3                 1000     0.8442208  0.8575  0.6231602
##   0.01     5                  100     0.8504437  0.9025  0.5017316
##   0.01     5                  150     0.8508766  0.8875  0.5437229
##   0.01     5                  200     0.8503950  0.8750  0.5718615
##   0.01     5                  250     0.8477219  0.8725  0.5857143
##   0.01     5                  300     0.8466937  0.8675  0.5995671
##   0.01     5                  350     0.8456710  0.8650  0.6041126
##   0.01     5                  400     0.8460606  0.8600  0.6179654
##   0.01     5                  450     0.8451299  0.8625  0.6367965
##   0.01     5                  500     0.8440693  0.8575  0.6322511
##   0.01     5                  550     0.8425649  0.8550  0.6324675
##   0.01     5                  600     0.8412392  0.8525  0.6279221
##   0.01     5                  650     0.8401786  0.8550  0.6374459
##   0.01     5                  700     0.8384740  0.8550  0.6329004
##   0.01     5                  750     0.8367045  0.8550  0.6281385
##   0.01     5                  800     0.8355736  0.8550  0.6281385
##   0.01     5                  850     0.8342857  0.8500  0.6329004
##   0.01     5                  900     0.8339069  0.8450  0.6281385
##   0.01     5                  950     0.8325325  0.8475  0.6190476
##   0.01     5                 1000     0.8312446  0.8450  0.6190476
##   0.01     7                  100     0.8470238  0.9050  0.5110390
##   0.01     7                  150     0.8491829  0.8775  0.5580087
##   0.01     7                  200     0.8481494  0.8700  0.5857143
##   0.01     7                  250     0.8492154  0.8675  0.6045455
##   0.01     7                  300     0.8477543  0.8625  0.6181818
##   0.01     7                  350     0.8471916  0.8625  0.6277056
##   0.01     7                  400     0.8458225  0.8575  0.6370130
##   0.01     7                  450     0.8440801  0.8525  0.6277056
##   0.01     7                  500     0.8429978  0.8500  0.6376623
##   0.01     7                  550     0.8395400  0.8500  0.6376623
##   0.01     7                  600     0.8385985  0.8475  0.6424242
##   0.01     7                  650     0.8372998  0.8425  0.6426407
##   0.01     7                  700     0.8365152  0.8425  0.6471861
##   0.01     7                  750     0.8346320  0.8425  0.6331169
##   0.01     7                  800     0.8331006  0.8400  0.6331169
##   0.01     7                  850     0.8314556  0.8400  0.6333333
##   0.01     7                  900     0.8294535  0.8400  0.6283550
##   0.01     7                  950     0.8278571  0.8425  0.6333333
##   0.01     7                 1000     0.8286905  0.8400  0.6240260
##   0.10     1                  100     0.8461688  0.8550  0.5761905
##   0.10     1                  150     0.8455411  0.8700  0.5720779
##   0.10     1                  200     0.8441883  0.8550  0.5670996
##   0.10     1                  250     0.8429437  0.8575  0.5954545
##   0.10     1                  300     0.8415963  0.8625  0.5952381
##   0.10     1                  350     0.8443236  0.8550  0.5956710
##   0.10     1                  400     0.8424892  0.8500  0.5952381
##   0.10     1                  450     0.8405465  0.8525  0.6188312
##   0.10     1                  500     0.8400487  0.8550  0.6138528
##   0.10     1                  550     0.8391234  0.8425  0.6279221
##   0.10     1                  600     0.8377110  0.8450  0.6324675
##   0.10     1                  650     0.8373539  0.8425  0.6186147
##   0.10     1                  700     0.8348431  0.8375  0.6283550
##   0.10     1                  750     0.8314340  0.8425  0.6283550
##   0.10     1                  800     0.8313853  0.8400  0.6145022
```

```
##   0.10      1               850   0.8325758  0.8375  0.6285714
##   0.10      1               900   0.8278409  0.8400  0.6188312
##   0.10      1               950   0.8287392  0.8400  0.6235931
##   0.10      1              1000   0.8249946  0.8350  0.6374459
##   0.10      3               100   0.8326948  0.8600  0.6138528
##   0.10      3               150   0.8270725  0.8550  0.6043290
##   0.10      3               200   0.8249892  0.8500  0.5997835
##   0.10      3               250   0.8234686  0.8450  0.5952381
##   0.10      3               300   0.8142154  0.8350  0.5859307
##   0.10      3               350   0.8138690  0.8350  0.5768398
##   0.10      3               400   0.8076461  0.8250  0.5954545
##   0.10      3               450   0.8074080  0.8250  0.5861472
##   0.10      3               500   0.8038528  0.8225  0.5811688
##   0.10      3               550   0.8000108  0.8225  0.5861472
##   0.10      3               600   0.7992478  0.8225  0.5863636
##   0.10      3               650   0.7974784  0.8200  0.5961039
##   0.10      3               700   0.7984145  0.8150  0.5865801
##   0.10      3               750   0.7965801  0.8150  0.5909091
##   0.10      3               800   0.7960335  0.8100  0.5958874
##   0.10      3               850   0.7960119  0.8100  0.5818182
##   0.10      3               900   0.7947511  0.8100  0.5722944
##   0.10      3               950   0.7933820  0.8075  0.5818182
##   0.10      3              1000   0.7914935  0.8025  0.5818182
##   0.10      5               100   0.8298323  0.8500  0.5956710
##   0.10      5               150   0.8194968  0.8250  0.5816017
##   0.10      5               200   0.8096645  0.8300  0.6051948
##   0.10      5               250   0.8098539  0.8200  0.6004329
##   0.10      5               300   0.8071158  0.8225  0.6047619
##   0.10      5               350   0.8046537  0.8250  0.6002165
##   0.10      5               400   0.8023918  0.8200  0.6051948
##   0.10      5               450   0.8030411  0.8125  0.6004329
##   0.10      5               500   0.7985931  0.8100  0.6192641
##   0.10      5               550   0.8007251  0.8075  0.6051948
##   0.10      5               600   0.7985011  0.8150  0.5915584
##   0.10      5               650   0.7948431  0.8125  0.5958874
##   0.10      5               700   0.7937067  0.8075  0.5913420
##   0.10      5               750   0.7968506  0.8050  0.6051948
##   0.10      5               800   0.7954058  0.8075  0.6004329
##   0.10      5               850   0.7927110  0.8000  0.6049784
##   0.10      5               900   0.7937933  0.8000  0.6142857
##   0.10      5               950   0.7923377  0.8000  0.6004329
##   0.10      5              1000   0.7921916  0.8025  0.5958874
##   0.10      7               100   0.8169048  0.8350  0.5764069
##   0.10      7               150   0.8106385  0.8350  0.5534632
##   0.10      7               200   0.8029870  0.8225  0.5770563
##   0.10      7               250   0.8041721  0.8275  0.5679654
##   0.10      7               300   0.8028355  0.8150  0.5722944
##   0.10      7               350   0.8017208  0.8175  0.5770563
##   0.10      7               400   0.7966883  0.8200  0.5772727
##   0.10      7               450   0.7978084  0.8175  0.5818182
##   0.10      7               500   0.7985444  0.8150  0.5774892
##   0.10      7               550   0.7994048  0.8150  0.5722944
##   0.10      7               600   0.7972457  0.8175  0.5718615
##   0.10      7               650   0.7993506  0.8150  0.5720779
```

```
##   0.10       7                      700     0.7990097  0.8125  0.5816017
##   0.10       7                      750     0.7968777  0.8100  0.5718615
##   0.10       7                      800     0.7967316  0.8075  0.5722944
##   0.10       7                      850     0.7946104  0.8100  0.5677489
##   0.10       7                      900     0.7936472  0.8125  0.5820346
##   0.10       7                      950     0.7961959  0.8150  0.5820346
##   0.10       7                     1000     0.7966775  0.8175  0.5820346
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 250, interaction.depth =
##  3, shrinkage = 0.01 and n.minobsinnode = 10.
```

```r
################# Elastinet #####################
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(.01, .2, length = 40))
set.seed(1)
glmnFit <- train(diabetes ~., data = train_data,
                method = "glmnet",
                tuneGrid = glmnGrid,
                preProcess = c("center","scale", "BoxCox"),
                metric = "ROC",
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
glmnFit
```

```
## glmnet
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), Box-Cox transformation (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      ROC        Sens    Spec
##   0.0    0.01000000  0.8533063  0.8675  0.590476190
##   0.0    0.01487179  0.8533063  0.8675  0.590476190
##   0.0    0.01974359  0.8533063  0.8675  0.590476190
##   0.0    0.02461538  0.8533063  0.8675  0.585714286
##   0.0    0.02948718  0.8528355  0.8700  0.585714286
##   0.0    0.03435897  0.8527327  0.8700  0.585714286
##   0.0    0.03923077  0.8527381  0.8700  0.585714286
##   0.0    0.04410256  0.8526190  0.8775  0.580952381
##   0.0    0.04897436  0.8518182  0.8800  0.576190476
##   0.0    0.05384615  0.8518236  0.8825  0.576190476
##   0.0    0.05871795  0.8514665  0.8825  0.576190476
##   0.0    0.06358974  0.8508820  0.8850  0.576190476
##   0.0    0.06846154  0.8500866  0.8875  0.571428571
##   0.0    0.07333333  0.8506764  0.8900  0.571428571
##   0.0    0.07820513  0.8503301  0.8900  0.571428571
##   0.0    0.08307692  0.8506818  0.8925  0.571428571
```

```
##   0.0   0.08794872  0.8509199  0.8950  0.566666667
##   0.0   0.09282051  0.8508009  0.8950  0.566666667
##   0.0   0.09769231  0.8510335  0.8950  0.566666667
##   0.0   0.10256410  0.8510281  0.8950  0.566666667
##   0.0   0.10743590  0.8511580  0.8950  0.566666667
##   0.0   0.11230769  0.8510390  0.8975  0.566666667
##   0.0   0.11717949  0.8511526  0.8975  0.566666667
##   0.0   0.12205128  0.8510390  0.9025  0.566666667
##   0.0   0.12692308  0.8508117  0.9050  0.557142857
##   0.0   0.13179487  0.8509199  0.9075  0.557142857
##   0.0   0.13666667  0.8510335  0.9075  0.557142857
##   0.0   0.14153846  0.8508009  0.9075  0.557142857
##   0.0   0.14641026  0.8506872  0.9075  0.552597403
##   0.0   0.15128205  0.8506981  0.9075  0.552597403
##   0.0   0.15615385  0.8505844  0.9075  0.552597403
##   0.0   0.16102564  0.8508171  0.9075  0.547835498
##   0.0   0.16589744  0.8505844  0.9075  0.547835498
##   0.0   0.17076923  0.8508171  0.9075  0.547835498
##   0.0   0.17564103  0.8509307  0.9075  0.547835498
##   0.0   0.18051282  0.8506981  0.9100  0.547835498
##   0.0   0.18538462  0.8510444  0.9100  0.533766234
##   0.0   0.19025641  0.8510444  0.9100  0.524242424
##   0.0   0.19512821  0.8509307  0.9150  0.524242424
##   0.0   0.20000000  0.8510444  0.9150  0.524242424
##   0.1   0.01000000  0.8523593  0.8650  0.599567100
##   0.1   0.01487179  0.8522511  0.8675  0.585714286
##   0.1   0.01974359  0.8523701  0.8675  0.595021645
##   0.1   0.02461538  0.8529437  0.8675  0.595021645
##   0.1   0.02948718  0.8529437  0.8700  0.585714286
##   0.1   0.03435897  0.8527165  0.8775  0.580952381
##   0.1   0.03923077  0.8523755  0.8775  0.585714286
##   0.1   0.04410256  0.8521483  0.8800  0.580952381
##   0.1   0.04897436  0.8521483  0.8825  0.580952381
##   0.1   0.05384615  0.8525108  0.8825  0.580952381
##   0.1   0.05871795  0.8527489  0.8850  0.580952381
##   0.1   0.06358974  0.8523918  0.8850  0.580952381
##   0.1   0.06846154  0.8523972  0.8875  0.580952381
##   0.1   0.07333333  0.8522835  0.8900  0.580952381
##   0.1   0.07820513  0.8520563  0.8925  0.571428571
##   0.1   0.08307692  0.8519426  0.8925  0.566666667
##   0.1   0.08794872  0.8520563  0.8925  0.561904762
##   0.1   0.09282051  0.8519426  0.8925  0.561904762
##   0.1   0.09769231  0.8518290  0.8950  0.561904762
##   0.1   0.10256410  0.8513582  0.9025  0.561904762
##   0.1   0.10743590  0.8507738  0.9050  0.561904762
##   0.1   0.11230769  0.8507738  0.9050  0.557359307
##   0.1   0.11717949  0.8506548  0.9075  0.548051948
##   0.1   0.12205128  0.8507684  0.9075  0.543290043
##   0.1   0.12692308  0.8507738  0.9150  0.538528139
##   0.1   0.13179487  0.8508929  0.9150  0.529004329
##   0.1   0.13666667  0.8510119  0.9150  0.529004329
##   0.1   0.14153846  0.8510119  0.9150  0.524242424
##   0.1   0.14641026  0.8511310  0.9150  0.519696970
##   0.1   0.15128205  0.8506710  0.9150  0.515151515
```

```
##    0.1     0.15615385  0.8504383  0.9150  0.515151515
##    0.1     0.16102564  0.8502002  0.9150  0.515151515
##    0.1     0.16589744  0.8504329  0.9150  0.515151515
##    0.1     0.17076923  0.8504383  0.9150  0.515151515
##    0.1     0.17564103  0.8504383  0.9175  0.515151515
##    0.1     0.18051282  0.8503193  0.9175  0.515151515
##    0.1     0.18538462  0.8502002  0.9175  0.515151515
##    0.1     0.19025641  0.8498539  0.9175  0.515151515
##    0.1     0.19512821  0.8502002  0.9175  0.510606061
##    0.1     0.20000000  0.8502002  0.9175  0.506060606
##    0.2     0.01000000  0.8523485  0.8675  0.599567100
##    0.2     0.01487179  0.8523485  0.8675  0.585714286
##    0.2     0.01974359  0.8529221  0.8675  0.595021645
##    0.2     0.02461538  0.8538636  0.8700  0.599783550
##    0.2     0.02948718  0.8530736  0.8750  0.590259740
##    0.2     0.03435897  0.8523810  0.8825  0.585714286
##    0.2     0.03923077  0.8523918  0.8825  0.585714286
##    0.2     0.04410256  0.8525162  0.8825  0.580952381
##    0.2     0.04897436  0.8526353  0.8825  0.576406926
##    0.2     0.05384615  0.8529870  0.8825  0.576406926
##    0.2     0.05871795  0.8525108  0.8850  0.566883117
##    0.2     0.06358974  0.8526353  0.8850  0.562121212
##    0.2     0.06846154  0.8525216  0.8850  0.557359307
##    0.2     0.07333333  0.8522944  0.8900  0.557359307
##    0.2     0.07820513  0.8525216  0.8950  0.557359307
##    0.2     0.08307692  0.8526299  0.9025  0.557359307
##    0.2     0.08794872  0.8523864  0.9025  0.557359307
##    0.2     0.09282051  0.8525054  0.9025  0.552597403
##    0.2     0.09769231  0.8522781  0.9025  0.538744589
##    0.2     0.10256410  0.8522835  0.9075  0.529220779
##    0.2     0.10743590  0.8522944  0.9075  0.529220779
##    0.2     0.11230769  0.8522998  0.9100  0.524675325
##    0.2     0.11717949  0.8518290  0.9125  0.524675325
##    0.2     0.12205128  0.8512500  0.9150  0.524675325
##    0.2     0.12692308  0.8513745  0.9150  0.524675325
##    0.2     0.13179487  0.8509145  0.9150  0.524675325
##    0.2     0.13666667  0.8505574  0.9150  0.519913420
##    0.2     0.14153846  0.8503301  0.9175  0.519913420
##    0.2     0.14641026  0.8503409  0.9175  0.519913420
##    0.2     0.15128205  0.8502165  0.9175  0.515151515
##    0.2     0.15615385  0.8498701  0.9175  0.515151515
##    0.2     0.16102564  0.8498647  0.9175  0.506060606
##    0.2     0.16589744  0.8498647  0.9175  0.506060606
##    0.2     0.17076923  0.8497457  0.9200  0.501298701
##    0.2     0.17564103  0.8498647  0.9225  0.501298701
##    0.2     0.18051282  0.8495076  0.9275  0.501298701
##    0.2     0.18538462  0.8492803  0.9275  0.492207792
##    0.2     0.19025641  0.8488258  0.9275  0.492207792
##    0.2     0.19512821  0.8489394  0.9300  0.482683983
##    0.2     0.20000000  0.8484740  0.9350  0.478138528
##    0.4     0.01000000  0.8524784  0.8675  0.595021645
##    0.4     0.01487179  0.8511797  0.8675  0.600000000
##    0.4     0.01974359  0.8513961  0.8700  0.595021645
##    0.4     0.02461538  0.8527327  0.8775  0.585714286
```

```
## 0.4 0.02948718 0.8534416 0.8800 0.585714286
## 0.4 0.03435897 0.8537825 0.8850 0.580952381
## 0.4 0.03923077 0.8537825 0.8850 0.571428571
## 0.4 0.04410256 0.8534416 0.8875 0.566666667
## 0.4 0.04897436 0.8535606 0.8900 0.566666667
## 0.4 0.05384615 0.8532089 0.8900 0.562121212
## 0.4 0.05871795 0.8527489 0.8925 0.557575758
## 0.4 0.06358974 0.8526299 0.8925 0.552813853
## 0.4 0.06846154 0.8526407 0.8950 0.552813853
## 0.4 0.07333333 0.8521807 0.8975 0.538744589
## 0.4 0.07820513 0.8524134 0.8975 0.529437229
## 0.4 0.08307692 0.8520671 0.9000 0.529437229
## 0.4 0.08794872 0.8516017 0.9000 0.529437229
## 0.4 0.09282051 0.8517154 0.9025 0.524891775
## 0.4 0.09769231 0.8518452 0.9025 0.524891775
## 0.4 0.10256410 0.8513799 0.9100 0.524891775
## 0.4 0.10743590 0.8511526 0.9125 0.515367965
## 0.4 0.11230769 0.8510281 0.9175 0.510822511
## 0.4 0.11717949 0.8508009 0.9200 0.506277056
## 0.4 0.12205128 0.8504383 0.9250 0.506277056
## 0.4 0.12692308 0.8498593 0.9250 0.492424242
## 0.4 0.13179487 0.8486905 0.9275 0.473809524
## 0.4 0.13666667 0.8476407 0.9300 0.473809524
## 0.4 0.14153846 0.8476353 0.9350 0.473809524
## 0.4 0.14641026 0.8476407 0.9425 0.464285714
## 0.4 0.15128205 0.8468344 0.9425 0.459740260
## 0.4 0.15615385 0.8453139 0.9450 0.450432900
## 0.4 0.16102564 0.8442695 0.9475 0.427056277
## 0.4 0.16589744 0.8437987 0.9500 0.413203463
## 0.4 0.17076923 0.8426461 0.9550 0.403679654
## 0.4 0.17564103 0.8424080 0.9550 0.394372294
## 0.4 0.18051282 0.8413690 0.9550 0.389610390
## 0.4 0.18538462 0.8407792 0.9550 0.366233766
## 0.4 0.19025641 0.8401948 0.9550 0.352380952
## 0.4 0.19512821 0.8397348 0.9575 0.347619048
## 0.4 0.20000000 0.8387825 0.9600 0.329220779
## 0.6 0.01000000 0.8516558 0.8675 0.604329004
## 0.6 0.01487179 0.8512067 0.8700 0.595238095
## 0.6 0.01974359 0.8530790 0.8800 0.595021645
## 0.6 0.02461538 0.8545942 0.8825 0.590259740
## 0.6 0.02948718 0.8541342 0.8825 0.590259740
## 0.6 0.03435897 0.8544968 0.8875 0.576190476
## 0.6 0.03923077 0.8543885 0.8900 0.571645022
## 0.6 0.04410256 0.8542749 0.8925 0.571645022
## 0.6 0.04897436 0.8534524 0.8925 0.571645022
## 0.6 0.05384615 0.8526515 0.8950 0.562337662
## 0.6 0.05871795 0.8525108 0.9000 0.552813853
## 0.6 0.06358974 0.8521483 0.9000 0.538744589
## 0.6 0.06846154 0.8521483 0.9025 0.529437229
## 0.6 0.07333333 0.8516883 0.9025 0.524675325
## 0.6 0.07820513 0.8508820 0.9025 0.520129870
## 0.6 0.08307692 0.8498106 0.9100 0.520129870
## 0.6 0.08794872 0.8495671 0.9100 0.515367965
## 0.6 0.09282051 0.8478139 0.9175 0.496969697
```

```
##    0.6    0.09769231  0.8468777  0.9225  0.492424242
##    0.6    0.10256410  0.8458279  0.9275  0.487878788
##    0.6    0.10743590  0.8438636  0.9325  0.478571429
##    0.6    0.11230769  0.8431548  0.9350  0.469264069
##    0.6    0.11717949  0.8430465  0.9425  0.464718615
##    0.6    0.12205128  0.8416396  0.9425  0.455194805
##    0.6    0.12692308  0.8394372  0.9475  0.436796537
##    0.6    0.13179487  0.8388366  0.9500  0.422727273
##    0.6    0.13666667  0.8378842  0.9500  0.408658009
##    0.6    0.14153846  0.8368344  0.9525  0.390043290
##    0.6    0.14641026  0.8349621  0.9525  0.380735931
##    0.6    0.15128205  0.8352002  0.9525  0.348051948
##    0.6    0.15615385  0.8340260  0.9550  0.325108225
##    0.6    0.16102564  0.8335335  0.9600  0.311038961
##    0.6    0.16589744  0.8324784  0.9700  0.287445887
##    0.6    0.17076923  0.8314989  0.9725  0.269047619
##    0.6    0.17564103  0.8298485  0.9750  0.255194805
##    0.6    0.18051282  0.8285552  0.9775  0.246103896
##    0.6    0.18538462  0.8260877  0.9775  0.227272727
##    0.6    0.19025641  0.8237446  0.9775  0.208225108
##    0.6    0.19512821  0.8215314  0.9775  0.194155844
##    0.6    0.20000000  0.8191775  0.9775  0.175541126
##    0.8    0.01000000  0.8519859  0.8675  0.618398268
##    0.8    0.01487179  0.8531981  0.8775  0.604545455
##    0.8    0.01974359  0.8549405  0.8825  0.604329004
##    0.8    0.02461538  0.8533009  0.8875  0.608874459
##    0.8    0.02948718  0.8540206  0.8900  0.595021645
##    0.8    0.03435897  0.8529870  0.8950  0.590259740
##    0.8    0.03923077  0.8527489  0.8950  0.571645022
##    0.8    0.04410256  0.8522727  0.8975  0.571645022
##    0.8    0.04897436  0.8513366  0.9000  0.557575758
##    0.8    0.05384615  0.8500433  0.9025  0.548051948
##    0.8    0.05871795  0.8500271  0.9025  0.534199134
##    0.8    0.06358974  0.8485119  0.9050  0.534199134
##    0.8    0.06846154  0.8481656  0.9075  0.524891775
##    0.8    0.07333333  0.8461797  0.9150  0.515584416
##    0.8    0.07820513  0.8443074  0.9175  0.511038961
##    0.8    0.08307692  0.8422024  0.9275  0.497186147
##    0.8    0.08794872  0.8409199  0.9300  0.488095238
##    0.8    0.09282051  0.8407846  0.9300  0.483333333
##    0.8    0.09769231  0.8399621  0.9400  0.474242424
##    0.8    0.10256410  0.8389123  0.9425  0.459956710
##    0.8    0.10743590  0.8380574  0.9500  0.436796537
##    0.8    0.11230769  0.8353409  0.9500  0.432251082
##    0.8    0.11717949  0.8342911  0.9525  0.395021645
##    0.8    0.12205128  0.8322727  0.9550  0.385497835
##    0.8    0.12692308  0.8307468  0.9600  0.357359307
##    0.8    0.13179487  0.8278084  0.9650  0.315584416
##    0.8    0.13666667  0.8242911  0.9675  0.306060606
##    0.8    0.14153846  0.8219156  0.9700  0.292207792
##    0.8    0.14641026  0.8192532  0.9750  0.273809524
##    0.8    0.15128205  0.8162392  0.9775  0.250432900
##    0.8    0.15615385  0.8130195  0.9775  0.236363636
##    0.8    0.16102564  0.8087013  0.9800  0.217532468
```

```
##   0.8    0.16589744  0.8062608  0.9825  0.193939394
##   0.8    0.17076923  0.8037635  0.9825  0.166450216
##   0.8    0.17564103  0.8014205  0.9825  0.115584416
##   0.8    0.18051282  0.8000244  0.9925  0.092424242
##   0.8    0.18538462  0.7997944  0.9925  0.064718615
##   0.8    0.19025641  0.7996699  0.9950  0.041558442
##   0.8    0.19512821  0.7993128  0.9975  0.013852814
##   0.8    0.20000000  0.7989556  1.0000  0.000000000
##   1.0    0.01000000  0.8528517  0.8725  0.613852814
##   1.0    0.01487179  0.8546158  0.8825  0.608874459
##   1.0    0.01974359  0.8541288  0.8875  0.608658009
##   1.0    0.02461538  0.8527381  0.8900  0.599350649
##   1.0    0.02948718  0.8535606  0.8925  0.589826840
##   1.0    0.03435897  0.8522673  0.8950  0.580735931
##   1.0    0.03923077  0.8517911  0.8950  0.566666667
##   1.0    0.04410256  0.8510714  0.9000  0.566666667
##   1.0    0.04897436  0.8493290  0.9000  0.548051948
##   1.0    0.05384615  0.8472132  0.9050  0.534199134
##   1.0    0.05871795  0.8459145  0.9100  0.520346320
##   1.0    0.06358974  0.8434794  0.9150  0.515800866
##   1.0    0.06846154  0.8413745  0.9225  0.501948052
##   1.0    0.07333333  0.8412446  0.9300  0.492857143
##   1.0    0.07820513  0.8390043  0.9300  0.488095238
##   1.0    0.08307692  0.8374297  0.9350  0.474242424
##   1.0    0.08794872  0.8356710  0.9350  0.459956710
##   1.0    0.09282051  0.8330574  0.9450  0.450865801
##   1.0    0.09769231  0.8295400  0.9475  0.427705628
##   1.0    0.10256410  0.8259091  0.9525  0.404329004
##   1.0    0.10743590  0.8223810  0.9575  0.372077922
##   1.0    0.11230769  0.8195779  0.9575  0.334415584
##   1.0    0.11717949  0.8157738  0.9675  0.320346320
##   1.0    0.12205128  0.8096266  0.9675  0.315800866
##   1.0    0.12692308  0.8062608  0.9750  0.278354978
##   1.0    0.13179487  0.8036039  0.9750  0.254761905
##   1.0    0.13666667  0.8007819  0.9775  0.231385281
##   1.0    0.14153846  0.7998295  0.9800  0.208008658
##   1.0    0.14641026  0.7991991  0.9800  0.184848485
##   1.0    0.15128205  0.7991937  0.9800  0.138528139
##   1.0    0.15615385  0.7989556  0.9875  0.092424242
##   1.0    0.16102564  0.7989556  0.9925  0.055411255
##   1.0    0.16589744  0.7989556  0.9950  0.022943723
##   1.0    0.17076923  0.7989556  1.0000  0.004761905
##   1.0    0.17564103  0.7989556  1.0000  0.000000000
##   1.0    0.18051282  0.7989556  1.0000  0.000000000
##   1.0    0.18538462  0.7989556  1.0000  0.000000000
##   1.0    0.19025641  0.7989556  1.0000  0.000000000
##   1.0    0.19512821  0.7989556  1.0000  0.000000000
##   1.0    0.20000000  0.7989556  1.0000  0.000000000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.8 and lambda = 0.01974359.
```

```
############################ Nearest Shrunken Centroids ##############################
nscGrid <- data.frame(.threshold = 0:25)
```

```r
set.seed(1)
nscFit <- train(diabetes ~., data = train_data,
                method = "pam",
                tuneGrid = nscGrid,
                preProcess = c("center","scale", "BoxCox"),
                metric = "ROC",
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
```

```
## 1
```

```r
nscFit
```

```
## Nearest Shrunken Centroids
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), Box-Cox transformation (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   threshold  ROC        Sens  Spec
##    0         0.8426948  0.94  0.427489177
##    1         0.8431764  0.96  0.301082251
##    2         0.8369968  0.99  0.134199134
##    3         0.8315693  1.00  0.004545455
##    4         0.8064069  1.00  0.000000000
##    5         0.7988420  1.00  0.000000000
##    6         0.7989556  1.00  0.000000000
##    7         0.5000000  1.00  0.000000000
##    8         0.5000000  1.00  0.000000000
##    9         0.5000000  1.00  0.000000000
##   10         0.5000000  1.00  0.000000000
##   11         0.5000000  1.00  0.000000000
##   12         0.5000000  1.00  0.000000000
##   13         0.5000000  1.00  0.000000000
##   14         0.5000000  1.00  0.000000000
##   15         0.5000000  1.00  0.000000000
##   16         0.5000000  1.00  0.000000000
##   17         0.5000000  1.00  0.000000000
##   18         0.5000000  1.00  0.000000000
##   19         0.5000000  1.00  0.000000000
##   20         0.5000000  1.00  0.000000000
##   21         0.5000000  1.00  0.000000000
##   22         0.5000000  1.00  0.000000000
##   23         0.5000000  1.00  0.000000000
##   24         0.5000000  1.00  0.000000000
##   25         0.5000000  1.00  0.000000000
##
## ROC was used to select the optimal model using the largest value.
```

```
## The final value used for the model was threshold = 1.
```

```
############################ LDA ##############################
set.seed(1)
ldaFit <- train(diabetes ~., data = train_data,
                method = "lda",
                metric = "ROC",
                preProcess = c("center","scale", "BoxCox"),
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
ldaFit
```

```
## Linear Discriminant Analysis
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), Box-Cox transformation (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results:
##
##   ROC        Sens    Spec
##   0.8528409  0.8625  0.6229437
```

```
#Compare ROC Value by Training Model
allmodels <- list(Logistic_Regression = lr_train_data, Random_Forest = rf_train_data, KNN = knn_train_da
allmodels2 <- list(NSC = nscFit, LDA = ldaFit, Boost = gbmFit, ENet = glmnFit)
trainresults <- resamples(allmodels)
trainresults2 <- resamples(allmodels2)
#Box Plot: Training Models' ROC Values
#Logistic Regression Performed Best on Training Data
bwplot(trainresults, metric="ROC")
bwplot(trainresults2, metric= "ROC")
```

```
###########################Test Data###########################
#Logistic Regression: Testing Data
set.seed(1)
lrpredict <- predict(lr_train_data, test_data)
#Confusion Matrix Accuracy
lrconfusion <- confusionMatrix(lrpredict, test_data$diabetes, positive="pos")
lrconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  85  23
##        pos  15  30
##
##                Accuracy : 0.7516
##                  95% CI : (0.6754, 0.8179)
```

```
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.005891
##
##                   Kappa : 0.4313
##
##  Mcnemar's Test P-Value : 0.256145
##
##             Sensitivity : 0.5660
##             Specificity : 0.8500
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7870
##              Prevalence : 0.3464
##          Detection Rate : 0.1961
##    Detection Prevalence : 0.2941
##       Balanced Accuracy : 0.7080
##
##        'Positive' Class : pos
##
```

```
#Random Forest: Testing Data
set.seed(1)
rfpredict <- predict(rf_train_data, test_data)
#Confusion Matrix Accuracy
rfconfusion <- confusionMatrix(rfpredict, test_data$diabetes, positive="pos")
rfconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  79  23
##        pos  21  30
##
##                Accuracy : 0.7124
##                  95% CI : (0.6338, 0.7826)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.07283
##
##                   Kappa : 0.3592
##
##  Mcnemar's Test P-Value : 0.88017
##
##             Sensitivity : 0.5660
##             Specificity : 0.7900
##          Pos Pred Value : 0.5882
##          Neg Pred Value : 0.7745
##              Prevalence : 0.3464
##          Detection Rate : 0.1961
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.6780
##
##        'Positive' Class : pos
##
```

```r
#K Nearest Neighbor: Testing Data
set.seed(1)
knnpredict <- predict(knn_train_data, test_data)
#Confusion Matrix Accuracy
knnconfusion <- confusionMatrix(knnpredict, test_data$diabetes, positive="pos")
knnconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  87  29
##        pos  13  24
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.03543
##
##                   Kappa : 0.3475
##
##  Mcnemar's Test P-Value : 0.02064
##
##             Sensitivity : 0.4528
##             Specificity : 0.8700
##          Pos Pred Value : 0.6486
##          Neg Pred Value : 0.7500
##              Prevalence : 0.3464
##          Detection Rate : 0.1569
##    Detection Prevalence : 0.2418
##       Balanced Accuracy : 0.6614
##
##        'Positive' Class : pos
##
```

```r
#Classification and Regression Trees (CART): Testing Data
set.seed(1)
cartpredict <- predict(cart_train_data, test_data)
#Confusion Matrix Accuracy
cartconfusion <- confusionMatrix(cartpredict, test_data$diabetes, positive="pos")
cartconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  74  28
##        pos  26  25
##
##                Accuracy : 0.6471
##                  95% CI : (0.5658, 0.7225)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.6037
```

```
##
##                   Kappa : 0.2136
##
##  Mcnemar's Test P-Value : 0.8918
##
##             Sensitivity : 0.4717
##             Specificity : 0.7400
##          Pos Pred Value : 0.4902
##          Neg Pred Value : 0.7255
##              Prevalence : 0.3464
##          Detection Rate : 0.1634
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.6058
##
##        'Positive' Class : pos
##
```

```
#Neural Net: Testing Data
set.seed(1)
nnetpredict <- predict(nnet_train_data, test_data)
#Confusion Matrix Accuracy
nnetconfusion <- confusionMatrix(nnetpredict, test_data$diabetes, positive="pos")
nnetconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  87  22
##        pos  13  31
##
##                Accuracy : 0.7712
##                  95% CI : (0.6965, 0.8352)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.001098
##
##                   Kappa : 0.4738
##
##  Mcnemar's Test P-Value : 0.176296
##
##             Sensitivity : 0.5849
##             Specificity : 0.8700
##          Pos Pred Value : 0.7045
##          Neg Pred Value : 0.7982
##              Prevalence : 0.3464
##          Detection Rate : 0.2026
##    Detection Prevalence : 0.2876
##       Balanced Accuracy : 0.7275
##
##        'Positive' Class : pos
##
```

```r
#Support Vector Machines
set.seed(1)
svmpredict <- predict(svmFit, test_data)
#Confusion Matrix Accuracy
svmconfusion <- confusionMatrix(svmpredict, test_data$diabetes, positive="pos")
svmconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  86  24
##        pos  14  29
##
##                Accuracy : 0.7516
##                  95% CI : (0.6754, 0.8179)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.005891
##
##                   Kappa : 0.4261
##
##  Mcnemar's Test P-Value : 0.144292
##
##             Sensitivity : 0.5472
##             Specificity : 0.8600
##          Pos Pred Value : 0.6744
##          Neg Pred Value : 0.7818
##              Prevalence : 0.3464
##          Detection Rate : 0.1895
##    Detection Prevalence : 0.2810
##       Balanced Accuracy : 0.7036
##
##        'Positive' Class : pos
##
```

```r
#Boost
set.seed(1)
gbmpredict <- predict(gbmFit, test_data)
#Confusion Matrix Accuracy
gbmconfusion <- confusionMatrix(gbmpredict, test_data$diabetes, positive="pos")
gbmconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  83  26
##        pos  17  27
##
##                Accuracy : 0.719
##                  95% CI : (0.6407, 0.7886)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.05152
```

```
##
##                  Kappa : 0.3535
##
##  Mcnemar's Test P-Value : 0.22247
##
##            Sensitivity : 0.5094
##            Specificity : 0.8300
##         Pos Pred Value : 0.6136
##         Neg Pred Value : 0.7615
##             Prevalence : 0.3464
##         Detection Rate : 0.1765
##   Detection Prevalence : 0.2876
##      Balanced Accuracy : 0.6697
##
##       'Positive' Class : pos
##
```

```
# Elastinet
set.seed(1)
glmnpredict <- predict(glmnFit, test_data)
#Confusion Matrix Accuracy
glmnconfusion <- confusionMatrix(glmnpredict, test_data$diabetes, positive="pos")
glmnconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  87  24
##        pos  13  29
##
##               Accuracy : 0.7582
##                 95% CI : (0.6824, 0.8237)
##    No Information Rate : 0.6536
##    P-Value [Acc > NIR] : 0.003479
##
##                  Kappa : 0.4386
##
##  Mcnemar's Test P-Value : 0.100178
##
##            Sensitivity : 0.5472
##            Specificity : 0.8700
##         Pos Pred Value : 0.6905
##         Neg Pred Value : 0.7838
##             Prevalence : 0.3464
##         Detection Rate : 0.1895
##   Detection Prevalence : 0.2745
##      Balanced Accuracy : 0.7086
##
##       'Positive' Class : pos
##
```

```r
# Nearest Shrunken Centroid
set.seed(1)
nscpredict <- predict(nscFit, test_data)
#Confusion Matrix Accuracy
nscconfusion <- confusionMatrix(nscpredict, test_data$diabetes, positive="pos")
nscconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  93  38
##        pos   7  15
##
##                Accuracy : 0.7059
##                  95% CI : (0.6269, 0.7767)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.1002
##
##                   Kappa : 0.247
##
##  Mcnemar's Test P-Value : 7.744e-06
##
##             Sensitivity : 0.28302
##             Specificity : 0.93000
##          Pos Pred Value : 0.68182
##          Neg Pred Value : 0.70992
##              Prevalence : 0.34641
##          Detection Rate : 0.09804
##    Detection Prevalence : 0.14379
##       Balanced Accuracy : 0.60651
##
##        'Positive' Class : pos
##
```

```r
#LDA
set.seed(1)
ldapredict <- predict(ldaFit, test_data)
#Confusion Matrix Accuracy
ldaconfusion <- confusionMatrix(ldapredict, test_data$diabetes, positive="pos")
ldaconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  85  23
##        pos  15  30
##
##                Accuracy : 0.7516
##                  95% CI : (0.6754, 0.8179)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.005891
```

```
##
##                   Kappa : 0.4313
##
##  Mcnemar's Test P-Value : 0.256145
##
##             Sensitivity : 0.5660
##             Specificity : 0.8500
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7870
##              Prevalence : 0.3464
##          Detection Rate : 0.1961
##    Detection Prevalence : 0.2941
##       Balanced Accuracy : 0.7080
##
##        'Positive' Class : pos
##
```

```
#Comparing Test Results
lrfinal<- c(lrconfusion$byClass['Sensitivity'], lrconfusion$byClass['Specificity'], lrconfusion$byClass
            lrconfusion$byClass['Recall'], lrconfusion$byClass['F1'])
rffinal <- c(rfconfusion$byClass['Sensitivity'], rfconfusion$byClass['Specificity'], rfconfusion$byClas
             rfconfusion$byClass['Recall'], rfconfusion$byClass['F1'])
knnfinal <- c(knnconfusion$byClass['Sensitivity'], knnconfusion$byClass['Specificity'], knnconfusion$by
              knnconfusion$byClass['Recall'], knnconfusion$byClass['F1'])
cartfinal <- c(cartconfusion$byClass['Sensitivity'], cartconfusion$byClass['Specificity'], cartconfusio
               cartconfusion$byClass['Recall'], cartconfusion$byClass['F1'])
nnetfinal <- c(nnetconfusion$byClass['Sensitivity'], nnetconfusion$byClass['Specificity'], nnetconfusio
               nnetconfusion$byClass['Recall'], nnetconfusion$byClass['F1'])
svmfinal <- c(svmconfusion$byClass['Sensitivity'], svmconfusion$byClass['Specificity'], svmconfusion$by
              svmconfusion$byClass['Recall'], svmconfusion$byClass['F1'])
gbmfinal <- c(gbmconfusion$byClass['Sensitivity'], gbmconfusion$byClass['Specificity'], gbmconfusion$by
              gbmconfusion$byClass['Recall'], gbmconfusion$byClass['F1'])
glmnfinal <- c(glmnconfusion$byClass['Sensitivity'], glmnconfusion$byClass['Specificity'], glmnconfusio
              glmnconfusion$byClass['Recall'], glmnconfusion$byClass['F1'])
nscfinal <- c(nscconfusion$byClass['Sensitivity'], nscconfusion$byClass['Specificity'], nscconfusion$by
              nscconfusion$byClass['Recall'], nscconfusion$byClass['F1'])
ldafinal <- c(ldaconfusion$byClass['Sensitivity'], ldaconfusion$byClass['Specificity'], ldaconfusion$by
              ldaconfusion$byClass['Recall'], ldaconfusion$byClass['F1'])
allmodelsfinal <- data.frame(rbind(lrfinal, rffinal, knnfinal, cartfinal, nnetfinal, svmfinal, gbmfinal
names(allmodelsfinal) <- c("Sensitivity", "Specificity", "Precision", "Recall", "F1")
allmodelsfinal
```

```
##           Sensitivity Specificity Precision    Recall        F1
## lrfinal     0.5660377        0.85 0.6666667 0.5660377 0.6122449
## rffinal     0.5660377        0.79 0.5882353 0.5660377 0.5769231
## knnfinal    0.4528302        0.87 0.6486486 0.4528302 0.5333333
## cartfinal   0.4716981        0.74 0.4901961 0.4716981 0.4807692
## nnetfinal   0.5849057        0.87 0.7045455 0.5849057 0.6391753
## svmfinal    0.5471698        0.86 0.6744186 0.5471698 0.6041667
## gbmfinal    0.5094340        0.83 0.6136364 0.5094340 0.5567010
## nscfinal    0.2830189        0.93 0.6818182 0.2830189 0.4000000
## ldafinal    0.5660377        0.85 0.6666667 0.5660377 0.6122449
```

```r
lrfinal2<- c(lrconfusion$overall['Accuracy'],lrconfusion$byClass['Sensitivity'], lrconfusion$byClass['Sp
            lrconfusion$byClass['Recall'], lrconfusion$byClass['F1'])
nnetfinal2 <- c(nnetconfusion$overall['Accuracy'],nnetconfusion$byClass['Sensitivity'], nnetconfusion$by
               nnetconfusion$byClass['Recall'], nnetconfusion$byClass['F1'])
knnfinal2 <- c(knnconfusion$overall['Accuracy'],knnconfusion$byClass['Sensitivity'], knnconfusion$byCla
              knnconfusion$byClass['Recall'], knnconfusion$byClass['F1'])
cartfinal2 <- c(cartconfusion$overall['Accuracy'],cartconfusion$byClass['Sensitivity'], cartconfusion$by
               cartconfusion$byClass['Recall'], cartconfusion$byClass['F1'])
allmodelsfinal2 <- data.frame(rbind(lrfinal2, nnetfinal2, knnfinal2, cartfinal2))
names(allmodelsfinal2) <- c("Accuracy","Sensitivity", "Specificity", "Precision", "Recall", "F1")
allmodelsfinal
```

```
##           Sensitivity Specificity Precision    Recall        F1
## lrfinal     0.5660377        0.85 0.6666667 0.5660377 0.6122449
## rffinal     0.5660377        0.79 0.5882353 0.5660377 0.5769231
## knnfinal    0.4528302        0.87 0.6486486 0.4528302 0.5333333
## cartfinal   0.4716981        0.74 0.4901961 0.4716981 0.4807692
## nnetfinal   0.5849057        0.87 0.7045455 0.5849057 0.6391753
## svmfinal    0.5471698        0.86 0.6744186 0.5471698 0.6041667
## gbmfinal    0.5094340        0.83 0.6136364 0.5094340 0.5567010
## nscfinal    0.2830189        0.93 0.6818182 0.2830189 0.4000000
## ldafinal    0.5660377        0.85 0.6666667 0.5660377 0.6122449
```

```r
allmodelsfinal2
```

```
##             Accuracy Sensitivity Specificity Precision    Recall        F1
## lrfinal2   0.7516340   0.5660377        0.85 0.6666667 0.5660377 0.6122449
## nnetfinal2 0.7712418   0.5849057        0.87 0.7045455 0.5849057 0.6391753
## knnfinal2  0.7254902   0.4528302        0.87 0.6486486 0.4528302 0.5333333
## cartfinal2 0.6470588   0.4716981        0.74 0.4901961 0.4716981 0.4807692
```

```r
lrfinal3<- c(lrconfusion$overall['Accuracy'],lrconfusion$byClass['Sensitivity'], lrconfusion$byClass['Sp
            lrconfusion$byClass['Recall'], lrconfusion$byClass['F1'])
rffinal3 <- c(rfconfusion$overall['Accuracy'],rfconfusion$byClass['Sensitivity'], rfconfusion$byClass['S
             rfconfusion$byClass['Recall'], rfconfusion$byClass['F1'])
knnfinal3 <- c(knnconfusion$overall['Accuracy'],knnconfusion$byClass['Sensitivity'], knnconfusion$byCla
              knnconfusion$byClass['Recall'], knnconfusion$byClass['F1'])
cartfinal3 <- c(cartconfusion$overall['Accuracy'],cartconfusion$byClass['Sensitivity'], cartconfusion$by
               cartconfusion$byClass['Recall'], cartconfusion$byClass['F1'])
nnetfinal3 <- c(nnetconfusion$overall['Accuracy'],nnetconfusion$byClass['Sensitivity'], nnetconfusion$by
               nnetconfusion$byClass['Recall'], nnetconfusion$byClass['F1'])
svmfinal3 <- c(svmconfusion$overall['Accuracy'],svmconfusion$byClass['Sensitivity'], svmconfusion$byCla
              svmconfusion$byClass['Recall'], svmconfusion$byClass['F1'])
gbmfinal3 <- c(gbmconfusion$overall['Accuracy'],gbmconfusion$byClass['Sensitivity'], gbmconfusion$byCla
              gbmconfusion$byClass['Recall'], gbmconfusion$byClass['F1'])
glmnfinal3 <- c(glmnconfusion$overall['Accuracy'],glmnconfusion$byClass['Sensitivity'], glmnconfusion$by
              glmnconfusion$byClass['Recall'], glmnconfusion$byClass['F1'])
nscfinal3 <- c(nscconfusion$overall['Accuracy'],nscconfusion$byClass['Sensitivity'], nscconfusion$byCla
              nscconfusion$byClass['Recall'], nscconfusion$byClass['F1'])
ldafinal3 <- c(ldaconfusion$overall['Accuracy'],ldaconfusion$byClass['Sensitivity'], ldaconfusion$byCla
              ldaconfusion$byClass['Recall'], ldaconfusion$byClass['F1'])
allmodelsfinal3 <- data.frame(rbind(lrfinal3, rffinal3, knnfinal3, cartfinal3, nnetfinal3, svmfinal3, gl
```

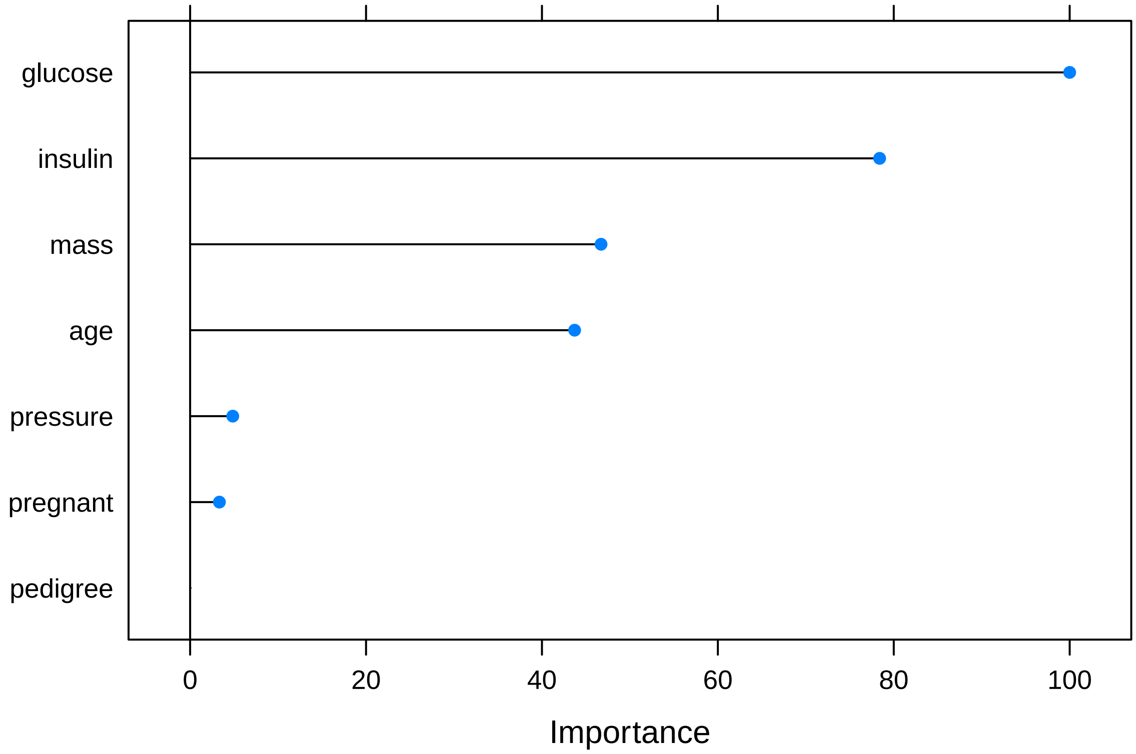
```r
names(allmodelsfinal3) <- c("Accuracy","Sensitivity", "Specificity", "Precision", "Recall", "F1")
allmodelsfinal3
```

```
##             Accuracy Sensitivity Specificity Precision    Recall        F1
## lrfinal3   0.7516340   0.5660377        0.85 0.6666667 0.5660377 0.6122449
## rffinal3   0.7124183   0.5660377        0.79 0.5882353 0.5660377 0.5769231
## knnfinal3  0.7254902   0.4528302        0.87 0.6486486 0.4528302 0.5333333
## cartfinal3 0.6470588   0.4716981        0.74 0.4901961 0.4716981 0.4807692
## nnetfinal3 0.7712418   0.5849057        0.87 0.7045455 0.5849057 0.6391753
## svmfinal3  0.7516340   0.5471698        0.86 0.6744186 0.5471698 0.6041667
## gbmfinal3  0.7189542   0.5094340        0.83 0.6136364 0.5094340 0.5567010
## nscfinal3  0.7058824   0.2830189        0.93 0.6818182 0.2830189 0.4000000
## ldafinal3  0.7516340   0.5660377        0.85 0.6666667 0.5660377 0.6122449
```

```r
#To find the Most Important Predictors within the Diabetes Dataset from within the Average Neural Netwo
set.seed(1)
nnetImp <- caret::varImp(nnet_train_data, importance=TRUE)
nnetImp
```

```
## ROC curve variable importance
##
##          Importance
## glucose     100.000
## insulin      78.391
## mass         46.718
## age          43.712
## pressure      4.849
## pregnant      3.332
## pedigree      0.000
```

```r
plot(caret::varImp(nnet_train_data)) #plot based on the univariate ROC curves generated using
```


glucose
insulin
mass
age
pressure
pregnant
pedigree
0
20
40
60
80
100
Importance