Final Project

Group 1

2/12/2022

```
#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
```

```
#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
```

```
#List Zero Markers: 6 out of 9 Variables have zero markers for Predictor Variables
list( Column = colSums(df==0),
      Row = sum(rowSums(df==0)))
```

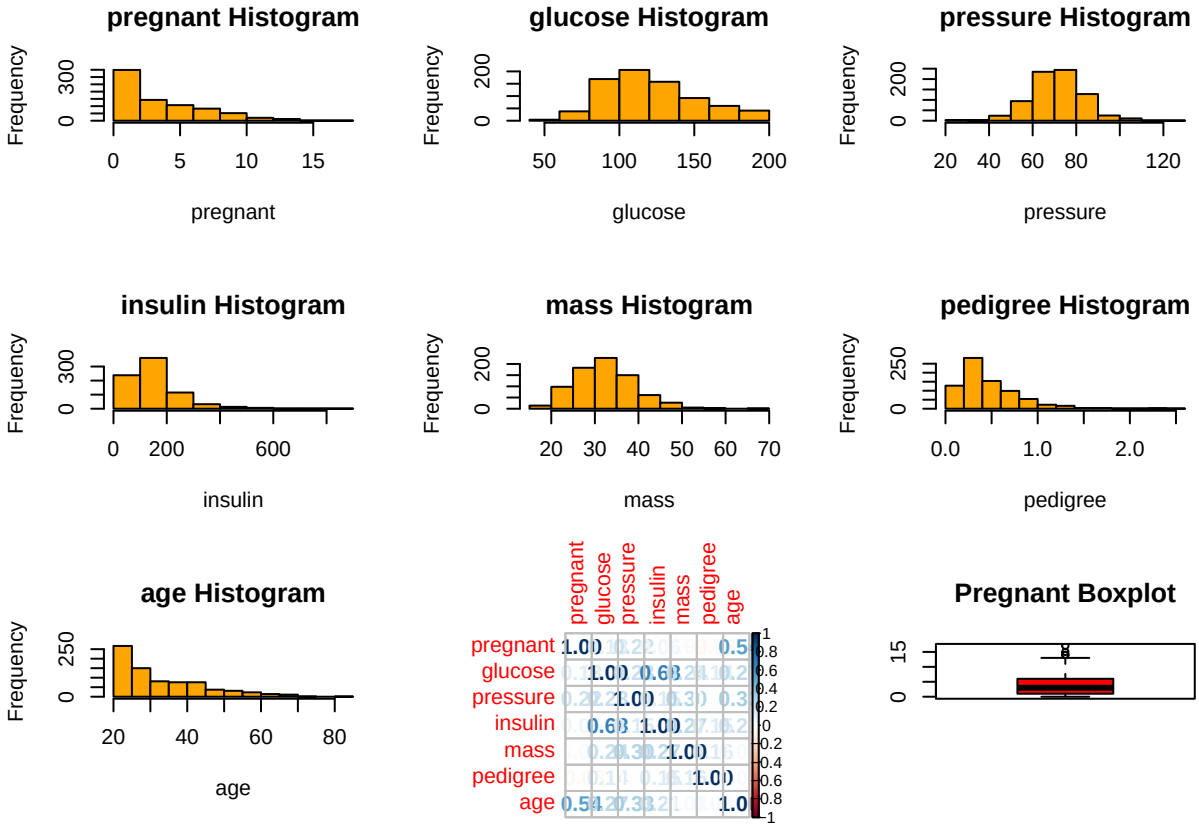
```
## $Column
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##      111        5       35      227      374       11        0        0
## diabetes
##        0
##
## $Row
## [1] 763
```

```
#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 use numerical mean
#pressure- only 35 values are missing, will keep predictor variable, will use numerical mean
#triceps- approximately 30% of the data contains 0 values, will keep predictor variable, will use numer
#insulin- almost 50% of the data has 0 values, will keep predictor variable, will use numerical mean
#mass- only 11 values are missing, will keep predictor variable

#Predictor Variables After Review of Summary Statistics and Zero Markers
#1.pregnant
#2.glucose
#3.pressure
#4.mass
#5.pedigree
#6.age
#7.triceps
#8.insulin

#Outcome Variable
#1.diabetes

# drop triceps as this does not seem to improve the predictions
df <- df[,-4]
#df

# 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
```

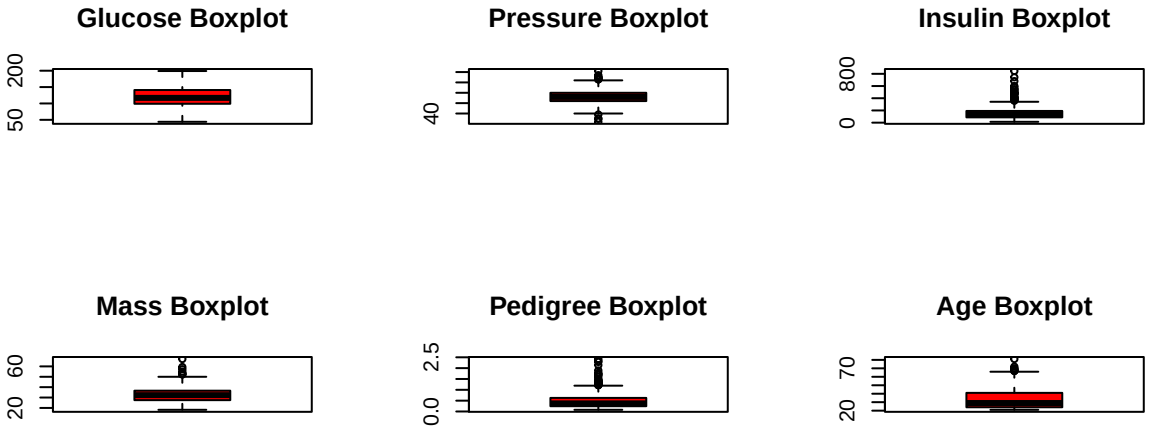
```
#Histograms of Diabetes: Predictor Variables
n <-df[,1:(ncol(df)-1)] #Predictors are variables 1-8
par(mfrow = c(3,3)) #Histograms will be 3x3
for (i in 1:ncol(n))
{hist(n[ ,i], xlab = names(n[i]), main = paste(names(n[i]), "Histogram"), col="orange")
}

#Correlation Plot of Diabetes: Predictor Variables
x <- cor(df[1:ncol(df)-1])
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")
#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")


#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
```




```
###################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"))

lr_train_data
```

```
## Generalized Linear Model
##
## 615 samples
```

```
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 553, 554, 553, 554, 554, ...
## Resampling results:
##
##   ROC        Sens    Spec
##   0.8438799  0.8875  0.5991342
```

```
summary(lr_train_data)
```

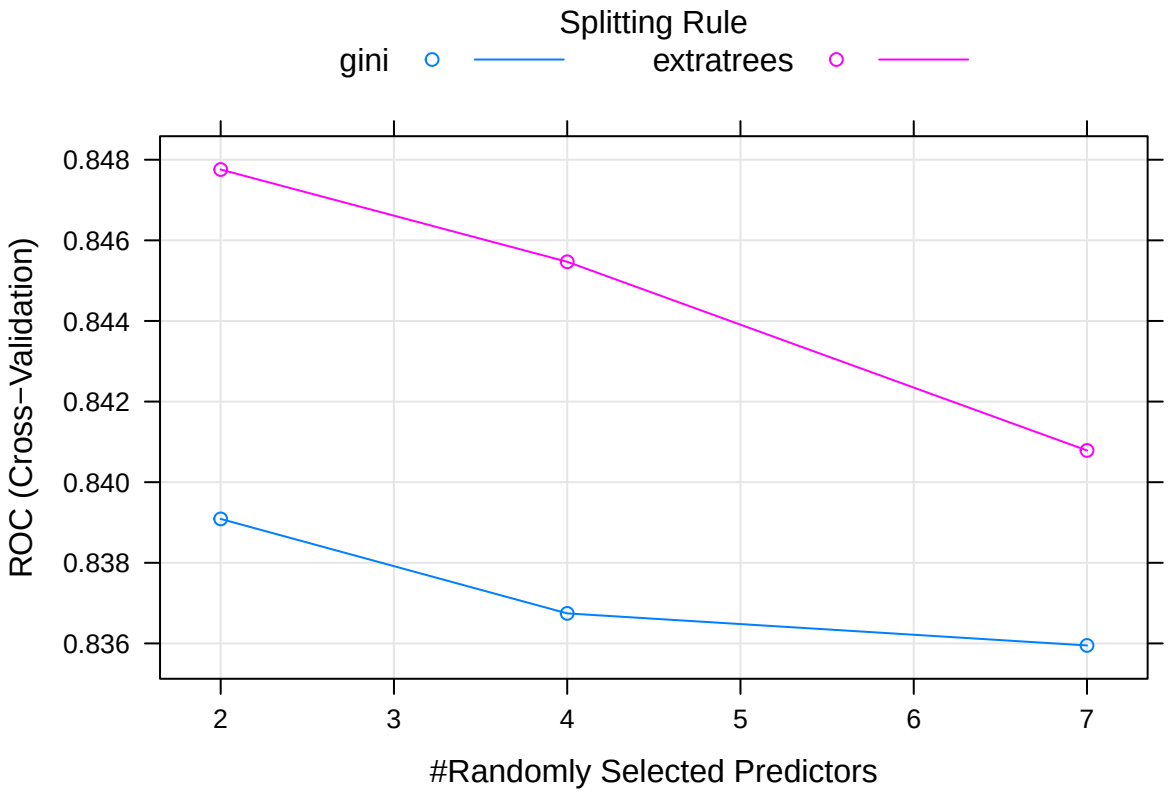
```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6721  -0.6993  -0.3742   0.6711   2.4459
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.87479    0.11052  -7.915 2.47e-15 ***
## pregnant     0.43455    0.12514   3.473 0.000515 ***
## glucose      1.22066    0.15519   7.866 3.67e-15 ***
## pressure    -0.11276    0.12028  -0.937 0.348507
## insulin     -0.07607    0.13773  -0.552 0.580740
## mass         0.69370    0.12401   5.594 2.22e-08 ***
## pedigree     0.34431    0.10779   3.194 0.001402 **
## age          0.13041    0.12721   1.025 0.305279
## ---
## 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: 554.74  on 607  degrees of freedom
## AIC: 570.74
##
## 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)
```

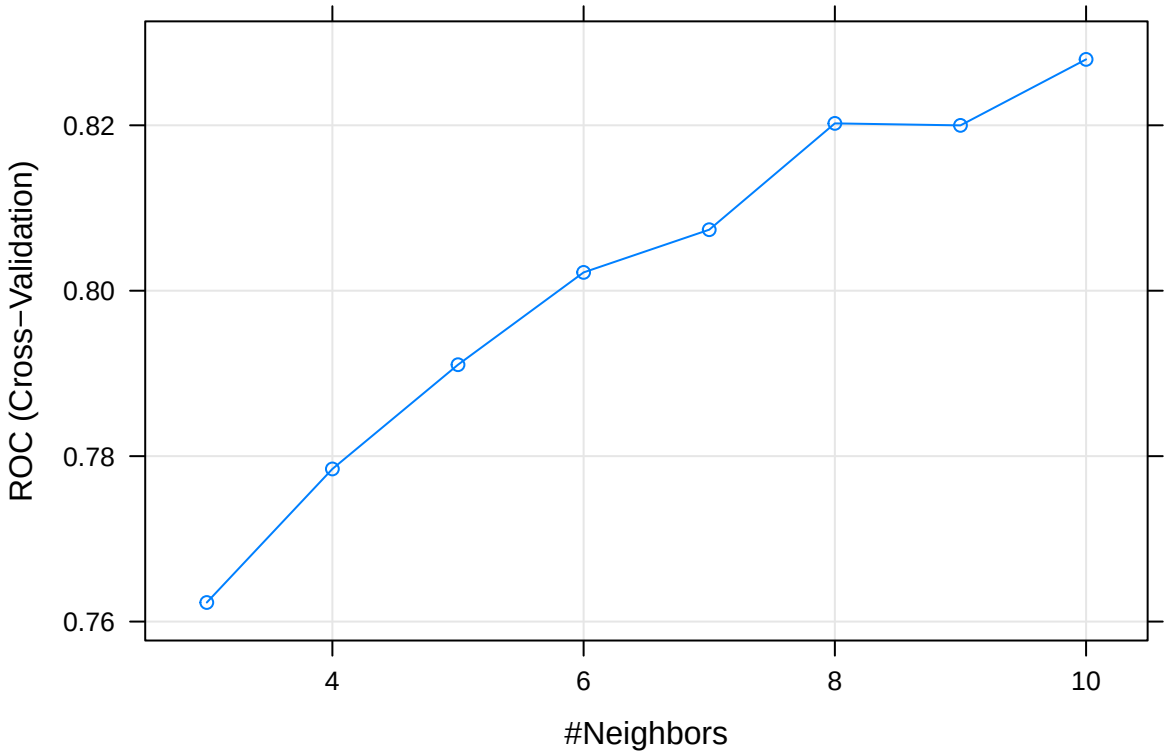



```
FinalTree = rf_train_data$finalModel$importance.mode


#K Nearest Neighbor: Training Model
knn_train_data <- caret::train(diabetes ~., data = train_data,
```

```
                          method = "knn",
                          metric = "ROC",
                          tuneGrid = expand.grid(.k = c(3:10)),
                          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  0.8300  0.5770563
##    5  0.7910552  0.8250  0.5816017
##    6  0.8022159  0.8375  0.5965368
##    7  0.8073755  0.8325  0.6153680
##    8  0.8202327  0.8525  0.6196970
##    9  0.8199892  0.8475  0.6101732
##   10  0.8279708  0.8600  0.5919913
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 10.
```

```
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", "pca"))

cart_train_data
```

```
## CART
##
## 615 samples
##   7 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (7), scaled (7), principal component signal
##  extraction (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 554, 553, 554, 553, 553, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens    Spec
##   0.00000000  0.7852841  0.8150  0.5770563
##   0.01272950  0.7860931  0.8250  0.6145022
```

```
##   0.02545900  0.7876272  0.8000  0.6190476
##   0.03818849  0.7876650  0.8075  0.6335498
##   0.05091799  0.7866261  0.8600  0.5506494
##   0.06364749  0.7837689  0.8550  0.5506494
##   0.07637699  0.7876975  0.8550  0.5554113
##   0.08910649  0.7721374  0.7650  0.6612554
##   0.10183599  0.7660011  0.7350  0.6976190
##   0.11456548  0.7572511  0.6800  0.7690476
##   0.12729498  0.7487284  0.6375  0.8599567
##   0.14002448  0.7487284  0.6375  0.8599567
##   0.15275398  0.7487284  0.6375  0.8599567
##   0.16548348  0.7487284  0.6375  0.8599567
##   0.17821297  0.6805465  0.6875  0.6735931
##   0.19094247  0.6805465  0.6875  0.6735931
##   0.20367197  0.6546374  0.7175  0.5917749
##   0.21640147  0.6120238  0.8050  0.4190476
##   0.22913097  0.5652976  0.8925  0.2380952
##   0.24186047  0.5470833  0.9275  0.1666667
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.07637699.
```

```
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, 554, 553, 554, 553, 554, ...
## Resampling results across tuning parameters:
##
##   decay  size  ROC        Sens    Spec
##   0.00    1    0.8472511  0.8625  0.6415584
##   0.00    2    0.8486526  0.8625  0.6274892
##   0.00    3    0.8399567  0.8575  0.6179654
##   0.00    4    0.8307792  0.8600  0.6038961
##   0.00    5    0.8213528  0.8550  0.5627706
##   0.00    6    0.8124621  0.8350  0.6043290
##   0.00    7    0.8236634  0.8550  0.6370130
##   0.00    8    0.8033983  0.8475  0.5809524
##   0.00    9    0.8261526  0.8575  0.6415584
##   0.00   10          NaN     NaN        NaN
##   0.01    1    0.8474784  0.8625  0.6415584
##   0.01    2    0.8490747  0.8550  0.6274892
##   0.01    3    0.8447403  0.8600  0.6229437
##   0.01    4    0.8347240  0.8450  0.5900433
##   0.01    5    0.8450325  0.8550  0.6034632
##   0.01    6    0.8191126  0.8275  0.5854978
##   0.01    7    0.8094968  0.8275  0.5441558
##   0.01    8    0.8168885  0.8325  0.5670996
##   0.01    9    0.8034740  0.8425  0.5582251
##   0.01   10          NaN     NaN        NaN
##   0.10    1    0.8473701  0.8625  0.6324675
##   0.10    2    0.8465530  0.8575  0.5995671
##   0.10    3    0.8432684  0.8750  0.5712121
##   0.10    4    0.8350108  0.8475  0.5813853
##   0.10    5    0.8234145  0.8450  0.5439394
##   0.10    6    0.8206981  0.8325  0.5530303
##   0.10    7    0.8108874  0.8450  0.5716450
##   0.10    8    0.8114881  0.8425  0.5673160
##   0.10    9    0.7991829  0.8175  0.5393939
##   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"),
                trControl = trainControl(method = "cv", number = 10,
                                         classProbs = TRUE, summaryFunction = twoClassSummary))
```

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

```
svmFit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 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:
##
##   C        ROC        Sens       Spec
##      0.25  0.8369589  0.8800000  0.5997835
##      0.50  0.8381006  0.8825000  0.6045455
##      1.00  0.8374459  0.8825000  0.5619048
##      2.00  0.8322294  0.8800000  0.5383117
##      4.00  0.8124513  0.8800000  0.5385281
##      8.00  0.7925595  0.8725000  0.5015152
##     16.00  0.7728409  0.8700000  0.4971861
##     32.00  0.7557413  0.8825000  0.4227273
##     64.00  0.7483225  0.8875000  0.3766234
##    128.00  0.7410931  0.9025000  0.3530303
##    256.00  0.7265476  0.8950000  0.3251082
##    512.00  0.7165530  0.8925000  0.3387446
##   1024.00  0.7099080  0.8950000  0.3255411
##   2048.00  0.7091811  0.8916667  0.3706109
##
## Tuning parameter 'sigma' was held constant at a value of 0.1445516
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1445516 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, 553, 553, 554, 554, 554, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  ROC        Sens    Spec
##   0.01       1                   100     0.8164989  0.9350  0.3770563
##   0.01       1                   150     0.8220157  0.9250  0.4465368
##   0.01       1                   200     0.8289908  0.9050  0.4930736
##   0.01       1                   250     0.8349702  0.8900  0.5251082
##   0.01       1                   300     0.8377165  0.8900  0.5296537
##   0.01       1                   350     0.8393561  0.8775  0.5387446
##   0.01       1                   400     0.8411310  0.8775  0.5528139
##   0.01       1                   450     0.8425379  0.8750  0.5530303
##   0.01       1                   500     0.8434848  0.8725  0.5668831
##   0.01       1                   550     0.8450054  0.8775  0.5627706
##   0.01       1                   600     0.8454924  0.8775  0.5670996
##   0.01       1                   650     0.8454762  0.8775  0.5718615
##   0.01       1                   700     0.8464069  0.8775  0.5718615
##   0.01       1                   750     0.8462825  0.8775  0.5861472
##   0.01       1                   800     0.8444426  0.8725  0.5816017
##   0.01       1                   850     0.8445887  0.8675  0.5863636
##   0.01       1                   900     0.8454113  0.8650  0.5954545
##   0.01       1                   950     0.8462121  0.8600  0.5954545
##   0.01       1                  1000     0.8465963  0.8625  0.5956710
##   0.01       3                   100     0.8359307  0.9175  0.4735931
##   0.01       3                   150     0.8392749  0.8875  0.5298701
##   0.01       3                   200     0.8425974  0.8800  0.5389610
##   0.01       3                   250     0.8441396  0.8750  0.5532468
##   0.01       3                   300     0.8432035  0.8675  0.5582251
##   0.01       3                   350     0.8420509  0.8650  0.5718615
##   0.01       3                   400     0.8438474  0.8625  0.5904762
##   0.01       3                   450     0.8435065  0.8625  0.6000000
##   0.01       3                   500     0.8420022  0.8625  0.6229437
##   0.01       3                   550     0.8411959  0.8600  0.6274892
##   0.01       3                   600     0.8410931  0.8600  0.6324675
##   0.01       3                   650     0.8388690  0.8600  0.6324675
##   0.01       3                   700     0.8379004  0.8600  0.6322511
##   0.01       3                   750     0.8349838  0.8600  0.6324675
##   0.01       3                   800     0.8348593  0.8625  0.6277056
##   0.01       3                   850     0.8339394  0.8625  0.6324675
```

```
##   0.01       3                900     0.8325541  0.8600  0.6324675
##   0.01       3                950     0.8324567  0.8625  0.6372294
##   0.01       3               1000     0.8312987  0.8550  0.6324675
##   0.01       5                100     0.8412284  0.9100  0.4924242
##   0.01       5                150     0.8430249  0.8800  0.5530303
##   0.01       5                200     0.8411959  0.8750  0.5809524
##   0.01       5                250     0.8428571  0.8800  0.6043290
##   0.01       5                300     0.8431277  0.8725  0.5995671
##   0.01       5                350     0.8397781  0.8675  0.6088745
##   0.01       5                400     0.8391180  0.8625  0.6134199
##   0.01       5                450     0.8374513  0.8600  0.6134199
##   0.01       5                500     0.8363258  0.8600  0.6229437
##   0.01       5                550     0.8347890  0.8600  0.6183983
##   0.01       5                600     0.8350379  0.8600  0.6279221
##   0.01       5                650     0.8339015  0.8600  0.6279221
##   0.01       5                700     0.8338041  0.8575  0.6279221
##   0.01       5                750     0.8329762  0.8550  0.6279221
##   0.01       5                800     0.8322781  0.8550  0.6279221
##   0.01       5                850     0.8313203  0.8550  0.6279221
##   0.01       5                900     0.8291017  0.8500  0.6324675
##   0.01       5                950     0.8276082  0.8475  0.6324675
##   0.01       5               1000     0.8256439  0.8425  0.6279221
##   0.01       7                100     0.8437446  0.8975  0.4974026
##   0.01       7                150     0.8445022  0.8750  0.5619048
##   0.01       7                200     0.8424946  0.8725  0.5995671
##   0.01       7                250     0.8420887  0.8675  0.6043290
##   0.01       7                300     0.8403030  0.8650  0.6372294
##   0.01       7                350     0.8400541  0.8675  0.6463203
##   0.01       7                400     0.8383387  0.8675  0.6322511
##   0.01       7                450     0.8371861  0.8675  0.6277056
##   0.01       7                500     0.8360011  0.8650  0.6370130
##   0.01       7                550     0.8345292  0.8625  0.6326840
##   0.01       7                600     0.8338528  0.8500  0.6279221
##   0.01       7                650     0.8323647  0.8475  0.6192641
##   0.01       7                700     0.8310714  0.8475  0.6145022
##   0.01       7                750     0.8302760  0.8450  0.6145022
##   0.01       7                800     0.8285498  0.8350  0.6192641
##   0.01       7                850     0.8262608  0.8325  0.6238095
##   0.01       7                900     0.8248647  0.8400  0.6281385
##   0.01       7                950     0.8232359  0.8375  0.6281385
##   0.01       7               1000     0.8219426  0.8400  0.6238095
##   0.10       1                100     0.8452814  0.8625  0.6138528
##   0.10       1                150     0.8416342  0.8550  0.6229437
##   0.10       1                200     0.8381981  0.8525  0.6365801
##   0.10       1                250     0.8354275  0.8475  0.6090909
##   0.10       1                300     0.8339827  0.8525  0.6181818
##   0.10       1                350     0.8335552  0.8475  0.6324675
##   0.10       1                400     0.8280032  0.8600  0.6281385
##   0.10       1                450     0.8258171  0.8475  0.6233766
##   0.10       1                500     0.8241613  0.8500  0.6142857
##   0.10       1                550     0.8219372  0.8525  0.6142857
##   0.10       1                600     0.8204708  0.8375  0.6283550
##   0.10       1                650     0.8215855  0.8475  0.6192641
##   0.10       1                700     0.8192695  0.8550  0.6383117
```

```
##   0.10       1                750      0.8197511  0.8500  0.6238095
##   0.10       1                800      0.8205465  0.8550  0.6188312
##   0.10       1                850      0.8198268  0.8450  0.6099567
##   0.10       1                900      0.8183009  0.8475  0.6010823
##   0.10       1                950      0.8170400  0.8450  0.6099567
##   0.10       1               1000      0.8149838  0.8450  0.6147186
##   0.10       3                100      0.8329762  0.8675  0.6326840
##   0.10       3                150      0.8237716  0.8475  0.6235931
##   0.10       3                200      0.8186742  0.8375  0.6281385
##   0.10       3                250      0.8157846  0.8350  0.6095238
##   0.10       3                300      0.8139989  0.8275  0.6047619
##   0.10       3                350      0.8132197  0.8350  0.6190476
##   0.10       3                400      0.8067803  0.8225  0.6093074
##   0.10       3                450      0.8046104  0.8200  0.6138528
##   0.10       3                500      0.8043236  0.8175  0.6190476
##   0.10       3                550      0.8027922  0.8150  0.6136364
##   0.10       3                600      0.8015368  0.8200  0.6274892
##   0.10       3                650      0.7965584  0.8250  0.6093074
##   0.10       3                700      0.7956115  0.8250  0.6231602
##   0.10       3                750      0.7972998  0.8150  0.6136364
##   0.10       3                800      0.7976028  0.8150  0.6136364
##   0.10       3                850      0.7939556  0.8125  0.6041126
##   0.10       3                900      0.7938095  0.8075  0.6038961
##   0.10       3                950      0.7932413  0.8200  0.6134199
##   0.10       3               1000      0.7917370  0.8125  0.6179654
##   0.10       5                100      0.8276515  0.8350  0.6181818
##   0.10       5                150      0.8184578  0.8250  0.6463203
##   0.10       5                200      0.8133279  0.8125  0.6326840
##   0.10       5                250      0.8124242  0.8150  0.6422078
##   0.10       5                300      0.8093182  0.8225  0.6192641
##   0.10       5                350      0.8063474  0.8175  0.6465368
##   0.10       5                400      0.8043506  0.8175  0.6140693
##   0.10       5                450      0.8017154  0.8125  0.6188312
##   0.10       5                500      0.7999621  0.7975  0.6326840
##   0.10       5                550      0.8000379  0.8150  0.6235931
##   0.10       5                600      0.7963528  0.8100  0.6235931
##   0.10       5                650      0.7965043  0.8050  0.6283550
##   0.10       5                700      0.7954383  0.8050  0.6145022
##   0.10       5                750      0.7943615  0.8050  0.6097403
##   0.10       5                800      0.7926353  0.8000  0.6190476
##   0.10       5                850      0.7887554  0.8025  0.6097403
##   0.10       5                900      0.7892641  0.8000  0.6142857
##   0.10       5                950      0.7871050  0.8050  0.6233766
##   0.10       5               1000      0.7868723  0.8025  0.6190476
##   0.10       7                100      0.8150325  0.8450  0.5904762
##   0.10       7                150      0.8054870  0.8300  0.6138528
##   0.10       7                200      0.8031548  0.8175  0.6281385
##   0.10       7                250      0.8033063  0.8175  0.6281385
##   0.10       7                300      0.8008929  0.8075  0.6188312
##   0.10       7                350      0.7956061  0.8050  0.6233766
##   0.10       7                400      0.7931061  0.8100  0.6186147
##   0.10       7                450      0.7926677  0.8025  0.6140693
##   0.10       7                500      0.7920400  0.8050  0.6093074
##   0.10       7                550      0.7887933  0.7950  0.5909091
```

```
##   0.10       7                  600     0.7886688  0.8050  0.5861472
##   0.10       7                  650     0.7871591  0.7950  0.5857143
##   0.10       7                  700     0.7875054  0.7975  0.5906926
##   0.10       7                  750     0.7865422  0.7925  0.5766234
##   0.10       7                  800     0.7865909  0.7925  0.5718615
##   0.10       7                  850     0.7854221  0.7950  0.5904762
##   0.10       7                  900     0.7833442  0.8000  0.5861472
##   0.10       7                  950     0.7842641  0.7975  0.5813853
##   0.10       7                 1000     0.7846158  0.7950  0.5954545
##
## 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 = 1000, interaction.depth =
##  1, shrinkage = 0.01 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"),
                 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)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 554, 554, 553, 554, 553, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      ROC        Sens    Spec
##   0.0    0.01000000  0.8484199  0.8950  0.57229437
##   0.0    0.01487179  0.8484199  0.8950  0.57229437
##   0.0    0.01974359  0.8484199  0.8950  0.57229437
##   0.0    0.02461538  0.8485335  0.8950  0.57229437
##   0.0    0.02948718  0.8477273  0.8975  0.56731602
##   0.0    0.03435897  0.8474675  0.8975  0.56731602
##   0.0    0.03923077  0.8472240  0.8975  0.56255411
##   0.0    0.04410256  0.8469805  0.8975  0.56255411
##   0.0    0.04897436  0.8467370  0.8975  0.56255411
##   0.0    0.05384615  0.8462500  0.9000  0.56255411
##   0.0    0.05871795  0.8457630  0.9050  0.55779221
##   0.0    0.06358974  0.8458820  0.9050  0.55779221
##   0.0    0.06846154  0.8452976  0.9050  0.55779221
```

```
##   0.0    0.07333333  0.8452976  0.9050  0.55779221
##   0.0    0.07820513  0.8450595  0.9050  0.54848485
##   0.0    0.08307692  0.8448377  0.9050  0.54848485
##   0.0    0.08794872  0.8448377  0.9050  0.54848485
##   0.0    0.09282051  0.8448323  0.9050  0.54848485
##   0.0    0.09769231  0.8447132  0.9050  0.54393939
##   0.0    0.10256410  0.8444751  0.9050  0.54393939
##   0.0    0.10743590  0.8440097  0.9100  0.54393939
##   0.0    0.11230769  0.8437771  0.9125  0.54393939
##   0.0    0.11717949  0.8435335  0.9125  0.53939394
##   0.0    0.12205128  0.8431872  0.9125  0.53939394
##   0.0    0.12692308  0.8431926  0.9150  0.53939394
##   0.0    0.13179487  0.8428463  0.9150  0.53463203
##   0.0    0.13666667  0.8430736  0.9175  0.53008658
##   0.0    0.14153846  0.8427165  0.9175  0.52554113
##   0.0    0.14641026  0.8428193  0.9200  0.52099567
##   0.0    0.15128205  0.8425812  0.9200  0.52099567
##   0.0    0.15615385  0.8421158  0.9225  0.52099567
##   0.0    0.16102564  0.8417695  0.9225  0.51623377
##   0.0    0.16589744  0.8416558  0.9250  0.50692641
##   0.0    0.17076923  0.8415476  0.9250  0.50692641
##   0.0    0.17564103  0.8415530  0.9250  0.50238095
##   0.0    0.18051282  0.8416667  0.9250  0.49761905
##   0.0    0.18538462  0.8412013  0.9250  0.49761905
##   0.0    0.19025641  0.8413149  0.9250  0.49761905
##   0.0    0.19512821  0.8414340  0.9275  0.49761905
##   0.0    0.20000000  0.8415476  0.9275  0.48333333
##   0.1    0.01000000  0.8483387  0.8875  0.58138528
##   0.1    0.01487179  0.8479654  0.8900  0.58138528
##   0.1    0.01974359  0.8479600  0.8925  0.57683983
##   0.1    0.02461538  0.8480628  0.8950  0.57683983
##   0.1    0.02948718  0.8477219  0.8975  0.56753247
##   0.1    0.03435897  0.8479491  0.9000  0.56255411
##   0.1    0.03923077  0.8475920  0.9025  0.55779221
##   0.1    0.04410256  0.8474675  0.9050  0.55779221
##   0.1    0.04897436  0.8476894  0.9050  0.55303030
##   0.1    0.05384615  0.8470833  0.9075  0.55303030
##   0.1    0.05871795  0.8466342  0.9075  0.55303030
##   0.1    0.06358974  0.8470996  0.9075  0.54848485
##   0.1    0.06846154  0.8468615  0.9075  0.54848485
##   0.1    0.07333333  0.8467370  0.9075  0.54372294
##   0.1    0.07820513  0.8464989  0.9075  0.54372294
##   0.1    0.08307692  0.8454437  0.9075  0.54372294
##   0.1    0.08794872  0.8453355  0.9075  0.54372294
##   0.1    0.09282051  0.8455682  0.9100  0.54372294
##   0.1    0.09769231  0.8449838  0.9100  0.53917749
##   0.1    0.10256410  0.8447565  0.9100  0.53917749
##   0.1    0.10743590  0.8440476  0.9125  0.53441558
##   0.1    0.11230769  0.8436959  0.9175  0.52532468
##   0.1    0.11717949  0.8434578  0.9200  0.52532468
##   0.1    0.12205128  0.8432305  0.9200  0.52532468
##   0.1    0.12692308  0.8432305  0.9200  0.52532468
##   0.1    0.13179487  0.8428734  0.9200  0.52056277
##   0.1    0.13666667  0.8426353  0.9200  0.51580087
```

```
##   0.1    0.14153846  0.8422890  0.9225  0.51125541
##   0.1    0.14641026  0.8421699  0.9225  0.51125541
##   0.1    0.15128205  0.8424080  0.9225  0.50216450
##   0.1    0.15615385  0.8420671  0.9250  0.48852814
##   0.1    0.16102564  0.8417208  0.9275  0.48398268
##   0.1    0.16589744  0.8414935  0.9275  0.47922078
##   0.1    0.17076923  0.8414935  0.9275  0.47922078
##   0.1    0.17564103  0.8412608  0.9275  0.47922078
##   0.1    0.18051282  0.8412608  0.9275  0.47922078
##   0.1    0.18538462  0.8412608  0.9275  0.47445887
##   0.1    0.19025641  0.8412608  0.9300  0.46969697
##   0.1    0.19512821  0.8409145  0.9300  0.46969697
##   0.1    0.20000000  0.8408009  0.9300  0.46515152
##   0.2    0.01000000  0.8490476  0.8925  0.58138528
##   0.2    0.01487179  0.8482251  0.8925  0.58138528
##   0.2    0.01974359  0.8478247  0.8925  0.57683983
##   0.2    0.02461538  0.8480465  0.8950  0.57207792
##   0.2    0.02948718  0.8481602  0.8975  0.56731602
##   0.2    0.03435897  0.8477056  0.9025  0.55324675
##   0.2    0.03923077  0.8474621  0.9050  0.55324675
##   0.2    0.04410256  0.8465476  0.9050  0.54848485
##   0.2    0.04897436  0.8464394  0.9050  0.54848485
##   0.2    0.05384615  0.8462175  0.9075  0.54848485
##   0.2    0.05871795  0.8468994  0.9075  0.54848485
##   0.2    0.06358974  0.8465368  0.9075  0.54848485
##   0.2    0.06846154  0.8460714  0.9075  0.54372294
##   0.2    0.07333333  0.8464232  0.9075  0.54372294
##   0.2    0.07820513  0.8460768  0.9075  0.54372294
##   0.2    0.08307692  0.8458387  0.9150  0.53896104
##   0.2    0.08794872  0.8460660  0.9150  0.53441558
##   0.2    0.09282051  0.8450000  0.9150  0.53441558
##   0.2    0.09769231  0.8444264  0.9150  0.52987013
##   0.2    0.10256410  0.8443182  0.9150  0.52987013
##   0.2    0.10743590  0.8442100  0.9150  0.52056277
##   0.2    0.11230769  0.8442100  0.9175  0.52056277
##   0.2    0.11717949  0.8435227  0.9200  0.51580087
##   0.2    0.12205128  0.8430519  0.9225  0.51125541
##   0.2    0.12692308  0.8425920  0.9225  0.51125541
##   0.2    0.13179487  0.8426028  0.9250  0.50194805
##   0.2    0.13666667  0.8424892  0.9250  0.49740260
##   0.2    0.14153846  0.8420184  0.9250  0.48831169
##   0.2    0.14641026  0.8413312  0.9275  0.48831169
##   0.2    0.15128205  0.8415639  0.9275  0.48376623
##   0.2    0.15615385  0.8412175  0.9275  0.47922078
##   0.2    0.16102564  0.8406548  0.9275  0.47445887
##   0.2    0.16589744  0.8400703  0.9275  0.46969697
##   0.2    0.17076923  0.8398377  0.9300  0.46515152
##   0.2    0.17564103  0.8397240  0.9300  0.46060606
##   0.2    0.18051282  0.8393777  0.9300  0.46060606
##   0.2    0.18538462  0.8392695  0.9325  0.45151515
##   0.2    0.19025641  0.8392749  0.9350  0.44696970
##   0.2    0.19512821  0.8393939  0.9350  0.44696970
##   0.2    0.20000000  0.8386905  0.9350  0.43766234
##   0.4    0.01000000  0.8481277  0.8925  0.58138528
```

```
##   0.4    0.01487179  0.8481331  0.8925  0.58138528
##   0.4    0.01974359  0.8482197  0.8925  0.57186147
##   0.4    0.02461538  0.8476190  0.8950  0.57186147
##   0.4    0.02948718  0.8480844  0.8975  0.56277056
##   0.4    0.03435897  0.8480952  0.9025  0.55324675
##   0.4    0.03923077  0.8473918  0.9025  0.55324675
##   0.4    0.04410256  0.8465801  0.9025  0.55324675
##   0.4    0.04897436  0.8454383  0.9050  0.55324675
##   0.4    0.05384615  0.8452110  0.9075  0.55324675
##   0.4    0.05871795  0.8440693  0.9100  0.53917749
##   0.4    0.06358974  0.8437392  0.9125  0.53441558
##   0.4    0.06846154  0.8432846  0.9125  0.52987013
##   0.4    0.07333333  0.8435119  0.9125  0.52532468
##   0.4    0.07820513  0.8424784  0.9175  0.52077922
##   0.4    0.08307692  0.8422511  0.9175  0.51623377
##   0.4    0.08794872  0.8418019  0.9175  0.50670996
##   0.4    0.09282051  0.8416883  0.9200  0.50670996
##   0.4    0.09769231  0.8412392  0.9225  0.50216450
##   0.4    0.10256410  0.8412284  0.9225  0.50216450
##   0.4    0.10743590  0.8402002  0.9225  0.49761905
##   0.4    0.11230769  0.8398377  0.9275  0.48852814
##   0.4    0.11717949  0.8397240  0.9300  0.48398268
##   0.4    0.12205128  0.8396266  0.9300  0.47922078
##   0.4    0.12692308  0.8389123  0.9325  0.47012987
##   0.4    0.13179487  0.8389177  0.9325  0.46558442
##   0.4    0.13666667  0.8386688  0.9350  0.45151515
##   0.4    0.14153846  0.8375162  0.9375  0.43311688
##   0.4    0.14641026  0.8371807  0.9425  0.41904762
##   0.4    0.15128205  0.8365963  0.9425  0.40519481
##   0.4    0.15615385  0.8365206  0.9425  0.39134199
##   0.4    0.16102564  0.8359470  0.9425  0.37727273
##   0.4    0.16589744  0.8356006  0.9475  0.37727273
##   0.4    0.17076923  0.8351245  0.9500  0.37727273
##   0.4    0.17564103  0.8334957  0.9575  0.37727273
##   0.4    0.18051282  0.8332413  0.9575  0.36341991
##   0.4    0.18538462  0.8326732  0.9575  0.35865801
##   0.4    0.19025641  0.8323214  0.9575  0.34913420
##   0.4    0.19512821  0.8314015  0.9575  0.34004329
##   0.4    0.20000000  0.8296320  0.9600  0.33073593
##   0.6    0.01000000  0.8478950  0.8925  0.58138528
##   0.6    0.01487179  0.8490476  0.8925  0.57186147
##   0.6    0.01974359  0.8493939  0.8950  0.57186147
##   0.6    0.02461538  0.8491667  0.8975  0.56731602
##   0.6    0.02948718  0.8470942  0.8975  0.56255411
##   0.6    0.03435897  0.8461742  0.8975  0.54848485
##   0.6    0.03923077  0.8458550  0.9000  0.54848485
##   0.6    0.04410256  0.8452760  0.9050  0.54848485
##   0.6    0.04897436  0.8446050  0.9100  0.53917749
##   0.6    0.05384615  0.8429762  0.9125  0.53441558
##   0.6    0.05871795  0.8422890  0.9150  0.52987013
##   0.6    0.06358974  0.8411418  0.9150  0.52532468
##   0.6    0.06846154  0.8409253  0.9150  0.52077922
##   0.6    0.07333333  0.8400108  0.9175  0.51601732
##   0.6    0.07820513  0.8397835  0.9200  0.50692641
```

```
##   0.6    0.08307692  0.8393074  0.9225  0.50238095
##   0.6    0.08794872  0.8388420  0.9250  0.50238095
##   0.6    0.09282051  0.8374351  0.9250  0.49329004
##   0.6    0.09769231  0.8360390  0.9300  0.47922078
##   0.6    0.10256410  0.8335877  0.9325  0.47922078
##   0.6    0.10743590  0.8328734  0.9325  0.45606061
##   0.6    0.11230769  0.8312716  0.9375  0.44675325
##   0.6    0.11717949  0.8311310  0.9400  0.44675325
##   0.6    0.12205128  0.8299567  0.9450  0.42359307
##   0.6    0.12692308  0.8283333  0.9500  0.40974026
##   0.6    0.13179487  0.8272998  0.9500  0.39567100
##   0.6    0.13666667  0.8270725  0.9500  0.37705628
##   0.6    0.14153846  0.8257955  0.9525  0.36774892
##   0.6    0.14641026  0.8242965  0.9550  0.35844156
##   0.6    0.15128205  0.8230303  0.9575  0.35389610
##   0.6    0.15615385  0.8226948  0.9625  0.33528139
##   0.6    0.16102564  0.8229275  0.9650  0.32121212
##   0.6    0.16589744  0.8216613  0.9675  0.31190476
##   0.6    0.17076923  0.8201407  0.9675  0.30281385
##   0.6    0.17564103  0.8200325  0.9675  0.29826840
##   0.6    0.18051282  0.8187608  0.9725  0.28441558
##   0.6    0.18538462  0.8173810  0.9750  0.27987013
##   0.6    0.19025641  0.8170509  0.9750  0.26580087
##   0.6    0.19512821  0.8148593  0.9750  0.25627706
##   0.6    0.20000000  0.8116180  0.9750  0.23744589
##   0.8    0.01000000  0.8487013  0.8925  0.57662338
##   0.8    0.01487179  0.8490639  0.8975  0.57186147
##   0.8    0.01974359  0.8484957  0.8975  0.56731602
##   0.8    0.02461538  0.8465260  0.8975  0.56255411
##   0.8    0.02948718  0.8465476  0.9025  0.55779221
##   0.8    0.03435897  0.8447132  0.9025  0.55303030
##   0.8    0.03923077  0.8435768  0.9025  0.54393939
##   0.8    0.04410256  0.8416126  0.9100  0.53441558
##   0.8    0.04897436  0.8406872  0.9100  0.52532468
##   0.8    0.05384615  0.8407035  0.9100  0.52077922
##   0.8    0.05871795  0.8401407  0.9150  0.52077922
##   0.8    0.06358974  0.8383766  0.9150  0.51147186
##   0.8    0.06846154  0.8366071  0.9150  0.50692641
##   0.8    0.07333333  0.8344048  0.9200  0.48831169
##   0.8    0.07820513  0.8330032  0.9225  0.48831169
##   0.8    0.08307692  0.8306764  0.9225  0.47445887
##   0.8    0.08794872  0.8302219  0.9275  0.46060606
##   0.8    0.09282051  0.8287175  0.9300  0.46060606
##   0.8    0.09769231  0.8271916  0.9350  0.45129870
##   0.8    0.10256410  0.8251299  0.9375  0.42813853
##   0.8    0.10743590  0.8239827  0.9450  0.41406926
##   0.8    0.11230769  0.8228084  0.9450  0.40930736
##   0.8    0.11717949  0.8220130  0.9475  0.39112554
##   0.8    0.12205128  0.8202489  0.9550  0.38658009
##   0.8    0.12692308  0.8193398  0.9550  0.37251082
##   0.8    0.13179487  0.8173755  0.9650  0.35389610
##   0.8    0.13666667  0.8168236  0.9650  0.33073593
##   0.8    0.14153846  0.8127760  0.9650  0.32619048
##   0.8    0.14641026  0.8102110  0.9675  0.32164502
```

```
##   0.8    0.15128205  0.8088149  0.9675  0.30303030
##   0.8    0.15615385  0.8052273  0.9675  0.27510823
##   0.8    0.16102564  0.8038285  0.9725  0.26558442
##   0.8    0.16589744  0.8036607  0.9775  0.25151515
##   0.8    0.17076923  0.8024865  0.9775  0.23268398
##   0.8    0.17564103  0.8021889  0.9775  0.21883117
##   0.8    0.18051282  0.8024729  0.9800  0.19545455
##   0.8    0.18538462  0.8024729  0.9825  0.17207792
##   0.8    0.19025641  0.8022944  0.9850  0.15822511
##   0.8    0.19512821  0.8022944  0.9900  0.12099567
##   0.8    0.20000000  0.8022944  0.9925  0.10714286
##   1.0    0.01000000  0.8491883  0.8925  0.57186147
##   1.0    0.01487179  0.8481494  0.8975  0.57186147
##   1.0    0.01974359  0.8467749  0.8975  0.56255411
##   1.0    0.02461538  0.8448377  0.8975  0.56255411
##   1.0    0.02948718  0.8441613  0.9000  0.55324675
##   1.0    0.03435897  0.8410173  0.9025  0.54848485
##   1.0    0.03923077  0.8400054  0.9075  0.52987013
##   1.0    0.04410256  0.8397944  0.9075  0.52532468
##   1.0    0.04897436  0.8388420  0.9125  0.51623377
##   1.0    0.05384615  0.8365097  0.9125  0.50692641
##   1.0    0.05871795  0.8338095  0.9150  0.48831169
##   1.0    0.06358974  0.8311472  0.9175  0.49307359
##   1.0    0.06846154  0.8297673  0.9175  0.48398268
##   1.0    0.07333333  0.8275541  0.9200  0.47012987
##   1.0    0.07820513  0.8259416  0.9200  0.45584416
##   1.0    0.08307692  0.8242208  0.9250  0.45129870
##   1.0    0.08794872  0.8229329  0.9300  0.43722944
##   1.0    0.09282051  0.8202489  0.9325  0.41861472
##   1.0    0.09769231  0.8189989  0.9350  0.40952381
##   1.0    0.10256410  0.8171537  0.9425  0.39588745
##   1.0    0.10743590  0.8152056  0.9425  0.39112554
##   1.0    0.11230769  0.8104383  0.9500  0.37272727
##   1.0    0.11717949  0.8075487  0.9575  0.35887446
##   1.0    0.12205128  0.8047511  0.9625  0.35432900
##   1.0    0.12692308  0.8033009  0.9625  0.32619048
##   1.0    0.13179487  0.8030574  0.9650  0.31666667
##   1.0    0.13666667  0.8022944  0.9675  0.28896104
##   1.0    0.14153846  0.8022944  0.9750  0.27489177
##   1.0    0.14641026  0.8022944  0.9775  0.26103896
##   1.0    0.15128205  0.8022944  0.9775  0.23268398
##   1.0    0.15615385  0.8022944  0.9800  0.21406926
##   1.0    0.16102564  0.8022944  0.9800  0.17683983
##   1.0    0.16589744  0.8022944  0.9850  0.15346320
##   1.0    0.17076923  0.8022944  0.9900  0.10714286
##   1.0    0.17564103  0.8022944  0.9950  0.07878788
##   1.0    0.18051282  0.8022944  0.9950  0.04653680
##   1.0    0.18538462  0.8022944  0.9975  0.01861472
##   1.0    0.19025641  0.8022944  1.0000  0.00000000
##   1.0    0.19512821  0.8022944  1.0000  0.00000000
##   1.0    0.20000000  0.8022944  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"),
                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)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 554, 554, 553, 553, 554, ...
## Resampling results across tuning parameters:
##
##   threshold  ROC        Sens    Spec
##    0         0.8417370  0.9375  0.36839827
##    1         0.8407359  0.9625  0.27510823
##    2         0.8326515  0.9825  0.15865801
##    3         0.8187987  0.9975  0.01385281
##    4         0.8029329  1.0000  0.00000000
##    5         0.8038853  1.0000  0.00000000
##    6         0.8038853  1.0000  0.00000000
##    7         0.5000000  1.0000  0.00000000
##    8         0.5000000  1.0000  0.00000000
##    9         0.5000000  1.0000  0.00000000
##   10         0.5000000  1.0000  0.00000000
##   11         0.5000000  1.0000  0.00000000
##   12         0.5000000  1.0000  0.00000000
##   13         0.5000000  1.0000  0.00000000
##   14         0.5000000  1.0000  0.00000000
##   15         0.5000000  1.0000  0.00000000
##   16         0.5000000  1.0000  0.00000000
##   17         0.5000000  1.0000  0.00000000
##   18         0.5000000  1.0000  0.00000000
##   19         0.5000000  1.0000  0.00000000
##   20         0.5000000  1.0000  0.00000000
##   21         0.5000000  1.0000  0.00000000
##   22         0.5000000  1.0000  0.00000000
##   23         0.5000000  1.0000  0.00000000
##   24         0.5000000  1.0000  0.00000000
##   25         0.5000000  1.0000  0.00000000
##
## 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"),
                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)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 554, 554, 553, 553, 553, ...
## Resampling results:
##
##   ROC        Sens    Spec
##   0.8432035  0.8875  0.5733766
```

```
#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  86  27
##        pos  14  26
##
##                Accuracy : 0.732
##                  95% CI : (0.6545, 0.8003)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.02367
##
##                   Kappa : 0.372
```

```
##
##  Mcnemar's Test P-Value : 0.06092
##
##             Sensitivity : 0.4906
##             Specificity : 0.8600
##          Pos Pred Value : 0.6500
##          Neg Pred Value : 0.7611
##              Prevalence : 0.3464
##          Detection Rate : 0.1699
##    Detection Prevalence : 0.2614
##       Balanced Accuracy : 0.6753
##
##        '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  86  22
##        pos  14  31
##
##               Accuracy : 0.7647
##                 95% CI : (0.6894, 0.8294)
##    No Information Rate : 0.6536
##    P-Value [Acc > NIR] : 0.001988
##
##                  Kappa : 0.4613
##
##  Mcnemar's Test P-Value : 0.243345
##
##             Sensitivity : 0.5849
##             Specificity : 0.8600
##          Pos Pred Value : 0.6889
##          Neg Pred Value : 0.7963
##              Prevalence : 0.3464
##          Detection Rate : 0.2026
##    Detection Prevalence : 0.2941
##       Balanced Accuracy : 0.7225
##
##        '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  87  31
##        pos  13  22
##
##               Accuracy : 0.7124
##                 95% CI : (0.6338, 0.7826)
##    No Information Rate : 0.6536
##    P-Value [Acc > NIR] : 0.07283
##
##                  Kappa : 0.3098
##
##  Mcnemar's Test P-Value : 0.01038
##
##             Sensitivity : 0.4151
##             Specificity : 0.8700
##          Pos Pred Value : 0.6286
##          Neg Pred Value : 0.7373
##              Prevalence : 0.3464
##          Detection Rate : 0.1438
```

```
##    Detection Prevalence : 0.2288
##       Balanced Accuracy : 0.6425
##
##        '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  82  24
##        pos  18  29
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.03543
##
##                   Kappa : 0.3772
##
##  Mcnemar's Test P-Value : 0.44040
##
##             Sensitivity : 0.5472
##             Specificity : 0.8200
##          Pos Pred Value : 0.6170
##          Neg Pred Value : 0.7736
##              Prevalence : 0.3464
##          Detection Rate : 0.1895
##    Detection Prevalence : 0.3072
##       Balanced Accuracy : 0.6836
##
##        '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  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
##
```

```
#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  85  25
##        pos  15  28
##
##                Accuracy : 0.7386
##                  95% CI : (0.6615, 0.8062)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.01536
##
##                   Kappa : 0.3959
##
##  Mcnemar's Test P-Value : 0.15473
##
##             Sensitivity : 0.5283
##             Specificity : 0.8500
##          Pos Pred Value : 0.6512
##          Neg Pred Value : 0.7727
##              Prevalence : 0.3464
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.2810
##       Balanced Accuracy : 0.6892
##
##        'Positive' Class : pos
##
```

```
# Elastinet
glmnpredict <- predict(glmnFit, test_data)
#Confusion Matrix Accuracy
```

```r
glmnconfusion <- confusionMatrix(glmnpredict, test_data$diabetes, positive="pos")
glmnconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  86  29
##        pos  14  24
##
##                Accuracy : 0.719
##                  95% CI : (0.6407, 0.7886)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.05152
##
##                   Kappa : 0.3351
##
##  Mcnemar's Test P-Value : 0.03276
##
##             Sensitivity : 0.4528
##             Specificity : 0.8600
##          Pos Pred Value : 0.6316
##          Neg Pred Value : 0.7478
##              Prevalence : 0.3464
##          Detection Rate : 0.1569
##    Detection Prevalence : 0.2484
##       Balanced Accuracy : 0.6564
##
##        'Positive' Class : pos
##
```

```r
# 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  94  35
##        pos   6  18
##
##                Accuracy : 0.732
##                  95% CI : (0.6545, 0.8003)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.02367
##
##                   Kappa : 0.3209
##
##  Mcnemar's Test P-Value : 1.226e-05
##
##             Sensitivity : 0.3396
##             Specificity : 0.9400
```

```
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7287
##              Prevalence : 0.3464
##          Detection Rate : 0.1176
##    Detection Prevalence : 0.1569
##       Balanced Accuracy : 0.6398
##
##        '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  86  28
##        pos  14  25
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.03543
##
##                   Kappa : 0.3537
##
##  Mcnemar's Test P-Value : 0.04486
##
##             Sensitivity : 0.4717
##             Specificity : 0.8600
##          Pos Pred Value : 0.6410
##          Neg Pred Value : 0.7544
##              Prevalence : 0.3464
##          Detection Rate : 0.1634
##    Detection Prevalence : 0.2549
##       Balanced Accuracy : 0.6658
##
##        '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'], glmnconfusion
              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     0.4905660        0.86 0.6500000 0.4905660 0.5591398
## rffinal     0.5849057        0.85 0.6739130 0.5849057 0.6262626
## knnfinal    0.5849057        0.86 0.6888889 0.5849057 0.6326531
## cartfinal   0.4150943        0.87 0.6285714 0.4150943 0.5000000
## nnetfinal   0.5471698        0.82 0.6170213 0.5471698 0.5800000
## svmfinal    0.5094340        0.84 0.6279070 0.5094340 0.5625000
## gbmfinal    0.5283019        0.85 0.6511628 0.5283019 0.5833333
## nscfinal    0.3396226        0.94 0.7500000 0.3396226 0.4675325
## ldafinal    0.4716981        0.86 0.6410256 0.4716981 0.5434783
```