Final Project

Group 1

2/12/2022

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

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

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

```
#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.

```
# 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(zerO)
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.   : 14.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.: 86.78
```

```
##  Median : 3.000   Median :117.0   Median : 72.00   Median :134.52
##  Mean   : 3.845   Mean   :121.6   Mean   : 72.32   Mean   :155.08
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:191.75
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :846.00
##       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   :32.45   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?

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

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

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

```
## [1] 0.5302425
```

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

```
## [1] 0.1542848
```

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

```
## [1] 2.169879
```

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

```
## [1] 0.5964305
```

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

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

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

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

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

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

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

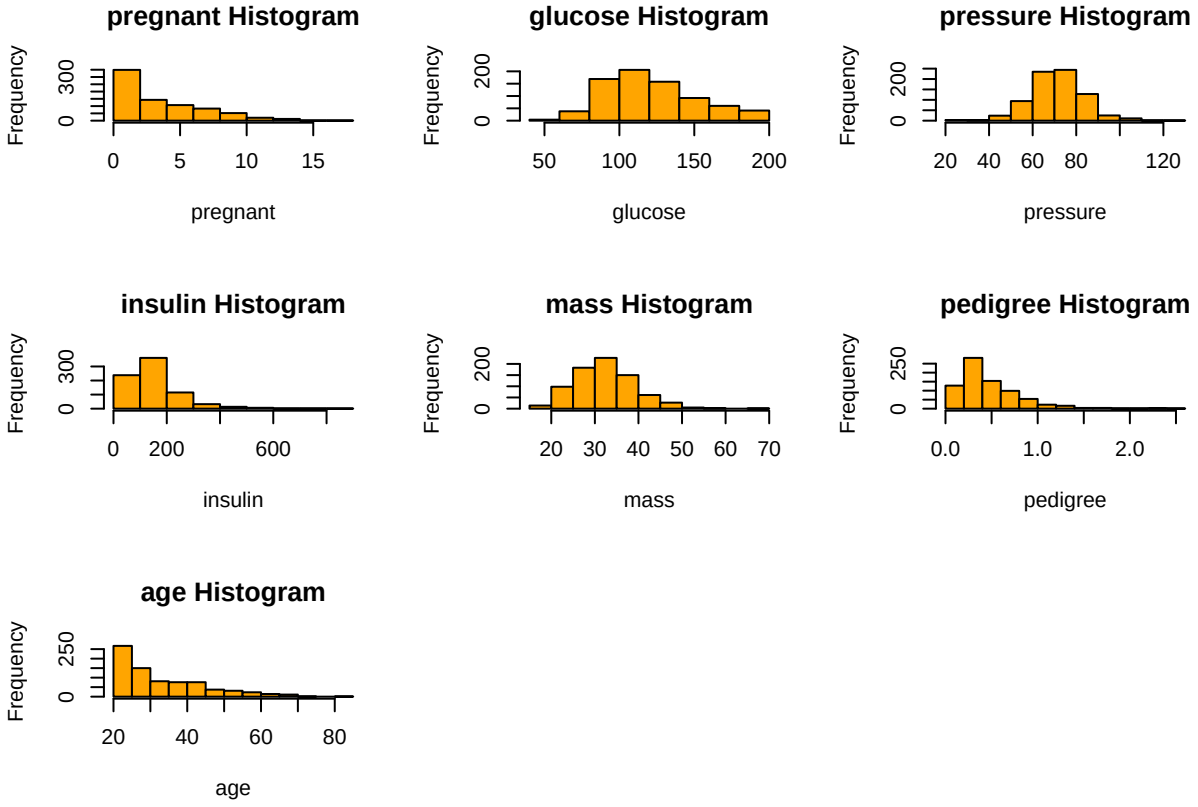
```
## [1] 0.1137321
```

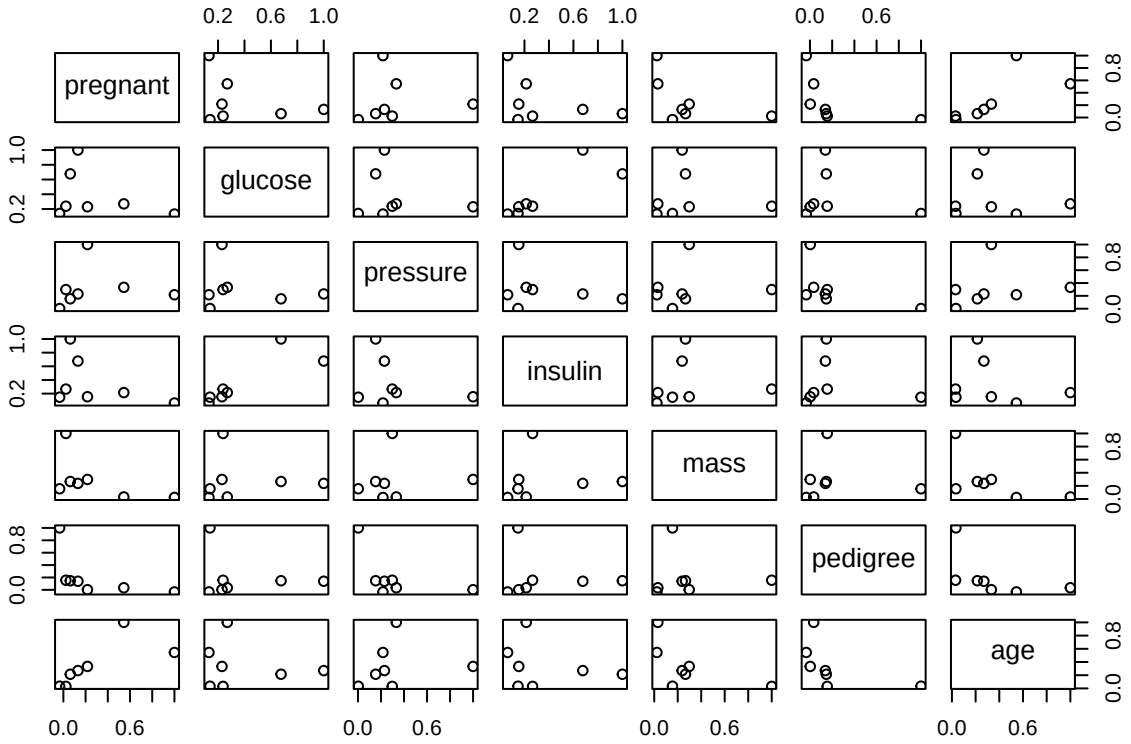
```
skewness(log(df$insulin))
```

```
## [1] -0.2351584
```

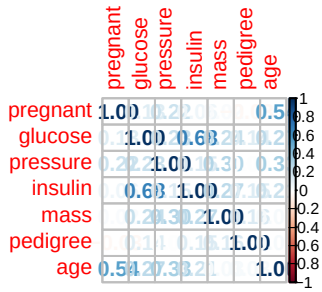
Graphical Review of data

```
#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)
```

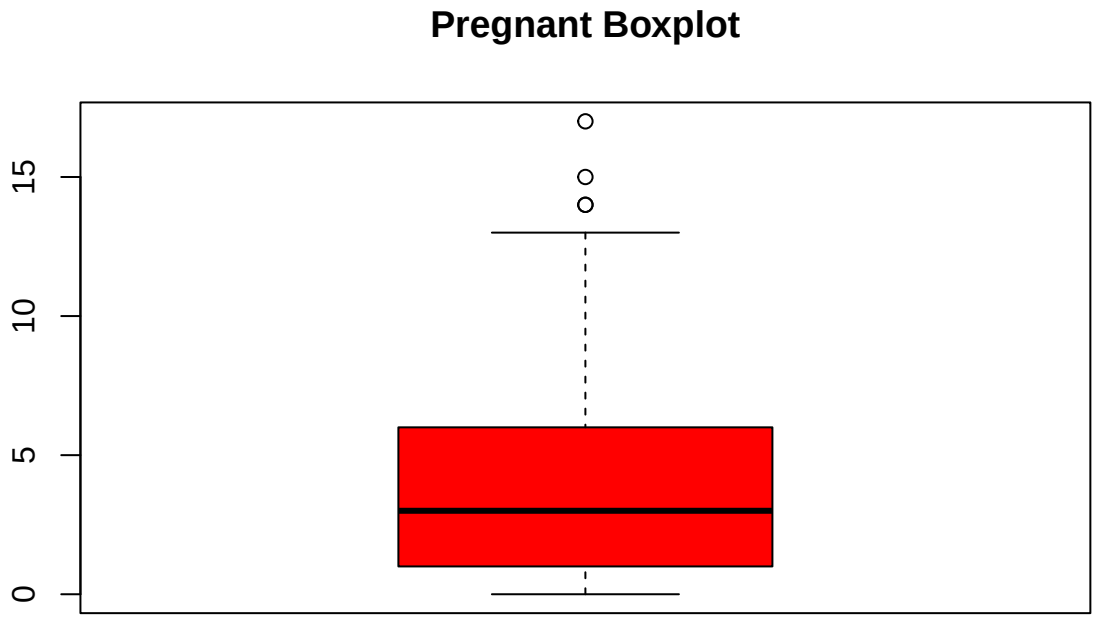




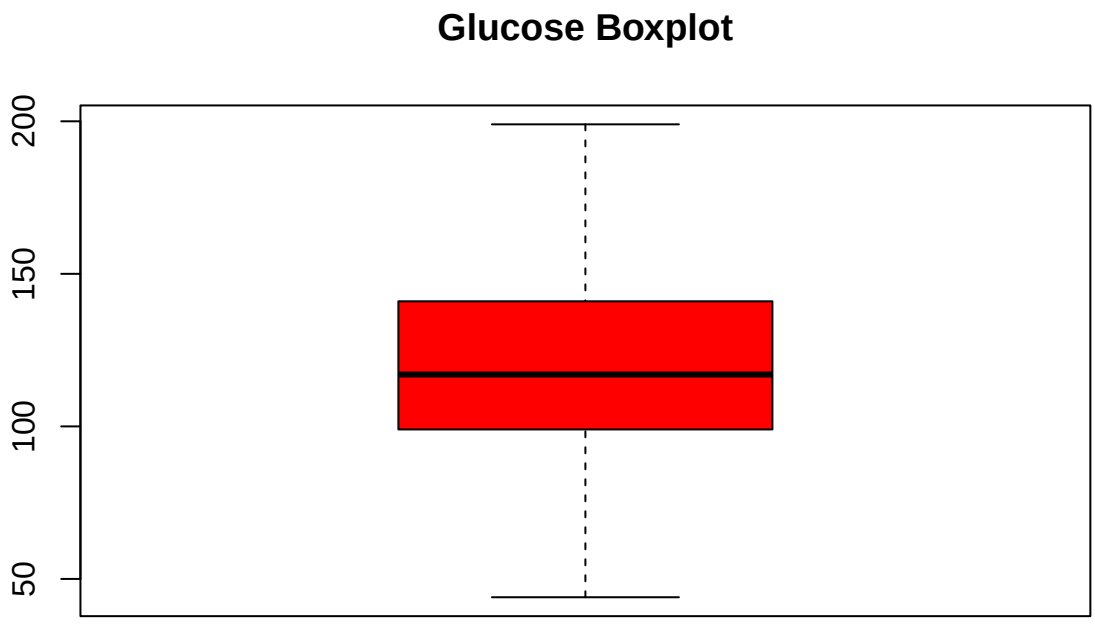

```
corrplot(x, method="number")
```



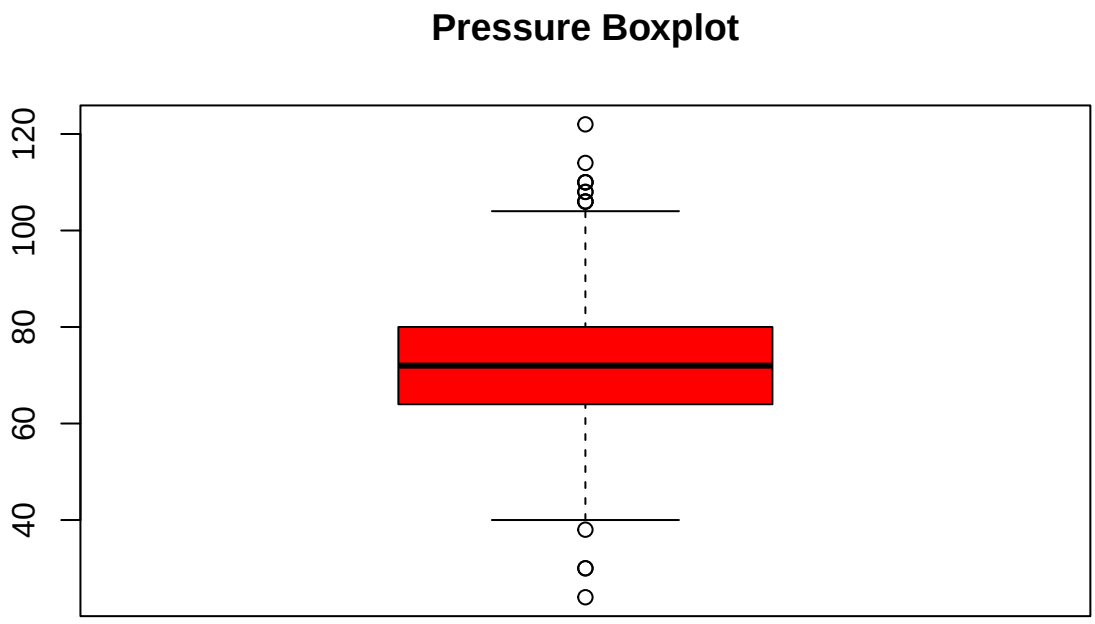

```r
#Box Plots of Diabetes: Predictor Variables
boxplot(df$pregnant, main = "Pregnant Boxplot", col = "red")
```



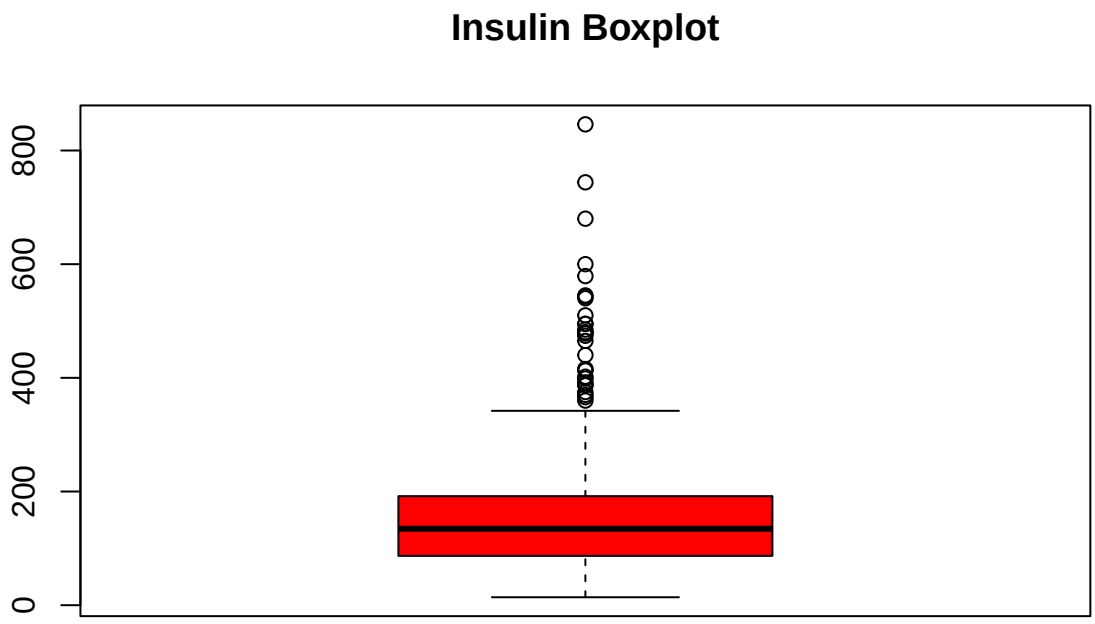

```
boxplot(df$glucose, main = "Glucose Boxplot", col = "red")
```



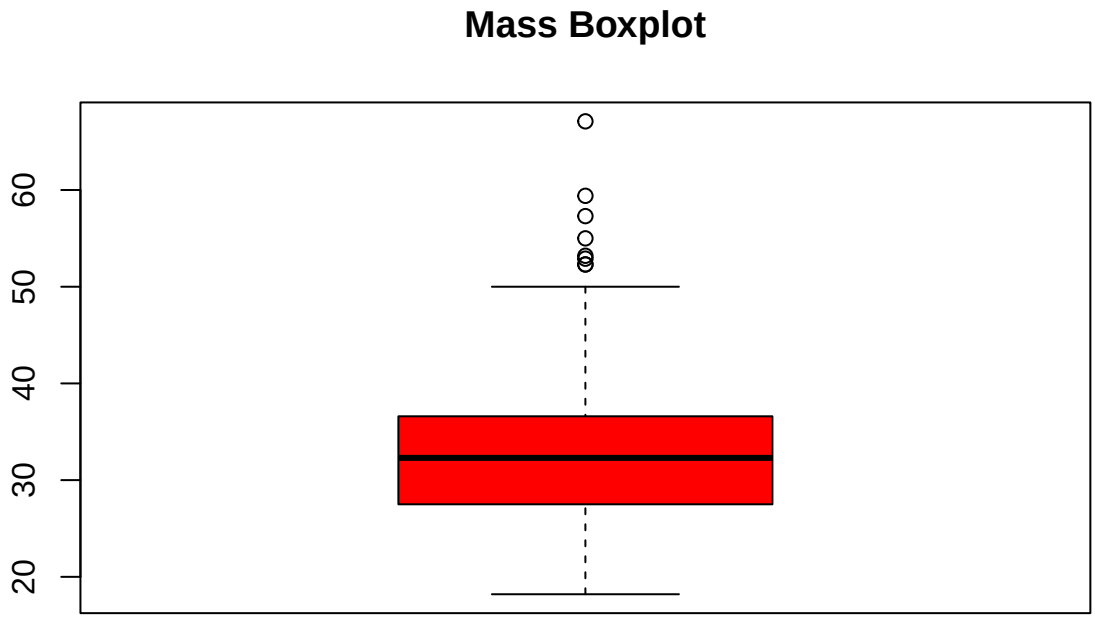

```
boxplot(df$pressure, main = "Pressure Boxplot", col = "red")
```



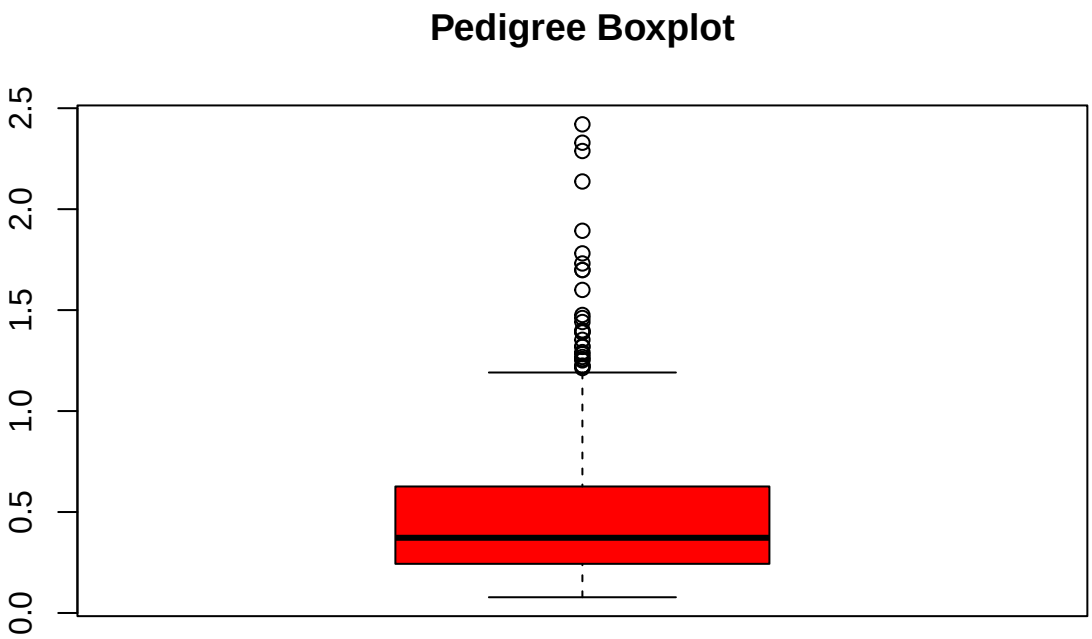

```
#boxplot(df$triceps, main = "Triceps Boxplot", col = "red")
boxplot(df$insulin, main = "Insulin Boxplot", col = "red")
```



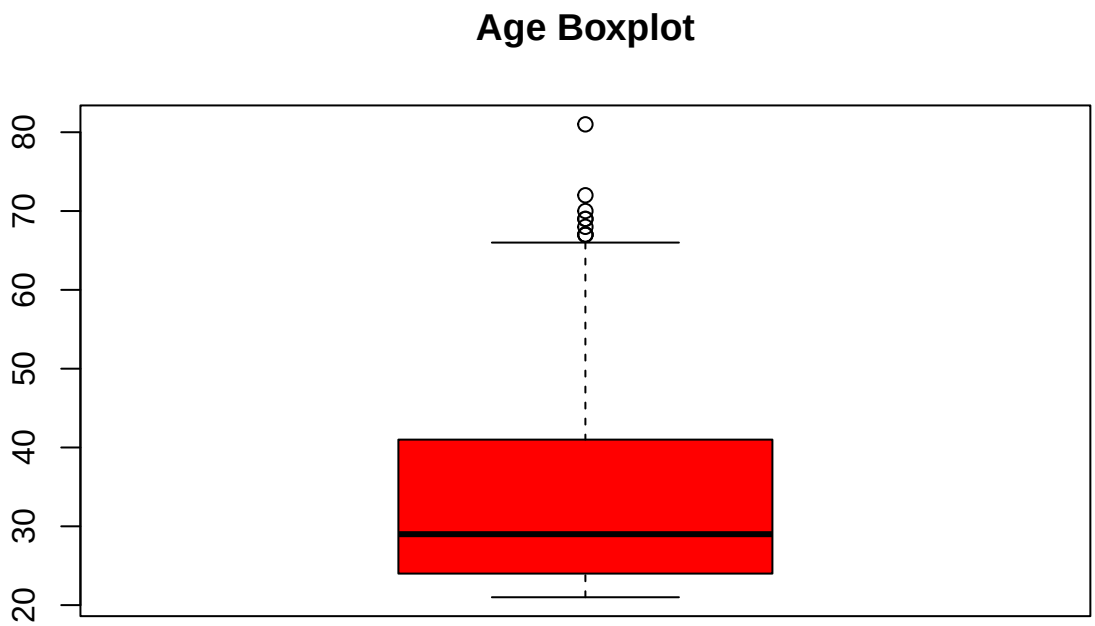

```
boxplot(df$mass, main = "Mass Boxplot", col = "red")
```




```
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(100)
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.   : 56.0   Min.   : 24.00   Min.   : 15.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.: 86.78
##  Median : 3.000   Median :117.0   Median : 72.00   Median :134.52
##  Mean   : 3.881   Mean   :121.8   Mean   : 72.54   Mean   :154.97
##  3rd Qu.: 6.000   3rd Qu.:140.0   3rd Qu.: 80.00   3rd Qu.:190.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :846.00
##       mass          pedigree           age         diabetes
##  Min.   :18.20   Min.   :0.0780   Min.   :21.00   neg:400
##  1st Qu.:27.60   1st Qu.:0.2370   1st Qu.:24.00   pos:215
```

```
##  Median :32.10   Median :0.3640   Median :29.00
##  Mean   :32.60   Mean   :0.4647   Mean   :33.41
##  3rd Qu.:36.85   3rd Qu.:0.6110   3rd Qu.:41.00
##  Max.   :67.10   Max.   :2.2880   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
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, 0.8, 0.1, 0.1, -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: 553, 553, 554, 553, 554, 554, ...
## Resampling results:
##
##   ROC        Sens  Spec
##   0.8507359  0.88  0.608658
```

```
summary(lr_train_data)
```

```
##
## Call:
## NULL
##
```

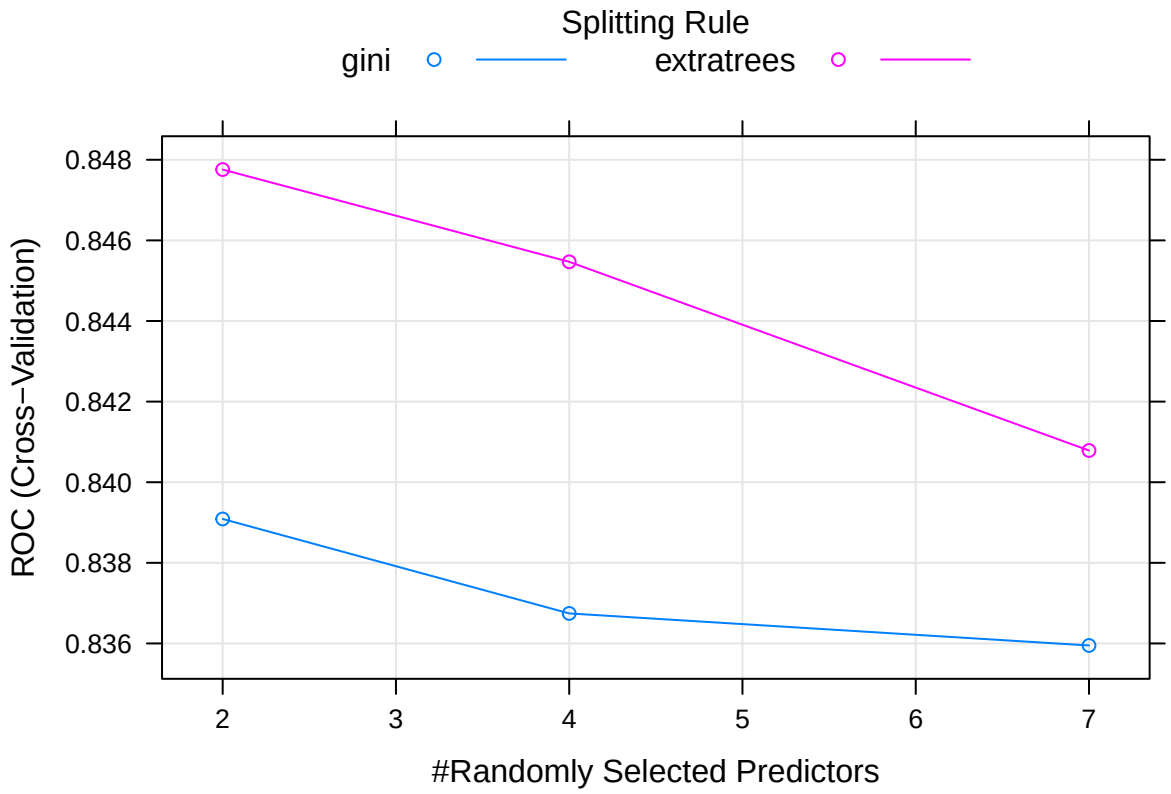
```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5188  -0.6817  -0.3342   0.6605   2.5871
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.98579    0.11756  -8.385  < 2e-16 ***
## pregnant     0.30478    0.13070   2.332 0.019701 *
## glucose      1.20776    0.16983   7.111 1.15e-12 ***
## pressure    -0.16588    0.12037  -1.378 0.168200
## insulin     -0.01901    0.16527  -0.115 0.908444
## mass         0.75482    0.12982   5.814 6.08e-09 ***
## pedigree     0.37903    0.11093   3.417 0.000633 ***
## age          0.38721    0.14329   2.702 0.006886 **
## ---
## 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: 541.68  on 607  degrees of freedom
## AIC: 557.68
##
## Number of Fisher Scoring iterations: 5
```

```
#Random Forest: Training Model
rf_train_data <- caret::train(diabetes ~., data = train_data,
                              method = "ranger",
                              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, 553, 554, 553, 553, 553, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens    Spec
##   2     gini        0.8390882  0.8425  0.6233766
##   2     extratrees  0.8477570  0.8675  0.6093074
##   4     gini        0.8367451  0.8400  0.6370130
##   4     extratrees  0.8454681  0.8550  0.6235931
##   7     gini        0.8359497  0.8375  0.6744589
##   7     extratrees  0.8407873  0.8600  0.6422078
##
## Tuning parameter 'min.node.size' 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 mtry = 2, splitrule = extratrees
##  and min.node.size = 1.
```

```
plot(rf_train_data)
```

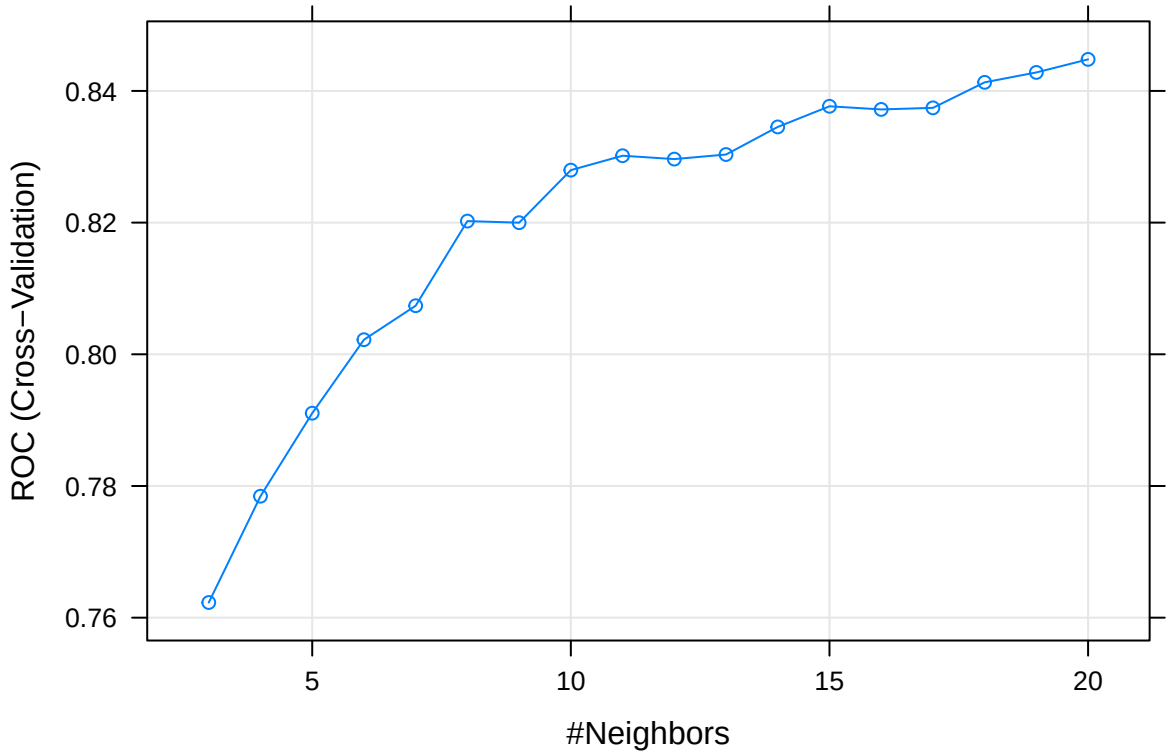



```
FinalTree = rf_train_data$finalModel$importance.mode


#K Nearest Neighbor: Training Model
knn_train_data <- caret::train(diabetes ~., data = train_data,
                           method = "knn",
                           metric = "ROC",
                           tuneGrid = expand.grid(.k = c(3:20)),
                           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, 554, 553, 553, 554, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens    Spec
##    3  0.7623052  0.8250  0.5952381
##    4  0.7784497  0.8175  0.5813853
##    5  0.7910552  0.8250  0.5816017
##    6  0.8022159  0.8275  0.5867965
##    7  0.8073755  0.8325  0.6153680
##    8  0.8202327  0.8500  0.6199134
##    9  0.8199892  0.8475  0.6101732
##   10  0.8279708  0.8675  0.5967532
##   11  0.8301569  0.8625  0.6006494
##   12  0.8296483  0.8675  0.6056277
##   13  0.8303382  0.8650  0.6101732
##   14  0.8345319  0.8625  0.5915584
##   15  0.8376650  0.8675  0.5917749
##   16  0.8371889  0.8700  0.5872294
##   17  0.8374269  0.8750  0.5820346
##   18  0.8413014  0.8600  0.5872294
##   19  0.8428111  0.8725  0.5965368
##   20  0.8447917  0.8650  0.6099567
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

```
plot(knn_train_data)
```




```
#Classification and Regression Trees (CART): Training Model
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: 553, 554, 553, 554, 553, 554, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens    Spec
##   0.00000000  0.7706899  0.8075  0.5917749
##   0.01272950  0.7558658  0.8325  0.6036797
##   0.02545900  0.7260173  0.8600  0.5385281
```

```
##   0.03818849  0.7359740  0.8550  0.5571429
##   0.05091799  0.7361472  0.8500  0.5664502
##   0.06364749  0.7379654  0.8550  0.5482684
##   0.07637699  0.7235714  0.8000  0.6216450
##   0.08910649  0.7235714  0.8000  0.6216450
##   0.10183599  0.7192532  0.7850  0.6307359
##   0.11456548  0.7143128  0.7550  0.6593074
##   0.12729498  0.7143128  0.7550  0.6593074
##   0.14002448  0.7137446  0.7500  0.6774892
##   0.15275398  0.7137446  0.7500  0.6774892
##   0.16548348  0.7137446  0.7500  0.6774892
##   0.17821297  0.7137446  0.7500  0.6774892
##   0.19094247  0.7137446  0.7500  0.6774892
##   0.20367197  0.7137446  0.7500  0.6774892
##   0.21640147  0.7137446  0.7500  0.6774892
##   0.22913097  0.6334037  0.8075  0.4593074
##   0.24186047  0.6334037  0.8075  0.4593074
##
## 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
)
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
```

```
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, 553, 553, 554, 553, 554, ...
## Resampling results across tuning parameters:
##
##   decay  size  ROC        Sens    Spec
##   0.00    1    0.8486418  0.8600  0.6290043
##   0.00    2    0.8508063  0.8750  0.6151515
##   0.00    3    0.8408874  0.8725  0.6290043
##   0.00    4    0.8211851  0.8550  0.6051948
##   0.00    5    0.8306494  0.8450  0.6285714
##   0.00    6    0.8270400  0.8600  0.6385281
##   0.00    7    0.8195779  0.8425  0.6047619
##   0.00    8    0.8030790  0.8475  0.5816017
##   0.00    9    0.8235444  0.8350  0.6149351
##   0.00   10          NaN     NaN        NaN
##   0.01    1    0.8491180  0.8625  0.6290043
##   0.01    2    0.8525649  0.8725  0.6294372
##   0.01    3    0.8520292  0.8675  0.6151515
##   0.01    4    0.8371050  0.8625  0.6292208
##   0.01    5    0.8335390  0.8600  0.6190476
##   0.01    6    0.8370887  0.8650  0.6043290
##   0.01    7    0.8328301  0.8375  0.6101732
##   0.01    8    0.8261255  0.8625  0.6054113
##   0.01    9    0.8102273  0.8350  0.5820346
##   0.01   10          NaN     NaN        NaN
##   0.10    1    0.8495779  0.8625  0.6290043
##   0.10    2    0.8520509  0.8700  0.6287879
##   0.10    3    0.8474838  0.8550  0.6145022
##   0.10    4    0.8397132  0.8675  0.6002165
##   0.10    5    0.8350649  0.8700  0.5820346
##   0.10    6    0.8398755  0.8600  0.5911255
##   0.10    7    0.8237229  0.8500  0.5725108
##   0.10    8    0.8088366  0.8500  0.6140693
##   0.10    9    0.8238528  0.8375  0.5913420
##   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 ######################

svmFit <- train(diabetes ~., data = train_data,
                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  0.8343939  0.8750  0.5813853
##      0.50  0.8365584  0.8750  0.5722944
##      1.00  0.8320779  0.8700  0.5675325
##      2.00  0.8208009  0.8675  0.5725108
##      4.00  0.8108658  0.8650  0.5491342
##      8.00  0.7954329  0.8725  0.5259740
##     16.00  0.7757955  0.8575  0.4937229
##     32.00  0.7652706  0.8775  0.4331169
##     64.00  0.7497078  0.8825  0.4099567
##    128.00  0.7513528  0.8700  0.4145022
##    256.00  0.7452868  0.9000  0.3445887
##    512.00  0.7442154  0.9000  0.3307359
##   1024.00  0.7456277  0.8850  0.3588745
##   2048.00  0.7416071  0.8725  0.4099567
##
## Tuning parameter 'sigma' was held constant at a value of 0.1173803
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1173803 and C = 0.5.
```

```
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)

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.
```

```
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: 553, 554, 554, 554, 553, 553, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  ROC        Sens    Spec
##   0.01       1                   100     0.8194291  0.9400  0.3822511
##   0.01       1                   150     0.8287744  0.9175  0.4517316
##   0.01       1                   200     0.8312879  0.8975  0.4841991
##   0.01       1                   250     0.8357278  0.8950  0.5307359
##   0.01       1                   300     0.8385146  0.8875  0.5352814
##   0.01       1                   350     0.8393696  0.8875  0.5398268
##   0.01       1                   400     0.8423755  0.8850  0.5398268
##   0.01       1                   450     0.8437608  0.8850  0.5491342
##   0.01       1                   500     0.8442587  0.8850  0.5493506
##   0.01       1                   550     0.8447240  0.8800  0.5536797
##   0.01       1                   600     0.8447294  0.8775  0.5675325
##   0.01       1                   650     0.8444913  0.8725  0.5770563
##   0.01       1                   700     0.8454275  0.8725  0.5816017
##   0.01       1                   750     0.8456656  0.8650  0.5816017
##   0.01       1                   800     0.8462500  0.8625  0.5816017
##   0.01       1                   850     0.8457955  0.8675  0.5954545
##   0.01       1                   900     0.8457035  0.8675  0.5909091
##   0.01       1                   950     0.8461851  0.8675  0.5952381
##   0.01       1                  1000     0.8454004  0.8650  0.6049784
##   0.01       3                   100     0.8413636  0.9125  0.4978355
##   0.01       3                   150     0.8438853  0.8925  0.5303030
##   0.01       3                   200     0.8463528  0.8825  0.5582251
##   0.01       3                   250     0.8463745  0.8775  0.5816017
##   0.01       3                   300     0.8466180  0.8750  0.5909091
##   0.01       3                   350     0.8460660  0.8650  0.6004329
##   0.01       3                   400     0.8457359  0.8625  0.6188312
##   0.01       3                   450     0.8463582  0.8650  0.6188312
##   0.01       3                   500     0.8449838  0.8625  0.6329004
##   0.01       3                   550     0.8458117  0.8625  0.6374459
##   0.01       3                   600     0.8437392  0.8575  0.6419913
##   0.01       3                   650     0.8445292  0.8650  0.6422078
##   0.01       3                   700     0.8423593  0.8625  0.6374459
##   0.01       3                   750     0.8428301  0.8600  0.6467532
##   0.01       3                   800     0.8423810  0.8600  0.6467532
##   0.01       3                   850     0.8418344  0.8550  0.6469697
##   0.01       3                   900     0.8410227  0.8575  0.6422078
##   0.01       3                   950     0.8393831  0.8500  0.6374459
##   0.01       3                  1000     0.8386688  0.8500  0.6422078
##   0.01       5                   100     0.8452110  0.9100  0.5071429
##   0.01       5                   150     0.8454329  0.8775  0.5532468
```

```
##   0.01       5                  200     0.8441667  0.8750  0.5816017
##   0.01       5                  250     0.8467641  0.8650  0.6006494
##   0.01       5                  300     0.8445887  0.8625  0.6097403
##   0.01       5                  350     0.8448593  0.8625  0.6190476
##   0.01       5                  400     0.8433063  0.8525  0.6188312
##   0.01       5                  450     0.8437879  0.8500  0.6324675
##   0.01       5                  500     0.8418236  0.8450  0.6279221
##   0.01       5                  550     0.8407846  0.8475  0.6372294
##   0.01       5                  600     0.8396591  0.8475  0.6324675
##   0.01       5                  650     0.8394426  0.8425  0.6372294
##   0.01       5                  700     0.8385281  0.8425  0.6419913
##   0.01       5                  750     0.8377381  0.8350  0.6329004
##   0.01       5                  800     0.8364665  0.8325  0.6326840
##   0.01       5                  850     0.8350433  0.8325  0.6419913
##   0.01       5                  900     0.8338799  0.8325  0.6417749
##   0.01       5                  950     0.8335606  0.8325  0.6372294
##   0.01       5                 1000     0.8339286  0.8325  0.6326840
##   0.01       7                  100     0.8461634  0.9075  0.5303030
##   0.01       7                  150     0.8422348  0.8850  0.5818182
##   0.01       7                  200     0.8448052  0.8725  0.5956710
##   0.01       7                  250     0.8445833  0.8625  0.6142857
##   0.01       7                  300     0.8447024  0.8525  0.6372294
##   0.01       7                  350     0.8438312  0.8500  0.6419913
##   0.01       7                  400     0.8440693  0.8425  0.6512987
##   0.01       7                  450     0.8425703  0.8450  0.6467532
##   0.01       7                  500     0.8407251  0.8450  0.6422078
##   0.01       7                  550     0.8384307  0.8475  0.6558442
##   0.01       7                  600     0.8389177  0.8450  0.6603896
##   0.01       7                  650     0.8370292  0.8400  0.6558442
##   0.01       7                  700     0.8355249  0.8400  0.6603896
##   0.01       7                  750     0.8334416  0.8375  0.6649351
##   0.01       7                  800     0.8319426  0.8425  0.6790043
##   0.01       7                  850     0.8308712  0.8350  0.6649351
##   0.01       7                  900     0.8300379  0.8375  0.6787879
##   0.01       7                  950     0.8291342  0.8375  0.6742424
##   0.01       7                 1000     0.8285768  0.8350  0.6742424
##   0.10       1                  100     0.8512067  0.8650  0.6051948
##   0.10       1                  150     0.8455844  0.8650  0.6238095
##   0.10       1                  200     0.8453084  0.8600  0.6512987
##   0.10       1                  250     0.8402219  0.8575  0.6608225
##   0.10       1                  300     0.8383496  0.8600  0.6424242
##   0.10       1                  350     0.8367857  0.8525  0.6471861
##   0.10       1                  400     0.8378139  0.8600  0.6560606
##   0.10       1                  450     0.8328517  0.8475  0.6469697
##   0.10       1                  500     0.8343939  0.8600  0.6658009
##   0.10       1                  550     0.8312392  0.8525  0.6378788
##   0.10       1                  600     0.8287284  0.8425  0.6374459
##   0.10       1                  650     0.8247998  0.8475  0.6142857
##   0.10       1                  700     0.8252002  0.8500  0.6329004
##   0.10       1                  750     0.8271212  0.8450  0.6285714
##   0.10       1                  800     0.8225487  0.8400  0.6093074
##   0.10       1                  850     0.8240639  0.8475  0.6190476
##   0.10       1                  900     0.8210552  0.8500  0.6190476
##   0.10       1                  950     0.8250920  0.8475  0.6235931
```

```
##   0.10       1                1000      0.8224513  0.8425  0.6419913
##   0.10       3                 100      0.8333604  0.8500  0.6701299
##   0.10       3                 150      0.8315422  0.8425  0.6601732
##   0.10       3                 200      0.8251190  0.8300  0.6419913
##   0.10       3                 250      0.8235173  0.8250  0.6422078
##   0.10       3                 300      0.8161364  0.8275  0.6374459
##   0.10       3                 350      0.8139123  0.8100  0.6326840
##   0.10       3                 400      0.8104924  0.8125  0.6372294
##   0.10       3                 450      0.8066342  0.8175  0.5954545
##   0.10       3                 500      0.8045671  0.8175  0.6047619
##   0.10       3                 550      0.8031656  0.8100  0.6372294
##   0.10       3                 600      0.7994805  0.8125  0.6231602
##   0.10       3                 650      0.7975812  0.8050  0.6138528
##   0.10       3                 700      0.7975379  0.8075  0.5954545
##   0.10       3                 750      0.7951569  0.8100  0.5952381
##   0.10       3                 800      0.7953788  0.8150  0.5859307
##   0.10       3                 850      0.7967965  0.8175  0.6000000
##   0.10       3                 900      0.7945779  0.8025  0.5766234
##   0.10       3                 950      0.7925162  0.7950  0.5859307
##   0.10       3                1000      0.7926190  0.8000  0.5952381
##   0.10       5                 100      0.8338528  0.8250  0.6283550
##   0.10       5                 150      0.8237771  0.8275  0.6419913
##   0.10       5                 200      0.8237067  0.8325  0.6380952
##   0.10       5                 250      0.8159794  0.8250  0.6049784
##   0.10       5                 300      0.8169751  0.8250  0.6097403
##   0.10       5                 350      0.8105844  0.8375  0.5958874
##   0.10       5                 400      0.8071483  0.8325  0.6056277
##   0.10       5                 450      0.8112987  0.8275  0.6099567
##   0.10       5                 500      0.8102868  0.8350  0.6329004
##   0.10       5                 550      0.8082089  0.8250  0.6140693
##   0.10       5                 600      0.8073052  0.8150  0.6283550
##   0.10       5                 650      0.8064123  0.8200  0.6285714
##   0.10       5                 700      0.8018669  0.8225  0.6097403
##   0.10       5                 750      0.8030195  0.8275  0.6238095
##   0.10       5                 800      0.8035444  0.8225  0.6333333
##   0.10       5                 850      0.8012175  0.8275  0.6285714
##   0.10       5                 900      0.8000649  0.8250  0.6101732
##   0.10       5                 950      0.8013203  0.8325  0.6145022
##   0.10       5                1000      0.7996591  0.8175  0.6006494
##   0.10       7                 100      0.8084524  0.8425  0.6186147
##   0.10       7                 150      0.8093831  0.8375  0.6186147
##   0.10       7                 200      0.8034307  0.8300  0.6097403
##   0.10       7                 250      0.8015476  0.8150  0.6183983
##   0.10       7                 300      0.8025325  0.8200  0.6469697
##   0.10       7                 350      0.8043344  0.8125  0.6235931
##   0.10       7                 400      0.8025433  0.8150  0.6190476
##   0.10       7                 450      0.7988095  0.8100  0.6051948
##   0.10       7                 500      0.7993236  0.8150  0.6049784
##   0.10       7                 550      0.7978247  0.8075  0.6140693
##   0.10       7                 600      0.7998160  0.8150  0.6145022
##   0.10       7                 650      0.8004816  0.8100  0.6238095
##   0.10       7                 700      0.8026515  0.8100  0.6190476
##   0.10       7                 750      0.8024351  0.8175  0.6238095
##   0.10       7                 800      0.8023647  0.8125  0.6331169
```

```
##   0.10       7                   850     0.8022294  0.8175  0.6194805
##   0.10       7                   900     0.8009145  0.8225  0.6051948
##   0.10       7                   950     0.8015314  0.8175  0.6099567
##   0.10       7                  1000     0.8019643  0.8150  0.6099567
##
## 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 = 100, interaction.depth =
##  1, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
################## Elastinet ####################
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(.01, .2, length = 40))

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, 553, 554, 553, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      ROC        Sens    Spec
##   0.0    0.01000000  0.8559578  0.8800  0.58290043
##   0.0    0.01487179  0.8559578  0.8800  0.58290043
##   0.0    0.01974359  0.8559578  0.8800  0.58290043
##   0.0    0.02461538  0.8559578  0.8800  0.58290043
##   0.0    0.02948718  0.8553571  0.8825  0.57835498
##   0.0    0.03435897  0.8553517  0.8850  0.57359307
##   0.0    0.03923077  0.8548972  0.8875  0.57359307
##   0.0    0.04410256  0.8552381  0.8875  0.57359307
##   0.0    0.04897436  0.8550054  0.8900  0.56904762
##   0.0    0.05384615  0.8545400  0.8900  0.56904762
##   0.0    0.05871795  0.8545455  0.8925  0.56428571
##   0.0    0.06358974  0.8538582  0.8925  0.55952381
##   0.0    0.06846154  0.8526948  0.8975  0.55952381
##   0.0    0.07333333  0.8526948  0.8975  0.55952381
##   0.0    0.07820513  0.8523323  0.8975  0.55952381
##   0.0    0.08307692  0.8515314  0.8950  0.55952381
##   0.0    0.08794872  0.8513041  0.8950  0.55952381
##   0.0    0.09282051  0.8510714  0.8975  0.55497835
```

```
##   0.0    0.09769231  0.8508387  0.8975  0.55497835
##   0.0    0.10256410  0.8500216  0.8975  0.55497835
##   0.0    0.10743590  0.8496861  0.9000  0.55497835
##   0.0    0.11230769  0.8497998  0.9000  0.55021645
##   0.0    0.11717949  0.8496807  0.9000  0.55021645
##   0.0    0.12205128  0.8492154  0.9025  0.55021645
##   0.0    0.12692308  0.8493344  0.9050  0.54567100
##   0.0    0.13179487  0.8496861  0.9050  0.54567100
##   0.0    0.13666667  0.8494481  0.9075  0.54090909
##   0.0    0.14153846  0.8496861  0.9075  0.54090909
##   0.0    0.14641026  0.8495725  0.9075  0.54090909
##   0.0    0.15128205  0.8494426  0.9075  0.54090909
##   0.0    0.15615385  0.8494372  0.9100  0.53614719
##   0.0    0.16102564  0.8494372  0.9100  0.53614719
##   0.0    0.16589744  0.8500216  0.9100  0.52705628
##   0.0    0.17076923  0.8500216  0.9125  0.52705628
##   0.0    0.17564103  0.8501407  0.9125  0.51753247
##   0.0    0.18051282  0.8499134  0.9125  0.51277056
##   0.0    0.18538462  0.8497944  0.9125  0.51277056
##   0.0    0.19025641  0.8497890  0.9125  0.51277056
##   0.0    0.19512821  0.8497890  0.9150  0.50800866
##   0.0    0.20000000  0.8494426  0.9150  0.50800866
##   0.1    0.01000000  0.8560444  0.8750  0.60129870
##   0.1    0.01487179  0.8557089  0.8750  0.59675325
##   0.1    0.01974359  0.8557143  0.8800  0.59220779
##   0.1    0.02461538  0.8553680  0.8800  0.58311688
##   0.1    0.02948718  0.8559361  0.8875  0.58311688
##   0.1    0.03435897  0.8559307  0.8875  0.57380952
##   0.1    0.03923077  0.8561580  0.8875  0.57380952
##   0.1    0.04410256  0.8559361  0.8875  0.56926407
##   0.1    0.04897436  0.8559253  0.8875  0.56450216
##   0.1    0.05384615  0.8558063  0.8900  0.56450216
##   0.1    0.05871795  0.8556872  0.8925  0.56450216
##   0.1    0.06358974  0.8552273  0.8950  0.56450216
##   0.1    0.06846154  0.8554545  0.8950  0.55974026
##   0.1    0.07333333  0.8549892  0.8950  0.55497835
##   0.1    0.07820513  0.8549946  0.8975  0.55021645
##   0.1    0.08307692  0.8548755  0.8975  0.55021645
##   0.1    0.08794872  0.8544156  0.8975  0.54567100
##   0.1    0.09282051  0.8534848  0.8975  0.54567100
##   0.1    0.09769231  0.8532522  0.9000  0.54090909
##   0.1    0.10256410  0.8530249  0.9025  0.54090909
##   0.1    0.10743590  0.8530249  0.9025  0.54090909
##   0.1    0.11230769  0.8523323  0.9025  0.54090909
##   0.1    0.11717949  0.8524513  0.9075  0.54090909
##   0.1    0.12205128  0.8518723  0.9075  0.54090909
##   0.1    0.12692308  0.8516396  0.9075  0.53636364
##   0.1    0.13179487  0.8516396  0.9075  0.53181818
##   0.1    0.13666667  0.8515260  0.9100  0.53181818
##   0.1    0.14153846  0.8516396  0.9100  0.53181818
##   0.1    0.14641026  0.8514123  0.9100  0.52705628
##   0.1    0.15128205  0.8515314  0.9125  0.52705628
##   0.1    0.15615385  0.8513041  0.9125  0.52251082
##   0.1    0.16102564  0.8511851  0.9150  0.51298701
```

```
##   0.1    0.16589744  0.8515314  0.9150  0.51298701
##   0.1    0.17076923  0.8512933  0.9150  0.51298701
##   0.1    0.17564103  0.8512933  0.9150  0.50822511
##   0.1    0.18051282  0.8510606  0.9150  0.50346320
##   0.1    0.18538462  0.8507197  0.9150  0.50346320
##   0.1    0.19025641  0.8503680  0.9175  0.50346320
##   0.1    0.19512821  0.8503680  0.9200  0.50346320
##   0.1    0.20000000  0.8500216  0.9225  0.50346320
##   0.2    0.01000000  0.8563907  0.8750  0.60129870
##   0.2    0.01487179  0.8562825  0.8775  0.59220779
##   0.2    0.01974359  0.8558225  0.8800  0.58766234
##   0.2    0.02461538  0.8555844  0.8850  0.58311688
##   0.2    0.02948718  0.8554600  0.8875  0.58311688
##   0.2    0.03435897  0.8553409  0.8875  0.57835498
##   0.2    0.03923077  0.8562771  0.8875  0.56450216
##   0.2    0.04410256  0.8566180  0.8900  0.56450216
##   0.2    0.04897436  0.8563961  0.8900  0.56450216
##   0.2    0.05384615  0.8564881  0.8900  0.56450216
##   0.2    0.05871795  0.8563582  0.8900  0.55974026
##   0.2    0.06358974  0.8559037  0.8900  0.55974026
##   0.2    0.06846154  0.8559037  0.8925  0.54545455
##   0.2    0.07333333  0.8560227  0.8950  0.54545455
##   0.2    0.07820513  0.8555465  0.8950  0.54545455
##   0.2    0.08307692  0.8554275  0.9000  0.54090909
##   0.2    0.08794872  0.8550866  0.9000  0.53636364
##   0.2    0.09282051  0.8549838  0.9025  0.53636364
##   0.2    0.09769231  0.8545184  0.9050  0.53636364
##   0.2    0.10256410  0.8546429  0.9050  0.53636364
##   0.2    0.10743590  0.8545238  0.9050  0.53636364
##   0.2    0.11230769  0.8543994  0.9050  0.53636364
##   0.2    0.11717949  0.8545238  0.9075  0.53636364
##   0.2    0.12205128  0.8543019  0.9075  0.53636364
##   0.2    0.12692308  0.8540693  0.9075  0.53636364
##   0.2    0.13179487  0.8536039  0.9125  0.52705628
##   0.2    0.13666667  0.8533604  0.9125  0.51774892
##   0.2    0.14153846  0.8531331  0.9150  0.51320346
##   0.2    0.14641026  0.8531331  0.9200  0.50865801
##   0.2    0.15128205  0.8525487  0.9225  0.50411255
##   0.2    0.15615385  0.8521970  0.9225  0.49956710
##   0.2    0.16102564  0.8517424  0.9225  0.49956710
##   0.2    0.16589744  0.8510498  0.9225  0.49956710
##   0.2    0.17076923  0.8510444  0.9250  0.49956710
##   0.2    0.17564103  0.8512662  0.9250  0.49480519
##   0.2    0.18051282  0.8512716  0.9250  0.49480519
##   0.2    0.18538462  0.8511634  0.9250  0.49004329
##   0.2    0.19025641  0.8515152  0.9275  0.48528139
##   0.2    0.19512821  0.8509361  0.9275  0.47575758
##   0.2    0.20000000  0.8507035  0.9275  0.47575758
##   0.4    0.01000000  0.8561526  0.8775  0.59199134
##   0.4    0.01487179  0.8561580  0.8775  0.59220779
##   0.4    0.01974359  0.8568398  0.8800  0.58311688
##   0.4    0.02461538  0.8565097  0.8875  0.58290043
##   0.4    0.02948718  0.8568452  0.8900  0.56883117
##   0.4    0.03435897  0.8561364  0.8900  0.56883117
```

```
##   0.4    0.03923077  0.8559091  0.8900  0.55952381
##   0.4    0.04410256  0.8555519  0.8900  0.55952381
##   0.4    0.04897436  0.8553193  0.8900  0.55952381
##   0.4    0.05384615  0.8556764  0.8900  0.55497835
##   0.4    0.05871795  0.8548755  0.8900  0.55021645
##   0.4    0.06358974  0.8543994  0.8925  0.55021645
##   0.4    0.06846154  0.8539286  0.8950  0.54090909
##   0.4    0.07333333  0.8538149  0.8950  0.53636364
##   0.4    0.07820513  0.8533604  0.8975  0.52727273
##   0.4    0.08307692  0.8527922  0.8975  0.52727273
##   0.4    0.08794872  0.8527976  0.9000  0.52727273
##   0.4    0.09282051  0.8529275  0.9050  0.51818182
##   0.4    0.09769231  0.8530411  0.9100  0.50865801
##   0.4    0.10256410  0.8522240  0.9125  0.50411255
##   0.4    0.10743590  0.8522240  0.9125  0.50411255
##   0.4    0.11230769  0.8519968  0.9150  0.50411255
##   0.4    0.11717949  0.8516342  0.9150  0.50411255
##   0.4    0.12205128  0.8503517  0.9200  0.50411255
##   0.4    0.12692308  0.8496537  0.9225  0.49956710
##   0.4    0.13179487  0.8489610  0.9225  0.49956710
##   0.4    0.13666667  0.8476786  0.9250  0.48073593
##   0.4    0.14153846  0.8466342  0.9275  0.47164502
##   0.4    0.14641026  0.8454600  0.9325  0.46233766
##   0.4    0.15128205  0.8446591  0.9350  0.45757576
##   0.4    0.15615385  0.8438582  0.9400  0.45757576
##   0.4    0.16102564  0.8424784  0.9425  0.43419913
##   0.4    0.16589744  0.8414232  0.9425  0.42489177
##   0.4    0.17076923  0.8403788  0.9425  0.41082251
##   0.4    0.17564103  0.8392154  0.9450  0.40627706
##   0.4    0.18051282  0.8392100  0.9475  0.39718615
##   0.4    0.18538462  0.8393236  0.9500  0.37359307
##   0.4    0.19025641  0.8388420  0.9550  0.35952381
##   0.4    0.19512821  0.8383658  0.9600  0.35021645
##   0.4    0.20000000  0.8371050  0.9600  0.34112554
##   0.6    0.01000000  0.8559199  0.8775  0.60108225
##   0.6    0.01487179  0.8565855  0.8775  0.58744589
##   0.6    0.01974359  0.8574134  0.8800  0.58290043
##   0.6    0.02461538  0.8567154  0.8850  0.57813853
##   0.6    0.02948718  0.8562608  0.8900  0.57337662
##   0.6    0.03435897  0.8554275  0.8925  0.56428571
##   0.6    0.03923077  0.8546374  0.8925  0.55519481
##   0.6    0.04410256  0.8531494  0.8925  0.55519481
##   0.6    0.04897436  0.8532738  0.8950  0.55519481
##   0.6    0.05384615  0.8539773  0.8975  0.54588745
##   0.6    0.05871795  0.8529329  0.8975  0.54588745
##   0.6    0.06358974  0.8528084  0.9000  0.54588745
##   0.6    0.06846154  0.8511959  0.9000  0.54134199
##   0.6    0.07333333  0.8502760  0.9050  0.52727273
##   0.6    0.07820513  0.8492370  0.9075  0.52251082
##   0.6    0.08307692  0.8485444  0.9075  0.52251082
##   0.6    0.08794872  0.8471429  0.9100  0.51774892
##   0.6    0.09282051  0.8465639  0.9100  0.51320346
##   0.6    0.09769231  0.8453193  0.9100  0.49458874
##   0.6    0.10256410  0.8427435  0.9150  0.48549784
```

```
##   0.6    0.10743590  0.8419210  0.9175  0.48549784
##   0.6    0.11230769  0.8400595  0.9275  0.46666667
##   0.6    0.11717949  0.8379762  0.9350  0.46666667
##   0.6    0.12205128  0.8368019  0.9400  0.46190476
##   0.6    0.12692308  0.8348323  0.9425  0.45281385
##   0.6    0.13179487  0.8345942  0.9450  0.43852814
##   0.6    0.13666667  0.8339989  0.9450  0.41969697
##   0.6    0.14153846  0.8336742  0.9475  0.41060606
##   0.6    0.14641026  0.8321429  0.9525  0.38290043
##   0.6    0.15128205  0.8309632  0.9550  0.35974026
##   0.6    0.15615385  0.8296753  0.9575  0.34112554
##   0.6    0.16102564  0.8287392  0.9650  0.33160173
##   0.6    0.16589744  0.8267587  0.9675  0.30346320
##   0.6    0.17076923  0.8253355  0.9725  0.28484848
##   0.6    0.17564103  0.8238203  0.9750  0.27077922
##   0.6    0.18051282  0.8219372  0.9750  0.24285714
##   0.6    0.18538462  0.8202976  0.9750  0.23354978
##   0.6    0.19025641  0.8177056  0.9750  0.21948052
##   0.6    0.19512821  0.8167641  0.9750  0.20064935
##   0.6    0.20000000  0.8159524  0.9775  0.17705628
##   0.8    0.01000000  0.8554275  0.8800  0.60129870
##   0.8    0.01487179  0.8567045  0.8775  0.59675325
##   0.8    0.01974359  0.8565909  0.8850  0.59220779
##   0.8    0.02461538  0.8559091  0.8875  0.58268398
##   0.8    0.02948718  0.8553193  0.8875  0.56883117
##   0.8    0.03435897  0.8539232  0.8900  0.56428571
##   0.8    0.03923077  0.8541829  0.8925  0.56428571
##   0.8    0.04410256  0.8526948  0.8950  0.55974026
##   0.8    0.04897436  0.8510931  0.8975  0.55497835
##   0.8    0.05384615  0.8501677  0.8975  0.54588745
##   0.8    0.05871795  0.8472673  0.9000  0.53658009
##   0.8    0.06358974  0.8455032  0.9050  0.53658009
##   0.8    0.06846154  0.8450325  0.9100  0.53181818
##   0.8    0.07333333  0.8422132  0.9100  0.51774892
##   0.8    0.07820513  0.8402489  0.9125  0.50865801
##   0.8    0.08307692  0.8387446  0.9150  0.49935065
##   0.8    0.08794872  0.8367532  0.9175  0.48982684
##   0.8    0.09282051  0.8356061  0.9250  0.48051948
##   0.8    0.09769231  0.8351353  0.9300  0.47575758
##   0.8    0.10256410  0.8339773  0.9375  0.47099567
##   0.8    0.10743590  0.8325595  0.9425  0.44761905
##   0.8    0.11230769  0.8311472  0.9425  0.43398268
##   0.8    0.11717949  0.8290314  0.9450  0.42943723
##   0.8    0.12205128  0.8255465  0.9500  0.39675325
##   0.8    0.12692308  0.8243561  0.9550  0.36839827
##   0.8    0.13179487  0.8211742  0.9650  0.34069264
##   0.8    0.13666667  0.8188258  0.9675  0.31255411
##   0.8    0.14153846  0.8164665  0.9725  0.29848485
##   0.8    0.14641026  0.8156006  0.9750  0.28441558
##   0.8    0.15128205  0.8140747  0.9750  0.27987013
##   0.8    0.15615385  0.8113041  0.9750  0.24675325
##   0.8    0.16102564  0.8092884  0.9750  0.23744589
##   0.8    0.16589744  0.8077895  0.9800  0.20930736
##   0.8    0.17076923  0.8050244  0.9800  0.17229437
```

```
##   0.8    0.17564103  0.8047321  0.9850  0.15346320
##   0.8    0.18051282  0.8047321  0.9900  0.14415584
##   0.8    0.18538462  0.8044345  0.9925  0.10194805
##   0.8    0.19025641  0.8044913  0.9925  0.06969697
##   0.8    0.19512821  0.8044913  0.9950  0.05129870
##   0.8    0.20000000  0.8044913  1.0000  0.02294372
##   1.0    0.01000000  0.8558820  0.8800  0.60129870
##   1.0    0.01487179  0.8571699  0.8825  0.59675325
##   1.0    0.01974359  0.8568182  0.8875  0.59199134
##   1.0    0.02461538  0.8551840  0.8875  0.58290043
##   1.0    0.02948718  0.8545022  0.8900  0.57813853
##   1.0    0.03435897  0.8525703  0.8925  0.56428571
##   1.0    0.03923077  0.8511851  0.8975  0.56428571
##   1.0    0.04410256  0.8492154  0.8975  0.55497835
##   1.0    0.04897436  0.8466288  0.9000  0.55043290
##   1.0    0.05384615  0.8440801  0.9050  0.54567100
##   1.0    0.05871795  0.8410390  0.9075  0.52229437
##   1.0    0.06358974  0.8389394  0.9100  0.51320346
##   1.0    0.06846154  0.8367262  0.9125  0.49913420
##   1.0    0.07333333  0.8353463  0.9150  0.49437229
##   1.0    0.07820513  0.8346320  0.9200  0.48051948
##   1.0    0.08307692  0.8315963  0.9275  0.48051948
##   1.0    0.08794872  0.8289123  0.9275  0.46147186
##   1.0    0.09282051  0.8257413  0.9325  0.44761905
##   1.0    0.09769231  0.8226786  0.9325  0.44329004
##   1.0    0.10256410  0.8204383  0.9400  0.42012987
##   1.0    0.10743590  0.8175649  0.9475  0.38744589
##   1.0    0.11230769  0.8160606  0.9500  0.36385281
##   1.0    0.11717949  0.8136824  0.9600  0.34523810
##   1.0    0.12205128  0.8107873  0.9650  0.31233766
##   1.0    0.12692308  0.8081412  0.9650  0.31233766
##   1.0    0.13179487  0.8052246  0.9725  0.28896104
##   1.0    0.13666667  0.8045482  0.9725  0.25627706
##   1.0    0.14153846  0.8044913  0.9775  0.23722944
##   1.0    0.14641026  0.8044913  0.9800  0.20930736
##   1.0    0.15128205  0.8044913  0.9825  0.16753247
##   1.0    0.15615385  0.8044913  0.9875  0.14415584
##   1.0    0.16102564  0.8044913  0.9900  0.09718615
##   1.0    0.16589744  0.8044913  0.9950  0.05584416
##   1.0    0.17076923  0.8044913  1.0000  0.03225108
##   1.0    0.17564103  0.8044913  1.0000  0.01363636
##   1.0    0.18051282  0.8044913  1.0000  0.00000000
##   1.0    0.18538462  0.8044913  1.0000  0.00000000
##   1.0    0.19025641  0.8044913  1.0000  0.00000000
##   1.0    0.19512821  0.8044913  1.0000  0.00000000
##   1.0    0.20000000  0.8044913  1.0000  0.00000000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.6 and lambda = 0.01974359.
```

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

```
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, 553, 553, 553, 554, 553, ...
## Resampling results across tuning parameters:
##
##   threshold  ROC        Sens    Spec
##    0         0.8445022  0.9375  0.405194805
##    1         0.8423485  0.9550  0.316666667
##    2         0.8365314  0.9825  0.158441558
##    3         0.8254708  1.0000  0.009090909
##    4         0.7985254  1.0000  0.000000000
##    5         0.7981926  1.0000  0.000000000
##    6         0.7994453  1.0000  0.000000000
##    7         0.5000000  1.0000  0.000000000
##    8         0.5000000  1.0000  0.000000000
##    9         0.5000000  1.0000  0.000000000
##   10         0.5000000  1.0000  0.000000000
##   11         0.5000000  1.0000  0.000000000
##   12         0.5000000  1.0000  0.000000000
##   13         0.5000000  1.0000  0.000000000
##   14         0.5000000  1.0000  0.000000000
##   15         0.5000000  1.0000  0.000000000
##   16         0.5000000  1.0000  0.000000000
##   17         0.5000000  1.0000  0.000000000
##   18         0.5000000  1.0000  0.000000000
##   19         0.5000000  1.0000  0.000000000
##   20         0.5000000  1.0000  0.000000000
##   21         0.5000000  1.0000  0.000000000
##   22         0.5000000  1.0000  0.000000000
##   23         0.5000000  1.0000  0.000000000
##   24         0.5000000  1.0000  0.000000000
##   25         0.5000000  1.0000  0.000000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was threshold = 0.
```

```
############################## LDA ################################
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, 553, 554, 554, 553, 554, ...
## Resampling results:
##
##   ROC        Sens    Spec
##   0.8514719  0.8725  0.6376623
```

```
#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
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  81  24
##        pos  19  29
##
##                Accuracy : 0.719
##                  95% CI : (0.6407, 0.7886)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.05152
##
##                   Kappa : 0.3653
##
##  Mcnemar's Test P-Value : 0.54187
##
```

```
##             Sensitivity : 0.5472
##             Specificity : 0.8100
##          Pos Pred Value : 0.6042
##          Neg Pred Value : 0.7714
##              Prevalence : 0.3464
##          Detection Rate : 0.1895
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.6786
##
##        'Positive' Class : pos
##
```

```
#Random Forest: Testing Data
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  85  22
##        pos  15  31
##
##                Accuracy : 0.7582
##                  95% CI : (0.6824, 0.8237)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.003479
##
##                   Kappa : 0.4488
##
##  Mcnemar's Test P-Value : 0.323940
##
##             Sensitivity : 0.5849
##             Specificity : 0.8500
##          Pos Pred Value : 0.6739
##          Neg Pred Value : 0.7944
##              Prevalence : 0.3464
##          Detection Rate : 0.2026
##    Detection Prevalence : 0.3007
##       Balanced Accuracy : 0.7175
##
##        'Positive' Class : pos
##
```

```
#K Nearest Neighbor: Testing Data
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  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
##
```

```
#Classification and Regression Trees (CART): Testing Data
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  21
##        pos  26  32
##
##               Accuracy : 0.6928
##                 95% CI : (0.6132, 0.7648)
##    No Information Rate : 0.6536
##    P-Value [Acc > NIR] : 0.1753
##
##                  Kappa : 0.3363
##
##  Mcnemar's Test P-Value : 0.5596
##
##             Sensitivity : 0.6038
##             Specificity : 0.7400
##          Pos Pred Value : 0.5517
##          Neg Pred Value : 0.7789
##              Prevalence : 0.3464
##          Detection Rate : 0.2092
##    Detection Prevalence : 0.3791
##       Balanced Accuracy : 0.6719
##
```

```
##        'Positive' Class : pos
##
```

```
#Neural Net: Testing Data
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  81  23
##        pos  19  30
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.03543
##
##                   Kappa : 0.3828
##
##  Mcnemar's Test P-Value : 0.64343
##
##             Sensitivity : 0.5660
##             Specificity : 0.8100
##          Pos Pred Value : 0.6122
##          Neg Pred Value : 0.7788
##              Prevalence : 0.3464
##          Detection Rate : 0.1961
##    Detection Prevalence : 0.3203
##       Balanced Accuracy : 0.6880
##
##        'Positive' Class : pos
##
```

```
#Support Vector Machines
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  82  25
##        pos  18  28
##
##                Accuracy : 0.719
##                  95% CI : (0.6407, 0.7886)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.05152
##
##                   Kappa : 0.3595
```

```
##
##  Mcnemar's Test P-Value : 0.36020
##
##             Sensitivity : 0.5283
##             Specificity : 0.8200
##          Pos Pred Value : 0.6087
##          Neg Pred Value : 0.7664
##              Prevalence : 0.3464
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3007
##       Balanced Accuracy : 0.6742
##
##        'Positive' Class : pos
##
```

```
#Boost
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  87  26
##        pos  13  27
##
##                Accuracy : 0.7451
##                  95% CI : (0.6684, 0.812)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.009661
##
##                   Kappa : 0.4026
##
##  Mcnemar's Test P-Value : 0.054664
##
##             Sensitivity : 0.5094
##             Specificity : 0.8700
##          Pos Pred Value : 0.6750
##          Neg Pred Value : 0.7699
##              Prevalence : 0.3464
##          Detection Rate : 0.1765
##    Detection Prevalence : 0.2614
##       Balanced Accuracy : 0.6897
##
##        'Positive' Class : pos
##
```

```
# Elastinet
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  84  26
##        pos  16  27
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.03543
##
##                   Kappa : 0.3656
##
##  Mcnemar's Test P-Value : 0.16491
##
##             Sensitivity : 0.5094
##             Specificity : 0.8400
##          Pos Pred Value : 0.6279
##          Neg Pred Value : 0.7636
##              Prevalence : 0.3464
##          Detection Rate : 0.1765
##    Detection Prevalence : 0.2810
##       Balanced Accuracy : 0.6747
##
##        'Positive' Class : pos
##
```

```
# Nearest Shrunken Centroid
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  34
##        pos   7  19
##
##                Accuracy : 0.732
##                  95% CI : (0.6545, 0.8003)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.02367
##
##                   Kappa : 0.3277
##
##  Mcnemar's Test P-Value : 4.896e-05
##
##             Sensitivity : 0.3585
##             Specificity : 0.9300
##          Pos Pred Value : 0.7308
##          Neg Pred Value : 0.7323
##              Prevalence : 0.3464
##          Detection Rate : 0.1242
```

```
##    Detection Prevalence : 0.1699
##       Balanced Accuracy : 0.6442
##
##        'Positive' Class : pos
##
```

```
#LDA
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  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
##
```

```
#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$byClass['Specificity'], rfconfusion$byClass
             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'], cartconfusion
               cartconfusion$byClass['Recall'], cartconfusion$byClass['F1'])

nnetfinal <- c(nnetconfusion$byClass['Sensitivity'], nnetconfusion$byClass['Specificity'], nnetconfusion
               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     0.5471698        0.81 0.6041667 0.5471698 0.5742574
## rffinal     0.5849057        0.85 0.6739130 0.5849057 0.6262626
## knnfinal    0.5660377        0.85 0.6666667 0.5660377 0.6122449
## cartfinal   0.6037736        0.74 0.5517241 0.6037736 0.5765766
## nnetfinal   0.5660377        0.81 0.6122449 0.5660377 0.5882353
## svmfinal    0.5283019        0.82 0.6086957 0.5283019 0.5656566
## gbmfinal    0.5094340        0.87 0.6750000 0.5094340 0.5806452
## nscfinal    0.3584906        0.93 0.7307692 0.3584906 0.4810127
## ldafinal    0.5094340        0.83 0.6136364 0.5094340 0.5567010
```