Stats 315B: Homework 1

Joe Higgins, Austin Wang, Jessica Wetstone

Due 5/20/2018

Question 1

Random forests predict with an ensemble of bagged trees each trained on a bootstrap sample randomly drawn from the original training data. Additional random variation among the trees is induced by choosing the variable for each split from a small randomly chosen subset of all of the predictor variables when building each tree. What are the advantages and disadvantages of this random variable selection strategy? How can one introduce additional tree variation in the forest without randomly selecting subsets of variables?

The advantages of choosing a random subset of predictor variables are:

- The random variation introduced by choosing the variable for each split from a randomly chosen subset de-correlates the resulting trees in the forest, forcing them to be different from each other. This has the effect of decreasing the variance of the ensemble's estimate, since the variance of the mean of a collection random variables is directly related to the correlation of those variables. Decreasing the correlation of the trees therefore decreases the variance of the forest.
- Because we are no longer doing an exhaustive search over all predictor variables, each split requires less processing power during the training process. Given modern computer architecture, this advantage matters less since each tree in the ensemble is independent and can be computed simultaneously in a different parallel process.

The disadvantage of this approach is that it can result in trees with very weak predictive power. Suppose that none of the best predictors were available for a given tree's first several splits – this would result in a highly sub-optimal tree. However, if this method decreases the variance of the forest enough to offset potential increases in bias, it will lower our model's error.

Another way to inject randomness into the process and de-correlate the trees is to decrease the size of the bootstrapped sample of the training data. Each individual tree is now more likely to overfit the smaller sample, increasing tree variation within the forest.

Question 2

Why is it necessary to use regularization in linear regression when the number of predictor variables is greater than the number of observations in the training sample? Explain how regularization helps in this case. Are there other situations where regularization might help? What is the potential disadvantage of introducing regularization? Why is sparsity a reasonable assumption in the boosting context. Is it always? If not, why not?

If the number of predictor variables n is greater than the number of observations in the training sample N, then there is an infinite number of solutions that will satisfy the linear regression problem – and there will be very high variance in the solution derived from that training sample. There are certain penalties (L1) that will set coefficients equal to zero, reducing the number of predictor variables in the solution.

Regularization always decreases the size of the function class, reducing variance. Therefore, in situations where our model is likely to have high variance, such as when we have a small training set, regularization would help.

A potential disadvantage of regularization is that it could increase bias - with strong penalties, the increase in squared bias could overtake the decrease in variance, ultimately increasing the mean-squared error.

In the boosting context, each predictor is a possible tree that could be built on the training data from the set of all possible trees. It is reasonable to assume that most trees - out of the set of all possible trees (all combinations of variables and all possible splits) - will do very poorly on the overall prediction problem. So we would want most of our coefficients in the linear regression to be 0.

Sparsity is not always a necessary assumption in linear regression, especially if your feature space is small. For example, if we were predicting house prices from zip code and square footage, we would expect both variables to be important.

Question 3

Show that the convex members of the power family of penalties, except for the lasso, have the property that solutions to $\hat{a}(\lambda) = argmin_a \hat{R}(a) + \lambda P_\gamma(a)$ have nonzero values for all coefficients at each path point indexed by λ. By contrast the convex members of the elastic net (except ridge) can produce solutions with many zero valued coefficients at various path points.

Question 4

Show that the variable x_{j^*} that has the maximum absolute correlation with $j^* = argmax_{1\leq j\leq J}|E(yx_j)|$ is the same as the one that best predicts y using squared—error loss $j^* = argmin_{1\leq j\leq J}\ min_\rho E[y-\rho x_j]^2$. This shows that the base learner most correlated with the generalized residual is the one that best predicts it with squared—error loss.

Let $f(\rho,\ x_j,\ y) = E[y-\rho x_j]^2$. We first find $min_\rho\ f(\rho,\ x_j,\ y)$ by taking the partial derivative of f with respect to ρ and setting it equal to 0:

$$\begin{aligned}
f(\rho,\ x_j,\ y) &= E[y-\rho x_j]^2\\
&= E[y^2 - 2\rho y x_j + \rho^2 x_j^2]\\
&= E[y^2] - 2\rho E[yx_j] + \rho^2 E[x_j^2]\\
&= E[y^2] - 2\rho E[yx_j] + \rho^2 \qquad \text{since } E[x_j^2]=1
\end{aligned}$$

$$\begin{aligned}
\implies \frac{\partial f}{\partial \rho} &= -2E[yx_j] + 2\rho\\
\implies 0 &= -2E[yx_j] + 2\rho^*\\
\implies \rho^* &= E[yx_j]
\end{aligned}$$

$$\begin{aligned}
\implies min_\rho\ f(\rho,\ x_j,\ y) &= f(\rho^*,\ x_j,\ y)\\
&= E[y-\rho^* x_j]^2\\
&= E[y^2] - 2\rho^* E[yx_j] + {\rho^*}^2\\
&= E[y^2] - 2E[yx_j]E[yx_j] + (E[yx_j])^2\\
&= E[y^2] - (E[yx_j])^2
\end{aligned}$$

Note that this is a unique minimum value because f is strictly convex in ρ. We know this because $\frac{\partial^2 f}{\partial \rho^2} > 0$:

$$\begin{aligned}\frac{\partial f}{\partial \rho} &= -2E[yx_j] + 2\rho \\ \implies \frac{\partial^2 f}{\partial \rho^2} &= 2 \\ &> 0\end{aligned}$$

Then:

$$\begin{aligned}argmin_{1\leq j\leq J}\ min_\rho E[y-\rho x_j]^2 &= argmin_{1\leq j\leq J}\ (E[y^2] - (E[yx_j])^2) \\ &= argmin_{1\leq j\leq J}\ -(E[yx_j])^2 \\ &= argmax_{1\leq j\leq J}\ (E[yx_j])^2 \\ &= argmax_{1\leq j\leq J}\ |E[yx_j]|^2 \\ &= argmax_{1\leq j\leq J}\ |E[yx_j]| \qquad \text{since } f(x)=\sqrt{x} \text{ is monotonically increasing for } x>0\end{aligned}$$

Therefore, $argmax_{1\leq j\leq J}\ |E[yx_j]| = argmin_{1\leq j\leq J}\ min_\rho E[y-\rho x_j]^2$, so the base learner most correlated with the generalized residual is the one that best predicts it with squared—error loss. ■

Question 5

Let $\mathbf{z}_l = \{z_1, ..., z_l\}$ be a subset of the predictor variables $\mathbf{x} = \{x_1, ..., x_n\}$ and $\mathbf{z}_{\setminus l}$ the complement subset, i.e. $\mathbf{z}_l \cup \mathbf{z}_{\setminus l} = \mathbf{x}$. Show that if a function $F(\mathbf{x})$ is additive in $\mathbf{z}_l$ and $\mathbf{z}_{\setminus l}$, i.e.

$$F(\mathbf{x}) = F_l(\mathbf{z}_l) + F_{\setminus l}(\mathbf{z}_{\setminus l})$$

then the partial dependence of $F(\mathbf{x})$ on $\mathbf{z}_l$ is $F_l(\mathbf{z}_l)$ up to an additive constant. This is the dependence of $F(\mathbf{x})$ on $\mathbf{z}_l$ accounting for the effect of the other variables $\mathbf{z}_{\setminus l}$. Show that this need not be the case for $E[F(\mathbf{x}) \mid \mathbf{z}_l]$ which is the dependence of $F(\mathbf{x})$ on $\mathbf{z}_l$ ignoring the other variables $\mathbf{z}_{\setminus l}$. Under what conditions would the two be the same?

```r
#Utility functions
misclass_rate <- function(y_hat, y){
  correct <- y_hat == y
  pct_correct = sum(correct)/length(correct)
  return(1 - pct_correct)
}

percent <- function(x, digits = 1, format = "f", ...) {
  paste0(formatC(100 * x, format = format, digits = digits, ...), "%")
}
```

Question 6

Binary classification: Spam Email. The data set for this problem is spam_stats315B.csv, with documentation files spam_stats315B_info.txt and spam_stats315B_names,txt. The data set is a collection of 4601 emails of which 1813 were considered spam, i.e. unsolicited commercial email. The data set consists of 58 attributes of which 57 are continuous predictors and one is a class label that indicates whether the email was considered spam (1) or not (0). Among the 57 predictor attributes are: percentage of the word "free" in the email, percentage of exclamation marks in the email, etc. See file spam_stats315B_names.txt for the full list of attributes. The goal is, of course, to predict whether or not an email is "spam". This data set is used for illustration in the tutorial Boosting with R Programming. The data set spam_stats315B_train.csv represents a subsample of these emails randomly selected from spam_stats315B.csv to be used for training. The file spam_stats315B_test.csv contains the remaining emails to be used for evaluating results.

```r
#remove all variables except for functions
rm(list = setdiff(ls(), lsf.str()))

#data labels
rflabs<-c("make", "address", "all", "3d", "our", "over", "remove",
  "internet","order", "mail", "receive", "will",
  "people", "report", "addresses","free", "business",
  "email", "you", "credit", "your", "font","000","money",
  "hp", "hpl", "george", "650", "lab", "labs",
  "telnet", "857", "data", "415", "85", "technology", "1999",
  "parts","pm", "direct", "cs", "meeting", "original", "project",
  "re","edu", "table", "conference", ";", "(", "[", "!", "$", "#",
  "CAPAVE", "CAPMAX", "CAPTOT","type")

#load the data
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)
train <- read.csv(file="spam_stats315B_train.csv", header=FALSE, sep=",")
test <- read.csv(file="spam_stats315B_test.csv", header=FALSE, sep=",")
colnames(train)<-rflabs
colnames(test)<-rflabs

#set labels and variables to avoid confusion
spam <- 1
nonspam <- 0
threshold = 0.50

#randomize order of data
set.seed(131)
x <- train[sample(nrow(train)),]
```

(a) Based on the training data, fit a gbm model for predicting whether or not an email is "spam", following the example in the tutorial. What is your estimate of the misclassification rate? Of all the spam emails of the test set what percentage was misclassified, and of all the non-spam emails in the test set what percentage was misclassified?

```r
#fit model on training data
set.seed(444) # random for bag.fraction
gbm0 <- gbm(type~., data=x, train.fraction=1,
            interaction.depth=4, shrinkage=.05,
            n.trees=2500, bag.fraction=0.5, cv.folds=5,
            distribution="bernoulli", verbose=F)

#Ground truths
y_train <- x$type
y_test <- test$type

#Predictions
nTreesForPrediction0 <- gbm.perf(gbm0, plot.it=FALSE, method="cv")
y_hat_train <- predict(gbm0, x, type="response", n.trees=nTreesForPrediction0)
y_hat_test <- predict(gbm0, test, type="response", n.trees=nTreesForPrediction0)

y_hat_train[y_hat_train >  threshold] <- spam
y_hat_train[y_hat_train <= threshold] <- nonspam
y_hat_test[y_hat_test >  threshold] <- spam
y_hat_test[y_hat_test <= threshold] <- nonspam

#Training misclassification rates:
cat("TRAIN overall misclassification rate: ",
    percent(misclass_rate(y_hat_train, y_train)), "\n")
cat("TRAIN spam    misclassification rate: ",
    percent(misclass_rate(y_hat_train[y_train==spam], y_train[y_train==spam])), "\n")
cat("TRAIN nonspam misclassification rate: ",
    percent(misclass_rate(y_hat_train[y_train==nonspam], y_train[y_train==nonspam])), "\n")


#Spam misclassification rates:
cat("TEST  overall misclassification rate: ",
    percent(misclass_rate(y_hat_test, y_test)), "\n")
cat("TEST  spam    misclassification rate: ",
    percent(misclass_rate(y_hat_test[y_test==spam], y_test[y_test==spam])), "\n")
#Non-Spam misclassiciation rates:
cat("TEST  nonspam misclassification rate: ",
    percent(misclass_rate(y_hat_test[y_test==nonspam], y_test[y_test==nonspam])), "\n")
```

```
## TRAIN overall misclassification rate:  0.2%
## TRAIN spam    misclassification rate:  0.4%
## TRAIN nonspam misclassification rate:  0.1%
## TEST  overall misclassification rate:  2.3%
## TEST  spam    misclassification rate:  2.6%
## TEST  nonspam misclassification rate:  2.2%
```

(b) Your classifier in part (a) can be used as a spam filter. One of the possible disadvantages of such a spam filter is that it might filter out too many good (non-spam) emails. Therefore, a better spam filter might be the one that penalizes misclassifying non-spam emails more heavily than the spam ones. Suppose that you want to build a spam filter that "throws out" no more that 0.3% of the good (non-spam) emails. You have to find and use a cost matrix that penalizes misclassifying "good" emails as"spam" more than misclassifying "spam" emails as "good" by the method of trial and error. Once you have constructed your final spam filter with the property described above, answer the following questions:

```
#set weights, fit model
w <- x$type
w[x$type == spam] <- 1
w[x$type == nonspam] <- 200

set.seed(444) # random for bag.fraction
gbm1 <- gbm(type ~ ., data=x,  cv.folds=20, train.fraction=1,
            interaction.depth=1,
            shrinkage=.05,
            n.trees=3000,
            bag.fraction=0.5,
            distribution="bernoulli", verbose=F,
            weights = w)
```

(i) What is the overall misclassification error of your final filter and what is the percentage of good emails and spam emails that were misclassified respectively?

```
#predictions
nTreesForPrediction1 <- gbm.perf(gbm1, plot.it=FALSE,method="cv")
y_hat_train <- predict(gbm1, x, type="response", n.trees=nTreesForPrediction1)
y_hat_test <- predict(gbm1, test, type="response", n.trees=nTreesForPrediction1)

y_hat_train[y_hat_train >  threshold] <- spam
y_hat_train[y_hat_train <= threshold] <- nonspam
y_hat_test[y_hat_test >  threshold] <- spam
y_hat_test[y_hat_test <= threshold] <- nonspam

#Training misclassification rates:
cat("TRAIN overall misclassification rate: ",
    percent(misclass_rate(y_hat_train, y_train)), "\n")
cat("TRAIN spam    misclassification rate: ",
    percent(misclass_rate(y_hat_train[y_train==spam], y_train[y_train==spam])), "\n")
cat("TRAIN nonspam misclassification rate: ",
    percent(misclass_rate(y_hat_train[y_train==nonspam], y_train[y_train==nonspam])), "\n")

#Spam misclassification rates:
cat("TEST  overall misclassification rate: ",
    percent(misclass_rate(y_hat_test, y_test)), "\n")
cat("TEST  spam    misclassification rate: ",
    percent(misclass_rate(y_hat_test[y_test==spam], y_test[y_test==spam])), "\n")
#Non-Spam misclassiciation rates:
cat("TEST  nonspam misclassification rate: ",
    percent(misclass_rate(y_hat_test[y_test==nonspam], y_test[y_test==nonspam])), "\n")
```

```
## TRAIN overall misclassification rate:  22.7%
## TRAIN spam    misclassification rate:  57.1%
## TRAIN nonspam misclassification rate:  0.0%
## TEST  overall misclassification rate:  24.1%
## TEST  spam    misclassification rate:  59.2%
## TEST  nonspam misclassification rate:  0.3%
```

(ii) What are the important variables in discriminating good emails from spam for your spam filter?

```r
#Unweighted filter
influential_vars <- summary.gbm(gbm0,main="RELATIVE INFLUENCE",order=TRUE,plotit=FALSE)
top5_unweighted <- gsub("`","",influential_vars$var[1:5])

#Weighted filter
influential_vars <- summary.gbm(gbm1,main="RELATIVE INFLUENCE",order=TRUE,plotit=FALSE)
top5_weighted <- gsub("`","",influential_vars$var[1:5])

cat("Top 5 influential variables, unweighted model: ",
    top5_unweighted, "\n")
```

```
## Top 5 influential variables, unweighted model:  ! $ remove hp free
```

```r
cat("Top 5 influential variables, weighted model: ",
    top5_weighted, "\n")
```

```
## Top 5 influential variables, weighted model:  CAPAVE remove CAPMAX credit business
```

(iii) Using the interpreting tools provided by gbm, describe the dependence of the response on the most important attributes.

Partial dependence plots show how a model's output is affected when varying a subset of variables (accounting for the other complementary subset of variables). Looking at the partial dependence, we can understand what the effect of changing a particular subset of variables' values has on the model's output (accounting for all other variables). This gives us a better understanding of the behavior of otherwise "black box" models. For this problem, we consider single variable partial dependence plots.

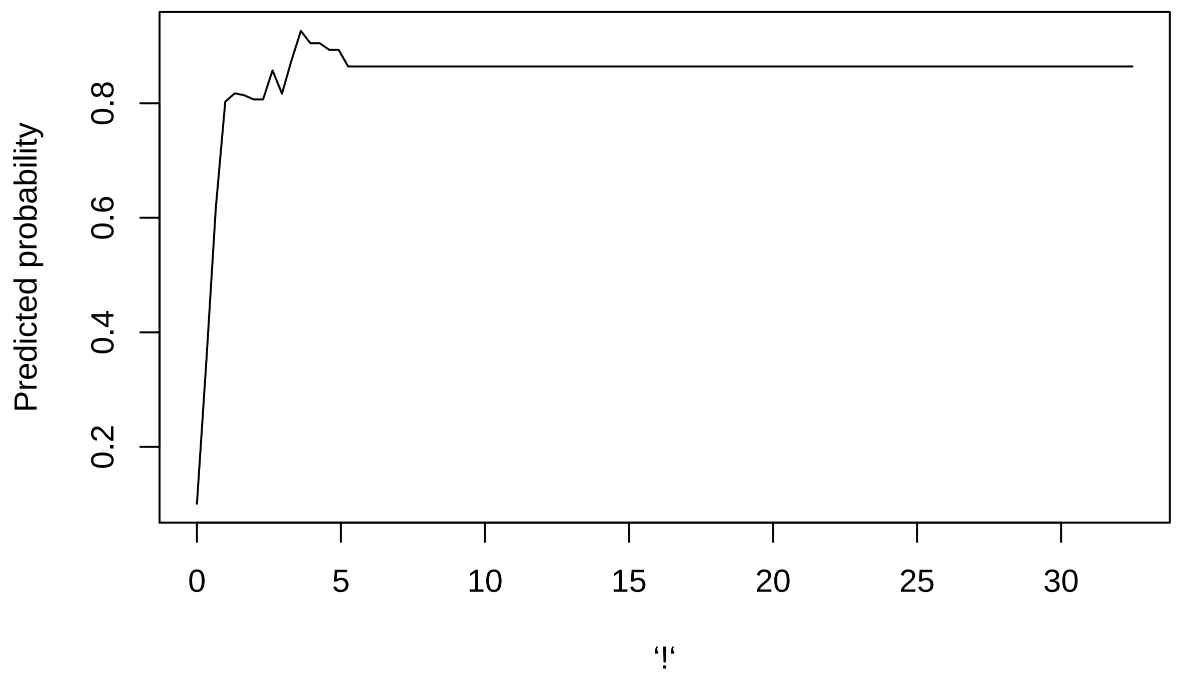
Unweighted model: Looking at the variable with the most relative influence, *"!"*, we can see that as we increase the frequency of *"!"* as a proportion of an email's length, it increases the model's output probability that it is spam. This means that emails with a high frequency of *"!"* relative to their length are indicative of spam since spam is the positve class and nonspam is the zero class.

We see opposite behavior in *"hp"*. Since George and his colleagues worked at Hewlett-Packard, emails with "hp" display the opposite effect and the higher we move "hp" in frequency the more likely it is that the email is legitimate and non-spam (according to our model).

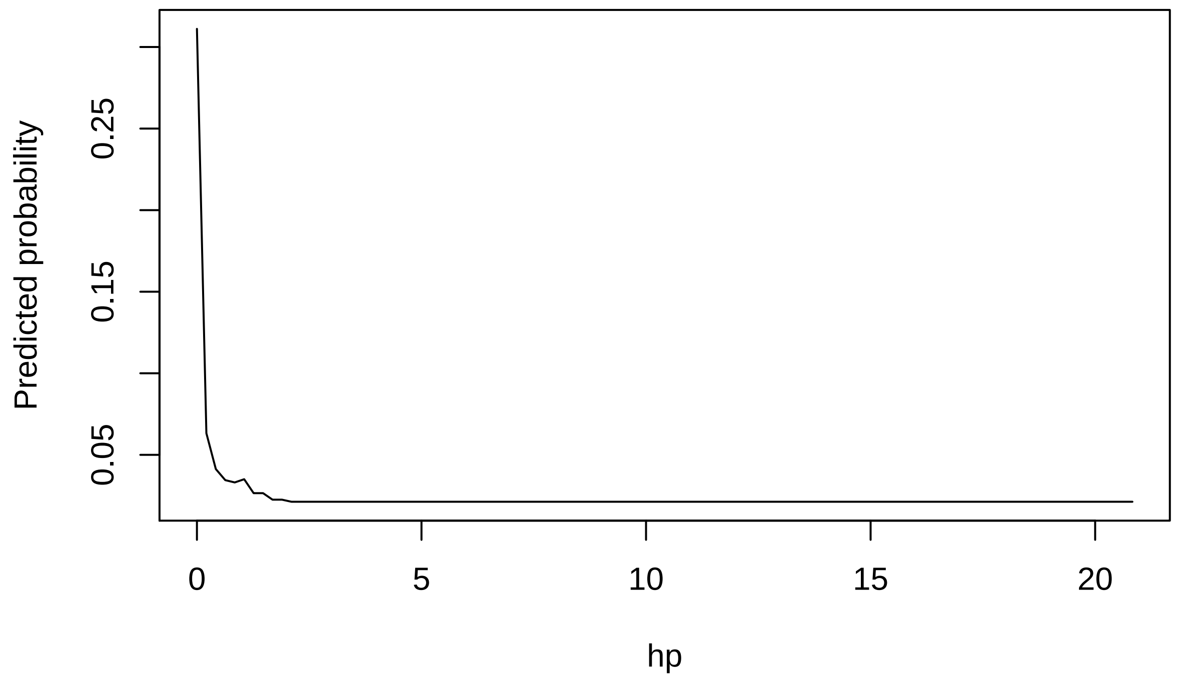
The other top influential variables have similar dependence plots: they influence the model's output to a particular class as its value is increased (taking into account the effects of the other variables).

Weighted model: The partial dependence plots of the weighted model are harder to interpret. We have used the weights to incentivize our model to classify emails as non-spam, even in the evidence of what humans might consider spammy. Let's look at the plot for `CAPAVE`, or `capital_run_length_average`. A partial dependence plot of our weighted Bernoulli model's link value (logit) shows a shape reminiscent of *"hp"*, However, look at the y-axis values. The difference of maximum and minimum values on this log odds plot doesn't mean much in terms of probability, which we can see in the following plot of `CAPAVE`'s response value.

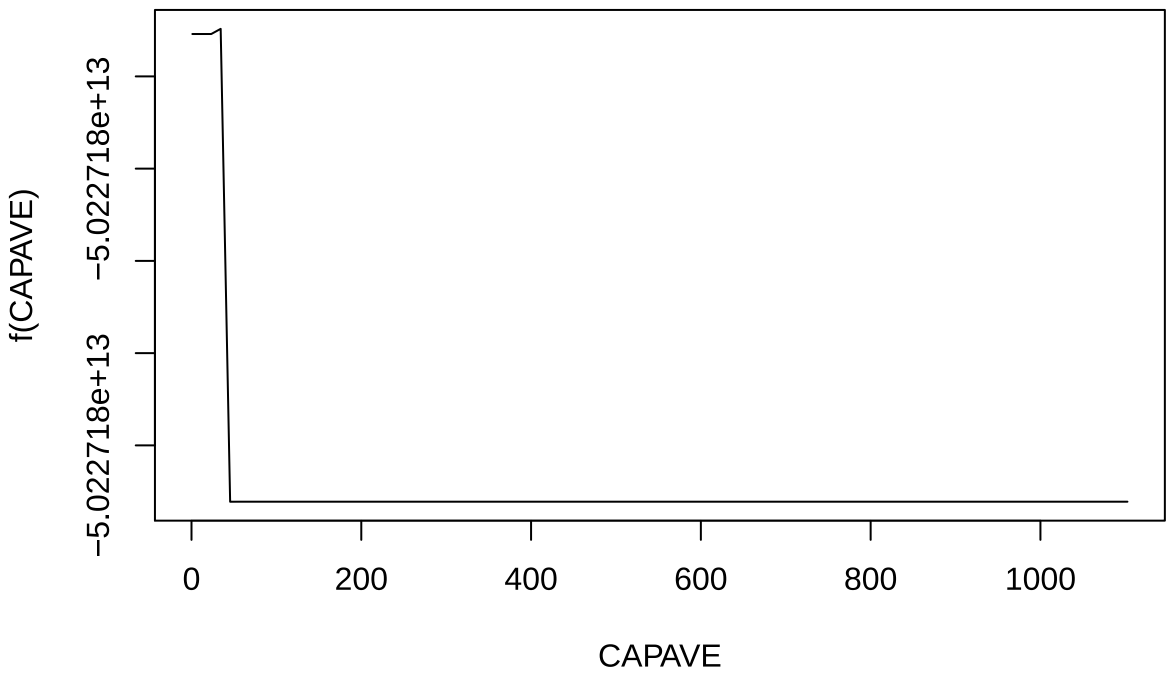
```r
plot(gbm0,match("!", rflabs),  nTreesForPrediction0, type="response",height=2)
```



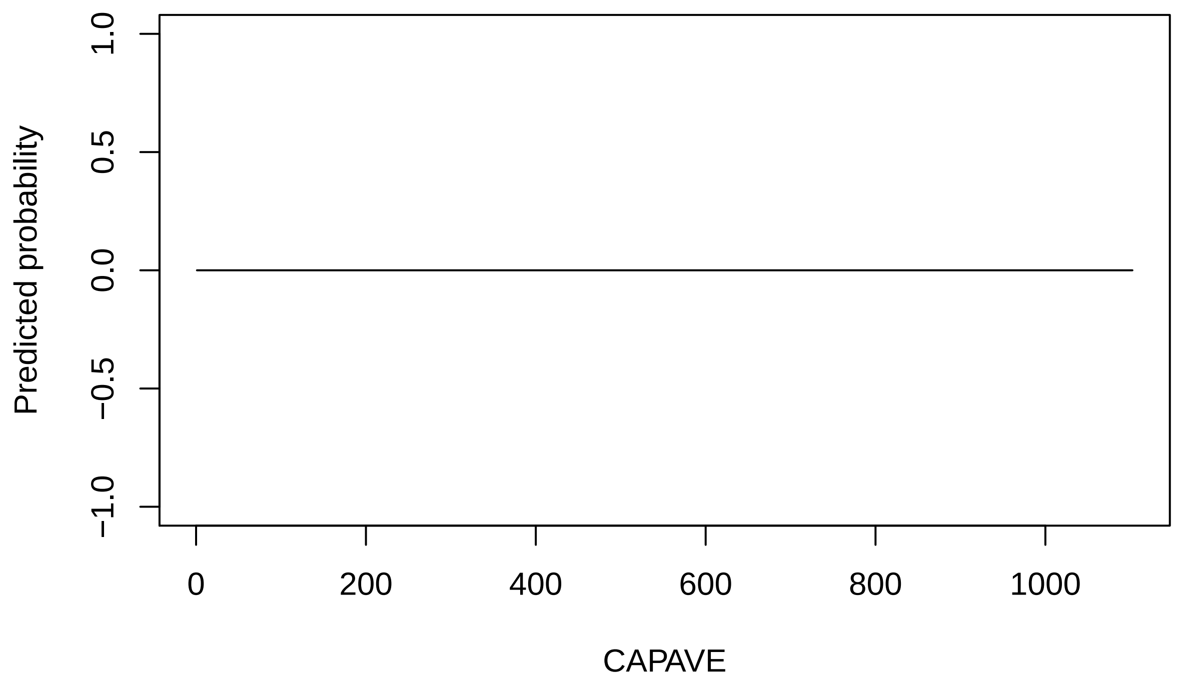
```r
plot(gbm0,match("hp", rflabs),  nTreesForPrediction0, type="response",height=3)
```



```
plot(gbm1,match("CAPAVE", rflabs), nTreesForPrediction1, type="link",height=4)
```



```
plot(gbm1,match("CAPAVE", rflabs), nTreesForPrediction1, type="response",height=4)
```



Question 7

Regression: California Housing. The data set calif_stats315B.csv consists of aggregated data from 20,640 California census blocks (from the 1990 census). The goal is to predict the median house value in each neighborhood from the others described in calif_stats315B.txt. Fit a gbm model to the data and write a short report that should include at least

```
#reset all variables except user-defined functions
rm(list = setdiff(ls(), lsf.str()))

#data labls
labels <-c(
  "house_value",
  "median_income",
  "housing_median_age",
  "average_no_rooms",
  "average_no_bedrooms",
  "population",
  "average_occupancy",
  "latitude",
  "longitude"
)

#load the data
```

```r
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)
data <- read.csv(file="calif_stats315B.csv", header=FALSE, sep=",")
colnames(data) <- labels

#fit model on training data
set.seed(444) # random for bag.fraction
model <- gbm(house_value~., data=data, train.fraction=1,
            interaction.depth=4, shrinkage=.05,
            n.trees=2500, bag.fraction=0.5, cv.folds=5,
            distribution="gaussian", verbose=F)
```

(a) The prediction accuracy of gbm on the data set.

The MSE of the response variable house_value is ~ 0.15. Since the response is in units of \$100,000, this means on average the squared error of the median house_value predicted by our model is $\sim 0.154 \times \$100,000$, or $\sim \$15,400$.

```r
#make predictions
nTreesForPrediction <- gbm.perf(model, plot.it=FALSE, method="cv")
y_hat <- predict(model, data, type="response", n.trees=nTreesForPrediction)

#Calculate MSE on of regression
y <- data$house_value
MSE <- sum((y_hat - y)^2)/length(y)
MSE
```

(b) Identification of the most important variables.

The most important variable for predicting house_value for a particular neighborhood in California is median_income, which intuitively makes sense.

The second-tier of variables in terms of importance had similar levels of importance: average_occupancy, latitude and longitude.

The 4 variables that all had the similar levels of insignificance in predicting house_value for a given neighborhood: housing_median_age, average_no_rooms, population, average_no_bedrooms.

```r
influential_vars <- summary.gbm(model,main="RELATIVE INFLUENCE",order=TRUE)
top4 <- gsub("`","",influential_vars$var[1:4])
cat("Top 4 influential variables: ",
    top4, "\n")
```

(c) Comments on the dependence of the response on the most important variables (you may want to consider partial dependence plots (plot) on single and pairs of variables, etc.).

This section refer to plots included following the discussion.

We can see from the univariate dependence plots that as `median_income` increases, our model's output for `home_value` also increases (accounting effects of other variables).

Additionally, the univariate dependence plots of `longitude` and `latitude` show that `home_value` tends to decrease as homes get further from the coast. It also shows that `home_value` tends to decrease as homes get further north (which makes sense since the Bay area is at the ∼half-way mark of California in terms of latitude). However, there is important nuance in the interaction between these two variables.

The bivariate partial dependence of the model on `latitude` and `longitude` shows us something the univariate plots alone do not. 2-dimensional partial dependence plots show the model's output as a heat map over a grid of 2 variables. Pink is a lower output value, blue is higher. The border between the pink and blue zones have interior corner points at the Bay Area ($latlng = (38, -122)$) and Los Angeles ($latlng = (34, -118)$). It is not surprising that these major metropolitan areas influence `home_value`.

```
nTreesForPrediction <- gbm.perf(model, plot.it=FALSE, method ='cv')
top4_indexes <- match(top4, labels) - 1 #-1 because first label is response var

#univariate partial dependence
for(i in top4_indexes){ plot(model,i,nTreesForPrediction) }

#longitude, latitude 2d partial dependence
plot(model,c(match("longitude",labels)-1,match("latitude",labels)-1),
     nTreesForPrediction)
```

Question 8

Regression: Marketing data. The data set age_stats315B.csv was already used in Homework 1. Review age_stats315B.txt for the information about order of attributes etc.

(a) Fit a gbm model for predicting age form the other demographic attributes and compare the accuracy with the accuracy of your best single tree from Homework 1.

Our GBM achieves a better accuracy in this regression problem, with an MSE of 0.5648001 compared to our best single tree which has an MSE of 0.6992926.

```
#reset all variables except user-defined functions
rm(list = setdiff(ls(), lsf.str()))

#Read and type data
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)
age_data <- read.csv('age_stats315B.csv')
factor_columns <- c(
  'Occup',
  'TypeHome',
  'sex',
  'MarStat',
  'DualInc',
  'HouseStat',
  'Ethnic',
  'Lang'
)
age_data[factor_columns] <- lapply(age_data[factor_columns], as.factor)

#fit a GBM
set.seed(444) # random for bag.fraction
model_gbm <- gbm(age~., data=age_data, train.fraction=1,
            interaction.depth=4, shrinkage=.05,
            n.trees=2500, bag.fraction=0.5, cv.folds=5,
            distribution="gaussian", verbose=F)
nTreesForPrediction <- gbm.perf(model_gbm, method='cv', plot.it=FALSE)

#fit a tree
model_tree <- rpart(age ~ ., data = age_data, method = "anova",
                    control=rpart.control(minbucket = 10,
                                          xval = 10,
                                          maxsurrogate = 5,
                                          usesurrogate = 2,
                                          cp=0.0001))
# Find the minimum cross-validation error + one SD
min_error_window <- min(model_tree$cptable[,"xerror"] + model_tree$cptable[,"xstd"])
# Find the simplest model with xerror within the min_error_window
best_cp <- first(model_tree$cptable[which(model_tree$cptable[,"xerror"] < min_error_window),"CP"])
best_single_tree <- prune(model_tree, cp = best_cp)

#Make predictions
y_hat_gbm <- predict(model_gbm, age_data, type="response", n.trees=nTreesForPrediction)
y_hat_tree <- predict(best_single_tree, age_data)

#Compare errors
y <- age_data$age
MSE_gbm <- sum((y_hat_gbm - y)^2)/length(y)
MSE_tree <- sum((y_hat_tree - y)^2)/length(y)

cat("MSE, gbm: ", MSE_gbm, "\n")
cat("MSE, tree: ", MSE_tree, "\n")
```

(b) Identify the most important variables.

The most important variables were: `MarStat, Occup, HouseStat, and Edu`

These variables are also identified as important by the best single tree from homework 1.

```
influential_vars <- summary.gbm(model_gbm,main="RELATIVE INFLUENCE",order=TRUE)
top4 <- gsub("`","",influential_vars$var[1:4])
cat("GBM top 4 influential variables: ",top4 , "\n")

important_vars_single_tree <- names(best_single_tree$variable.importance)
cat("Single tree top 6 important variables: ",important_vars_single_tree[1:6], "\n")
```

Question 9

Multiclass classification: marketing data. The data set occup_stats315B.csv comes from the same marketing database used in Homework 1. The description of the attributes can be found in occup_stats315B.txt. The goal in this problem is to fit a gbm model to predict the type of occupation from the 13 other demographic variables.

```
rm(list = setdiff(ls(), lsf.str()))

#Read and type data
```

```r
labels <-c(
  "Occup",
  "TypeHome",
  "sex",
  "MarStat",
  "age",
  "Edu",
  "Income",
  "LiveBA",
  "DualInc",
  "Persons",
  "Under18",
  "HouseStat",
  "Ethnic",
  "Lang"
)
factor_columns <- c(
  'TypeHome',
  'sex',
  'MarStat',
  'Occup',
  'LiveBA',
  'DualInc',
  'HouseStat',
  'Ethnic',
  'Lang'
)

occup_labels <- c(
  "Professional/Managerial",
  "Sales Worker",
  "Factory Worker/Laborer/Driver",
  "Clerical/Service Worker",
  "Homemaker",
  "Student, HS or College",
  "Military",
  "Retired",
  "Unemployed"
)

#load the data
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)
occup_data <- read.csv('occup_stats315B.csv', header=FALSE, sep=",")
colnames(occup_data) <- labels
occup_data[factor_columns] <- lapply(occup_data[factor_columns], as.factor)

#fit GBM model for occupation
set.seed(444) # random for bag.fraction
model <- gbm(Occup ~ ., data=occup_data, train.fraction=1,
            interaction.depth=4, shrinkage=.05,
            n.trees=2500, bag.fraction=0.5, cv.folds=5,
            distribution="multinomial", verbose=F)
```

(a) Report the test set misclassification error for gbm on the data set, and also the misclassification error for each class.

```r
nTreesForPrediction <- gbm.perf(model, method='cv')

#determine misclassification rate
y_hat_p <- predict(model, occup_data, type="response", n.trees=nTreesForPrediction)
y_hat <- apply(y_hat,1,which.max)
y <- occup_data$Occup

#overall
cat("Overall misclassification rate: ", percent(misclass_rate(y_hat, y)), "\n")

#by class
for(occupation in levels(occup_data$Occup)){
  occupation_i <- strtoi(occupation)
  cat(percent(misclass_rate(y_hat[y==occupation_i], y[y==occupation_i])), ":",
    occup_labels[occupation_i], "\n")
}
```

(b) Identify the most important variables.

The most important variables overall are `age`, texttt{income}, and texttt{education}.

Occupation is hard to predict with the data given, as can be seen by the overall high misclassification rate. However the three classes we are able to predict better than most: Professional/Managerial, Student, and Retired.

The three most important variables actually do a good job to distinguish these classes from the rest, that is income for professionals, age and education for students, and age for retirees.

```r
influential_vars <- summary.gbm(model,main="RELATIVE INFLUENCE",order=TRUE)
cat("Top 3 influential variables: ",gsub("`","",influential_vars$var[1:3]) , "\n")
```