Problem Set 2 Solution

I. Case study in predicting a major smoking caused disease using random forests and logistic regression

0. Data set up

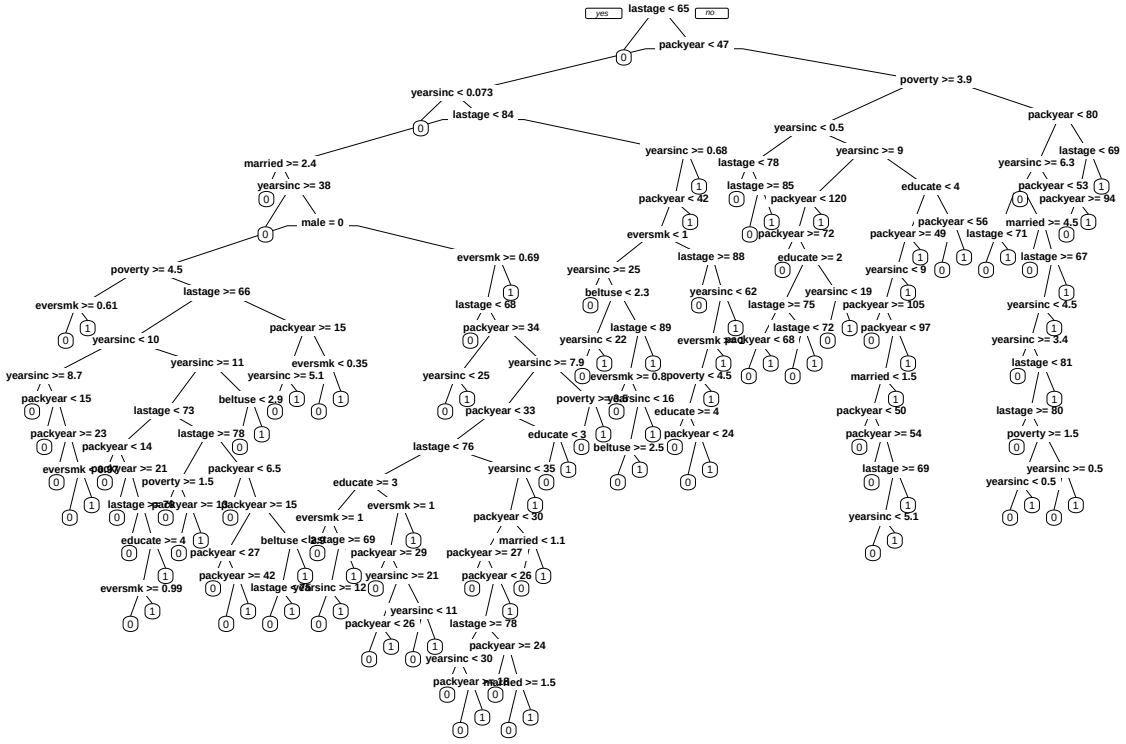
Prior to any data analysis we will be imputing the missing values using the rfImpute function. Then we will utilize this imputed data for the CART, parametric model and random forest. You may have a different approach to handle the imputation! The code below does the imputation for the missing values in the predictor variables and then creates the training and testing data.

```
## Read data in
load('./nmes.rdata')
nmes[nmes=='.'] = NA

## Prepare the data
nmes = nmes %>%
        mutate(mscd = as.factor(ifelse(lc5 + chd5 > 0, 1, 0)),
           lastage = as.numeric(lastage),
           eversmk = as.numeric(eversmk),
           packyears = as.numeric(packyears),
           yearsince = as.numeric(yearsince),
           beltuse = as.numeric(beltuse),
           educate = as.numeric(educate),
           married = as.numeric(marital),
           poverty = as.numeric(povstalb),
           male = as.factor(male)) %>%
    select(mscd, lastage, eversmk, packyears,
           yearsince, beltuse, educate, married,
           poverty, male)

## The imputation takes a few minutes
## I ran this once and then saved the output
## then commented this code when I compiled the Rmd file
#set.seed(10)
#dat1 = rfImpute(nmes[, 2 : 10], nmes[,1])
#names(dat1)[1] = "mscd"
#save(dat1,file="ImputedNMES.rda")
load("ImputedNMES.rda")

## Construct the training and testing dataset
orig0 = which(dat1$mscd == 0)
orig1 = which(dat1$mscd == 1)
#orig1 = orig1[sample(seq(1, length(orig1)), size = 2 * length(orig1), replace = TRUE)]

## Create 70-30 split within subset
set.seed(429)
```

```
train.mscd.0 <- sample(1:length(orig0), floor(length(orig0)*0.7))
train.mscd.1 <- sample(1:length(orig1), floor(length(orig1)*0.7))

## Define the training and testing subsets
dat1.train = rbind(dat1[orig0, ][train.mscd.0, ], dat1[orig1, ][train.mscd.1, ])
dat1.test = rbind(dat1[orig0, ][-train.mscd.0, ], dat1[orig1, ][-train.mscd.1, ])

# Set up additional definitions of the
# training and testing data for use in
# CART and RF
par(mfrow = c(1, 1))
x.train = dat1.train[, 2 : 10]
y.train = dat1.train[, 1]
x.test = dat1.test[, 2 : 10]
y.test = dat1.test[, 1]
```

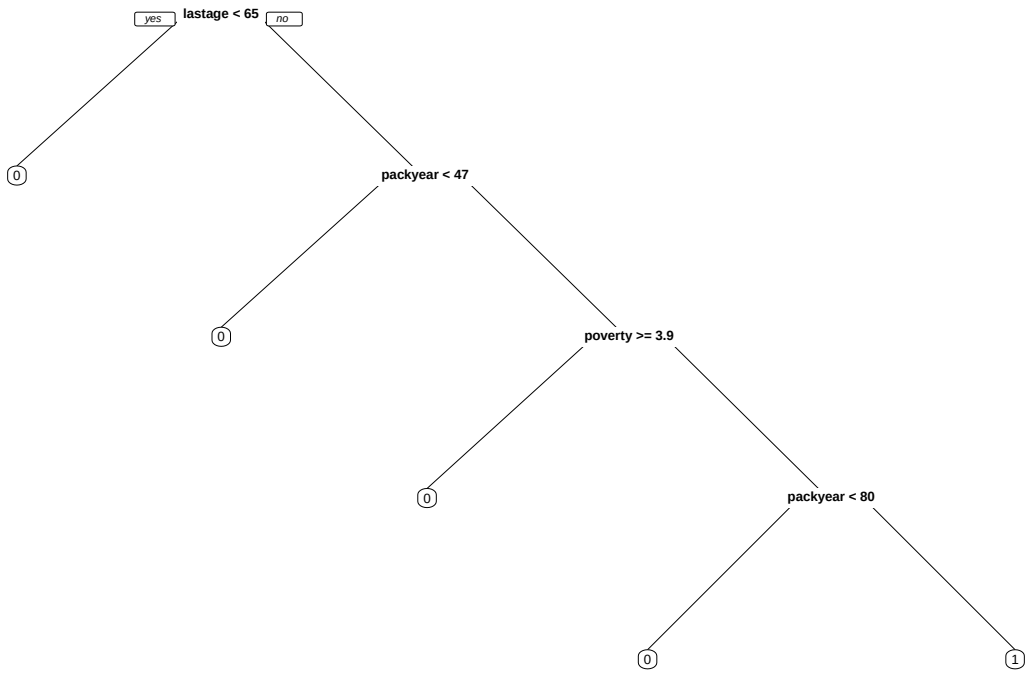
1. Use a CART to predict the probability that a person has a major smoking caused disease in the NMES data set. Key predictors are smoking history, age, sex, education, and poverty status.

```
cart.predict = rpart(mscd~., data = dat1.train,
                     method = "class",
                     control = rpart.control(minsplit=5, cp=0.001))
prp(cart.predict, faclen = 0, cex = 0.3, extra = 0)
```



Based on the result, we prune the CART

```
keep.cp = cart.predict$cptable[which.min(cart.predict$cptable[,"xerror"]), "CP"]
cart.prune <- prune(cart.predict, cp = keep.cp)
prp(cart.prune, faclen = 0, cex = 0.4, extra = 0)
```




```
print(cart.prune$cptable)
```

```
##         CP nsplit rel error xerror    xstd
## 1 0.004004      0     1.000 1.0000 0.02802
## 2 0.001779      4     0.984 0.9956 0.02796
```

```
print(cart.prune$variable.importance)
```

```
##   lastage packyears   educate   poverty yearsince   married   eversmk
##  94.35786  39.08772  10.77954  10.77059   9.29216   9.24497   0.07164
```

```
# CART prediction
cart.pred = 1 - as.integer(predict(cart.prune, newdata = dat1.test, type = "class"))
pred.rate = c(mean(cart.pred == dat1.test$mscd))
names(pred.rate) = c("CART")
print(pred.rate)
```

```
##   CART
## 0.8789
```

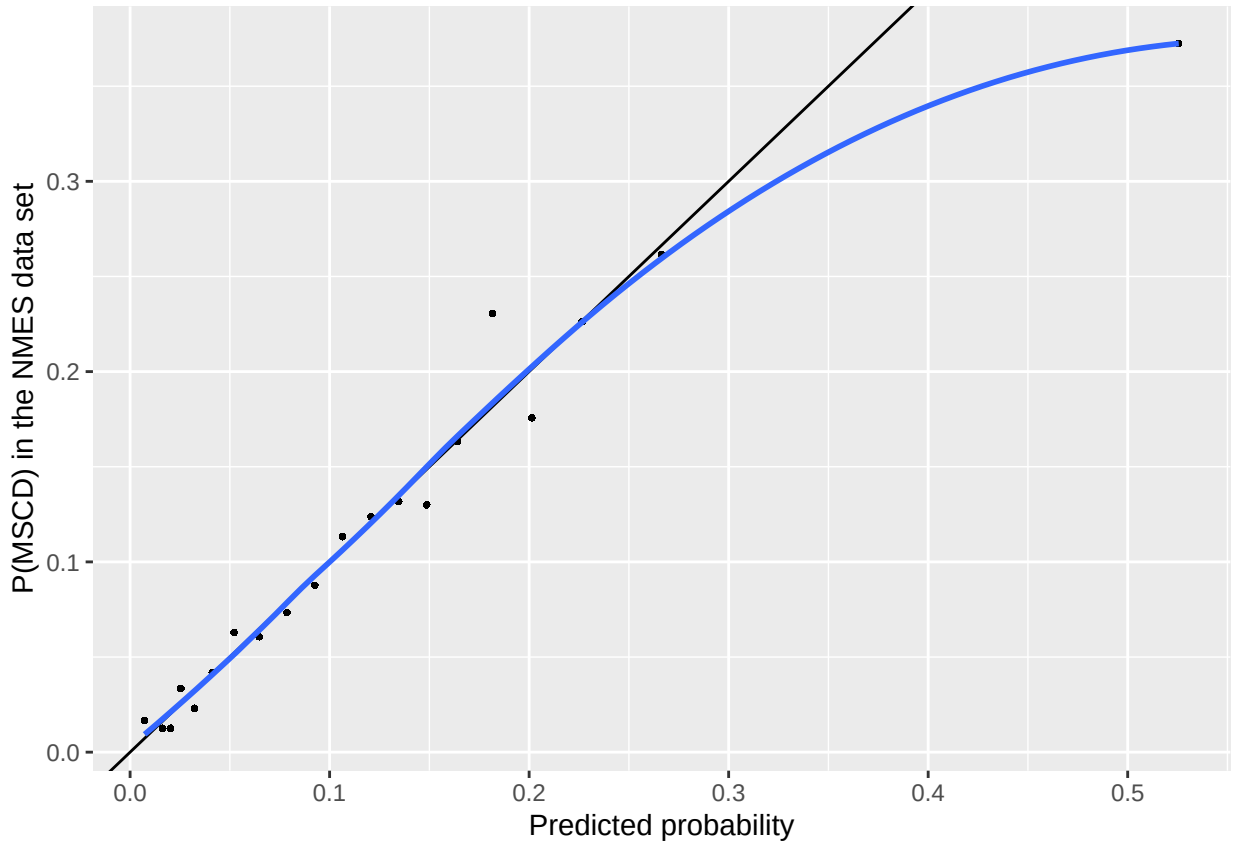
2. Use the CART results and prior health services knowledge to pose a logistic regression model to achieve the same prediction aim. Estimate its coefficients and check the model for consistency with the observations by comparing the observed rates within several bins of predicted rates. Check for extremely influential observations in your final model.

```r
model.binomial <- glm(mscd ~ ns(lastage,3) + packyears * yearsince +
                        educate + married + poverty + male + beltuse,
                      family = binomial(), data = dat1.train)
summary(model.binomial)
```

```
##
## Call:
## glm(formula = mscd ~ ns(lastage, 3) + packyears * yearsince +
##     educate + married + poverty + male + beltuse, family = binomial(),
##     data = dat1.train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.629  -0.568  -0.380  -0.207   3.033
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -4.153183   0.306547  -13.55  < 2e-16 ***
## ns(lastage, 3)1      2.329467   0.173222   13.45  < 2e-16 ***
## ns(lastage, 3)2      4.317025   0.472184    9.14  < 2e-16 ***
## ns(lastage, 3)3      2.440349   0.189669   12.87  < 2e-16 ***
## packyears            0.013134   0.001393    9.43  < 2e-16 ***
## yearsince            0.002225   0.003747    0.59  0.55261
## educate             -0.036883   0.039493   -0.93  0.35034
## married             -0.040956   0.036547   -1.12  0.26244
## poverty             -0.079010   0.028082   -2.81  0.00490 **
## male1                0.248079   0.073772    3.36  0.00077 ***
## beltuse              0.030405   0.043213    0.70  0.48168
## packyears:yearsince  0.000165   0.000149    1.11  0.26610
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6920.6  on 9551  degrees of freedom
## Residual deviance: 6119.5  on 9540  degrees of freedom
## AIC: 6144
##
## Number of Fisher Scoring iterations: 6
```

```r
par(mfrow = c(1,1))
#compute probabilities from fitted logOR
expit <- function(x){
  p <- exp(x)/(1+exp(x))
  return(p)
}
dat1.train = dat1.train %>%
    mutate(prob1 = expit(fitted.values(model.binomial)),
           pred = predict(model.binomial, type="response"))

step = 0.05
fit1c = cut(dat1.train$pred,
           breaks = c(0,
            quantile(dat1.train$pred, p = seq(step, 1 - step, by = step)),
```

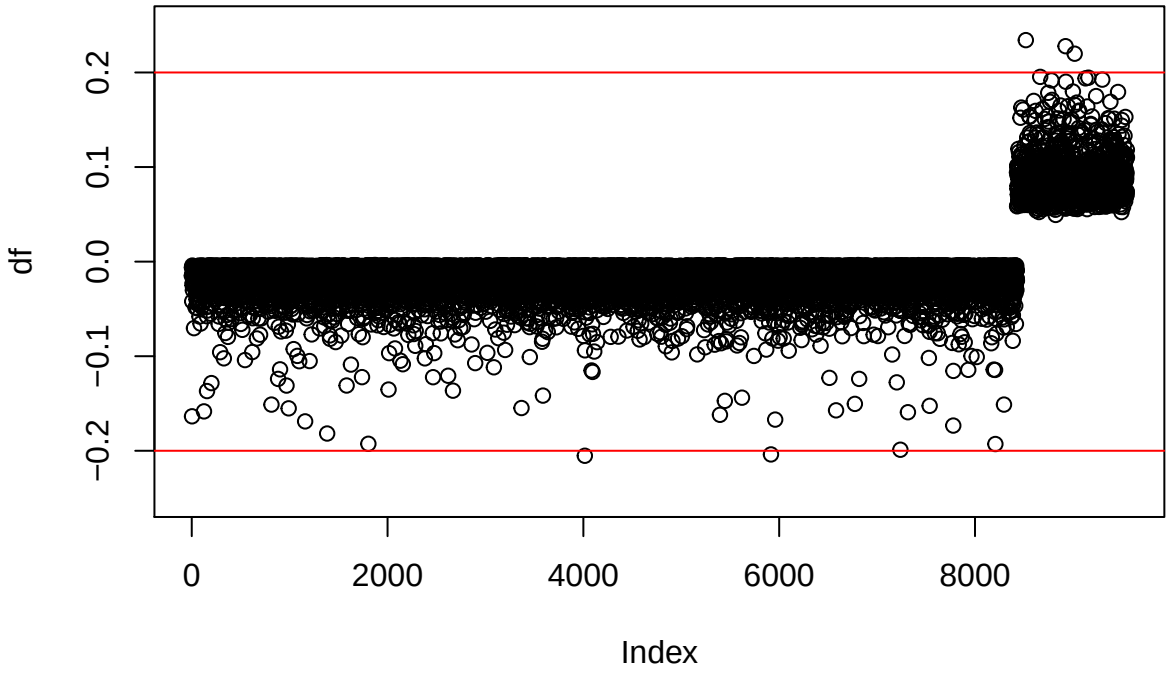
```
            max(dat1.train$pred)))

obs_mid = c()
obsx = c(0, quantile(dat1.train$pred, p = seq(step, 1 - step, by = step)), max(dat1.train$pred))
for(i in 1 : (length(obsx)-1)){
  obs_mid = c(obs_mid, mean(c(obsx[i], obsx[i + 1])))
}

dat1.train$bin = fit1c
nlevel = length(levels(fit1c))
pred.bin = c()
for(i in 1 : nlevel){
  dat = dat1.train[factor(fit1c) == levels(fit1c)[i], ]
  pred.bin = c(pred.bin, mean(dat$mscd == 1))
}
dat1.train$midobs = obs_mid[as.numeric(dat1.train$bin)]
dat1.train$pred_bin = pred.bin[as.numeric(dat1.train$bin)]

ggplot(data = dat1.train) +
  geom_point(aes(x = midobs, y = pred_bin), alpha = 1, size = .5) +
  geom_abline() +
  geom_smooth(aes(midobs, pred_bin), method = "loess") +
  ylab('P(MSCD) in the NMES data set') +
  xlab('Predicted probability')
```



The model exhibits good consistency, since the smooth curve (blue solid line) of observed rates - predicted

rates for 80 bins are similar to the 45° line.

```
df = dffits(model.binomial)
plot(df, ylim = c(-0.25, 0.25))
abline(h = -0.2, col = "red")
abline(h = 0.2, col = "red")
```

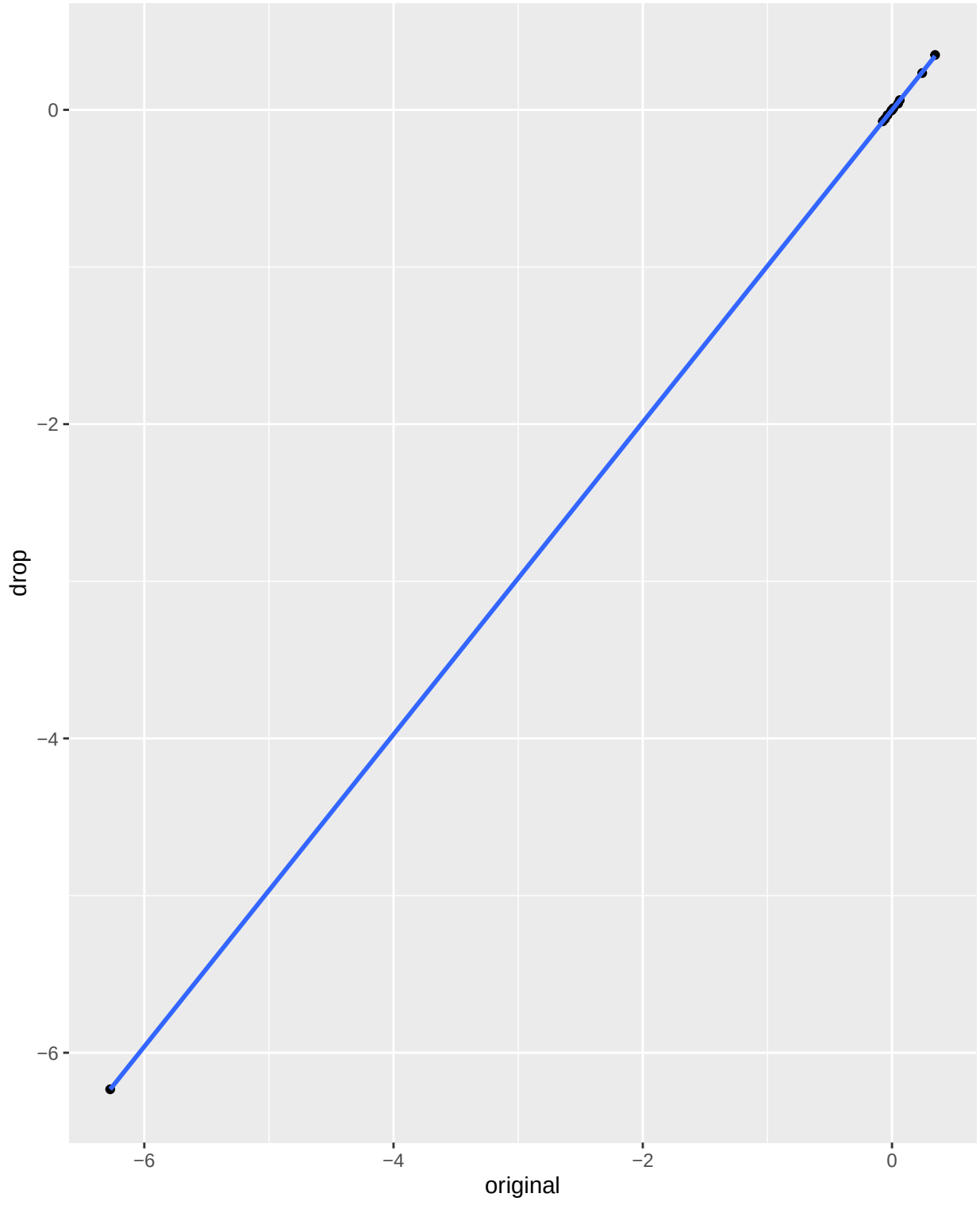



```
dropdata = dat1.train[abs(df) < 0.2, ]
```

To explore the extremely influential observations, we can use "dffits" to drop those outliers. Here we use absolute value of diffits(fit)>=0.2 as the criteria and dropped 6 observations. Next the model was refitted using the new dataset and the predicted values of the models before and after dropping the outliers were compared. The comparsion was illustrated in the plots below.

```
model.binomial <- glm(mscd ~ eversmk + lastage + packyears * yearsince +
                        beltuse + educate + married + poverty + male,
                      family = binomial(), data = dat1.train)

model.binomial.drop <- glm(mscd ~ eversmk + lastage + packyears * yearsince +
                             beltuse + educate + married + poverty + male,
                           family = binomial(), data = dropdata)

fit.smooth = lm(coef(model.binomial) ~ coef(model.binomial.drop))
coef.data = data.frame(original = coef(model.binomial.drop),
                       drop = coef(model.binomial))

coef.data %>%
    ggplot(aes(x = original, y = drop)) +
```

```
    geom_point() +
    geom_smooth(method='lm', formula = y ~ x)
```

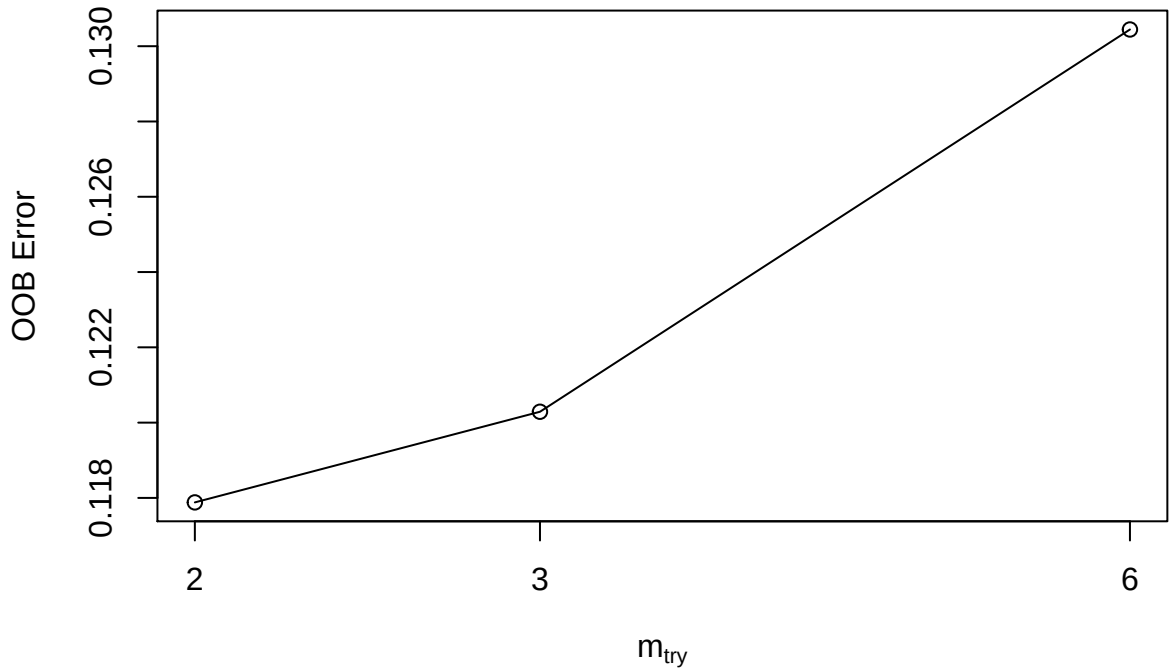


3. Use a random forest to predict the probability that a person has a major smoking caused disease in the NMES data set. Select an appropriate number of variables to randomly sample

as candidates at each split. Use the output to identify the variable importance.

```
# test for different mtry values (default, 3, 4, 5)
set.seed(10640)
## Tuned random forest result
tune = tuneRF(dat1.train[, 2 : 10], dat1.train[, 1], ntreeTry = 500)
```

```
## mtry = 3  OOB error = 12.03%
## Searching left ...
## mtry = 2     OOB error = 11.79%
## 0.02002 0.05
## Searching right ...
## mtry = 6     OOB error = 13.04%
## -0.08442 0.05
```



500 decision trees (a forest) was built using the Random Forest algorithm. We here choose to use 2 variables randomly sampled as candidates at each split, as it minimizes the OOB (out of bag) error estimate.

We summarize the result as below:

```
set.seed(10640)
rf2 = randomForest(dat1.train[, 2 : 10], dat1.train[, 1],
                   mtry = 2, ntree = 500, keep.forest=TRUE)
print(rf2)
```

```
##
## Call:
##  randomForest(x = dat1.train[, 2:10], y = dat1.train[, 1], ntree = 500,      mtry = 2, keep.forest =
##                Type of random forest: classification
```

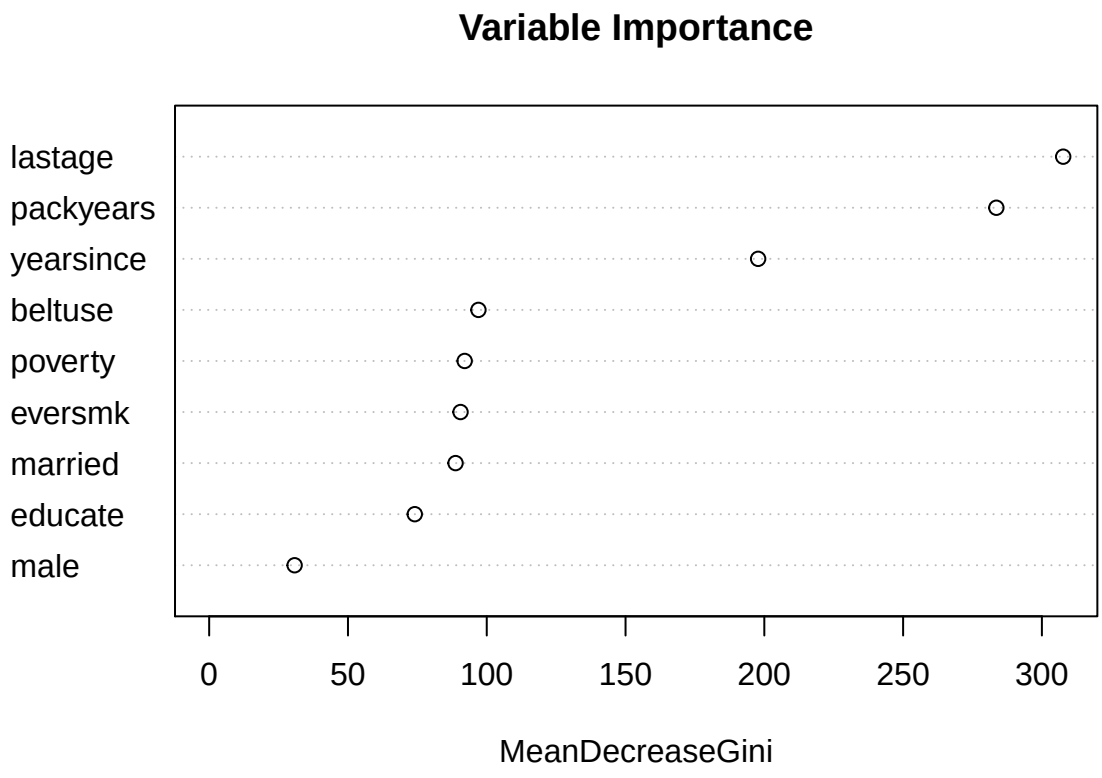
```
##                       Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 11.87%
## Confusion matrix:
##      0  1 class.error
## 0 8400 28    0.003322
## 1 1106 18    0.983986
```

```r
# Random forest prediction
pred.rate = rf2$err.rate[, 1]
pred.rate = 1 - rf2$err.rate[length(rf2$err.rate[,1])]
names(pred.rate) = c("RandomForest")
print(pred.rate)
```

```
## RandomForest
##       0.8813
```

We generated a random forest to predict the probability that a person has a major smoking caused disease in the NMES data set. Based on the variable importance calculated from the forest, key predictors of having a major smoking caused disease include age, packyears, yearsince, poverty status and beltuse. Next we look at variable importance across the whole forest. The table and plot below gives us a single score per variable aggregated across the whole forest.

```r
# Variable importance plot
varImpPlot(rf2, sort = T, main = "Variable Importance", n.var = 9, type = 2)
```

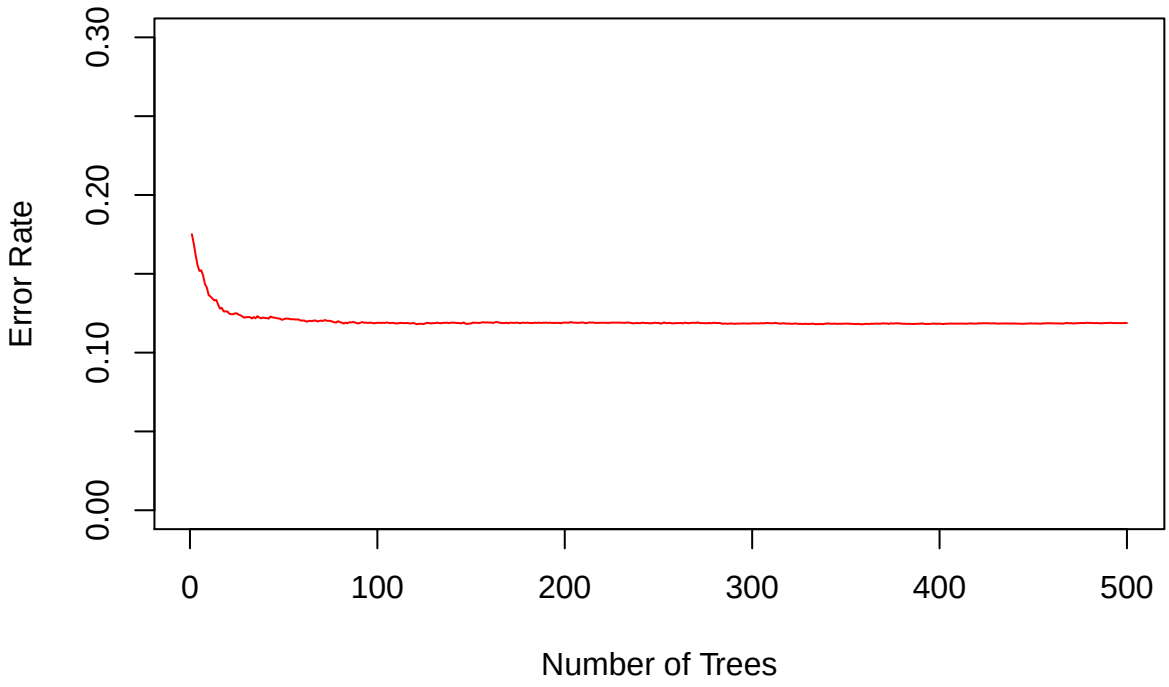


```r
# Variable Importance Table
var.imp <- data.frame(importance(rf2, type=2))
var.imp$Variables <- row.names(var.imp)
var.imp[order(var.imp$MeanDecreaseGini, decreasing = T), ]
```

```
##           MeanDecreaseGini Variables
## lastage             307.67   lastage
## packyears           283.59 packyears
## yearsince           197.77 yearsince
## beltuse              97.00   beltuse
## poverty              92.04   poverty
## eversmk              90.55   eversmk
## married              88.76   married
## educate              74.07   educate
## male                 30.77      male
```

From the plot and table, we can see that the order of importance of variables is arranged using Gini index. Briefly, Gini importance measures the average gain of purity by splits of a given variable. Here, we can see that age is the most important variable, followed by packyears, yearsince, beltuse, poverty, married, eversmk, educate, and male. This means that, when predicting whether the person has a major smoking caused disease in the NMES data set, the person's age, packyears and yearsince quitting (if a former smoker) are the most important factors.

```r
plot(1 : 500, rf2$err.rate[, 1],
     col = "red",
     type = "l",
     xlab = "Number of Trees",
     ylab = "Error Rate", ylim = c(0, 0.3))
```



We explored the out of bag misclassification error rate as a function of B, the number of trees. We see that the error stabilizes after roughly 100 trees.

4. For each of your prediction methods, calculate the sensitivity and specificity for classifying a person as having a major smoking caused disease at a threshold of your choosing. Now calculate the bootstrapped or cross-validated Receiver Operator Curve and its AUC for each method (i.e. CART or logistic regression model or random forest).

```
out.plot = data.frame(
  pr.cart = predict(cart.predict, dat1.test[, 2 : 10])[, 2],
  pr.glm = predict(model.binomial, dat1.test[, 2 : 10], type="response"),
  pr.rf = predict(rf2, dat1.test[, 2 : 10], type = "prob")[, 2],
  true_class = dat1.test[, 1])

## Histogram plot for CART
cart.plot = out.plot %>%
  select(pr.cart, true_class) %>%
  ggplot(aes(pr.cart, fill = true_class)) +
  geom_histogram(position = 'identity', bins = 30) +
  scale_y_log10() +
  xlab("Pr(Y = 1)") +
  ylim(0, 800) +
  ggtitle('CART')
```

```
## Scale for 'y' is already present. Adding another scale for 'y',
## which will replace the existing scale.
```

```r
## Histogram plot for Logistic Regression
glm.plot = out.plot %>%
  select(pr.glm, true_class) %>%
  ggplot(aes(pr.glm, fill = true_class)) +
  geom_histogram(position = 'identity', bins = 30) +
  scale_y_log10() +
  xlab("Pr(Y = 1)") +
  ylim(0, 800) +
  ggtitle('Logistic Regression')
```

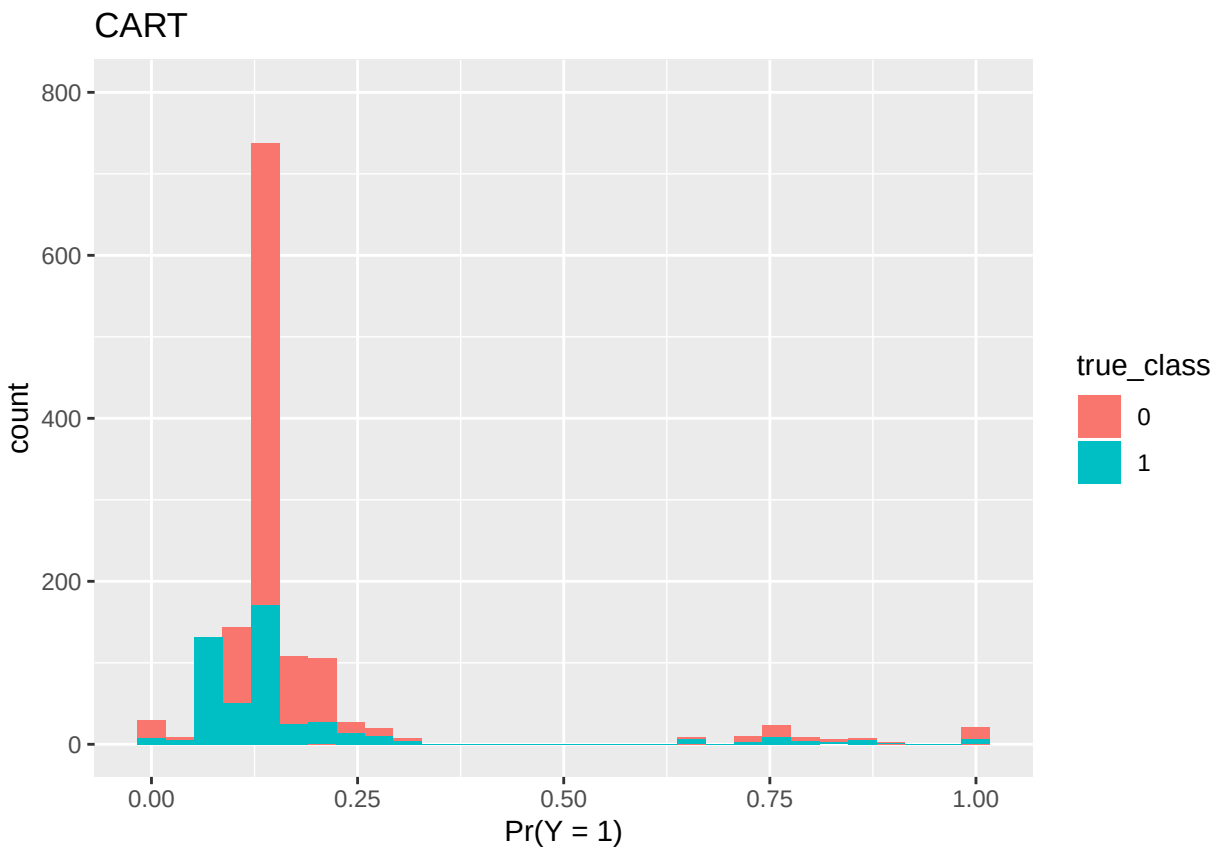
```
## Scale for 'y' is already present. Adding another scale for 'y',
## which will replace the existing scale.
```

```r
## Histogram plot for Random Forest
rf.plot = out.plot %>%
  select(pr.rf, true_class) %>%
  ggplot(aes(pr.rf, fill = true_class)) +
  geom_histogram(position = 'identity', bins = 30) +
  scale_y_log10() +
  xlab("Pr(Y = 1)") +
  ylim(0, 800) +
  ggtitle("Random Forest")
```

```
## Scale for 'y' is already present. Adding another scale for 'y',
## which will replace the existing scale.
```

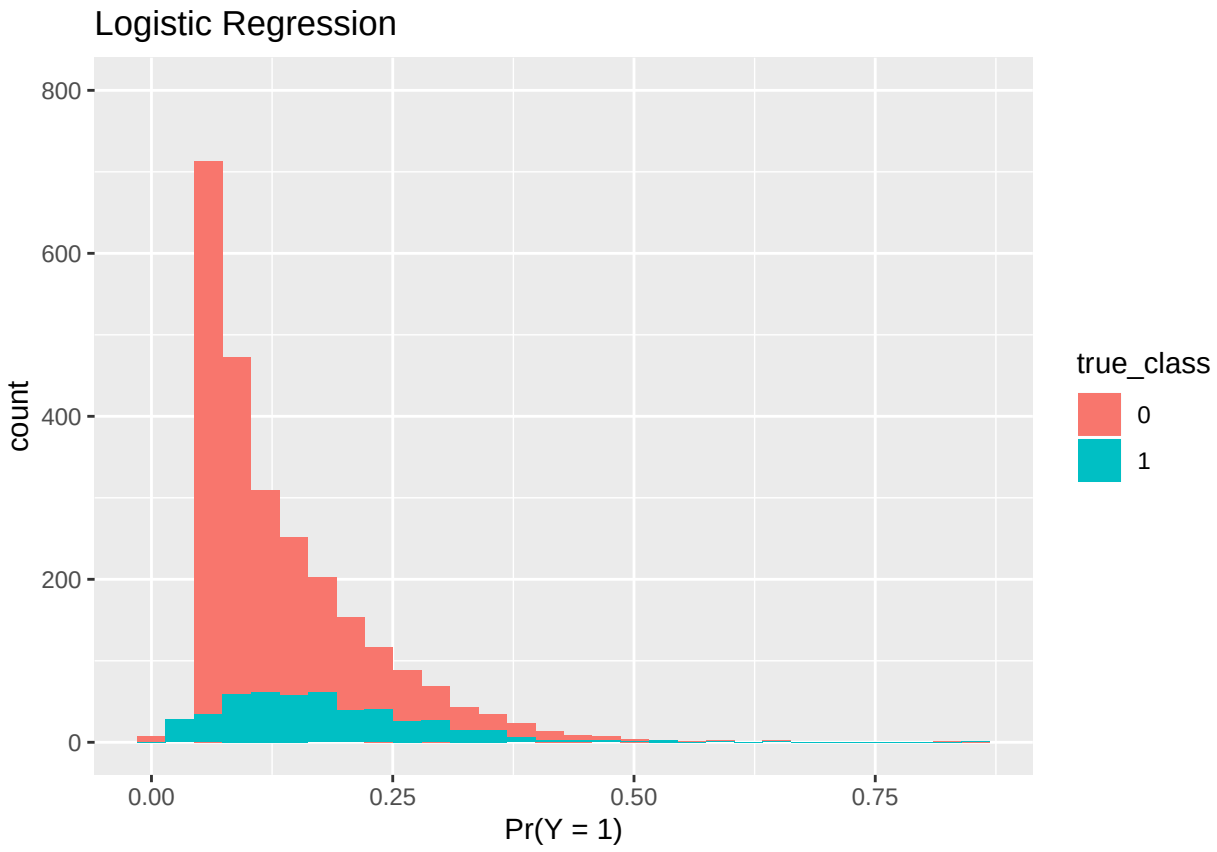
```r
cart.plot
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



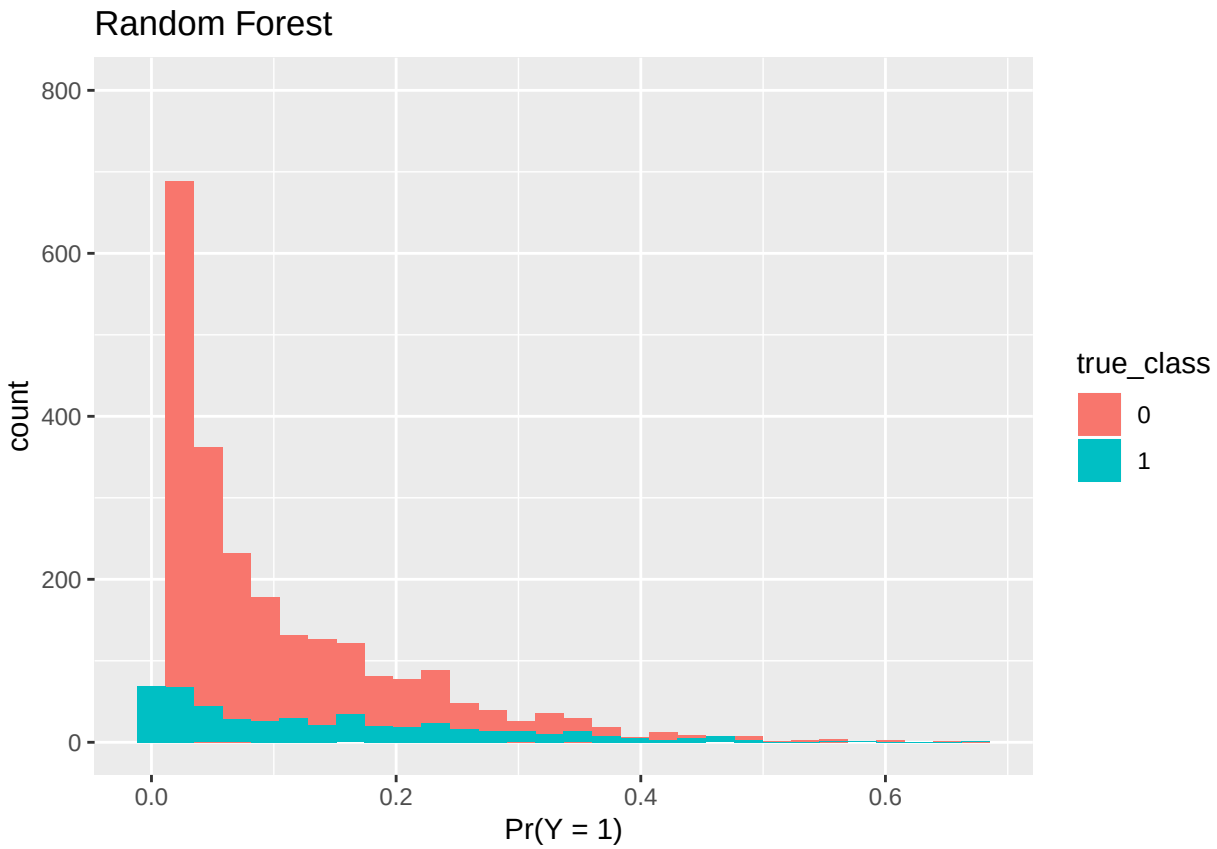

```
glm.plot
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```

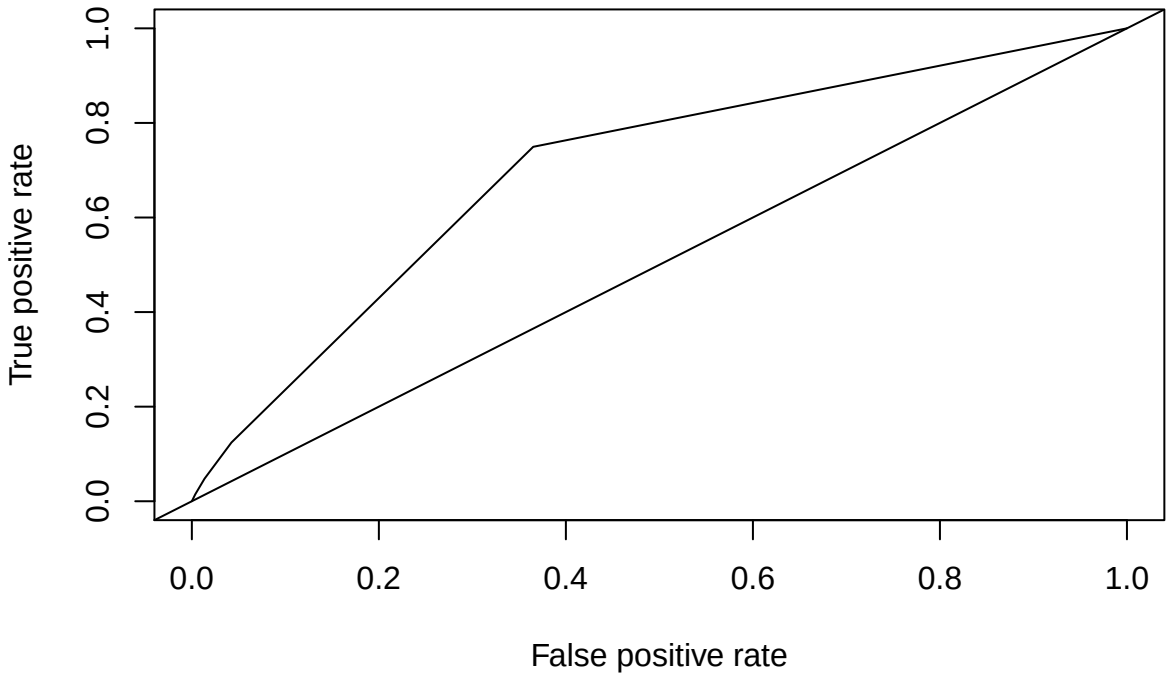



```
rf.plot
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



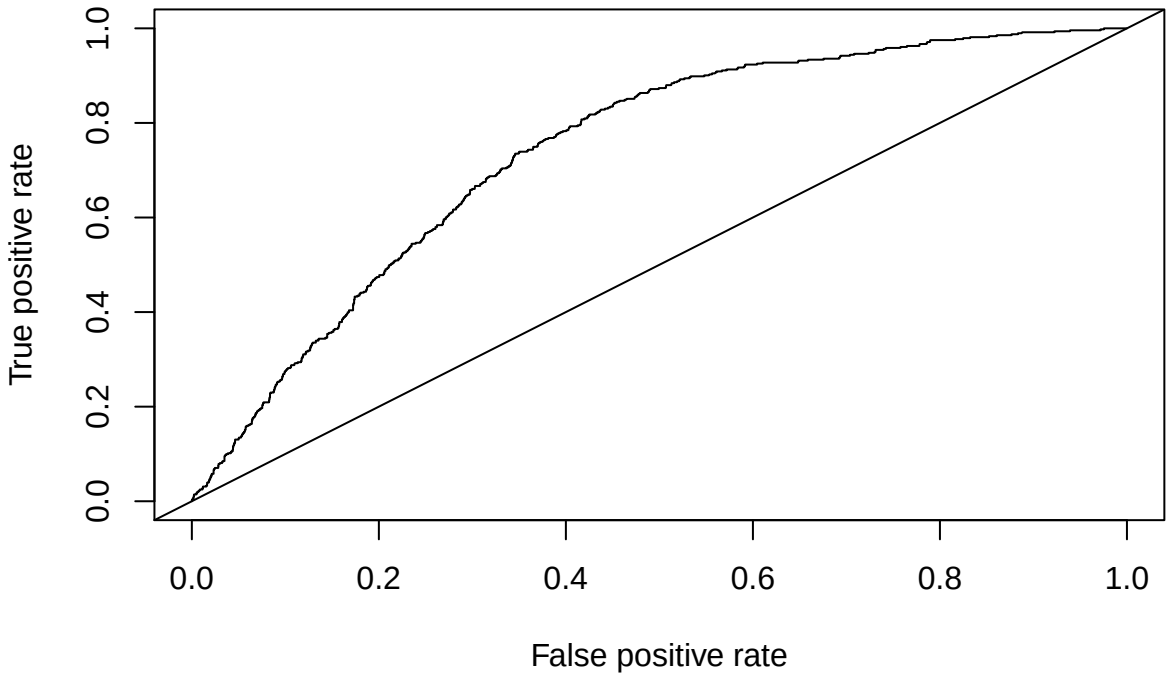

```
thresh = 0.3 #You choose your own threshold

cart.yhat = predict(cart.prune, newdata = dat1.test, type = "prob")
j.cart = table(cart.yhat[, 2] > thresh, dat1.test$mscd);
cart.sens = j.cart[4] / (j.cart[4] + j.cart[3]);
cart.spec = j.cart[1] / (j.cart[1] + j.cart[2]);
pred.cart.test = prediction(cart.yhat[, 2], dat1.test$mscd)
perf.cart.test = performance(pred.cart.test, "tpr", "fpr")
plot(perf.cart.test);
abline(0, 1)
```




```
auc.cart.test = performance(pred.cart.test, "auc")
auc.cart.test@y.values
```

```
## [[1]]
## [1] 0.6992
```

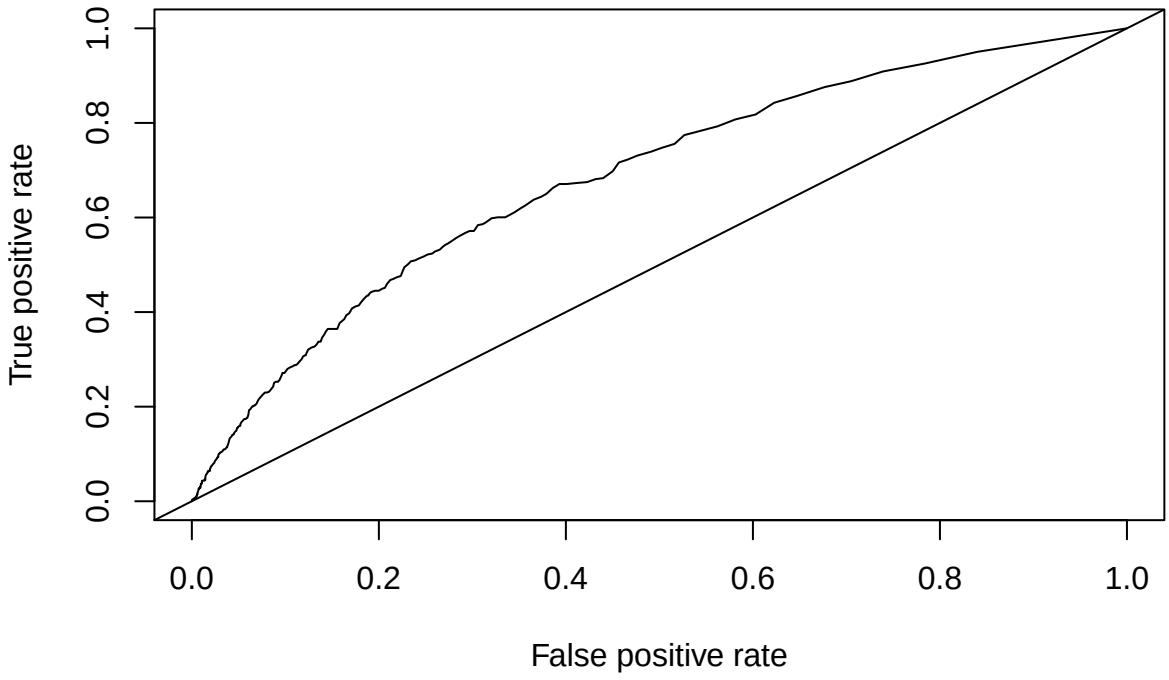
```
glm.yhat = predict.glm(model.binomial, newdata = dat1.test, type = "response")
j.glm = table(glm.yhat > thresh, dat1.test$mscd);
glm.sens = j.glm[4] / (j.glm[4] + j.glm[3]);
glm.spec = j.glm[1] / (j.glm[1] + j.glm[2]);
pred.glm.test = prediction(glm.yhat, dat1.test$mscd)
perf.glm.test = performance(pred.glm.test, "tpr", "fpr")
plot(perf.glm.test);
abline(0, 1)
```




```
auc.glm.test = performance(pred.glm.test, "auc")
auc.glm.test@y.values
```

```
## [[1]]
## [1] 0.7394
```

```
rf.yhat = predict(rf2, newdata = dat1.test, type = "prob")
j.rf = table(rf.yhat[, 2] > thresh, dat1.test$mscd);
rf.sens = j.rf[4] / (j.rf[4] + j.rf[3]);
rf.spec = j.rf[1] / (j.rf[1] + j.rf[2]);
pred.rf.test = prediction(rf.yhat[,2], dat1.test$mscd)
perf.rf.test = performance(pred.rf.test, "tpr", "fpr")
plot(perf.rf.test);
abline(0,1)
```




```
auc.rf.test = performance(pred.rf.test, "auc")
auc.rf.test@y.values
```

```
## [[1]]
## [1] 0.6845
```

```
# Cross-validated AUC
cart_auc = vector("list", 5)
lr_auc = vector("list", 5)
rf_auc = vector("list", 5)
cart_cv = lr_cv = rf_cv = mscd_cv = NULL
set.seed(2021)
folds_ind = createDataPartition(dat1.test$mscd, times = 5)
for(i in 1 : 5){
  train = dat1.test[-folds_ind[[i]], ]
  x.train = train[, 2 : 10]
  y.train = as.factor(train[, 1])
  test = dat1.test[folds_ind[[i]], ]
  x.test = test[, 2 : 10]
  y.test = as.factor(test[, 1])
  mscd_cv = c(mscd_cv,y.test)

  rf_fold = randomForest(train[, 2 : 10], train[, 1],
                         mtry = 2, ntree = 500, keep.forest=TRUE)
  cart_fold = rpart(mscd~., data = train, method = "class",
                    control = rpart.control(minsplit = 5, cp = 0.001))
  lr_fold = glm(data = train, mscd ~ ns(lastage,3) + packyears * yearsince +
```

```
                  beltuse + educate + married + poverty + male,
                family = binomial())

  cart_cv = c(cart_cv,predict(cart_fold,newdata=test,type="prob")[,2])
  lr_cv = c(lr_cv,predict.glm(lr_fold,newdata=test,type="response"))
  rf_cv = c(rf_cv,predict(rf_fold,newdata=test,type="prob")[,2])
}

## Compute the CV-AUC
  pred.cart.test = prediction(cart_cv, mscd_cv)
  auc.cart.test = performance(pred.cart.test, "auc")@y.values
  pred.glm.test = prediction(lr_cv,mscd_cv)
  auc.glm.test = performance(pred.glm.test, "auc")@y.values
  pred.rf.test = prediction(rf_cv,mscd_cv)
  auc.rf.test = performance(pred.rf.test, "auc")@y.values

print(paste("Cross-validated AUC for CART", round(as.numeric(auc.cart.test),3)))
```

```
## [1] "Cross-validated AUC for CART 0.598"
```

```
print(paste("Cross-validated AUC for logistic regression:", round(as.numeric(auc.glm.test),3)))
```

```
## [1] "Cross-validated AUC for logistic regression: 0.736"
```

```
print(paste("Cross-validated AUC for random forest", round(as.numeric(auc.rf.test),3)))
```

```
## [1] "Cross-validated AUC for random forest 0.708"
```

5. In a page or less, summarize your findings about the prediction of large expenditures and compare the three methods. Be numerate and avoid unnecessary statistical jargon.

The goal of our analysis was to predict the presence of a major smoking caused disease (mscd) as a function of smoking history, age, gender, poverty status, education background, seat belt use and marriage status. There was considerable missing data for the smoking history variables, so prior to any modeling we utilized an imputation approach that imputed missing values for the predictor variables using surrogate variables within a random forest model.

We considered three prediction modeling approaches: a classification tree, a parametric logistic regression model and a random forest. The dataset (including imputed values) was partitioned into a training and testing dataset according to a 70:30 split made separately within persons with and without a mscd.

We evaluated our models using the testing sample and calculated the sensitivity and specificity for each of the three approaches using a threshold of 0.3. For the classification tree, we estimated the sensitivity and specificity as 0.1242 and 0.9577, respectively. The sensitivity and specificity, respectively, were 0.118 and 0.9552 for the parametric logistic regression model, and 0.1346 and 0.9588 for the random forest model. The parametric logistic regression model had the highest sensitity at the 0.3 threshold. The cross-validated AUC from the ROC curve is 0.598, 0.736 and 0.708 for the classification tree, the parametric logistic regression model, and random forest, respectively. For our example. the parametric logistic regression model yielded the largest cross-validated AUC.

The strength of the classification tree is the ability to identify higher order interactions; with the disadvantage of requiring step functions to describe continuous variables. The strength of parametric logistic regression is that is versatile, allowing for smooth functions of continuous variables (eg. natural spline of age with 3 degrees of freedom), or to add interaction terms based on prior beliefs. It is my belief that the parametric logistic regression model is superior to the other two approaches in our data example based on the ability to smoothly model age, packyears and yearsince quitting smoking. The weaknesses are that it is often hard to identify higher order interactions and/or interpret these results from the fit of the model.

Random forests reduce overfitting by averaging several classification trees. Also, by using multiple trees, the chance of stumbling across a classifier that doesn't perform well because of the relationship between the training data for that tree is reduced. Weaknesses of the random forest approach includes the inability to translate the findings to a "model" that may be described to users of the prediction model and the potential to be computationally more expensive (although this did not hold in our example).