STATS 315B: Data Mining, Spring 2016
Homework 2, Due 5/22/2016
Completed by: Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

Problem 1

The advantage of random variable selection strategy in random forests is that we reduce the possibility of generating highly correlated bagged trees. Highly correlated trees lead to low prediction accuracy. By only considering a subset of split variables, we force ourselves to split differently at each tree. This enhances prediction because if we split on all variables we tend to still split on the same best variables, so there is little randomness and averaging the prediction value won't give us much of a different answer from just making one tree. The random variable selection strategy works best if the there are very few interactions among the features. Another advantage of the method is the increased computational efficiency due to less feature exploration.

The major disadvantage of random variable selection strategy is not being able to capture the interaction effects among the features in most of the bagged trees. An interaction between two variables is said to have been captured if they are on the samebranch of the generated tree. These interactions are not necessarily captured due to random variable selection for the split. Also, one does not have the liberty to split on the best feature due to the previous constraint. This leads to increasein the bias of the trees generated on average and may lead to decrease in prediction accuracy when many trees with low predictive power are averaged.

An alternate startegy to random variable selection in the random forests is to randomly change the size of bootstrap sample and allow all variables to be split at each node. This strategy introduces variation in the trees due to sampling and ensures capturing interaction effects among the features.

STATS 315B: Data Mining, Spring 2016
Homework 2, Due 5/22/2016
Completed by: Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

Problem 2

Linear regression uses least squares as the error function. When the number of predictors is greater than the number of observations in the training data, we have multiple perfect fits for the model and therefore, infinite solutions for the coefficient estimates of the least squares. Thus, the model would have a huge variance.

Regularization is necessary to find the unique coefficient estimates when the number of observations is less than the number of predictors. If there is high collinearity/correlation among the predictors in the model, then a regularization such as ridge penalty reduces the variation in coefficient estimates of the variables. Another case where regularization helps is when there exist multiple noisy variables in the data. In such a case, regularization such as a lasso penalty ensures that the coefficients of noisy variables are zero.

Regularization is disadvantageous when the number of predictors is extremely small and predictors are believed to be independent. In this case, we may want to pursue feature augmentation by incorporating the higher orders and interaction among parameters. But the idea of regularization goes against feature augmentation. Another example where regularization fails is when we have a perfect line/plane that fits the model for linear regression. Generally, regularization fails when the bias introduced by it does not lead to drastic decrease in the variance of prediction.

During boosting, we generate a linear combination of all possible trees. In this situation sparsity is assumed because out of all possible trees there are only a few trees that capture the signal in the data.

Most of the trees generated in boosting do not influence the outcome variable. If we assume sparsity and it exists then our predictions are good; otherwise, our predictions are inaccurate. However, if we do not assume sparsity and our variable space is larger than our sample space we are stuck. We simply do not have enough data to estimate a large number of parameters (all possible trees in the boosting context). Hence, the assumption of sparsity is always reasonable in boosting.

STATS 315B: Data Mining, Spring 2016
Homework 2, Due 5/22/2016
Completed by: Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

Problem 3

For convex empirical risk and convex penalities, the solution of $\hat{a}(\lambda)$ will be unique since it is a sum of convex functions. We will now attempt to minimize the sum of the empircal risk and the penalty. First, we consider minimizing the empirical risk:

$$\text{Let } L(y_i, a_0 + \sum_{j=1}^{n} a_j x_{ij}) = L \text{ and } a_0 + \sum_{j=1}^{n} a_j x_{ij} = f(x,a) = f$$

$$\frac{\partial \hat{R}}{\partial a_k} = \frac{1}{N} \sum_{j=1}^{N} \frac{\partial L}{\partial a_k}$$

$$\text{By the chain rule: } \frac{\partial L}{\partial a_k} = \frac{\partial L(y_i, f(x,a))}{\partial a_k} = \frac{\partial L(y_i, f(x,a))}{f(x,a)} \frac{\partial f(x,a)}{\partial a_k} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a_k}$$

$$\text{Now consider } \frac{\partial f}{\partial a_k} = \frac{\partial}{\partial a_k}[a_0 + \sum_{j=1}^{n} a_j x_{ij}]$$

Note that a_k only appears in the sumation and is multiplied by x_{ik}

So all values in f will evaluate to zero except $a_k x_{ik}$

$$\text{Then } \frac{\partial}{\partial a_k}[a_0 + \sum_{j=1}^{n} a_j x_{ij}] = x_{ik}$$

$$\text{And } \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L}{\partial a_k} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L}{\partial f} \frac{\partial f}{\partial a_k} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L}{\partial f} x_{ik}$$

Now we consider the penalty function for $\gamma \geq 1$:

$$\frac{\partial P_\gamma(a)}{\partial a_k} = \frac{\partial}{\partial a_k} \sum_{j=1}^{n} |a_j|^\gamma$$

Note, the only value in this sum that depends on a_k is $|a_k|^\gamma$

$$\text{So, } \frac{\partial}{\partial a_k} \sum_{j=1}^{n} |a_j|^\gamma = \frac{\partial}{\partial a_k} |a_k|^\gamma = \gamma |a_k|^{\gamma - 1} \frac{\partial |a_k|}{\partial a_k}, \text{ where } \frac{\partial |a_k|}{\partial a_k} \text{ is the subgradient and } \frac{\partial |a_k|}{\partial a_k} \in [-1, 1]$$

Note that the above holds for $\gamma = 1$. Finally, together we have:

$$\frac{\partial}{\partial a_k}[\hat{R}(a) + \lambda P_\gamma(a)] = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \gamma\lambda|a_k|^{\gamma-1}\frac{\partial|a_k|}{a_k}$$

To find the minimum, we would need to set the above equal to zero and find the values of a that satisfy the equality. Now, let's assume that $\gamma > 1$. We will show by contradiction that values of a are not forced to zero. Assume that some $a_k = 0$, then:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \gamma|a_k|^{\gamma-1}\frac{\partial|a_k|}{a_k} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \gamma\lambda \times 0^{\gamma-1} \times \frac{\partial|a_k|}{a_k} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik}$$

$$\text{Then } \frac{\partial}{\partial a_k}[\hat{R}(a) + \lambda P_\gamma(a)] = 0 \implies \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} = 0$$

Note that the above is just the derivative of the emperical risk set equal to zero. Its solution then will just be the standard unpenalized solution and will not necessarily equal zero. This is a contradiction to our assumption that $a_k = 0$, thus $a_k \neq 0$ and all coefficients along the path indexed by λ are not necessarily equal to zero.

We will now show that getting zero coefficients is possible for the lasso penalty. Assume $\gamma = 1$.

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \gamma\lambda|a_k|^{\gamma-1}\frac{\partial|a_k|}{a_k} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \lambda\frac{\partial|a_k|}{a_k}$$

$$\text{For } a_k \text{ near zero } -1 \leq \frac{\partial|a_k|}{a_k} \leq 1 \implies -\lambda \leq -\lambda\frac{\partial|a_k|}{a_k} \leq \lambda$$

$$\text{So } \frac{\partial}{\partial a_k}[\hat{R}(a) + \lambda P_\gamma(a)] = 0 \implies -\lambda\frac{\partial|a_k|}{a_k} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} \implies -\lambda \leq \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} \leq \lambda$$

So, for a_k near zero, we can bound the solution to the derivative of the unpenalized loss between plus and minus λ. This means that the solution will be equal to zero between those values of λ, so we have shown that a_k can be zero with the lasso penalty.

We will now show that the elastic net has the ability to set some coefficients to zero.

$$\frac{\partial P_\gamma(a)}{\partial a_k} = \frac{\partial}{\partial a_k}[\sum_{j=1}^{n}(\gamma - 1)\frac{a_j^2}{2} + (2-\gamma)|a_j|]$$

Again, the only values that will not evaluate to zero are the a_k values. So the above evaluates to:

$$(\gamma - 1)a_k + (2-\gamma)\frac{\partial|a_k|}{a_k}$$

We have shown what will happen for $\gamma = 1$ (Lasso penalty) and for $\gamma = 2$ (Ridge penalty). We only need to consider $1 < \gamma < 2$. With these constraints, we are interested in solving:

$$\frac{\partial}{\partial a_k}[\hat{R}(a) + \lambda P_\gamma(a)] = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \lambda(\gamma-1)a_k + \lambda(2-\gamma)\frac{\partial|a_k|}{a_k}$$

$$\text{For } a_k \text{ near zero } -1 \leq \frac{\partial|a_k|}{a_k} \leq 1 \implies -\lambda(2-\gamma) \leq -\lambda(2-\gamma)\frac{\partial|a_k|}{a_k} \leq \lambda(2-\gamma)$$

$$\text{Then } \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \lambda(\gamma-1)a_k + \lambda(2-\gamma)\frac{\partial|a_k|}{a_k} = 0 \implies$$

$$\implies \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \lambda(\gamma-1)a_k = -\lambda(2-\gamma)\frac{\partial|a_k|}{a_k} \implies$$

$$\implies -\lambda(2-\gamma) \leq \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial f}x_{ik} + \lambda(\gamma-1)a_k \leq \lambda(2-\gamma)$$

The bounded value is a ridge-like solution (with $\lambda' = (\gamma-1)\lambda$). a_k solutions to this near zero can be bounded by $\lambda(2-\gamma)$, which implies that some coefficients can be forced to zero (by the same logic as why the lasso forces some coefficients to zero).

STATS 315B: Data Mining, Spring 2016
Homework 2, Due 5/22/2016
Completed by: Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

Problem 4

Assume $E[x_j] = 0$ and $E[x_j^2] = 1$ for all x.

We want to show that the variable, x_j^* that has the maximum absolute correlation with y

$$j^* = \underset{1 \le j \le J}{\text{argmax}}\, |E(y \cdot x)| \tag{1}$$

is the same as the one that best predicts y using squared error loss

$$j^* = \underset{1 \le j \le J}{\text{argmin}}\, \min_{\rho} E[y - \rho x_j]^2 \tag{2}$$

We begin by expanding equation 2.

$$j^* = \underset{1 \le j \le J}{\text{argmin}}\, \min_{\rho} E[(y - \rho x_j)^T (y - \rho x_j)]$$

$$j^* = \underset{1 \le j \le J}{\text{argmin}}\, \min_{\rho} E[y^T y + \rho^2 x_j^T x_j - \rho x_j^T y - \rho y^T x_j]$$

Distributing the expectation as well as using our second assumption leaves us with

$$j^* = \underset{1 \le j \le J}{\text{argmin}}\, \min_{\rho} [E(y^2) + \rho^2 - 2\rho E(y \cdot x_j)]$$

Since y and ρ are real $\Rightarrow E(y^2) \ge 0$ and $\rho^2 \ge 0$
define $A = E(y^2) + \rho^2 \ge 0$

$$j^* = \underset{1 \le j \le J}{\text{argmin}}\, \min_{\rho} [A - 2\rho E(y \cdot x_j)]$$

if $E(y \cdot x_j) < 0$ want to minimize $\rho |E(y \cdot x_j)|$
if $E(y \cdot x_j) > 0$ want to maximize $\rho |E(y \cdot x_j)|$
First we want to minimize with respect to ρ to get ρ^*

$$\frac{\partial}{\partial \rho}\left[E(y^2) + \rho^2 - 2\rho E(y \cdot x_j)\right] = 2\rho - 2E(y \cdot x_j) \Rightarrow \rho^* = E(y \cdot x_j) \Rightarrow sign(\rho^*) = sign(E(y \cdot x_j))$$

if $E(y \cdot x_j) < 0$ want to minimize $-|\rho^*||E(y \cdot x_j)|$
if $E(y \cdot x_j) > 0$ want to maximize $|\rho^*||E(y \cdot x_j)|$
Therefore in either case we want to maximize $|E(y \cdot x_j)|$

$$\Rightarrow j^* = \underset{1 \le j \le J}{\text{argmax}}\, |E(y \cdot x_j)|$$

STATS 315B: Data Mining, Spring 2016
Homework 2, Due 5/22/2016
Completed by: Henry Neeb, Christopher Kurrus, Yash Vyas and Tyler Chase

Problem 5

Taking the expectation of our additive function, we have:

$$E_{z_{\setminus l}}[F(z_l, z_{\setminus l})] = E_{z_{\setminus l}}[F(z_l) + F(z_{\setminus l})] = E_{z_{\setminus l}}[F(z_l)] + E_{z_{\setminus l}}[F(z_{\setminus l})]$$
$$\text{Note that } E_{z_{\setminus l}}[F(z_{\setminus l})]\text{evaluates to a constant}$$
$$\text{Also, } E_{z_{\setminus l}}[F(z_l)] = F(z_l)$$
$$\text{because we are taking an expectation with respect to a variable not present in the function}$$
$$\text{So we have } E_{z_{\setminus l}}[F(z_l)] + E_{z_{\setminus l}}[F(z_{\setminus l})] = F(z_l) + constant$$

So we have shown that this expectation evaluates to a function of $F(z_l)$ up to an additive constant. Now we will show what $E[F(x)|z_l]$ evaluates to:

$$E_{x|z_l}[F(x)|z_l] = E_{x|z_l}[[F(z_l) + F(z_{\setminus l})]|z_l] = E_{x|z_l}[F(z_l)|z_l] + E_{x|z_l}[F(z_{\setminus l})|z_l]$$
$$\text{Note } E_{x|z_l}[F(z_l)|z_l] = F(z_l) \text{ and } E_{x|z_l}[F(z_{\setminus l})|z_l] \text{ is a function of } z_{\setminus l}$$
$$\text{If } z_l \text{ and } z_{\setminus l} \text{ are independent, then } E_{x|z_l}[F(z_{\setminus l})|z_l] = E_{x|z_l}[F(z_{\setminus l})]$$
$$\text{Then in general, } E_{x|z_l}[F(x)|z_l] = F(z_l) + E_{x|z_l}[F(z_{\setminus l})|z_l] = F(z_l) + E_{z_{\setminus l}}[F(z_{\setminus l})]$$
$$\text{However, if } z_l \text{ and } z_{\setminus l} \text{ are independent:}$$
$$E_{x|z_l}[F(x)|z_l] = F(z_l) + constant$$

Thus the dependence of $F(x)$ on z_l ignoring the other variables ($z_{\setminus l}$) will be the same as the partial dependence of $F(x)$ on z_l when the two are independent.

Problem 6 - Binary Classification: Spam Email

Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

May 22, 2016

Part A:

After fitting a gbm model on the training data and checking its misclassification on the testing dataset, we can see our overall estimate of misclassification rate is 0.0397653. Our misclassification percentage for spam in the testing set is 0.566343% and for not spam is 0.2838428%

Part B:

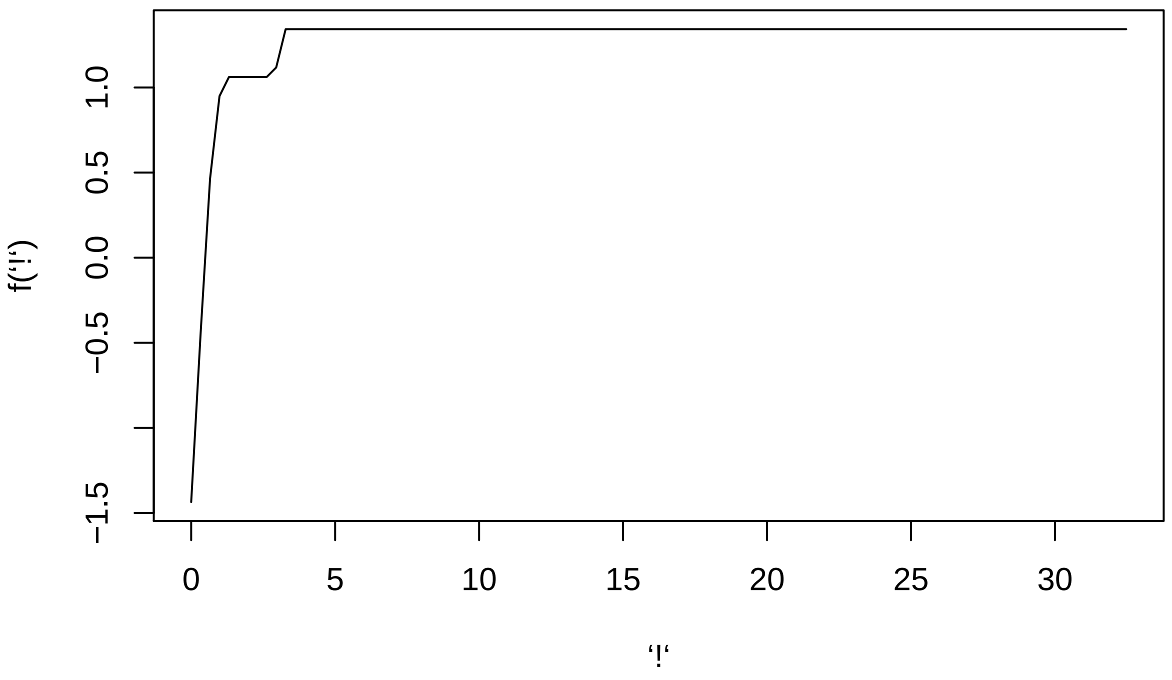
After editing our weights to obtain our desired misclassification rates, we see that with a very high relative cost we can reach the desired non-spam misclassification rate, but our overall misclassification rate has risen to 0.1577575. Our misclassification percentage for spam in the testing set is now 3.8834951% and for not spam is 0.0218341%.

```
##                   var      rel.inf
## `!`               `!` 1.975046e+01
## `$`               `$` 1.509120e+01
## remove         remove 1.461162e+01
## hp                 hp 8.697849e+00
## free             free 6.803539e+00
## CAPAVE         CAPAVE 5.747085e+00
## your             your 3.787023e+00
## CAPMAX         CAPMAX 3.586949e+00
## CAPTOT         CAPTOT 2.527999e+00
## money           money 2.391571e+00
## our               our 2.189312e+00
## edu               edu 1.807139e+00
## you               you 1.424414e+00
## george         george 1.414462e+00
## `000`           `000` 1.271262e+00
## will             will 8.492138e-01
## email           email 7.809329e-01
## `(`               `(` 6.624304e-01
## business     business 6.605465e-01
## re                 re 6.287674e-01
## receive       receive 6.231240e-01
## `;`               `;` 5.230270e-01
## `1999`         `1999` 5.057743e-01
## internet     internet 4.332845e-01
## `650`           `650` 4.316534e-01
## mail             mail 3.874840e-01
## meeting       meeting 3.555494e-01
## over             over 2.506832e-01
## technology technology 2.457473e-01
## all               all 2.404583e-01
## hpl               hpl 2.189405e-01
## report         report 1.274217e-01
## `[`                 `[` 1.096789e-01
## `3d`               `3d` 9.976536e-02
## pm                   pm 9.240302e-02
## data               data 7.196081e-02
## original       original 7.048919e-02
## labs               labs 6.427108e-02
## `#`                 `#` 5.731940e-02
## order             order 5.364514e-02
## `85`               `85` 5.207701e-02
## conference conference 4.942412e-02
## make               make 4.742877e-02
## font               font 4.637198e-02
## people           people 4.636174e-02
## project         project 4.585091e-02
## address         address 3.881539e-02
## lab                 lab 1.598947e-02
## addresses     addresses 4.483105e-03
## credit           credit 3.834119e-03
## direct           direct 1.209557e-03
## parts             parts 1.175796e-03
## cs                   cs 3.178756e-04
## telnet           telnet 2.125285e-04
## `857`             `857` 0.000000e+00
## `415`             `415` 0.000000e+00
## table             table 0.000000e+00
```

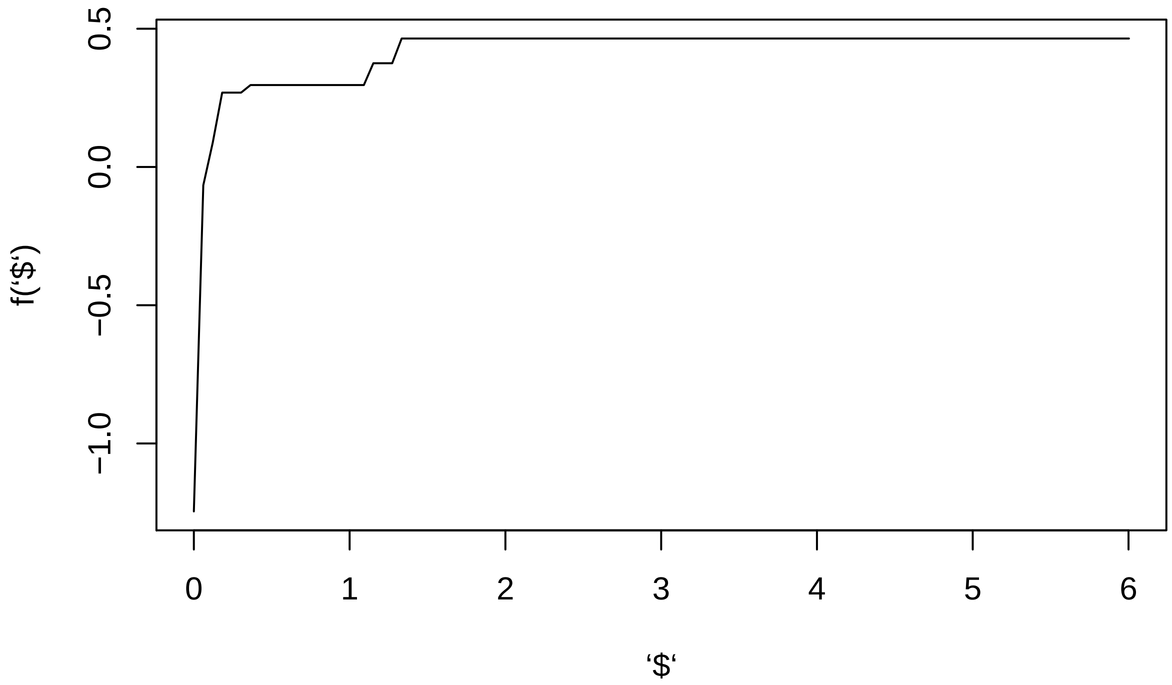
Using the recommended relative.influence method for determining importance, here we can see that the most important predictors are '!', '$' and 'remove', at 19.75, 15.09, and 14.61 rel.inf respectively. After this the influence drops nearly 41% to 8.69 for 'hp'.

Now after setting our number of trees to the best iteration, chosen by the Out-Of-Bag error, we can proceed to create our dependence plots.

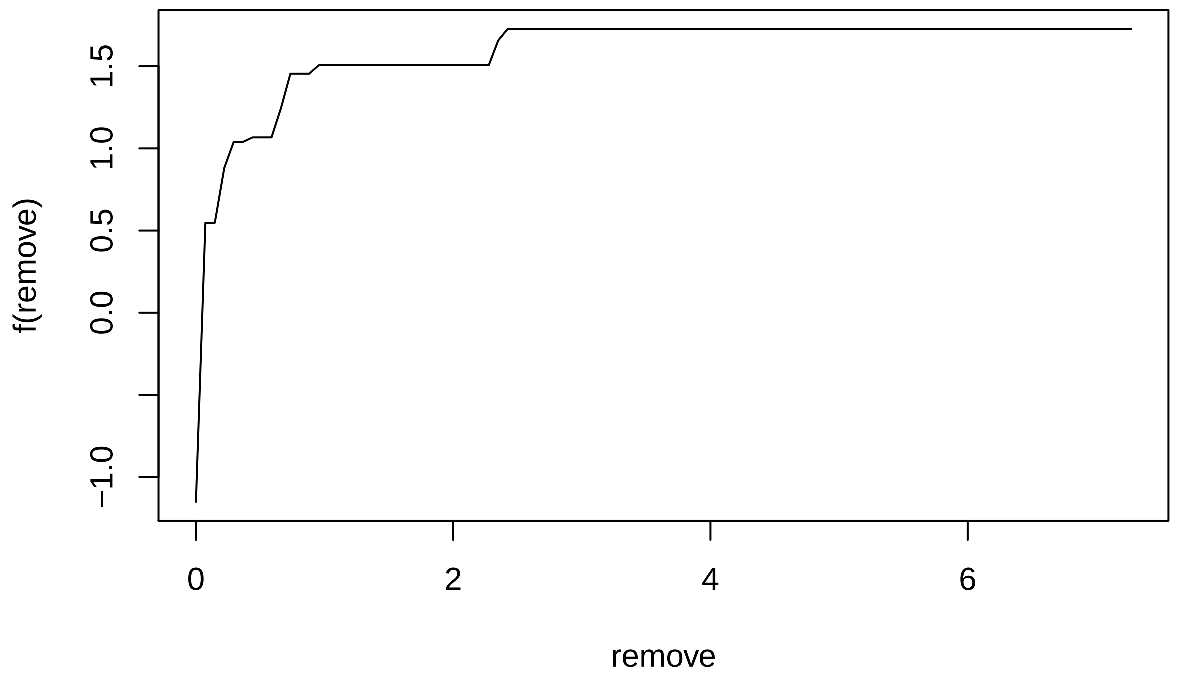
Looking at the plot of the most important variable, '!', we can see that the effect is almost binary. If an email contains no '!', that strongly suggests that it is not spam, but if it contains even one '!', it suggests that it is spam. Interestingly, the positive (is spam) effect levels off slightly at f('!') = 1, before rising to the limit of 1.5.



The plot for '\$' is very similar to the plot for '!', but with a smoother approach to the maximum value for f('\$'). Once again, a value even slightly >0 for '\$' suggests that the email is spam, though the prediction effect here is significantly weaker than it was for 'result' or '!'.



The plot for 'remove' suggests that even a slightly positive value of 'remove' strongly predicts that the email is spam. In addition, the prediction effect for remove appears to be a good deal stronger than the other two predictors, i.e. a positive value for remove suggests that the email is spam more so than '!' or '\$'.



Code Appendix:

```
library(ggplot2)
library(dplyr)
library(readr)
library(gbm)

set.seed(123)

data_path <- "data/"
file_tst <- "Spam.Test.txt"
file_trn <- "Spam_Train.txt"
file_tst <- paste(data_path, file_tst, sep = "")
file_trn <- paste(data_path, file_trn, sep = "")

tst <- read_csv(file = file_tst, col_names = FALSE)
trn <- read_csv(file = file_trn, col_names = FALSE)

cnms <- c("make", "address", "all", "3d", "our", "over", "remove", "internet","order", "mail", "receive

colnames(tst) <- cnms
colnames(trn) <- cnms

gbm0 <- gbm(type ~ . , data = trn, train.fraction = 1, interaction.depth=4, shrinkage=.05,  n.trees=250

gbm0.pred <- predict(gbm0, tst, type="response", n.trees=300)
df <- cbind(as.data.frame(gbm0.pred), tst$type)
df <- df %>% mutate(pred = ifelse(gbm0.pred >= .5, 1, 0), equ = pred == tst$type)
gbm0.tst.mc <- (1534 - sum(df$equ))/1534

df_spam <- df %>% filter(tst$type == 1)
gbm0.spam.mc <- (618 - sum(df_spam$equ))/618
```

```
df_not <- df %>% filter(tst$type == 0)
gbm0.not.mc <- (916 - sum(df_not$equ))/916

rm(df, df_spam, df_not)

wvec <- trn %>% mutate(wght = ifelse(type == 0, .99, .01)) %>% .$wght

gbm1 <- gbm(type ~ . , data = trn, train.fraction = 1, interaction.depth=4, shrinkage=.03,  n.trees=250

gbm1.pred <- predict(gbm1, tst, type="response", n.trees=300)

df <- cbind(as.data.frame(gbm1.pred), tst$type)
df <- df %>% mutate(pred = ifelse(gbm1.pred >= .5, 1, 0), equ = pred == tst$type)
gbm1.tst.mc <- (1534 - sum(df$equ))/1534

df_spam <- df %>% filter(tst$type == 1)
gbm1.spam.mc <- (618 - sum(df_spam$equ))/618

df_not <- df %>% filter(tst$type == 0)
gbm1.not.mc <- (916 - sum(df_not$equ))/916
gbm1.not.mc

summary(gbm0,main="Relative Influence for all Predictors")
best.iter<-gbm.perf(gbm0,method="OOB")

plot(x=gbm0,i.var=52,n.trees=best.iter, main="Partial Dependence of !")
plot(x=gbm0,i.var=53,n.trees=best.iter, main="Partial Dependence of $")
plot(x=gbm0,i.var=7,n.trees=best.iter, main="Partial Dependence of remove")
```

Problem 7 - Regression: California Housing

Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

May 22, 2016

Libraries

```r
library(ggplot2)
library(dplyr)
library(readr)
library(gbm)
```

File Parameters

```r
# Session -> set working directory -> Source file location

set.seed(131)

# Path for data
data_path <- "data/"
data_name <- "California_Data.txt"
data_name <- paste(data_path, data_name, sep = "")
```

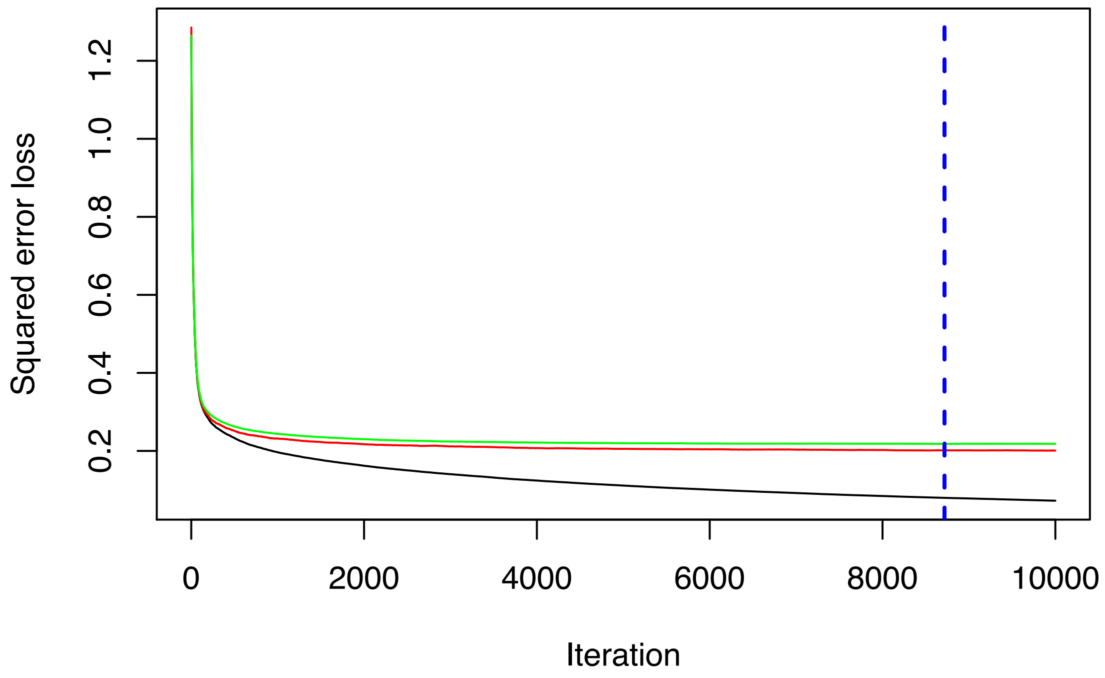
Read in Data

```r
data_0 <- read_csv(file = data_name, col_names = FALSE)


# Name data columns
column_names <- c("House_Value", "Median_Income", "Housing_Median_Age",
                  "Average_Rooms", "Average_Bedrooms", "Population",
                  "Average_Occupancy", "Lattitude", "Longitude")
colnames(data_0) <- column_names

# Make sure the samples are mixed
set.seed(131)
data <- data_0[sample(nrow(data_0)),]
```

Part (a)

We fit our data using a gbm model and determine the prediction accuracy.

```r
# Fit model on our data
set.seed(444) # Random for bag.fraction
gbm_model <- gbm(House_Value ~ ., data = data , train.fraction = 0.8, interaction.depth = 4,
                 shrinkage = 0.05, n.trees = 10000, bag.fraction = 0.5, cv.folds = 5,
                 distribution = "gaussian", verbose = F)

# Determine the optimal number of trees by cross validation
best.iter <- gbm.perf(gbm_model, method = "cv")
```



Determine the training error and the test error.

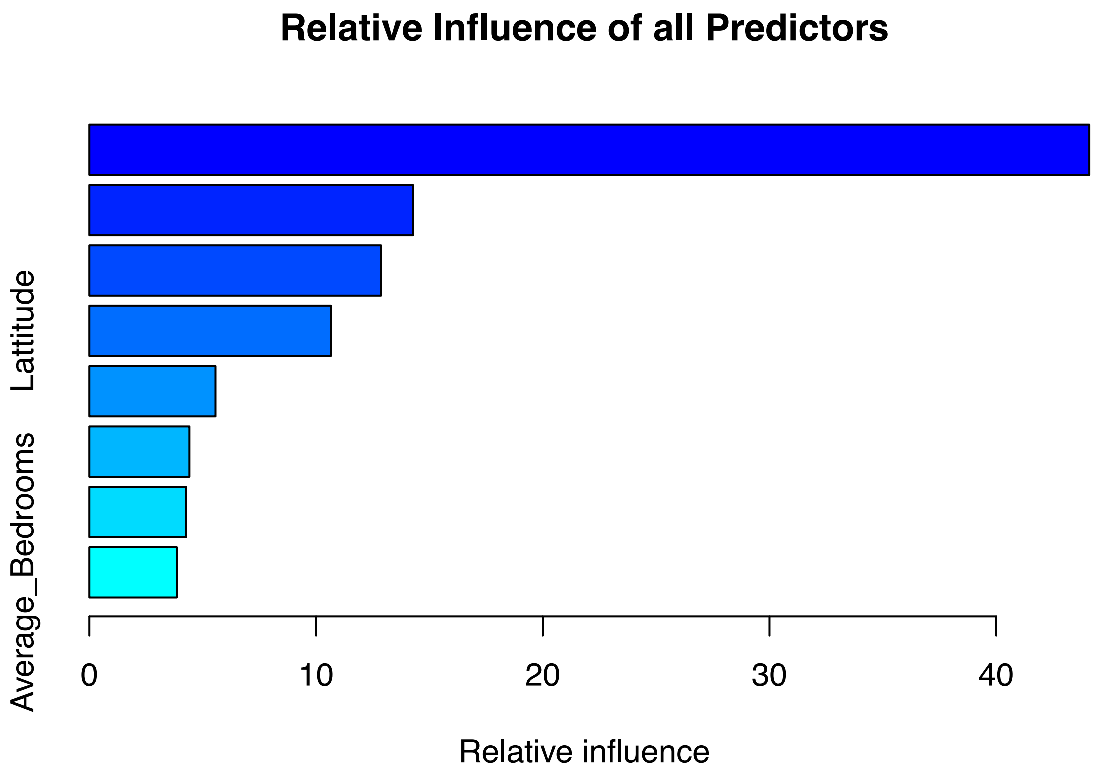
```
# Determine the training error and the test error
train_error = gbm_model$train.error[best.iter]
test_error = gbm_model$valid.error[best.iter]
```

We quantify the accuracy of the gbm on the data by reporting the mean square error for both the training set and the test set. The training error is 0.0797317, while the test error is 0.2015788.

Part (b)

Here we identify the most important variables.

```
summary(gbm_model, main = "Relative Influence of all Predictors")
```



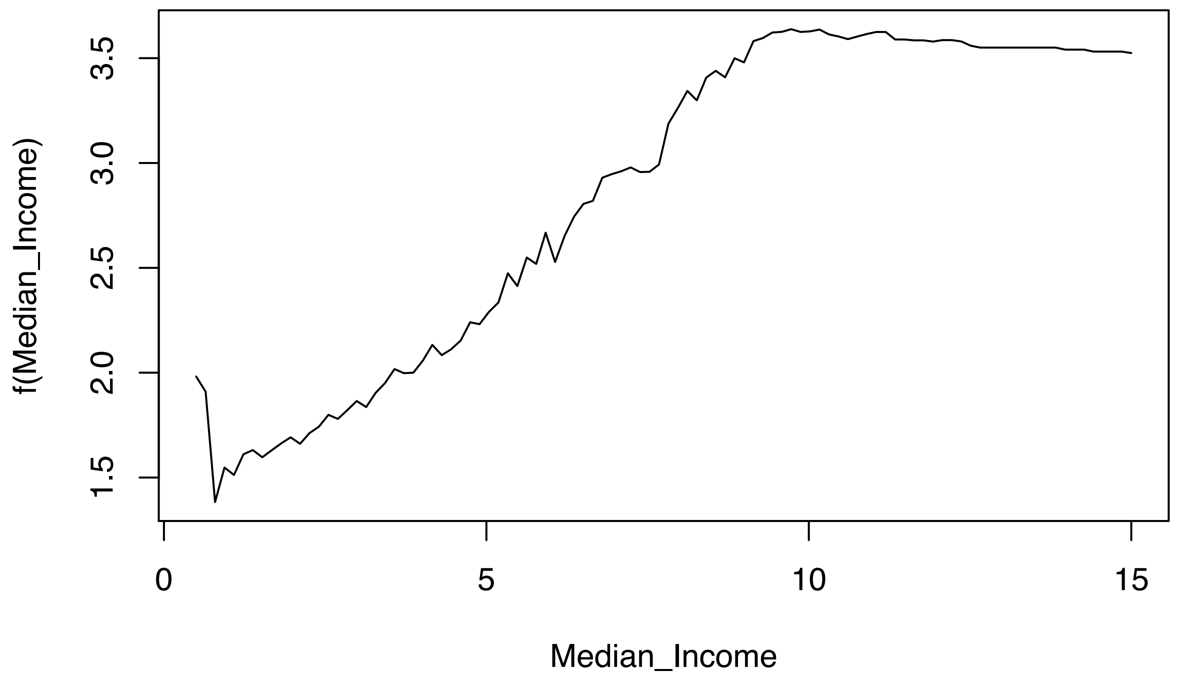
```
##                                    var   rel.inf
## Median_Income           Median_Income 44.107907
## Average_Occupancy   Average_Occupancy 14.270683
## Longitude                   Longitude 12.865632
## Lattitude                   Lattitude 10.652504
## Average_Rooms           Average_Rooms  5.563540
## Housing_Median_Age Housing_Median_Age  4.413439
## Population                 Population  4.271677
## Average_Bedrooms     Average_Bedrooms  3.854619
```

I will list the 4 variables that are over 10% in relative influence. Median income is the most influential variable with a relative influence of 44.1. Average Occupancy is the most second most influencial is the average occupancy which has a relative influence of 14.3. The third most influencial is the longitude with a relative influence of 12.9, followed by lattitude with a relative influence of 10.7.

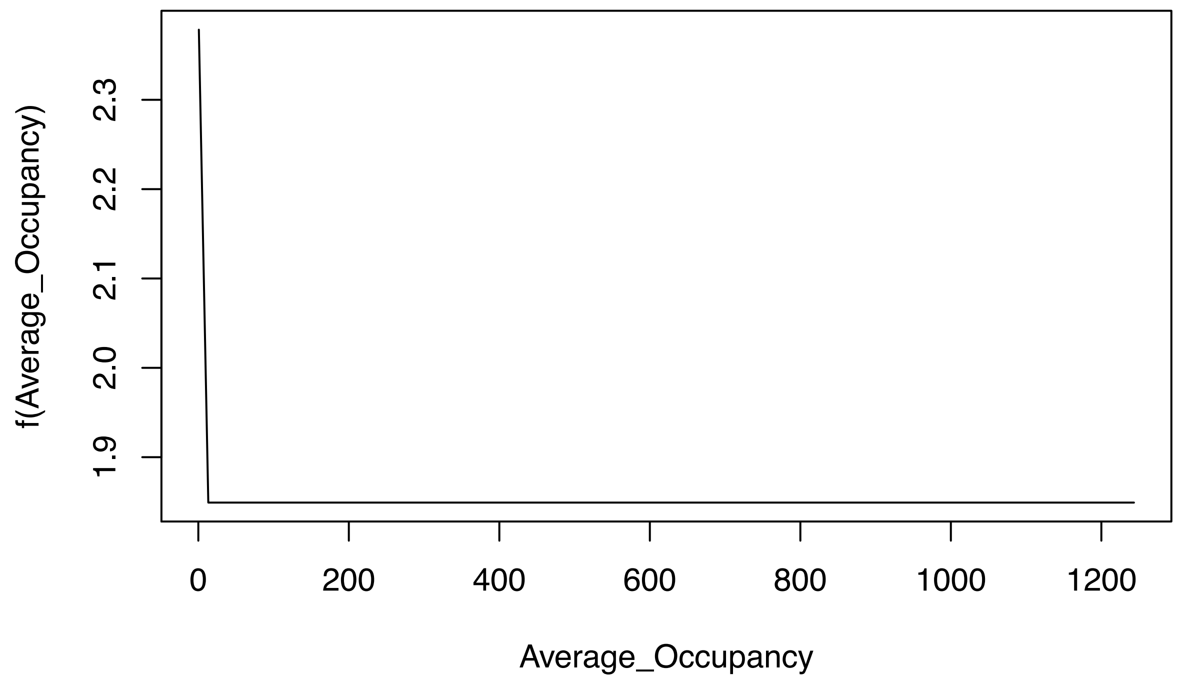
Part (c)

Here we plot the dependence of the response on the most important variables, and on the most important variable pairs.

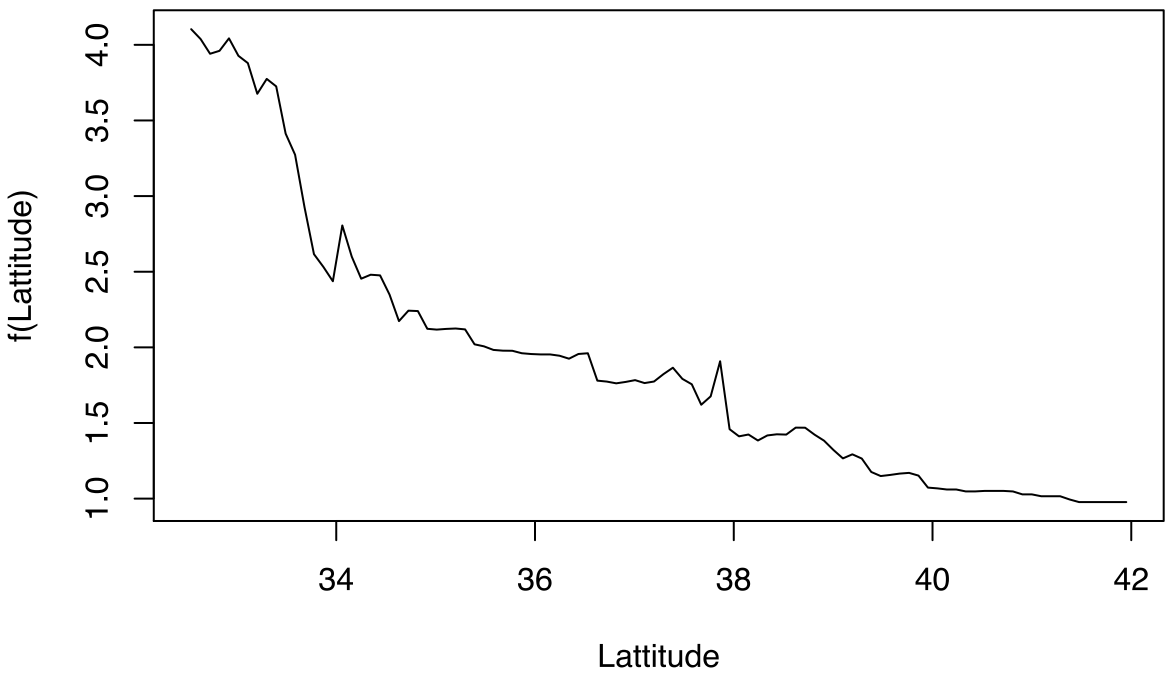
```
plot(x=gbm_model, i.var=1, n.trees=best.iter)
```



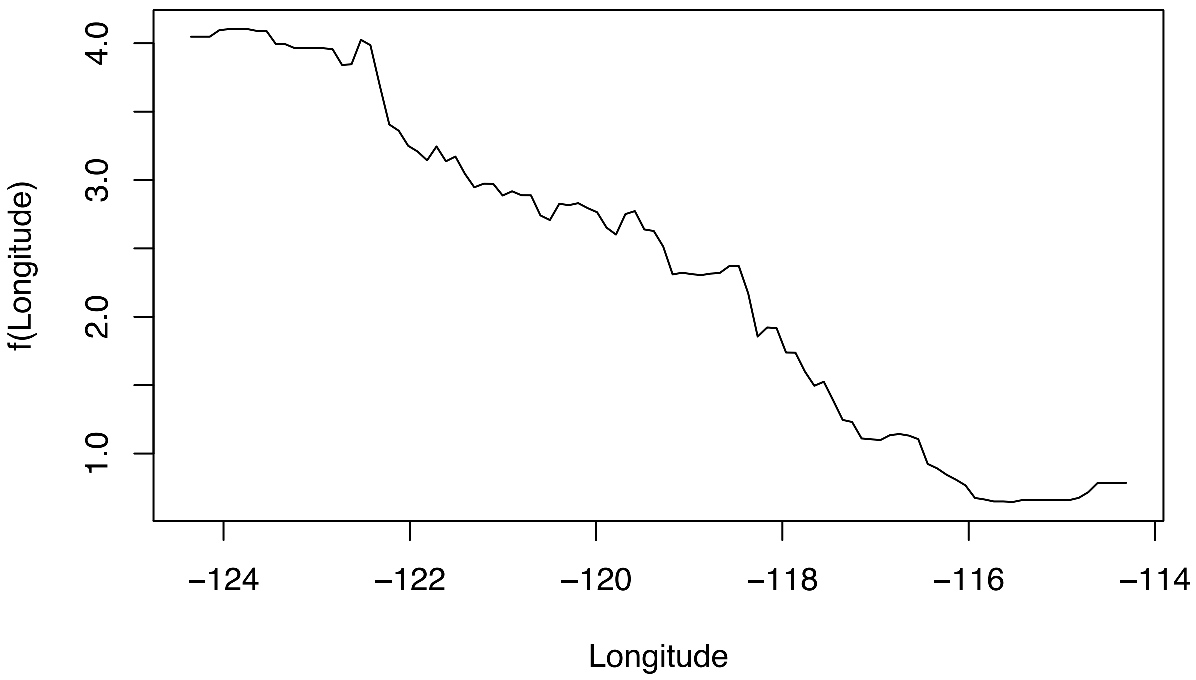
```r
plot(x=gbm_model, i.var=6, n.trees=best.iter)
```



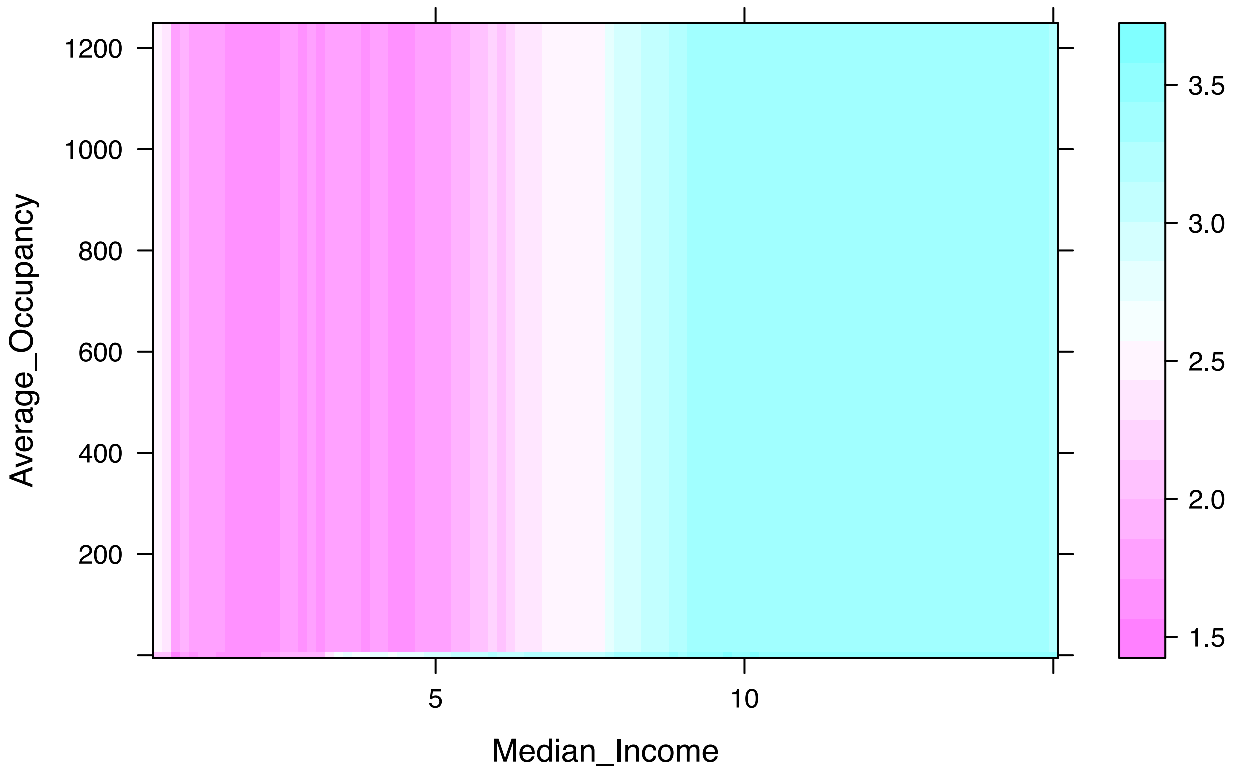
```r
plot(x=gbm_model, i.var=7, n.trees=best.iter)
```



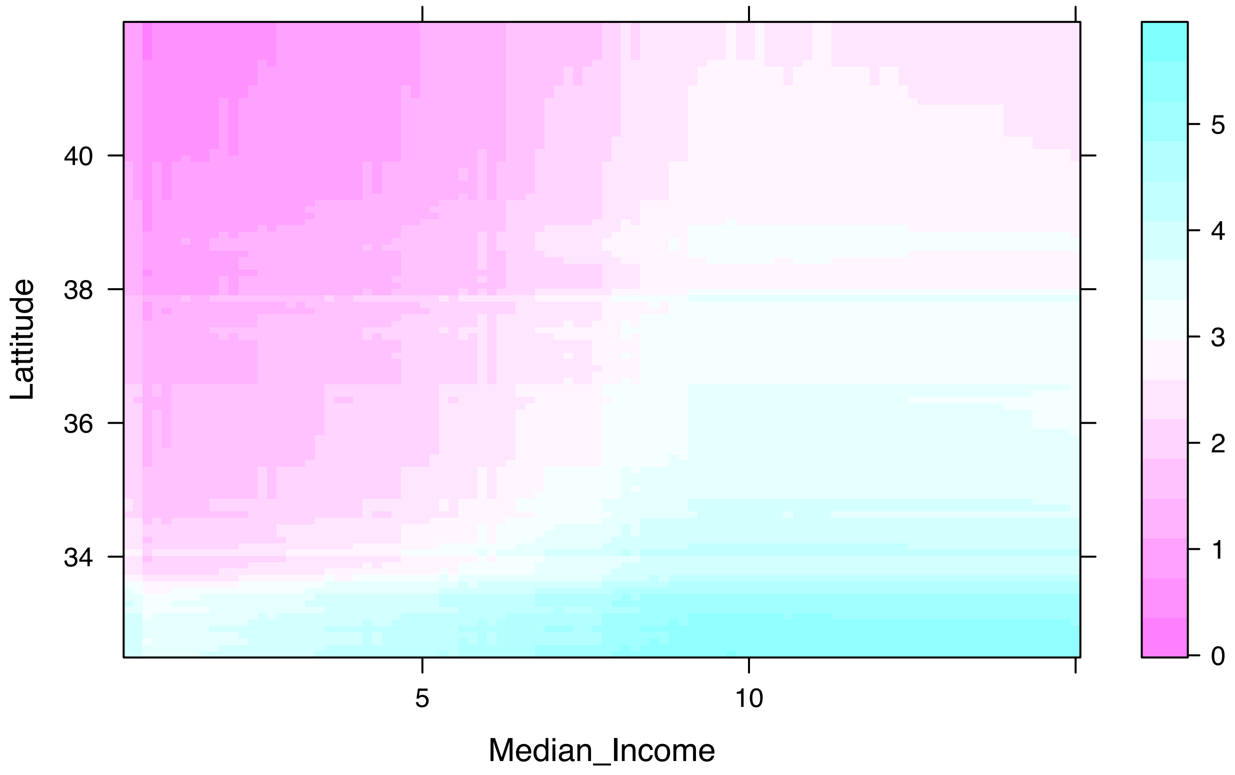
```r
plot(x=gbm_model, i.var=8, n.trees=best.iter)
```



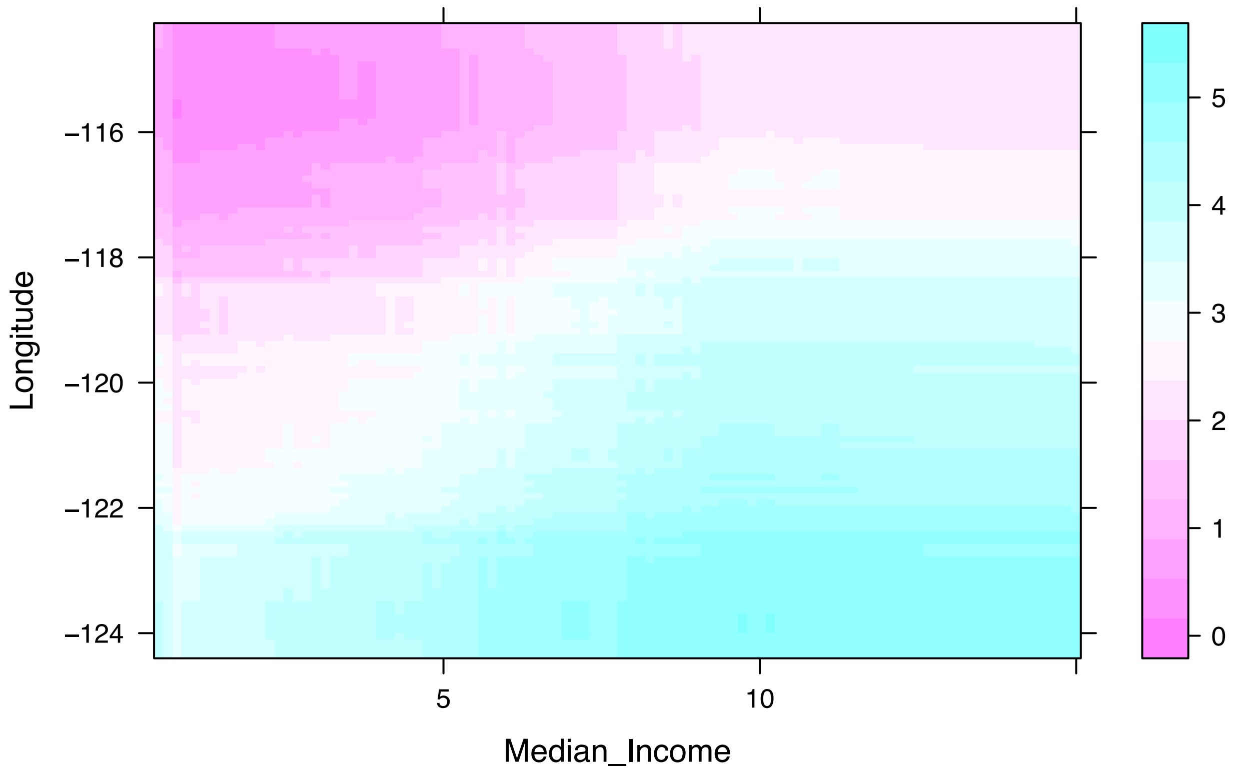
```r
plot(x=gbm_model, i.var=c(1,6), n.trees=best.iter)
```



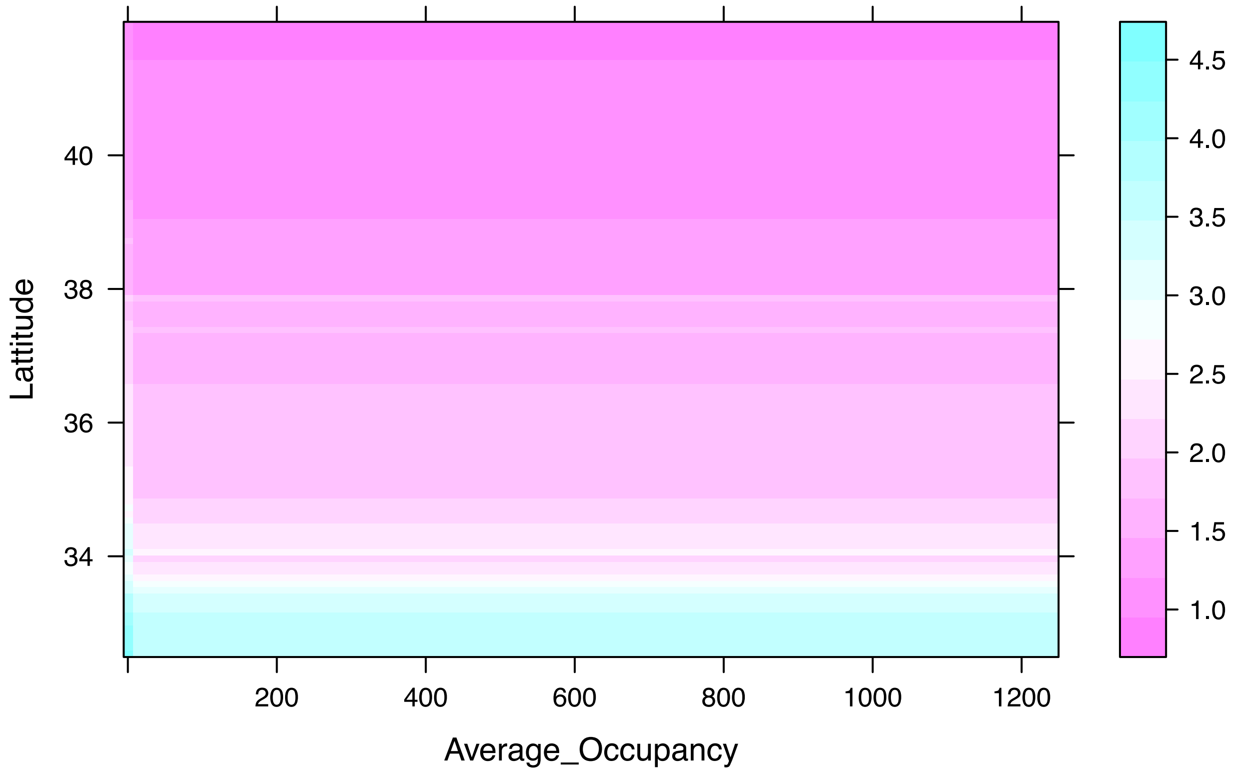
```
plot(x=gbm_model, i.var=c(1,7), n.trees=best.iter)
```



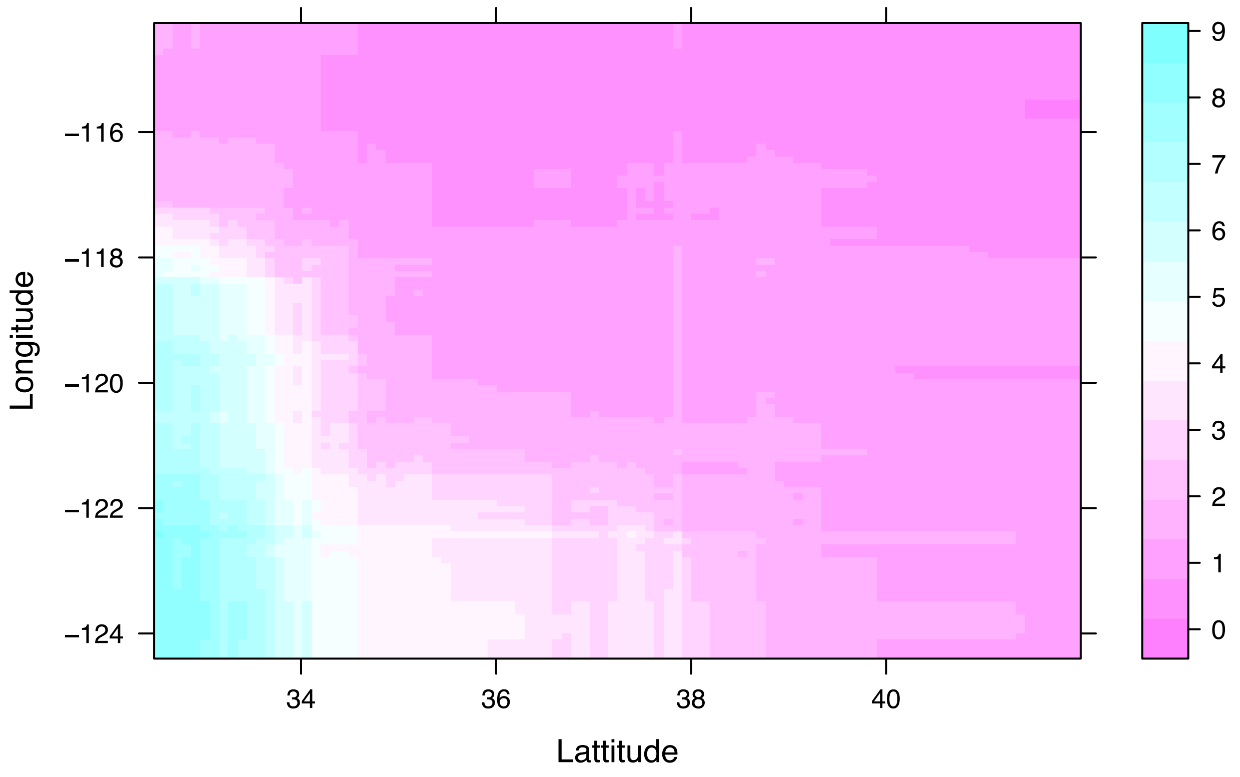
```r
plot(x=gbm_model, i.var=c(1,8), n.trees=best.iter)
```



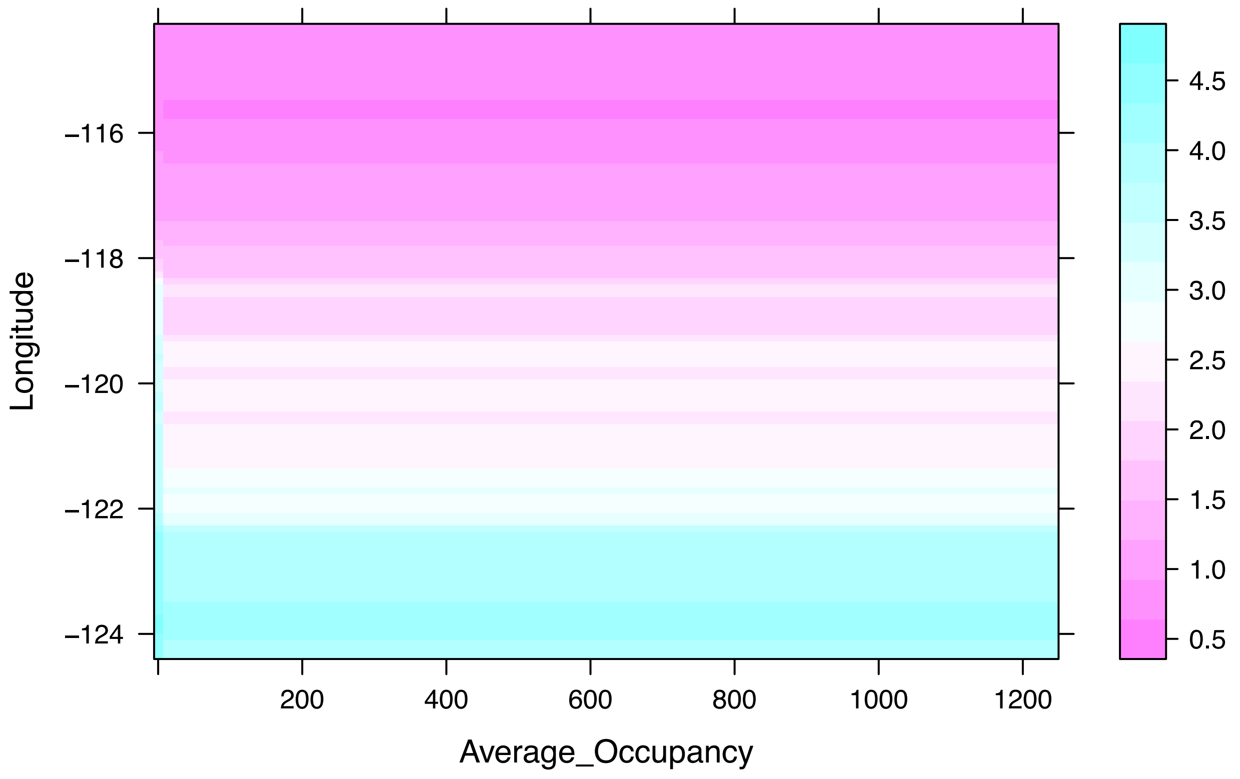
```r
plot(x=gbm_model, i.var=c(6,7), n.trees=best.iter)
```



```
plot(x=gbm_model, i.var=c(7,8), n.trees=best.iter)
```



```
plot(x=gbm_model, i.var=c(6,8), n.trees=best.iter)
```



We can see that median income has a positive correlation with house value. Average occupancey has a negative correlation with house value and has a steep drop at low occupancy. Lattitude has a netative correlation with house value. And Longitude also has a negative correlation with house value.

Occupancy and median income don't seem very correlated. There does seem to be a correlation between Lattitude and median income. The lower the lattitude and higher the median income the more the house value. There is also a correlation between longitude and median income. The lower the longitude and higher the median income the more the house value. There doesn't seem to be a correlation between Latitude and average occupancy. There does seem to be a correlation between longitude and lattitude. The lower the lattitude and longitude the more the house value.

Probelem 8 - Income Prediction

We get `mc_error` as our boosted tree error, which as compared to our missclassification error from problem 1 is a `improvement` decrease in misclassification. Our fit is marginally better. Our optimal number of trees that we fit is `bestIter_cv`. We selected this based on a 5-fold cross validation.

We fit trees with depth of 4 only because they appeared to fit better than deeper and even shallower trees. We bagged on only 50% of the data. Using 100% seemed to make our model worse.

The most important predictors for income in order are:

```
summary(fit)
```

```
##                   Length Class  Mode
## initF                 1  -none- numeric
## fit               80937  -none- numeric
## train.error        1000  -none- numeric
## valid.error        1000  -none- numeric
## oobag.improve      1000  -none- numeric
## trees              9000  -none- list
## c.splits          25036  -none- list
## bag.fraction          1  -none- numeric
## distribution          1  -none- list
## interaction.depth     1  -none- numeric
## n.minobsinnode        1  -none- numeric
## num.classes           1  -none- numeric
## n.trees               1  -none- numeric
## nTrain                1  -none- numeric
## train.fraction        1  -none- numeric
## response.name         1  -none- character
## shrinkage             1  -none- numeric
## var.levels           13  -none- list
## var.monotone         13  -none- numeric
## var.names            13  -none- character
## var.type             13  -none- numeric
## verbose               1  -none- logical
## classes               9  -none- character
## estimator         80937  -none- numeric
## data                  6  -none- list
## Terms                 3  terms  call
## cv.error           1000  -none- numeric
## cv.folds              1  -none- numeric
## call                 11  -none- call
## m                     5  -none- call
## cv.fitted         80937  -none- numeric
```

We notice that the variables that seem to be the most important (occupation, age, and housholder status) were also important in our homework 1 model. Namely, occupation was our primary split, as well as some of our nodes that predicted income. These variables having higher relative importance makes sense - income is primarily tied to the type of work you do. Also, typically the older you are in a certain occupation the more you earn. Further, being able to afford to own a home defines a certain threshold for income.

The fact that gender does not appear to be very important (it ranks second to last in relative imortance) is not indicative that it our results is inconsistent with the national average result. It is possible that being a woman correlates with another factor that predicts better across the sexes for their income level.

For example, one of the traditional explanations for why women on average earn less than men is traditionally women tend to be the homemaker, which there is generally no money in.

The plot labeld 1 is for men and 2 is for women. Homemaking is the 5 label on the x-axis (also note clerical which is label 4, also a typical low paying job). Notice that women constitute the majority of homemaking (and clerical). Also note that occupation was one of the most important variables in our model. It's possible that occupation and maybe some other variables already split out the differences between men and women before needing to split on sex directly.

Probelem 8 - Income Prediction

Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

May 22, 2016

Libraries

```r
library(dplyr)
library(readr)
library(gbm)
```

```
## Warning: package 'survival' was built under R version 3.2.5
```

```r
library(purrr)
library(ggplot2)
```

File Parameters

```r
# Session -> set working directory -> Source file location

# Path for data
data_path <- "data/"
file <- "Income_Data.txt"
ext <- paste(data_path, file, sep = "")
```

Read in Data

The data does not have any column headers, so we have to manually assign the column names. Our variables are also all categorical variables, some with an order relationship and some with out. We also have to import these values as factors, specifying if they have an order relationship or not. The following variables have an order relationship:

- Income
- Age
- Education
- Amount of time living in the bay area
- Household count
- Minors in household count

The remaining variables do not have an order relationship:

- Sex
- Marital status
- Occupation
- Dual income status
- Whether you rent or own a house

- Type of house you live in
- Ethnicity
- Languages

```
# Name of the variables imported
var_names = c("income",
              "sex",
              "marital_status",
              "age",
              "education",
              "occupation",
              "bay_duration",
              "dual_income",
              "household_total",
              "household_minors",
              "householder_status",
              "home_type",
              "ethnicity",
              "language")

# Type of the variable being imported
var_type <- cols(col_factor(sprintf("%i", 1:9), ordered =  TRUE), # income
                 col_factor(sprintf("%i", 1:2), ordered = FALSE), # sex
                 col_factor(sprintf("%i", 1:5), ordered = FALSE), # marital stat
                 col_factor(sprintf("%i", 1:7), ordered =  TRUE), # age
                 col_factor(sprintf("%i", 1:6), ordered =  TRUE), # education
                 col_factor(sprintf("%i", 1:9), ordered = FALSE), # occupation
                 col_factor(sprintf("%i", 1:5), ordered =  TRUE), # bay duration
                 col_factor(sprintf("%i", 1:3), ordered = FALSE), # dual income
                 col_factor(sprintf("%i", 1:9), ordered =  TRUE), # house count
                 col_factor(sprintf("%i", 0:9), ordered =  TRUE), # minor count
                 col_factor(sprintf("%i", 1:3), ordered = FALSE), # rent/own
                 col_factor(sprintf("%i", 1:5), ordered = FALSE), # house type
                 col_factor(sprintf("%i", 1:8), ordered = FALSE), # ethnicity
                 col_factor(sprintf("%i", 1:3), ordered = FALSE)) # language

df <- read_csv(file = ext,
               col_names = var_names,
               col_types = var_type)
```

Randomize the Dat

To use GBM's test/train functionality, we will randomize the data first. We do this because GBM will take the first x% specified as training data and the remaining as testing data. We want to ensure that the data is properly shuffled first to mimic random sampling across our data.

```
# Randomize the data
n <- dim(df)[1]
df <- df[sample(n),]

# Split 80% training and 20% test
train_n <- round(0.8 * n, 0)
test_n <- n - train_n
```

```
train <- df[0:train_n,]
test_x <- df[, 2:14]
test_y <- df[, 1]
#test_x <- df[(train_n + 1):n, 2:14]
#test_y <- df[(train_n + 1):n, 1]
```

Fitting the GBM

We now fit our first model. We will make our first model the same as that in the tutorial.

```
# Run with many tree iterations to be trimmed later
fit <- gbm(income~.,
           data = df,
           train.fraction = 1.0,
           interaction.depth = 4,
           shrinkage = .05,
           n.trees = 1000,
           bag.fraction = 0.5,
           cv.folds = 5,
           distribution = "multinomial",
           verbose = TRUE)
```

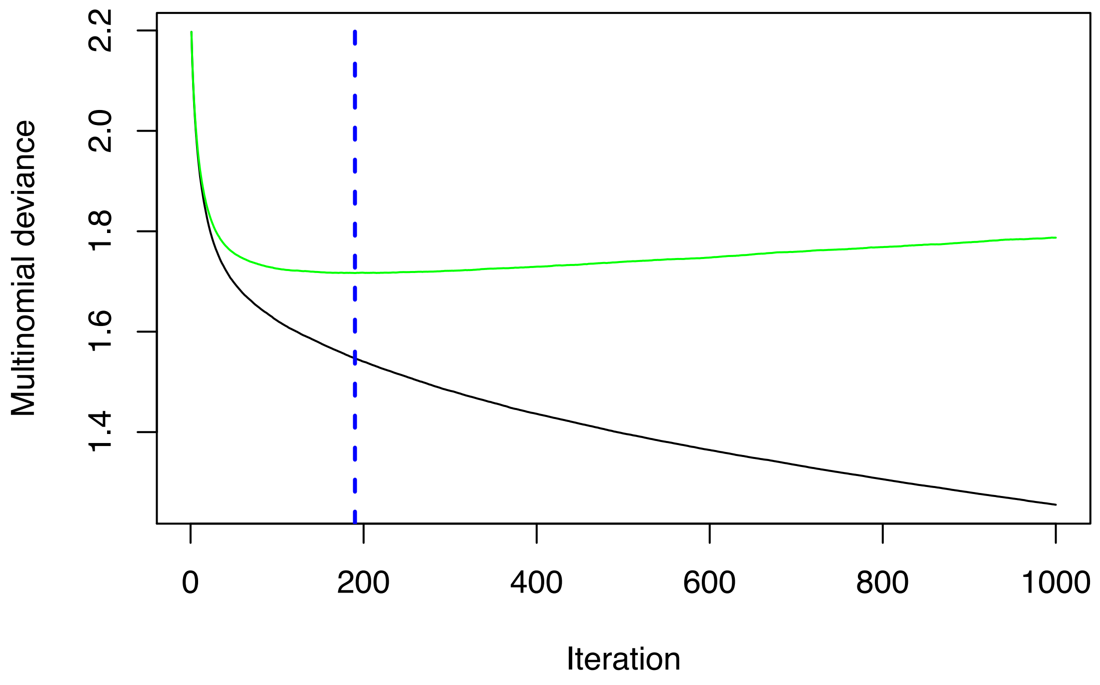
```
## Warning in if (nrow(x) != ifelse(class(y) == "Surv", nrow(y), length(y)))
## {: the condition has length > 1 and only the first element will be used
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        2.1972             nan     0.0500    0.1089
##      2        2.1379             nan     0.0500    0.0776
##      3        2.0940             nan     0.0500    0.0625
##      4        2.0575             nan     0.0500    0.0518
##      5        2.0272             nan     0.0500    0.0444
##      6        2.0012             nan     0.0500    0.0385
##      7        1.9786             nan     0.0500    0.0321
##      8        1.9582             nan     0.0500    0.0298
##      9        1.9408             nan     0.0500    0.0266
##     10        1.9246             nan     0.0500    0.0245
##     20        1.8189             nan     0.0500    0.0111
##     40        1.7247             nan     0.0500    0.0018
##     60        1.6774             nan     0.0500   -0.0002
##     80        1.6470             nan     0.0500   -0.0003
##    100        1.6222             nan     0.0500   -0.0007
##    120        1.6024             nan     0.0500   -0.0010
##    140        1.5854             nan     0.0500   -0.0008
##    160        1.5691             nan     0.0500   -0.0013
##    180        1.5538             nan     0.0500   -0.0018
##    200        1.5402             nan     0.0500   -0.0020
##    220        1.5275             nan     0.0500   -0.0015
##    240        1.5158             nan     0.0500   -0.0015
##    260        1.5041             nan     0.0500   -0.0015
##    280        1.4927             nan     0.0500   -0.0011
##    300        1.4823             nan     0.0500   -0.0016
##    320        1.4723             nan     0.0500   -0.0013
##    340        1.4630             nan     0.0500   -0.0019
##    360        1.4536             nan     0.0500   -0.0017
##    380        1.4443             nan     0.0500   -0.0016
##    400        1.4364             nan     0.0500   -0.0015
##    420        1.4284             nan     0.0500   -0.0011
##    440        1.4207             nan     0.0500   -0.0014
##    460        1.4129             nan     0.0500   -0.0020
##    480        1.4050             nan     0.0500   -0.0018
##    500        1.3974             nan     0.0500   -0.0017
##    520        1.3908             nan     0.0500   -0.0018
##    540        1.3837             nan     0.0500   -0.0015
##    560        1.3772             nan     0.0500   -0.0017
##    580        1.3706             nan     0.0500   -0.0016
##    600        1.3643             nan     0.0500   -0.0020
##    620        1.3581             nan     0.0500   -0.0020
##    640        1.3520             nan     0.0500   -0.0017
##    660        1.3462             nan     0.0500   -0.0016
##    680        1.3406             nan     0.0500   -0.0015
##    700        1.3345             nan     0.0500   -0.0016
##    720        1.3284             nan     0.0500   -0.0018
##    740        1.3226             nan     0.0500   -0.0017
##    760        1.3173             nan     0.0500   -0.0016
##    780        1.3118             nan     0.0500   -0.0015
##    800        1.3062             nan     0.0500   -0.0019
##    820        1.3006             nan     0.0500   -0.0015
##    840        1.2952             nan     0.0500   -0.0014
##    860        1.2906             nan     0.0500   -0.0021
##    880        1.2850             nan     0.0500   -0.0018
##    900        1.2800             nan     0.0500   -0.0018
##    920        1.2749             nan     0.0500   -0.0012
##    940        1.2702             nan     0.0500   -0.0015
##    960        1.2653             nan     0.0500   -0.0020
##    980        1.2601             nan     0.0500   -0.0013
##   1000        1.2553             nan     0.0500   -0.0014
```

```r
# Find best iteration by 5 fold crossvalidation
gbm.perf(object = fit, method = "cv")
```

```
## [1] 190
```

```r
bestIter_cv <- gbm.perf(object = fit, method = "cv")
```



```
bestIter_cv
```

```
## [1] 190
```

We use 5 fold cross validation to pick our best number of trees. We will then use this amount (`bestIter_test`) to predict on our holdout testing data.

Determine Misclassification Rate on Test

We now apply the predictions to our test data and generate the misclassification error.

```
# Determine Predictions based on best CV iteration
pred <- as.data.frame(predict(fit, test_x, bestIter_cv))

# Rename columns of our prediction
pred_name <- c("income 1",
               "income 2",
               "income 3",
               "income 4",
               "income 5",
               "income 6",
               "income 7",
               "income 8",
               "income 9")
colnames(pred) <- pred_name

# Return the column (class) with maximum value per observation. bind with test y
pred <- as.data.frame(max.col(pred))
pred <- cbind(pred, test_y) %>%
  rename(test = income, pred = `max.col(pred)`)
```

```r
# Create table of all correct classifications (true) vs misclass (false)
misclass <- pred %>%
  mutate(correct_class = (pred == test)) %>%
  count(correct_class)

# Compute misclassification error
tot_obs <- misclass$n[1] + misclass$n[2]
idx <- which.min(misclass$correct_class)
mc_error <- misclass$n[idx]/tot_obs
mc_error
```

```
## [1] 0.5560992
```

```r
hw1_error <- 0.65
improvement <- 1 - mc_error/hw1_error
```

Problem 9 - Occupation Prediction

Henry Neeb, Christopher Kurrus, Tyler Chase, and Yash Vyas

May 22, 2016

Libraries

```r
library(gbm)
library(ggplot2)
```

File Parameters

```r
setwd("Z:/Acads/3spr_2016/Stats315b/")
OccData.table = read.table(file = 'Occupation_Data.txt', header = F, sep = ',
')
```

Read in Data

We name the column headers and convert the values into categorical variables. The columns that have unordered categorical variables are:

- Occupation
- House Type
- Sex
- Marital status
- Dual income status
- Whether you rent or own a house
- Type of house residency
- Ethnicity
- Language spoken

While the columns that have categorical values with an ordered relationship are:

- Age
- Education
- Income
- Number of years of residency in the bay area
- Number of people in the house
- Number of people in house below 18 years

```r
# Name of the variables imported
colnames(OccData.table) = c("occu","HouseType", "sex", "marStatus", "age",
                             "education",  "income",
                             "ResYears", "dualInc", "housePeople",
                                "Below18Peop", "HouseResType",
                                "Ethini", "Lingo")

# Converting to unordered categorical variable
OccData.table$occu = factor(OccData.table$occu, levels = c(1:9))
OccData.table$HouseType = factor(OccData.table$HouseType, levels = c(1:5))
OccData.table$sex = factor(OccData.table$sex, levels = c(1:2))
OccData.table$marStatus = factor(OccData.table$marStatus, levels = c(1:5))
OccData.table$dualInc = factor(OccData.table$dualInc, levels = c(1:3))
OccData.table$HouseResType =factor(OccData.table$HouseResType, levels = c(1:3
))
OccData.table$Ethini = factor(OccData.table$Ethini, levels = c(1:8))
OccData.table$Lingo = factor(OccData.table$Lingo, levels = c(1:3))

# Converting to ordered categorical variable
OccData.table$age = ordered(OccData.table$age, levels = c(1:7))
OccData.table$education = ordered(OccData.table$education, levels = c(1:6))
OccData.table$income = ordered(OccData.table$income, levels = c(1:9))
OccData.table$ResYears = ordered(OccData.table$ResYears, levels = c(1:5))
OccData.table$housePeople = ordered(OccData.table$housePeople, levels = c(1:9
))
OccData.table$Below18Peop = ordered(OccData.table$Below18Peop, levels = c(0:9
))
```

Randomize the Data

We shuffle the data in order to ensure that gbm does not choose sequential rows in test or train datasets.

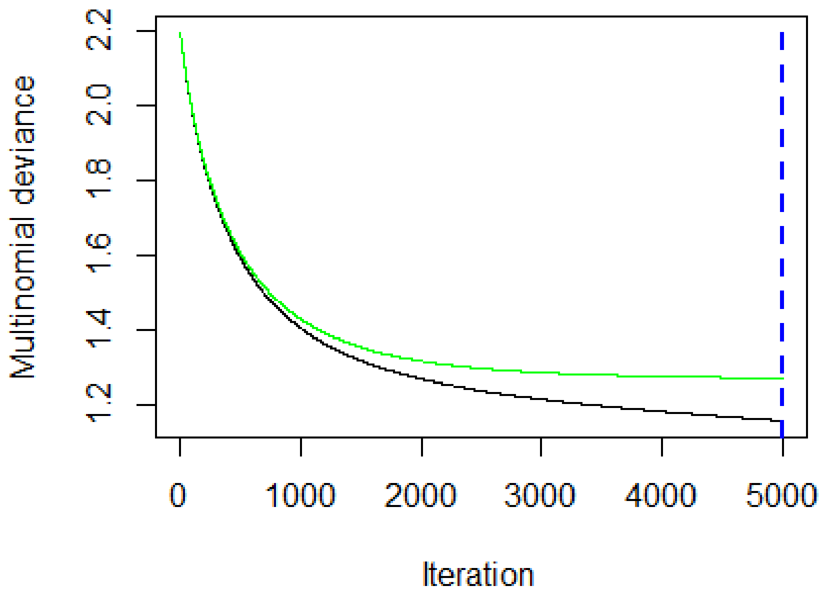
```
# Randomize the data
set.seed(1)
u = runif(nrow(OccData.table))
OccData.tableShuffle = OccData.table[order(u),]


# Split 70% training and 30% test
train = sample(1:nrow(OccData.tableShuffle), 70/100*nrow(OccData.tableShuffle
))
```

Fitting the GBM

We now fit our first model. We choose 2500 randomly as values for number of trees.

```
# Model
boost.occ = gbm(occu ~., data= OccData.tableShuffle[train,],
                distribution = "multinomial",n.trees =5000,
                interaction.depth =4, cv.folds = 5, verbose = TRUE)

# Find best iteration by 5 fold crossvalidation
gbm.perf(object = boost.occ, method = "cv")
```

```
## [1] 4997
bestIter_cv <- gbm.perf(object = boost.occ, method = "cv")
```



```
bestIter_cv
## [1] 4997
```

We use 5 fold cross validation to pick our best number of trees. We will then use this amount (4997) to predict on our holdout testing data.

Determine Misclassification Rate on Test data

We now apply the predictions to our test data and generate the misclassification error.

```
# Determine Predictions based on best CV iteration
occ.pred = max.col(as.data.frame
                     (predict(boost.occ, newdata = OccData.tableShuffle[-trai
n,],
                              n.trees = bestIter_cv)))

misclass.matrix = table(occ.pred, OccData.tableShuffle[-train,1])
misclass.rate = 1 - sum(diag(misclass.matrix))/nrow(OccData.tableShuffle[-tra
in,])

#Obtaining the misclassification rate for each class
```

```
err.class = NA

for (i in 1:9){
  err.class[i] = 1- misclass.matrix[i,i]/sum(misclass.matrix[,i])
}
```

```
occ.pred   1   2   3   4   5   6   7   8   9
       1 662  76  80 148  31  39  26  13  13
       2   9  13   5  10   3   9   1   1   4
       3  22  25  82  19   4   8  16   2  11
       4  44  27  12  95  18  12   4   1   6
       5  16   5   4   7 120   8   0  18   6
       6  27  57  33  47   6 365   6   1  45
       7   2   0   3   2   0   2  18   0   1
       8  40   2   5   9  20   1   2 164   3
       9   3  11   3  10   5   3   2   2  23
```

```
err.class

## [1] 0.1975758 0.9398148 0.6387665 0.7262248 0.4202899 0.1834452 0.7600000
## [8] 0.1881188 0.7946429
```

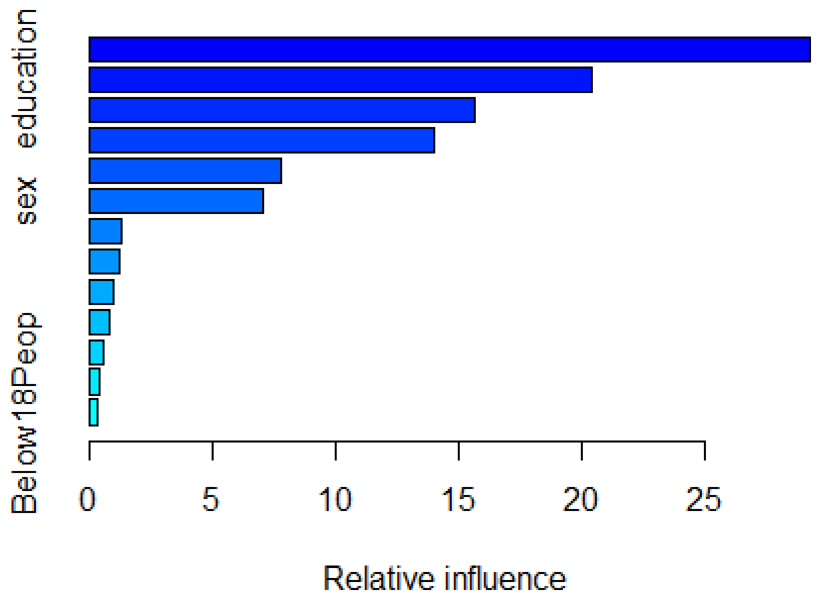
Occupation Prediction

We get `0.4206` as our overall boosted tree misclassification error. Our optimal number of trees that we fit is 4997. We selected this based on a 5-fold cross validation.

We have fit trees with depth of 4. And used the default shrinkage parameter as 0.001.

The most important predictors for income in order are:

```
summary(boost.occ)
```



```
##                      var    rel.inf
## age                  age 29.2317564
## education      education 20.4035857
## income            income 15.6619919
## HouseResType HouseResType 14.0238070
## dualInc          dualInc  7.8191364
## sex                  sex  7.0419062
## housePeople  housePeople  1.3595753
## HouseType      HouseType  1.2137822
## Ethini            Ethini  1.0384646
## marStatus      marStatus  0.8101742
## ResYears        ResYears  0.5922543
## Lingo              Lingo  0.4388237
## Below18Peop  Below18Peop  0.3647421
```

Conclusions

We notice that the most important variable in order to predict the occupation of the person is his age. Education level, income and the type of house a person lives in are also among the top variables that help in predicting the occupation of a person. The importance of these variables corroborates our beliefs. Age clearly determines the type of work that one pursues. Also, occupations are highly associated with education levels. Rarely would one find a person with low education level to be in a professional role. Also, all occupation levels do not offer same pay.

We also noticed that the type of occupation of a person also depends on whether there is a dual source of income in the house and the sex of the person.

Besides, we find that the misclassification rate for a person in one of the following classes is low:

- Professional/Managerial
- Homemaker
- Student,
- HS or College
- Retired

as compared to the misclassification rate for a person belonging to the following class:

- Sales Worker
- Factory Worker/Laborer/Driver
- Clerical/Service Worker
- Military
- Unemployed

The high prediction accuracy of occupation in the first group is due to high correlation between the certain input variables and occupation types. Professional/Managerial is linked with education, Retired and Student, HS or College is linked with age while Homemaker is linked with sex. There are no clear attributes for the occupation types in the second group.