Smoke Detector! - Binary Classification Using Random Forest and Logistic Regression

Pavlo Mysak

2023-12-13

Objective:

Develop a robust predictive classification model utilizing bio-signals to determine an individual's smoking status from a comprehensive dataset, including age, anthropometric measurements, blood pressure, cholesterol levels, and various biochemical markers. This project employs machine learning techniques (namely, Random Forest and Logistic Regression) to accurately identify smokers and non-smokers, shedding light on the intricate relationship between diverse bio-signals and smoking habits.

Loading the Data, Creating Training & Validation Sets and Quick Summaries

According to the summary of the training data, there are no missing values. We have 22 independent variables available at our disposal.

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
dt <- read.csv('/Users/pavlomysak/Downloads/Kaggle-SmokerBinaryClassification/train.c
sv')

train.index <- sample(1:nrow(dt), 70000)
train <- dt[train.index,]
valid <- dt[-train.index,]

train$smoking <- factor(train$smoking)
valid$smoking <- factor(valid$smoking)
# remove ID
train <- train[,-1]
valid <- valid[,-1]

summary(train)
```

```
##       age         height.cm.      weight.kg.       waist.cm.    
##  Min.   :20.0   Min.   :135.0   Min.   : 30.00   Min.   : 51.00  
##  1st Qu.:40.0   1st Qu.:160.0   1st Qu.: 60.00   1st Qu.: 77.00  
##  Median :40.0   Median :165.0   Median : 65.00   Median : 83.00  
##  Mean   :44.3   Mean   :165.3   Mean   : 67.18   Mean   : 83.02  
##  3rd Qu.:55.0   3rd Qu.:170.0   3rd Qu.: 75.00   3rd Qu.: 89.00  
##  Max.   :85.0   Max.   :190.0   Max.   :130.00   Max.   :127.00  
##  eyesight.left.  eyesight.right. hearing.left.   hearing.right. 
##  Min.   :0.100   Min.   :0.100   Min.   :1.000   Min.   :1.000  
##  1st Qu.:0.800   1st Qu.:0.800   1st Qu.:1.000   1st Qu.:1.000  
##  Median :1.000   Median :1.000   Median :1.000   Median :1.000  
##  Mean   :1.008   Mean   :1.002   Mean   :1.024   Mean   :1.023  
##  3rd Qu.:1.200   3rd Qu.:1.200   3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :9.900   Max.   :9.900   Max.   :2.000   Max.   :2.000  
##     systolic       relaxation    fasting.blood.sugar  Cholesterol   
##  Min.   : 81.0   Min.   : 46.0   Min.   : 46.00      Min.   : 77.0  
##  1st Qu.:114.0   1st Qu.: 70.0   1st Qu.: 90.00      1st Qu.:175.0  
##  Median :121.0   Median : 78.0   Median : 96.00      Median :196.0  
##  Mean   :122.5   Mean   : 76.9   Mean   : 98.38      Mean   :195.7  
##  3rd Qu.:130.0   3rd Qu.: 82.0   3rd Qu.:103.00      3rd Qu.:217.0  
##  Max.   :213.0   Max.   :121.0   Max.   :375.00      Max.   :393.0  
##   triglyceride        HDL              LDL           hemoglobin   
##  Min.   :  8.0   Min.   :  9.00   Min.   :   1.0   Min.   : 5.80  
##  1st Qu.: 77.0   1st Qu.: 45.00   1st Qu.:  95.0   1st Qu.:13.80  
##  Median :115.0   Median : 54.00   Median : 114.0   Median :15.00  
##  Mean   :127.6   Mean   : 55.83   Mean   : 114.5   Mean   :14.81  
##  3rd Qu.:165.0   3rd Qu.: 64.00   3rd Qu.: 133.0   3rd Qu.:15.80  
##  Max.   :766.0   Max.   :135.00   Max.   :1860.0   Max.   :21.00  
##  Urine.protein   serum.creatinine      AST                ALT         
##  Min.   :1.000   Min.   :0.1000   Min.   :  6.00   Min.   :   1.00  
##  1st Qu.:1.000   1st Qu.:0.8000   1st Qu.: 20.00   1st Qu.:  16.00  
##  Median :1.000   Median :0.9000   Median : 24.00   Median :  22.00  
##  Mean   :1.075   Mean   :0.8932   Mean   : 25.51   Mean   :  26.53  
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.: 29.00   3rd Qu.:  32.00  
##  Max.   :6.000   Max.   :7.4000   Max.   :656.00   Max.   :2914.00  
##       Gtp          dental.caries    smoking  
##  Min.   :  2.00   Min.   :0.0000   0:39317  
##  1st Qu.: 18.00   1st Qu.:0.0000   1:30683  
##  Median : 27.00   Median :0.0000            
##  Mean   : 36.27   Mean   :0.1962            
##  3rd Qu.: 44.00   3rd Qu.:0.0000            
##  Max.   :766.00   Max.   :1.0000            
```

A few of our independent variables are exhibiting suspicious values. triglyceride, LDL, AST, ALT and Gtp have suspiciously high values.

More with Outliers

Generally, we have to be careful with outlier treatment. We don't want to simply omit outliers if they are legitimate observations in our data. We also do not want to blindly alter them (Ex. Winsorization) without domain knowledge. In regard to some machine learning models like Random Forest, we actually want to leave the outliers untouched and perform minimal data cleaning/pre-processing.

To get a better idea of how many outliers we're dealing with, let's iterate through each variable and find how many observations are above the 99th percentile. When we do this, we find that our count of observations over the 99th percentile ranges from around 300 to 700. In a dataset with 70000 observations, this accounts for, at most, less than 1% of the total observations. For this reason, we will disregard these outliers.

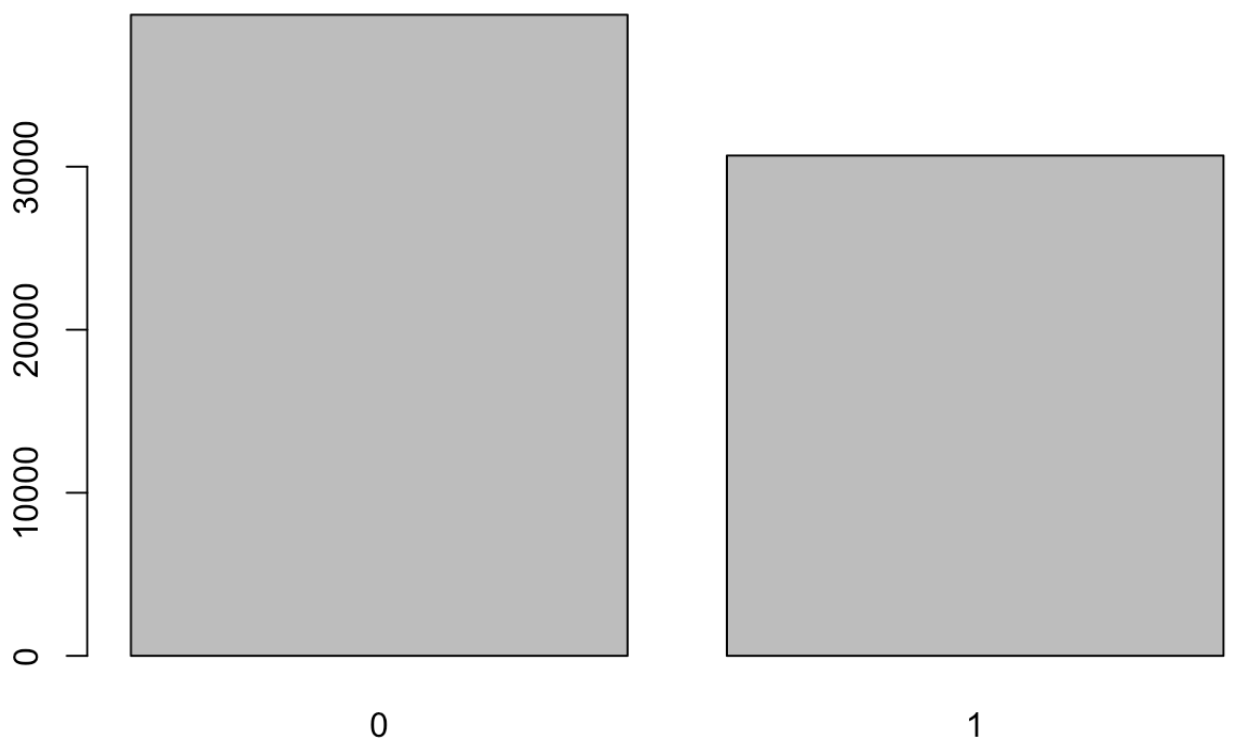
```
for( vari in colnames(train)){
  if(!is.factor(train[[vari]])){
  n_out = sum(ifelse(train[[vari]] > quantile(train[[vari]], .99),1,0))
  }
  cat(vari, ' has', n_out, ' outliers over the 99th percentile.', '\n')
}
```

```
## age  has 281  outliers over the 99th percentile.
## height.cm.  has 18  outliers over the 99th percentile.
## weight.kg.  has 369  outliers over the 99th percentile.
## waist.cm.  has 690  outliers over the 99th percentile.
## eyesight.left.  has 330  outliers over the 99th percentile.
## eyesight.right.  has 285  outliers over the 99th percentile.
## hearing.left.  has 0  outliers over the 99th percentile.
## hearing.right.  has 0  outliers over the 99th percentile.
## systolic  has 666  outliers over the 99th percentile.
## relaxation  has 369  outliers over the 99th percentile.
## fasting.blood.sugar  has 696  outliers over the 99th percentile.
## Cholesterol  has 660  outliers over the 99th percentile.
## triglyceride  has 697  outliers over the 99th percentile.
## HDL  has 592  outliers over the 99th percentile.
## LDL  has 655  outliers over the 99th percentile.
## hemoglobin  has 578  outliers over the 99th percentile.
## Urine.protein  has 249  outliers over the 99th percentile.
## serum.creatinine  has 361  outliers over the 99th percentile.
## AST  has 682  outliers over the 99th percentile.
## ALT  has 679  outliers over the 99th percentile.
## Gtp  has 684  outliers over the 99th percentile.
## dental.caries  has 0  outliers over the 99th percentile.
## smoking  has 0  outliers over the 99th percentile.
```

Distribution of our Dependant Variable

We see that our data has a good number of both smokers and non-smokers. This will give our models sufficient information on both categories to train on.

```
plot(train$smoking)
```

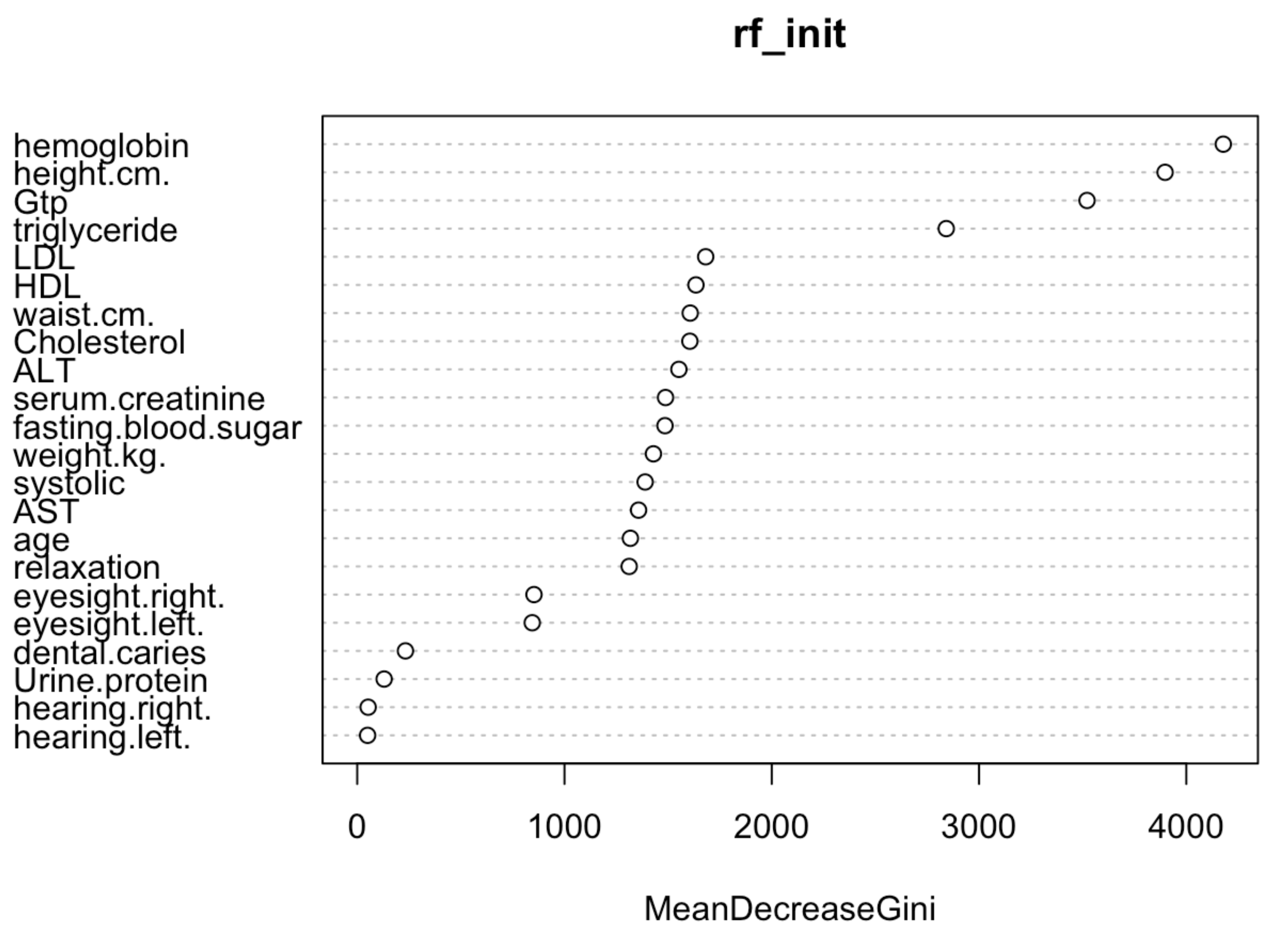


Random Forest

To begin our modelling process, let’s initialize a basic random forest model to get an idea of how this algorithm will react to our data and which variables are most important. Random Forest is a tree-based machine learning algorithm that grows a forest of (weak-ish) classification trees using a process called ‘bagging’. Bagging, or Bootstrap Aggregating is an ‘ensemble learning method’ that selects n uniform random samples with replacement from a training set and fits them to n classification models, respectively. In our case, it would create n ‘weak’ random forest trees. These trees are considered ‘weak’ because they may have limited predictive power on their own. The strength of the Random Forest algorithm materializes when we combine the classifications of multiple trees. To return our ultimate classification (Ensemble Classifier), each tree ‘votes’ on a classification. The ‘forest’ returns the classification that has received the most votes from the population of trees.

```
rf_init <- randomForest(smoking~.,
                        data = train)

varImpPlot(rf_init)
```



```
rf_init
```

```
##
## Call:
##  randomForest(formula = smoking ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 22.67%
## Confusion matrix:
##       0     1 class.error
## 0 29349  9968   0.2535290
## 1  5900 24783   0.1922889
```

Here we see that our initial model is built with a default of 500 trees and 4 variables in each tree.

We also see that based on the mean decrease in the gini impurity, the most important variables in this classification are: hemoglobin, height, Gtp and triglyceride. There is a pretty sharp decrease in importance once we get to LDL.

Mean Decrease in Gini Impurity

Gini Impurity is a measure commonly employed in decision tree algorithms, particularly in CART (Classification and Regression Tree). It quantifies the likelihood that a randomly chosen observation would be incorrectly classified if it were randomly labeled based on the distribution of categories in a set of data. The Gini Impurity ranges from 0 to 0.5, where 0 indicates perfect purity (all cases belong to a single category), and 0.5 implies that the classification is as good as a random guess. In other words, a lower Gini Impurity signifies a more homogeneous node in the decision tree, contributing to a more effective split during the tree-building process.

Finding the Optimal Random Forest Parameters

With these facts in mind, let’s find the optimal parameters for our Random Forest model. We will be optimizing the number of ‘weak’ trees ‘grown’ in the forest (ntree) and the number of independent variables included in each tree (mtry).

We will build 16 Random Forest models. The goal is to exhaust every desired ntree and mtry value then choose the best model.

```
k <- 1
rf_list <- list()
ntree_param <- c(400,500,600, 700)
mtry_param <- c(3,4,5,6)

for(trees in ntree_param){
  for(vars in mtry_param){
    model <- randomForest(smoking~.,
                          data = train,
                          ntree = trees,
                          mtry = vars)

    rf_list[[k]] <- model
    k = k+1

  }
}
```

Choosing the Optimal Model

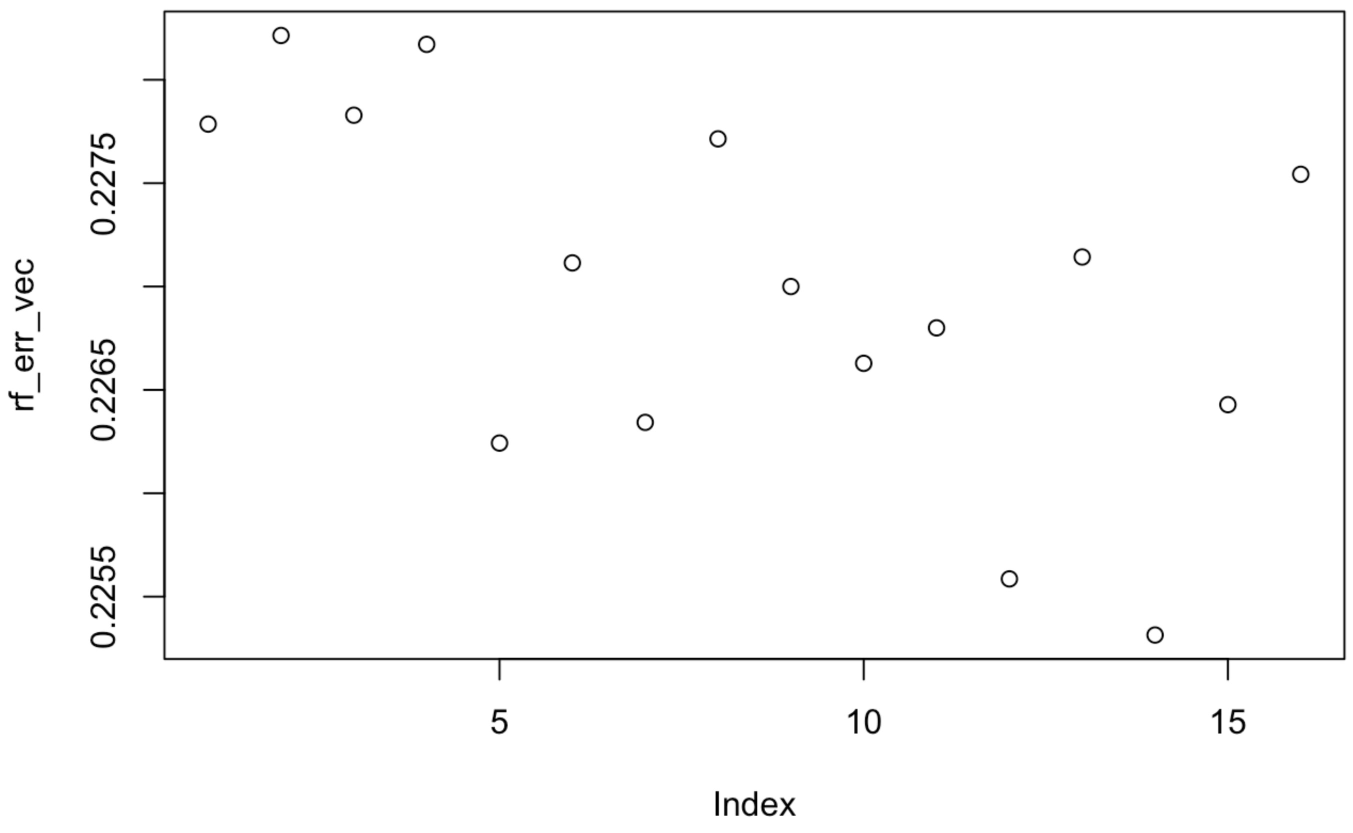
We will be using the Out-Of-Bag Error to find the best model. OOB Error also utilizes Bootstrap Aggregation. When we create those previously mentioned ‘weak’ trees using bootstrap samples, we can calculate an almost pseudo-out-of-sample error rate for each tree using the data that was NOT randomly selected to train the tree. How well will the tree predict the observations that it was not trained on… but that are still within the training data?

We will be looking at the OOB Error Rate for each of the 16 models and choosing the model with the lowest value.

```
rf_err_vec <- c()

for(i in 1:length(rf_list)){
  rf_err_vec[i] <- tail(rf_list[[i]]$err.rate[,'OOB'],1)
}

plot(rf_err_vec)
```



```
which.min(rf_err_vec) [redacted]
```

```
## [1] 14
```

```
[redacted]
rf_list[[which.min(rf_err_vec)]]
```

```
##
## Call:
##  randomForest(formula = smoking ~ ., data = train, ntree = trees,      mtry = vars
)
##                Type of random forest: classification
##                      Number of trees: 700
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 22.53%
## Confusion matrix:
##       0     1 class.error
## 0 29413  9904   0.2519012
## 1  5868 24815   0.1912460
```

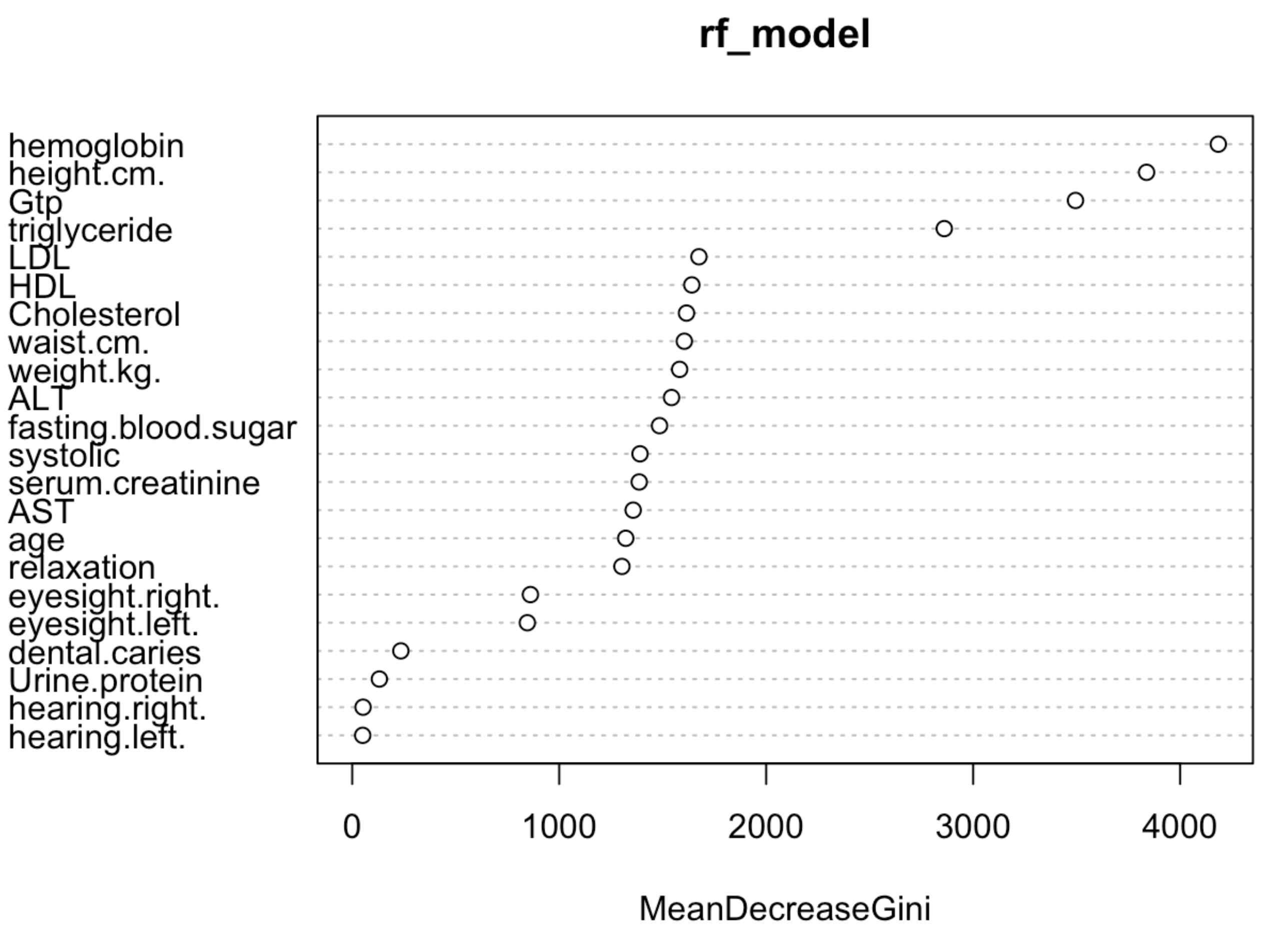
```
rf_model <- rf_list[[which.min(rf_err_vec)]]

rf_model
```

```
##
## Call:
##  randomForest(formula = smoking ~ ., data = train, ntree = trees,      mtry = vars
)
##                Type of random forest: classification
##                      Number of trees: 700
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 22.53%
## Confusion matrix:
##       0     1 class.error
## 0 29413  9904   0.2519012
## 1  5868 24815   0.1912460
```

Because Random Forest uses boostrapping and random sampling, every time we run the algorithm, we get different results. The parameters for the optimal tree (with the lowest OOB Error Rate) will be shown above.

```
varImpPlot(rf_model)
```



The variable importance has not changed much from our initial random forest model.

Logistic Regression

Now that we have an idea of what our important variables are based on Gini Impurity, we can begin thinking about fitting a logistic regression model.

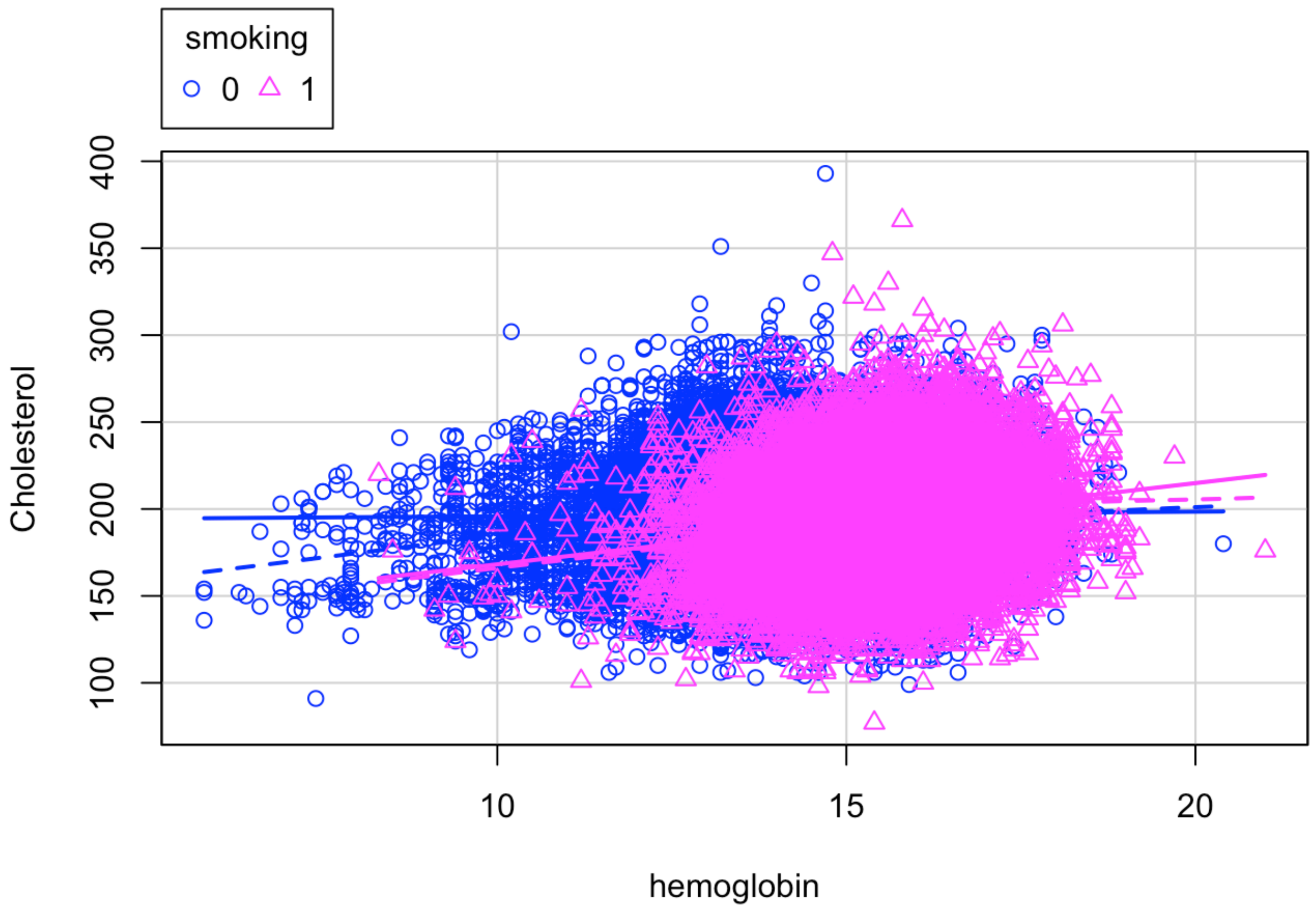
Logistic regression uses the sigmoid function (an S-shaped curve) and assigns probabilities to points on this curve. It is commonly used for classification problems. Logistic regression uses the MLE (Maximum Likelihood Estimate) method to estimate the parameters of an assumed probability distribution by maximizing the likelihood function.

To strengthen the power of this model, I'd like to include interaction terms. This is typically done via domain knowledge and is difficult to do via statistical tests. Using a stepwise logit regression is an option, but is computationally inefficient (it would take hours). Another option (not exactly perfect either) is to revert to visualizations. After iterating through many scatterplots, we find two possible interactions to include in our model.

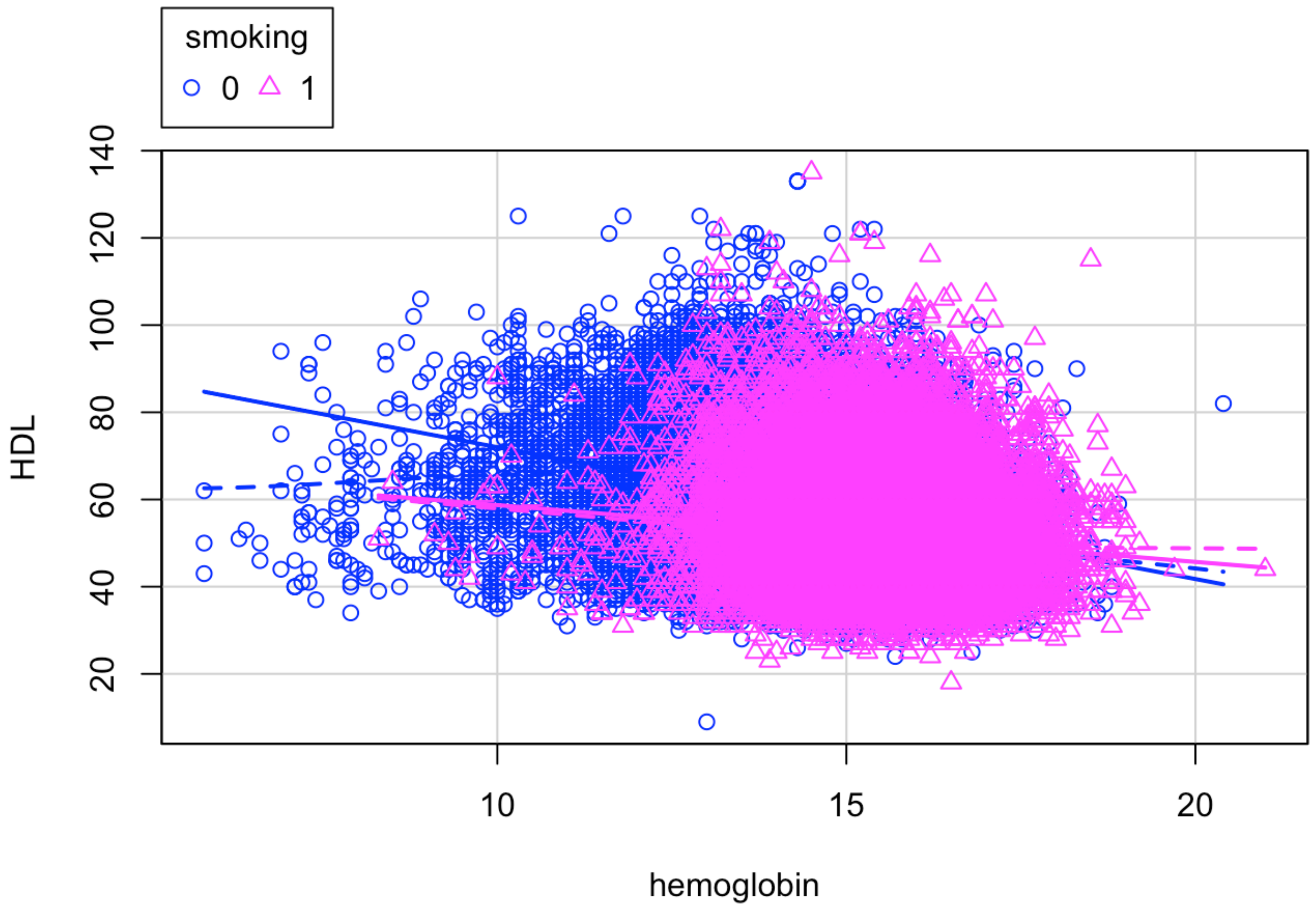
```
library(car)
```

```
## Loading required package: carData
```

```
scatterplot(formula = Cholesterol~hemoglobin | smoking , data=train)
```



```
scatterplot(formula = HDL~hemoglobin | smoking , data=train)
```



Let’s fit our first logit model with the previosuly found ‘important variables’ and these two interaction terms.

```
logit.1 <- glm(formula = smoking~hemoglobin+height.cm.+Gtp+triglyceride+LDL+HDL+Chole
sterol+Cholesterol:hemoglobin+HDL:hemoglobin,
            data = train,
            family = binomial)
summary(logit.1)
```

```
## 
## Call:
## glm(formula = smoking ~ hemoglobin + height.cm. + Gtp + triglyceride + 
##     LDL + HDL + Cholesterol + Cholesterol:hemoglobin + HDL:hemoglobin, 
##     family = binomial, data = train)
## 
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)    
## (Intercept)            -5.3277692  0.9660932  -5.515 3.49e-08 ***
## hemoglobin             -0.4467334  0.0621290  -7.190 6.46e-13 ***
## height.cm.              0.0737653  0.0013853  53.250  < 2e-16 ***
## Gtp                     0.0167457  0.0004262  39.290  < 2e-16 ***
## triglyceride            0.0071351  0.0002430  29.360  < 2e-16 ***
## LDL                    -0.0008717  0.0007837  -1.112    0.266    
## HDL                    -0.1246563  0.0100324 -12.425  < 2e-16 ***
## Cholesterol            -0.0438327  0.0044826  -9.778  < 2e-16 ***
## hemoglobin:Cholesterol  0.0022996  0.0002904   7.920 2.38e-15 ***
## hemoglobin:HDL          0.0082156  0.0006633  12.386  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 95973  on 69999  degrees of freedom
## Residual deviance: 69557  on 69990  degrees of freedom
## AIC: 69577
## 
## Number of Fisher Scoring iterations: 5
```

Each variable looks to be significant!

Logit 2

Now, just to be safe, let's run a logit model with only the main effects (no interaction terms).

```
logit.2 <- glm(formula = smoking~hemoglobin+height.cm.+Gtp+triglyceride+LDL+HDL+Chole
sterol,
               data = train,
               family = binomial)
summary(logit.2)
```

```
##
## Call:
## glm(formula = smoking ~ hemoglobin + height.cm. + Gtp + triglyceride +
##     LDL + HDL + Cholesterol, family = binomial, data = train)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.894e+01  2.393e-01 -79.142   <2e-16 ***
## hemoglobin    4.411e-01  9.264e-03  47.611   <2e-16 ***
## height.cm.    7.541e-02  1.385e-03  54.448   <2e-16 ***
## Gtp           1.704e-02  4.261e-04  39.993   <2e-16 ***
## triglyceride  7.090e-03  2.444e-04  29.010   <2e-16 ***
## LDL          -9.219e-04  7.939e-04  -1.161    0.246
## HDL          -1.928e-03  1.174e-03  -1.642    0.101
## Cholesterol  -9.169e-03  8.685e-04 -10.558   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95973  on 69999  degrees of freedom
## Residual deviance: 69811  on 69992  degrees of freedom
## AIC: 69827
##
## Number of Fisher Scoring iterations: 5
```

Once again, each variable is significant, but our AIC has increased.

Model Evaluation

Accuracy

A model's accuracy measures it's ability to detect correct positive and negative cases among all actual positive and negative cases.

Random Forest

The in-sample and out-of-sample accuracy's for our random forest model are fairly similar. However, our out-of-sample is a little higher. A great sign!

```
pred_in_rf <- predict(rf_model, type = 'response')
pred_out_rf <- predict(rf_model, newdata = valid, type = 'response')

# in-sample accuracy
sum(diag(rf_model$confusion[,c(1,2)]))/sum(rf_model$confusion[,c(1,2)])
```

```
## [1] 0.7746857
```

```
# out-of-sample accuracy
rf_out_confusion <- table(valid$smoking, pred_out_rf)
sum(diag(rf_out_confusion))/sum(rf_out_confusion)
```

```
## [1] 0.7765977
```

Logit

With both of our logistic regression models, our out-of-sample accuracy outperforms our in-sample accuracy. Logit.1 (Logistic Regression Model with Interaction Terms) is performing just a little better than Logit.2 (Logistic Regression Model with only Main Effects).

```
pred_in_logit1 <- predict(logit.1, type = 'response')
pred_out_logit1 <- predict(logit.1, newdata = valid, type = 'response')
logit1_in_confusion <- table(train$smoking, ifelse(pred_in_logit1>0.5,1,0))
logit1_out_confusion <- table(valid$smoking, ifelse(pred_out_logit1>0.5,1,0))
# in-sample logit.1
sum(diag(logit1_in_confusion))/sum(logit1_in_confusion)
```

```
## [1] 0.7403286
```

```
# out-of-sample logit.1
sum(diag(logit1_out_confusion))/sum(logit1_out_confusion)
```

```
## [1] 0.7416308
```

```
pred_in_logit2 <- predict(logit.2, type = 'response')
pred_out_logit2 <- predict(logit.2, newdata = valid, type = 'response')
logit2_in_confusion <- table(train$smoking, ifelse(pred_in_logit2>0.5,1,0))
logit2_out_confusion <- table(valid$smoking, ifelse(pred_out_logit2>0.5,1,0))
# in-sample logit.2
sum(diag(logit2_in_confusion))/sum(logit2_in_confusion)
```

```
## [1] 0.7393714
```

```
# out-of-sample logit.2
sum(diag(logit2_out_confusion))/sum(logit2_out_confusion)
```

```
## [1] 0.7408241
```

Sensitivity

A model’s sensitivity measures its ability to detect positive cases among all actual positive cases.

Our Random Forest model outperforms the Logistic Regression models in the sensitivity measure.

```
# out-of-sample RF
sum(ifelse(valid$smoking==1 & pred_out_rf==1,1,0))/sum(valid$smoking==1)
```

```
## [1] 0.8109571
```

```
# out-of-sample logit.1
sum(ifelse(valid$smoking==1 & pred_out_logit1>0.5,1,0))/sum(valid$smoking==1)
```

```
## [1] 0.7334873
```

```
# out-of-sample logit.2
sum(ifelse(valid$smoking==1 & pred_out_logit2>0.5,1,0))/sum(valid$smoking==1)
```

```
## [1] 0.7260457
```

Specificty

A model's specificity measures its ability to detect negative cases among all actual negative cases.

In specificity, our logit.2 (Logistic Regression with only Main Effects) model outperforms all other models.

```
# out-of-sample RF
sum(ifelse(valid$smoking==0 & pred_out_rf==0,1,0))/sum(valid$smoking==0)
```

```
## [1] 0.7499702
```

```
# out-of-sample logit.1
sum(ifelse(valid$smoking==0 & pred_out_logit1<=0.5,1,0))/sum(valid$smoking==0)
```

```
## [1] 0.7479418
```

```
# out-of-sample logit.2
sum(ifelse(valid$smoking==0 & pred_out_logit2<=0.5,1,0))/sum(valid$smoking==0)
```

```
## [1] 0.752277
```

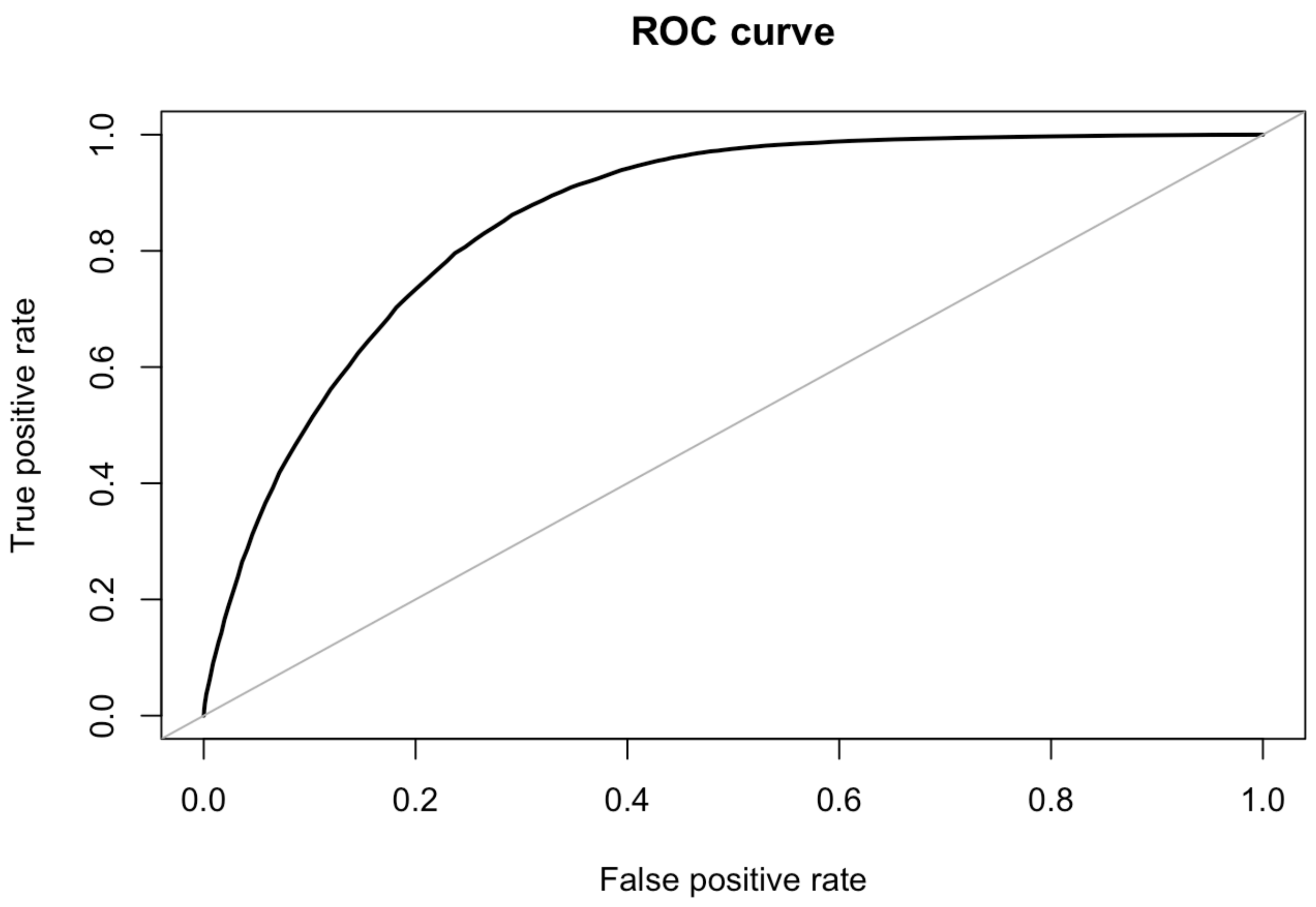
ROC Curves and AUC

Receiver Operating Characteristic (ROC) curves plot the performance of binary classification models. They illustrate the relationship between the True Positive Rate (Sensitivity) and the False Positive Rate (1 - Specificity) at varying threshold values for classifying the positive class. These thresholds are the points where the model transitions between designating a case as True or False for the dependent variable, often based on predicted probabilities.

The Area Under the Curve (AUC) represents the area under the ROC curve. A higher AUC generally indicates a stronger model, but it's important to note that a perfect model has an AUC of 1.0, and a random or ineffective model has an AUC of 0.5. Therefore, higher AUC values signify better discriminative ability.

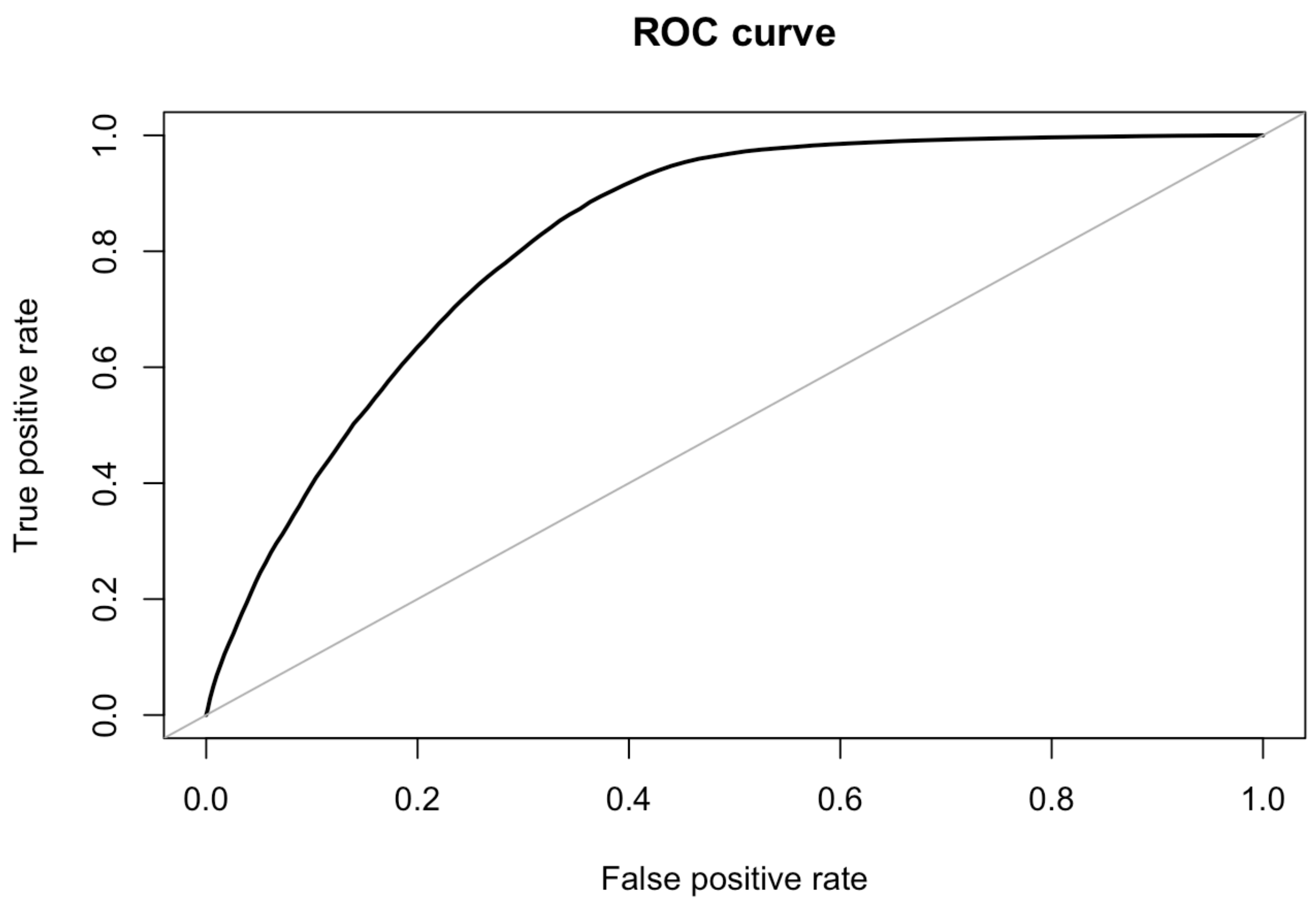
```r
prob_out_rf_prob <- predict(rf_model, newdata = valid, type = 'prob')[,2]

# Random Forest
roc.curve(response = valid$smoking, predicted = prob_out_rf_prob)
```



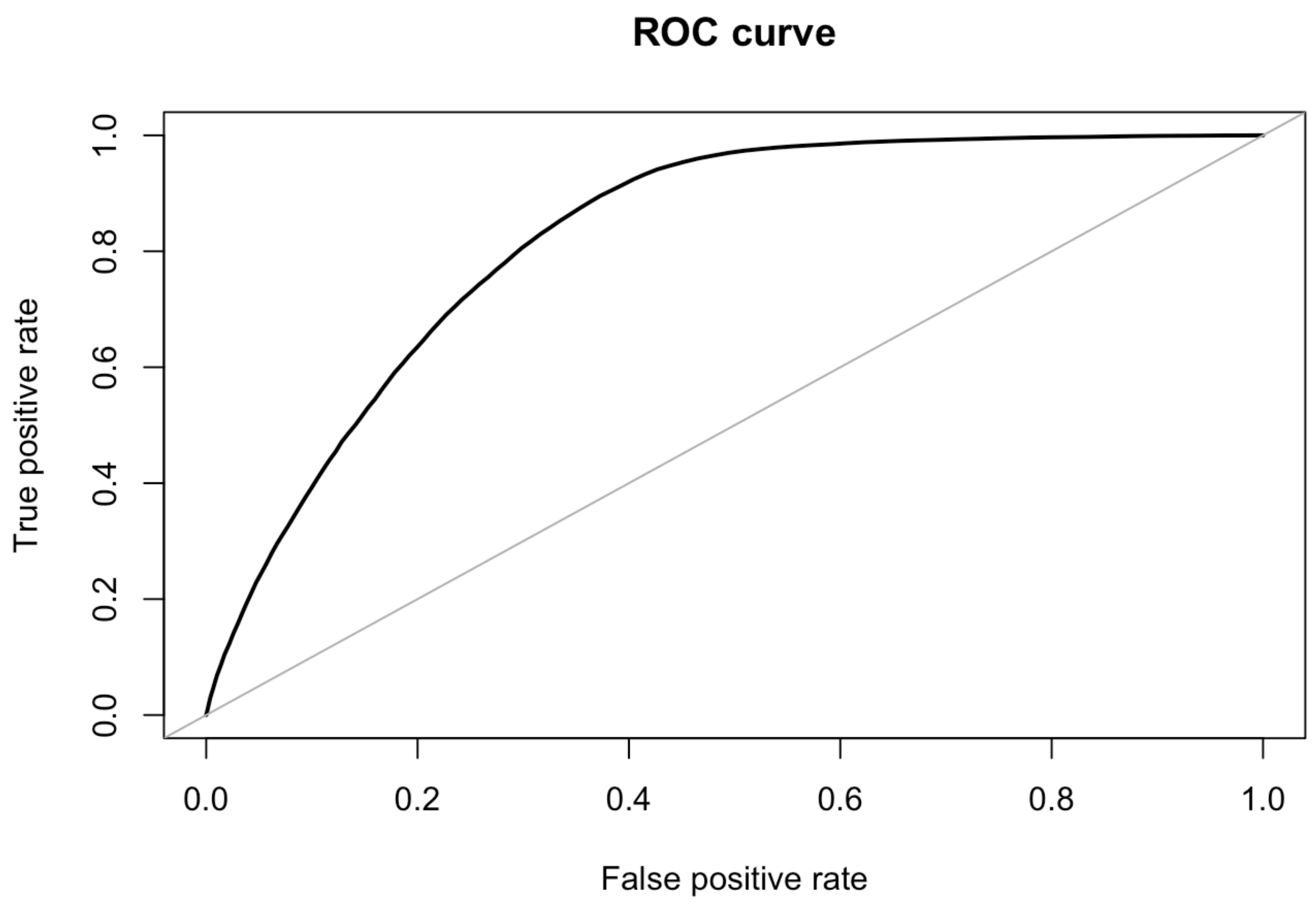
```
## Area under the curve (AUC): 0.858
```

```r
# Logit 1
roc.curve(response = valid$smoking, predicted = pred_out_logit1)
```



```
## Area under the curve (AUC): 0.825
```

```
# Logit 2
roc.curve(response = valid$smoking, predicted = pred_out_logit2)
```



```
## Area under the curve (AUC): 0.825
```

Our Random Forest model has the largest Area Under the Curve, indicating that it has the best discriminative power out of the three models.

While our Logistic Regression models demonstrated competitive performance, the Random Forest approach showcased its effectiveness in leveraging diverse bio-signals for accurate identification of smokers and non-smokers. This underscores the potential of machine learning, particularly Random Forest, in elucidating the nuanced relationship between bio-signals and smoking habits for enhanced predictive modeling.