Homework 1

Pittsburgh
Jingyi Guo (jingyig1)

1. (a) Given the loss table, we have

$$E(\mathcal{L}(Y, \hat{f}(x))) = E[E(L(Y, f'(x)|x))]$$
$$= E(\sum_{i=0}^{1} L(Y, \hat{f}(x)|x) P(Y=i|x)]$$

minimize the inside of the above expectation over $\hat{f}(x)$ for each value of x,

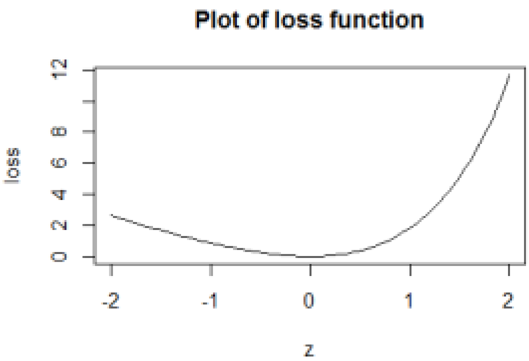
we have:

$$\hat{f}(x) = \underset{g\in\{0,1\}}{\arg\min} \sum_{k=0}^{1} \mathcal{L}(k,g) P(Y=k|X=x)$$
$$= \underset{g\in\{0,1\}}{\arg\min} \sum_{k=0}^{1} \mathbb{1}_{g\neq k} \mathcal{L}_{kg} P(Y=k|X=x)$$
$$= \underset{g\in\{0,1\}}{\arg\min} \sum_{k=0}^{1} (1-\mathbb{1}_{k=g}) \mathcal{L}_{kg} P(Y=k|X=x)$$
$$= \underset{g\in\{0,1\}}{\arg\min} \sum_{k=0}^{1} \mathcal{L}_{kg} P(Y=k|X=x)$$
$$= \underset{g\in\{0,1\}}{\arg\min} L_{0g}(1-p) + \overset{L_{1g}P}{\cancel{L_{0g}p}} \quad \text{where } p = P(Y=k|X=x)$$
$$= \begin{cases} 1 & \text{if } L_{0g}p > L_{0g}(1-p) \Leftrightarrow p > \frac{L_{01}}{L_{01}+L_{10}} \\ 0 & \text{otherwise} \end{cases}$$

2. Code:

```
a=1.1
b=2
# z = Y - f(X)
loss = function(z){
   return (b*(exp(a*z)-a*z-1))
}
plot(loss, from = -2, to = 2, xlab = "z", main = "Plot of loss function")
```

Result:



Comment:

The loss function curve is asymmetric. It increases very fast when z is positive, giving more penalty when there's underestimation.

2.(b) $E(\text{loss}) = E(L(Y, \hat{f}(x)))$

$$= E(b(e^{a(Y-\hat{f}(x))} - a(Y-\hat{f}(x)) - 1))$$

$$= bE[e^{aY} - e^{a\hat{f}(x)} - aY + a\hat{f}(x) - 1]$$

$$= bE(E(e^{aY}|x)e^{-af(x)} - aE(Y|X) + af(x) - 1)$$

take derivative of the inside of the outer expectation wrt. $f(x)$,

we have $-ae^{-af(x)}E(e^{aY}|x) + a = 0$

$\Rightarrow e^{af(x)} = E(e^{aY}|x)$

So $f(x) = \dfrac{\log(E(e^{aY}|X))}{a}$

(c) Suppose $Y|X=x \sim N(\beta \cdot x, \sigma^2)$ $\quad = E(e^{Zt})$

Use the fact $Z \sim N(\mu, \sigma^2) \Rightarrow M_Z(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}$, we have

$$f(x) = \frac{\log(E(e^{aY}|x))}{a} = \frac{\log e^{a\beta x + \frac{1}{2}\sigma^2 a^2}}{a} = \beta x + \frac{1}{2}a\sigma^2.$$

(d) Code:

```
beta = 0.5
b=2
sigma = 2 a = 1.1
#Estimation functions
#Estimation using the conditional expectation of Y|X f_condexp =
function(x){beta*x}
#TODO: Put your function in here. You can reference a,b,sigma, and it will just
pull them from # the outside namespace
f_yours = function(x){beta*x + (a*sigma^2)/2}
#Define the loss function, where z = y - yhat loss = function(z){b*(exp(a*z)-a*z-
1)}
#Simulation to see how you do
reps = 1000
#Just generate the X variables normally. You don't really care x =
rnorm(reps,0,1)
#Generate the Y variables from our normal model
y = rnorm(reps,x*beta,sigma)
#Compute the losses
```

condexp_loss = sapply(y-f_condexp(x),loss)
your_loss = sapply(y-f_yours(x),loss)
print(paste("Average loss of the conditional expectation:", round(mean(condexp_loss),3))) print(paste("Average loss of your method:", round(mean(your_loss),3)))
 # > print(paste("Average loss of the conditional expectation:", round(mean(condexp_loss),3)))
[1] "Average loss of the conditional expectation: 15.117"
 # > print(paste("Average loss of your method:", round(mean(your_loss),3)))
[1] "Average loss of your method: 4.476"

Comment:
Average loss is lower than the conditional mean. This is because loss function give larger penalty for underestimation.

3. (a) $Var(x^T\hat{\beta}) = x^T Var(\hat{\beta}) x$

Given $Var(\hat{\beta}) = Var((X^TX)^{-1}X^Ty)$

$$= (X^TX)^{-1}X^T Var(y)[(X^TX)^{-1}X^T]^T$$
$$= (X^TX)^{-1}X^T \sigma^2 I_n \times ((X^TX)^T)^{-1}$$
$$= X^{-1}(X^T)^{-1}X^T \sigma^2 I_n \times (X^TX)$$
$$= \sigma^2 (X^TX)^{-1}$$

$\therefore Var(\hat{\beta}) = \sigma^2 x^T (X^TX)^{-1} x$

(b) $\frac{1}{n}\sum_{i=1}^{n} Var(X_i^T\hat{\beta}) = \frac{1}{n}\sigma^2 Tr(X(X^TX)^{-1}X^T)$

$$= \frac{1}{n}\sigma^2 Tr(H) = \frac{P}{n}\sigma^2.$$

4. Code:

```
library(glmnet)
library(randomForest)
library(pROC)
marketing = read.csv(' /Users/apple/Desktop/ml2/marketing.csv')
set.seed(1)
idx.test = sample(1:nrow(marketing),floor(0.2*nrow(marketing))) test =
marketing[idx.test,]
train = marketing[-idx.test,]
```

(a) Using the training data, the fraction of successes:
mean(train$y=='yes')

[1] 0.1162874

(b) Using training data, fit a logistic regression using all varialbes to predict the outcome and calculate the misclassification rate

fit = glm(y~., data = train, family = 'binomial')
predict = predict(fit,newdata = test)
result = ifelse(predict>0,'yes','no')
mean(test$y != result)
[1] 0.1199956

(c) Using a silly classifier that guesses 'no' for all observations

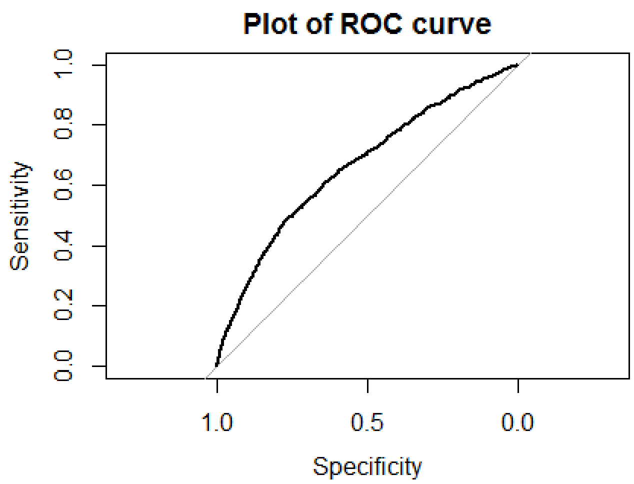
mean(test$y != 'no')
[1] 0.1197744
This rate is almost close to the logistic miscalculate rate.

(d) Using logistic model, find the 1000 clients in the test set that are most likely to score 'yes'

index = order(predict, decreasing = TRUE)
mean(test[index[1:1000],"y"] == "yes")
[1] 0.27
For this set of 1000 clients, the fraction of them actually say yes is 27%, whereas If we pick a random set of 1000 clients, the fraction will be as low as 11.6%.

(e)

roc(test$y, as.numeric(predict), plot = TRUE, main = "Plot of ROC curve")



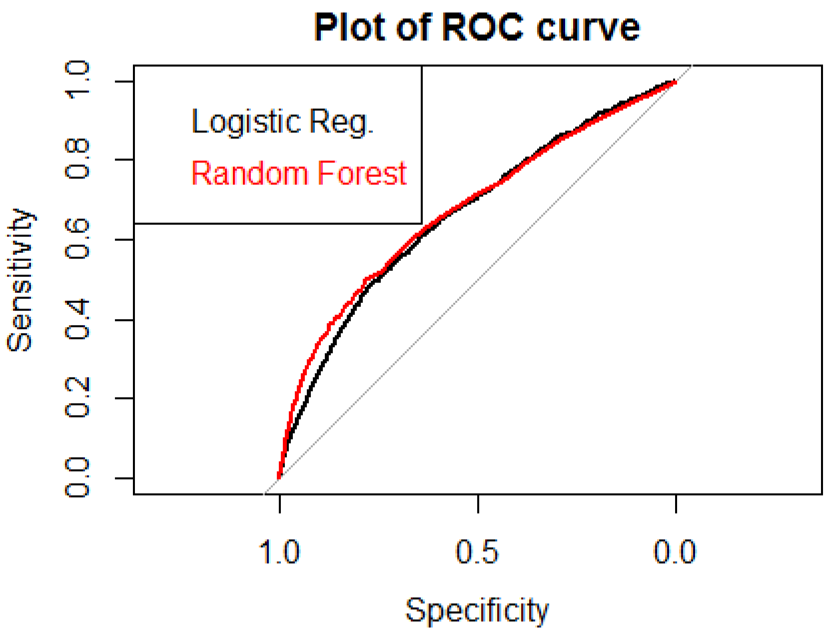
(f) fit a random forest to our training data

rf_fit= randomForest(train[,1:8], train$y)
rf_predict = predict(rf_fit, test[,1:8])
mean(test$y != rf_predict)
[1] 0.1192214

Select the top 1000 observations by using the predict function with type='prob' to get the vote fractions for each observation, and then sort

rf_predict = predict(rf_fit, test[,1:8],type="prob")
index = order(rf_predict[,2], decreasing = TRUE)
mean(test[index[1:1000],"y"] == "yes")
#[1] 0.334
roc(test$y,rf_predict[,2], plot=TRUE, add=TRUE, col='red')
legend("topleft",c("Logistic Reg.", "Random Forest"),
col = c('black','red'),text.col = c("black","red"))

Random Forest is a little bit better in lower corner, but didn't beat logistic overall.



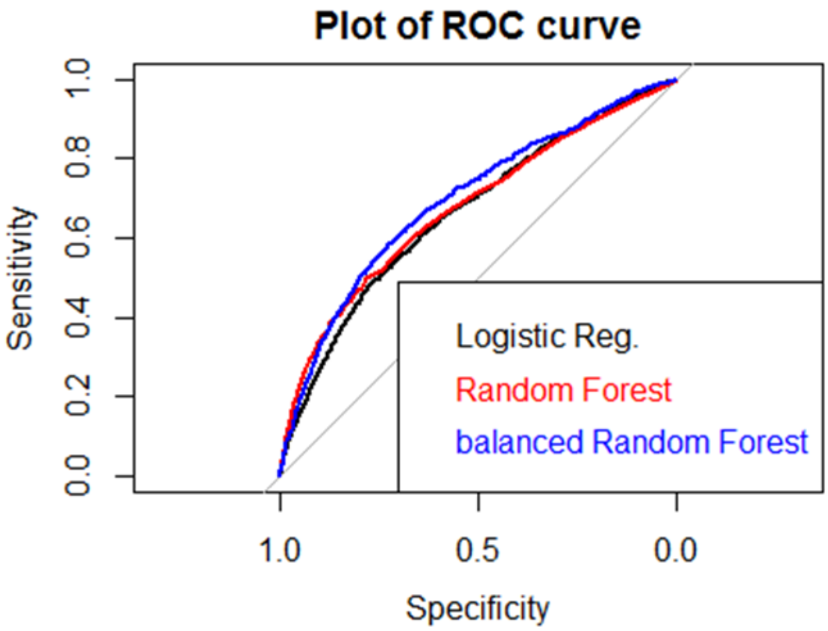
(g)

rf2_fit = randomForest(train[,1:8], train$y, strata=train$y, sampsize = c(sum(train$y=='yes'), sum(train$y=='yes')))
rf2_predict = predict(rf2_fit, test[,1:8])
mean(test$y != rf2_predict)
[1] 0.2837868

The misclassification rate is higher.

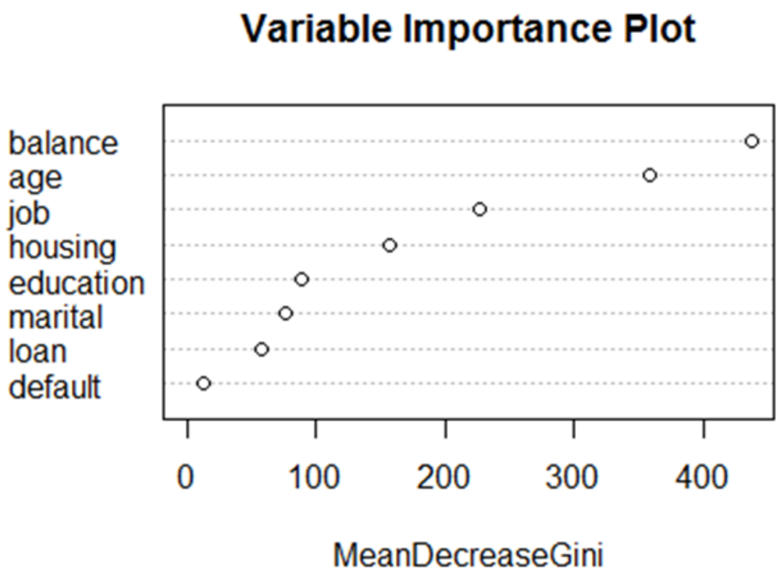
Select the top 1000 observations and then sort:

rf2_predict = predict(rf2_fit, test[,1:8],type="prob")
index = order(rf2_predict[,2], decreasing = TRUE)
mean(test[index[1:1000],"y"] == "yes")
[1] 0.31
roc(test$y,rf2_predict[,2], plot=TRUE, add=TRUE, col='blue')
legend("bottomright",c("Logistic Reg.", "Random Forest", "balanced Random Forest"),col = c('black','red', 'blue'),text.col = c("black","red", "blue"))



The ROC curve shows that balanced random forest performs better than others. This might come from increased sensitivity at lower values of specificity.

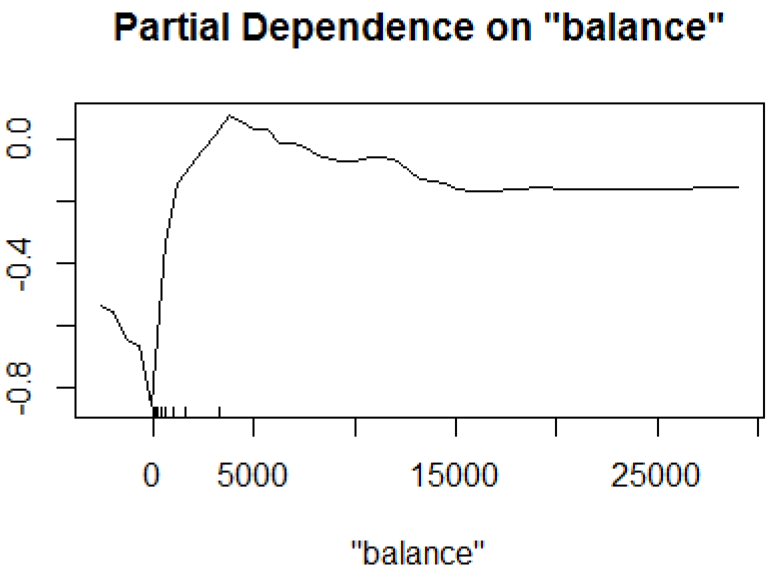
(h)
varImpPlot(rf2_fit,main ="Variable Importance Plot")



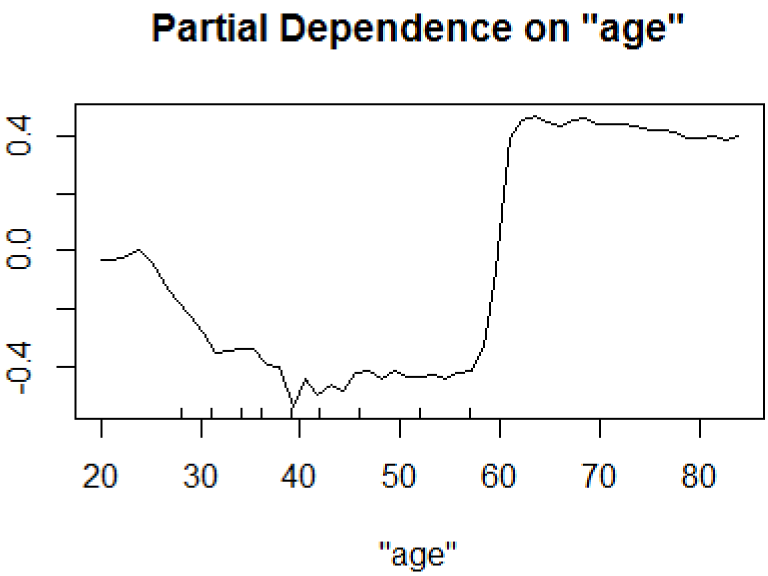
As shown above, balance and age are the two most important variables for classification.

(i)
partialPlot(rf2_fit, pred.data=train[sample(1:nrow(train),1000),], x.var='balance',

which.class='yes')



partialPlot(rf2_fit, pred.data=train[sample(1:nrow(train),1000),], x.var='age', which.class='yes')



The partial dependence of these two important variables are not linear, so randome forest model fits better than linear model.