Predictive Modeling with R

# Q1 Data Preparation

### 1a. Load the dataset insurance.csv into memory.

#set working directory  
setwd("/Users/marianneleano/Library/Mobile Documents/com~apple~CloudDocs/Desktop/dsci 512/")  
  
insurance <- read.csv(file = '512\_finalproject/insurance.csv')  
colnames(insurance)

## [1] "age" "sex" "bmi" "children" "smoker" "region" "charges"

### 1b. In the data frame, transform the variables charges by setting insurancechanges). Do not transform it outside of the data frame.

insurance$charges = log(insurance$charges)

### 1c. Using the data set from 1b, use the model.matrix() function to create another data set that uses dummy variables in place of categorical variables. Verify that the first column only has ones (1) as values, and then discard the column only after verifying it has only ones as values.

dummy <- model.matrix(~. -charges, data=insurance)

#remove first column with 1 as values  
dummy = dummy[, -1]  
colnames(dummy)

## [1] "age" "sexmale" "bmi" "children"   
## [5] "smokeryes" "regionnorthwest" "regionsoutheast" "regionsouthwest"

### 1d. Use the sample() function with set.seed equal to 1 to generate row indexes for your training and tests sets, with 2/3 of the row indexes for your training set and 1/3 for your test set. Do not use any method other than the sample() function for splitting your data.

set.seed(1)  
train\_size <- floor(2\*nrow(insurance)/3)  
train\_idx <- sample(seq\_len(nrow(insurance)), size=train\_size)

### 1e. Create a training and test data set from the data set created in 1b using the training and test row indexes created in 1d.

train\_1e <- insurance[train\_idx, ]  
test\_1e <- insurance[-train\_idx, ]

### 1f. Create a training and test data set from data set created in 1c using the training and test row indexes created in 1d.

train\_1f <- dummy[train\_idx, ]  
test\_1f <- dummy[-train\_idx, ]

# Q2 Build a multiple linear regression model

### 2a. Perform multiple linear regression with charges as the response and the predictors are age, sex, bmi, children, smoker, and region. Print out the results using the summary() function. Use the training dataset created in step 1e to train your model.

lm.fit = lm(charges ~ age + sex + bmi + children + smoker + region, data=insurance[train\_idx, ])  
summary(lm.fit)

##   
## Call:  
## lm(formula = charges ~ age + sex + bmi + children + smoker +   
## region, data = insurance[train\_idx, ])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.03533 -0.17138 -0.03868 0.06382 2.12159   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.042076 0.082891 84.956 < 2e-16 \*\*\*  
## age 0.034915 0.001030 33.911 < 2e-16 \*\*\*  
## sexmale -0.063228 0.028700 -2.203 0.027847 \*   
## bmi 0.011805 0.002441 4.836 1.56e-06 \*\*\*  
## children 0.100152 0.011894 8.420 < 2e-16 \*\*\*  
## smokeryes 1.533561 0.035475 43.229 < 2e-16 \*\*\*  
## regionnorthwest -0.082398 0.041040 -2.008 0.044972 \*   
## regionsoutheast -0.140025 0.040180 -3.485 0.000516 \*\*\*  
## regionsouthwest -0.107718 0.041108 -2.620 0.008934 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4259 on 883 degrees of freedom  
## Multiple R-squared: 0.7839, Adjusted R-squared: 0.7819   
## F-statistic: 400.4 on 8 and 883 DF, p-value: < 2.2e-16

### 2b. Is there a relationship between the predictors and the response?

To determine the relationship between **charges** and **age, sex, bmi, children, smoker and region**, we look at the p-value in our summary() output. If the p-value is greater than 5% their will be no indication of a relationship. If the p-value is less than 5%, than their is a relationship. As the p-value is *2.2e-16%*, which is less than 5%, which means that there is a relationship between the predictors and the response.

### 2c. Does sex have a statistically significant relationship to the response?

Sex does play a significant relationship to the response charges. We determine this by looking at the Pr(>|t|) values. Sex, specifically for male is *0.027847%* is less than 5% which means that sex has a significant relationship with response charges.

### 2d. Perform best subset selection using the stepAIC() function from the MASS library, choose best model based on AIC. For the “direction” parameter in the stepAIC() method, set direction = “backward”.

The best model will include the following predictors: **age, sex, bmi, children, smoker, region**.

library(MASS)  
  
full = lm(charges ~ age+sex+bmi+children+smoker+region, data = insurance[train\_idx, ])  
lm.bwd = stepAIC(full, direction = "backward")

## Start: AIC=-1513.99  
## charges ~ age + sex + bmi + children + smoker + region  
##   
## Df Sum of Sq RSS AIC  
## <none> 160.13 -1513.99  
## - sex 1 0.88 161.01 -1511.10  
## - region 3 2.39 162.52 -1506.80  
## - bmi 1 4.24 164.37 -1492.67  
## - children 1 12.86 172.99 -1447.09  
## - age 1 208.54 368.67 -772.14  
## - smoker 1 338.89 499.02 -502.09

lm.bwd

##   
## Call:  
## lm(formula = charges ~ age + sex + bmi + children + smoker +   
## region, data = insurance[train\_idx, ])  
##   
## Coefficients:  
## (Intercept) age sexmale bmi   
## 7.04208 0.03491 -0.06323 0.01180   
## children smokeryes regionnorthwest regionsoutheast   
## 0.10015 1.53356 -0.08240 -0.14002   
## regionsouthwest   
## -0.10772

### 2e. Compute the test error of the best model in 2d based on AIC using LOOCV using trainControl() and train() from the caret library. Report the MSE by squaring the reported RMSE.

The MSE for the multiple linear regression model in 2d is **0.1833683**.

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

train\_control = trainControl(method = "LOOCV")  
  
model\_2e <- train(charges ~ age+sex+bmi+children+smoker+region, data = insurance[train\_idx, ], trControl = train\_control, method = "lm")  
print(model\_2e)

## Linear Regression   
##   
## 892 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Leave-One-Out Cross-Validation   
## Summary of sample sizes: 891, 891, 891, 891, 891, 891, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.4282153 0.7792766 0.2626782  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

MSE = (0.4282153)^2  
MSE

## [1] 0.1833683

### 2f. Calculate the test error of the best model in 2d based on AIC using 10-fold Cross-Validation. Use train and trainControl from the caret library. Refer to model selected in 2d based on AIC. Report the MSE.

The MSE using the 10-fold CV model is **0.1792372**.

library(caret)  
train\_control\_2f <- trainControl(method = "CV", number = 10)  
  
model\_2f <- train(charges ~ age+sex+bmi+children+smoker+region, data = insurance[train\_idx, ], trControl = train\_control\_2f, method = "lm")  
print(model\_2f)

## Linear Regression   
##   
## 892 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 804, 803, 803, 801, 803, 802, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.4233641 0.7822236 0.2621516  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

MSE\_2f = (0.4233641)^2  
MSE\_2f

## [1] 0.1792372

### 2g. Calculate and report the test MSE using the best model from 2d and test data set created in step 1e.

The MSE using the best model from 2d and the test data set in 1e is **0.2347613**.

library(caret)  
train\_control = trainControl(method = "LOOCV")  
  
model\_2g <- train(charges ~ age+sex+bmi+children+smoker+region, data=insurance[-train\_idx, ], trControl=train\_control, method="lm")  
print(model\_2g)

## Linear Regression   
##   
## 446 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Leave-One-Out Cross-Validation   
## Summary of sample sizes: 445, 445, 445, 445, 445, 445, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.4845217 0.7309161 0.3181654  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

MSE\_2g = (0.4845217)^2  
MSE\_2g

## [1] 0.2347613

### 2h. Compare the test MSE calculated in step 2.f using 10-fold cross-validation with the test MSE calculated in step 2.g. How similar are they?

MSE\_2f in 2f using 10-fold CV is **0.1792372**. MSE\_2g in 2g using LOOCV is **0.2347613**. The MSE for both models are different from each other. We use the test data set to compare the predictions to the actual response variable in the test data so we are able to evaluate the model’s accuracy. The 10-fold CV performed better than the LOOCV because the MSE is closer to zero. This means the 10-fold CV has a good variance-bias tradeoff with low variance.

# Q3 Build a regression model tree.

### 3a. Build a regression tree model using function tree(), where charges is the response and the predictors are age, sex, bmi, children, smoker, and region.

Our final model will use the following predictors: **smoker, age, children and bmi**.

insurance$sex <- as.factor(insurance$sex)  
insurance$smoker <- as.factor(insurance$smoker)  
insurance$region <- as.factor(insurance$region)

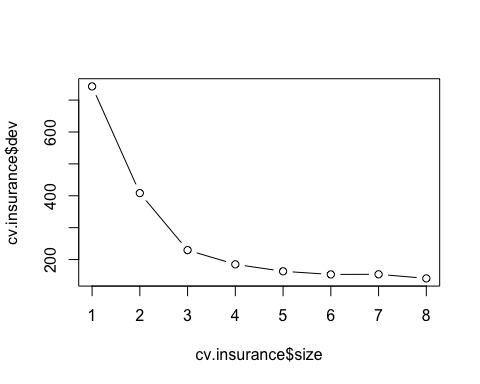
library(tree)  
tree.insurance = tree(charges ~ age+sex+bmi+children+smoker+region, data=insurance, subset=train\_idx)  
summary(tree.insurance)

##   
## Regression tree:  
## tree(formula = charges ~ age + sex + bmi + children + smoker +   
## region, data = insurance, subset = train\_idx)  
## Variables actually used in tree construction:  
## [1] "smoker" "age" "children" "bmi"   
## Number of terminal nodes: 8   
## Residual mean deviance: 0.1372 = 121.3 / 884   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.84080 -0.19220 -0.05068 0.00000 0.08704 2.40300

### 3b. Find the optimal tree by using cross-validation and display the results in a graphic. Report the best size.

The size for the best optimal model is **6**.

cv.insurance = cv.tree(tree.insurance)  
plot(cv.insurance$size, cv.insurance$dev, type='b')



### 3c. Justify the number you picked for the optimal tree with regard to the principle of variance-bias trade-off.

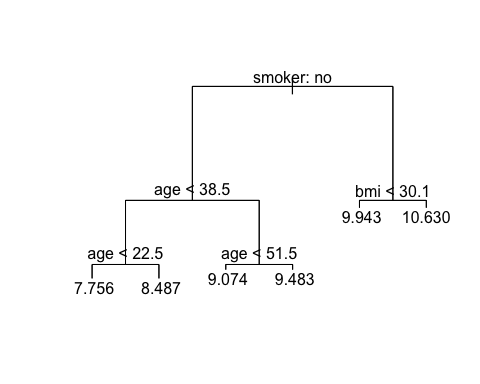
We would like to choose the size that has the least amount of error rate. The size for the best optimal model is 6. We would like to go with the size with the lowest error rate because we want to first test how well our model is able to get trained by some data and then predict the data it hasn’t seen. If our test error is high after pruning, that just means we can continue with a subtree to find a simpler model and lower test error. We want to find a balance between variance and bias, with respect to how simple our tree can be. It is important to do cross validation so we can select the optimal value that controls the trade-off between a subtree’s complexity and its fit to the training data.

### 3d. Prune the tree using the optimal size found in 3b.

prune.insurance = prune.tree(tree.insurance, best=6)

### 3e. Plot the best tree model and give labels.

plot(prune.insurance)  
text(prune.insurance, pretty=0)



### 3f. Calculate the test MSE for the best model.

The MSE using the regression tree model is **0.2019634**. Based on the two ways I computed test error after pruning, the test error before using the test data set is **0.2019634**. Before pruning the test error is 0.1890688.

#test error before pruning  
yhat0 = predict(tree.insurance, insurance[-train\_idx,])  
insurance.test0 = insurance[-train\_idx, "charges"]  
mean((yhat0 - insurance.test0)^2)

## [1] 0.1890688

#test error after pruning  
yhat = predict(prune.insurance, newdata=insurance[-train\_idx,])  
insurance.test = insurance[-train\_idx, "charges"]  
mean((yhat - insurance.test)^2)

## [1] 0.2019634

# Q4 Build a random forest model.

### 4a. Build a random forest model using function randomForest(), where charges is the response and the predictors are age, sex, bmi, children, smoker, and region.

## Loading required package: randomForest

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

Based on the randomForest() function, we can see that the MSR and % variance are based on OOB estimates, a device to get honest error estimates.

rf.insurance = randomForest(charges ~ age+sex+bmi+children+smoker+region, data=insurance, subset=train\_idx, importance=TRUE, na.action = na.exclude)  
rf.insurance

##   
## Call:  
## randomForest(formula = charges ~ age + sex + bmi + children + smoker + region, data = insurance, importance = TRUE, subset = train\_idx, na.action = na.exclude)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## Mean of squared residuals: 0.1362289  
## % Var explained: 83.6

### 4b. Compute the test error using the test data set.

The MSE using the random forest model is **0.1779684**.

yhat.rf = predict(rf.insurance, newdata = insurance[-train\_idx,])  
insurance.test = insurance[-train\_idx, "charges"]  
mean((yhat.rf - insurance.test)^2)

## [1] 0.1779684

### 

### 4c. Extract variable importance measure using the importance() function.

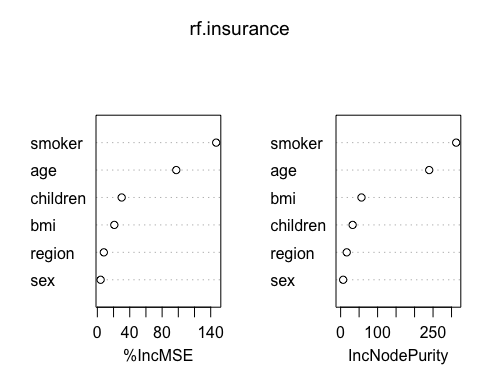
importance(rf.insurance)

## %IncMSE IncNodePurity  
## age 97.450967 239.563590  
## sex 4.043290 6.843952  
## bmi 20.931746 56.438873  
## children 30.047757 32.516463  
## smoker 146.762558 312.573128  
## region 8.013693 16.529764

### 4d. Plot the variable importance using the function, varImpPlot(). Which are the top 3 important predictors in this model?

The top 2 important predictors in this model is **smoker** and **age**, and the third important predictor is between **children** and **bmi** depending on the graphs. This plot ranks the usefulness of the variables. This means that smoker and age are the variables that will give the prediction and contribute most to the model. **Children** may be an important variable because if a policy holder has more children, the charges would be higher. On the other hand, **bmi** may be an important variable because a policy holder’s health background may affect the charges.

varImpPlot(rf.insurance)



# Q5 Build a support vector machine model.

### 5a. The response is charges and the predictors are age, sex, bmi, children, smoker, and region. Please use the svm() function with radial kernel and gamma=5 and cost = 50.

library(e1071)  
svm.fit = svm(charges ~ age+sex+bmi+children+smoker+region, data=insurance[train\_idx, ], kernel="radial", gamma=5, cost=50)  
summary(svm.fit)

##   
## Call:  
## svm(formula = charges ~ age + sex + bmi + children + smoker + region,   
## data = insurance[train\_idx, ], kernel = "radial", gamma = 5,   
## cost = 50)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 50   
## gamma: 5   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 715

### 5b. Perform a grid search to find the best model with potential cost: 1, 10, 50, 100 and potential gamma: 1,3 and 5 and potential kernel: “linear”,“radial” and “sigmoid”. And use the training set created in step 1e.

tune.out = tune(svm, charges ~ age+sex+bmi+children+smoker+region, data=insurance[train\_idx, ], ranges=list(cost=c(1,10,50,100), gamma=c(1,3,5), kernel=c("linear", "radial", "sigmoid")))

### 5c. Print out the model results. What are the best model parameters?

The best model parameters are **kernel=radial, cost=1, and gamma=1**.

summary(tune.out)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma kernel  
## 1 1 radial  
##   
## - best performance: 0.1906752

### 5d. Forecast charges using the test dataset and the best model found in 5c.

pred = predict(tune.out$best.model, newdata=insurance[-train\_idx, ])

### 5e. Compute the MSE (Mean Squared Error) on the test data.

The MSE for the SVM model is **0.2259622**.

trueobservation = insurance[-train\_idx, "charges"]

MSE\_5e = mean((trueobservation - pred)^2)  
MSE\_5e

## [1] 0.2259622

# Q6 Perform the k-means cluster analysis.

### 6a. Remove the sex, smoker, and region, since they are not numerical values.

insurance = insurance[, c(-2,-5,-6)]  
colnames(insurance)

## [1] "age" "bmi" "children" "charges"

### 6b. Determine the optimal number of clusters. Justify your answer. It may take longer running time since it uses a large dataset.

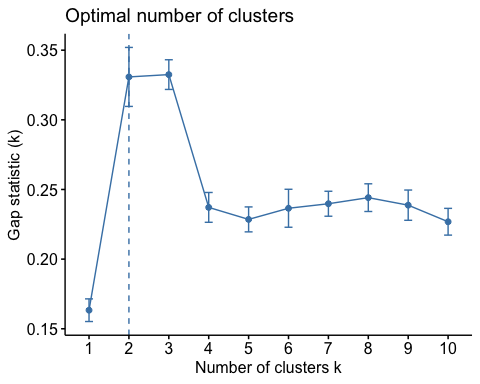
The optimal number of clusters is 3. This means that when k=3, this is considered a good cluster for which the within-cluster variation is small as possible.

library("cluster")  
library("factoextra")

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

set.seed(101)

fviz\_nbclust(insurance, kmeans, method = "gap\_stat")



### 6c. Perform k-means clustering using the 3 clusters.

km.res <- kmeans(insurance, 3, nstart = 25)  
km.res

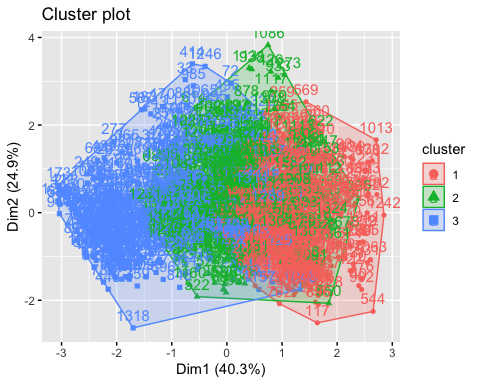
## K-means clustering with 3 clusters of sizes 442, 425, 471  
##   
## Cluster means:  
## age bmi children charges  
## 1 55.54977 31.72505 0.9819005 9.610162  
## 2 39.70353 30.41379 1.4588235 9.202735  
## 3 23.42251 29.89235 0.8726115 8.524738  
##   
## Clustering vector:  
## [1] 3 3 3 2 2 3 2 2 2 1 3 1 3 1 3 3 1 3 1 3 1 3 3 2 2 1 1 1 3 3 3 3 3 1 3 3 1  
## Within cluster sum of squares by cluster:  
## [1] 26777.03 25462.33 27807.97  
## (between\_SS / total\_SS = 74.7 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

### 

### 6d. Visualize the clusters in different colors.

Using fviz\_cluster() can be good to use to visualize our results and assess the choice of the number of clusters as well as comparing two different cluster analyses. fviz\_cluster() takes the k-means results and the original data as arguments. In the resulting plot below the observations are represented by points using principal components to find a low-dimensional representation of the observations that explains a good fraction of the variance. For the insurance data, we clustered into 3 sub groups that visualizes all 4 columns of data observations. The sum of squares summary looks at the the deviations of each point around its cluster mean. For our percentage from between sum of squares divided by total sum of squares it is 74.7 %. This is a percent variance that explains the cluster means. It is pretty high which means that the k we chose, which is meant to be the most optimal is doing a good job.

fviz\_cluster(km.res, data = insurance)



# Q7 Build a neural networks model.

### 7a. Remove the sex, smoker, and region, since they are not numerical values.

Occurred in question 6b.

### 7b. Standardize the inputs using the scale() function.

scaled.insurance = scale(insurance[,-5])  
scaled.insurance

## attr(,"scaled:center")  
## age bmi children charges   
## 39.207025 30.663397 1.094918 9.098659   
## attr(,"scaled:scale")  
## age bmi children charges   
## 14.0499604 6.0981869 1.2054927 0.9195271

### 7c. Convert the standardized inputs to a data frame using the as.data.frame() function.

scaled.insurance = as.data.frame(scaled.insurance)

### 7d. Split the dataset into a training set containing 80% of the original data and the test set containing the remaining 20%.

set.seed(101)  
index <- sample(1:nrow(scaled.insurance), 0.80\*nrow(scaled.insurance))  
train <- scaled.insurance[index, ]  
test <- scaled.insurance[-index, ]

### 7e. The response is charges and the predictors are age, bmi, and children. Please use 1 hidden layer with 1 neuron.

library(neuralnet)  
nn.model <- neuralnet(charges ~ age+bmi+children, data=train, hidden = 1)

### 7f. Plot the neural networks.

plot(nn.model)

### 7g. Forecast the charges in the test dataset.

predict.nn = compute(nn.model, test[, c("age", "bmi", "children")])

### 7h. Get the observed charges of the test dataset.

observ.test = test$charges

### 7i. Compute test error (MSE).

The MSE using the neural network model is **0.6754152**.

mean((observ.test - predict.nn$net.result)^2)

## [1] 0.6754152

# Q8 Putting it all together.

### 8a. For predicting insurance charges, your supervisor asks you to choose the best model among the multiple regression, regression tree, random forest, support vector machine, and neural network models. Compare the test MSEs of the models generated in steps 2.g, 3.f, 4.b, 5.e, and 7.i. Display the names for these types of these models, using these labels: Multiple Linear Regression, Regression Tree, Random Forest, Support Vector Machine, and Neural Network and their corresponding test MSEs in a data.frame. Label the column in your data frame with the labels as Model.Type, and label the column with the test MSEs as Test.MSE and round the data in this column to 4 decimal places. Present the formatted data to your supervisor and recommend which model is best and why.

|  |  |
| --- | --- |
| Model.Type | Test.MSE |
| Multiple Linear Regression | 0.2348 |
| Regression Tree | 0.2020 |
| Random Forest | 0.1780 |
| Support Vector Machine | 0.2260 |
| Neural Network | 0.6754 |

For the insurance data set, the best model to use is the **Random Forest Model** with a MSE of **0.1780**. Our goal is to find the model with the smallest MSE. The smallest MSE will represent that the predicted responses are very close to the true responses. We calculate based on MSE because it takes into consideration of the bias-variance trade off. The random forest model is the best because while achieving a MSE closer to zero, it simultaneously has low variance and low bias which are both non-negative.

### 

### 8b. Another supervisor from the sales department has requested your help to create a predictive model that his sales representatives can use to explain to clients what the potential costs could be for different kinds of customers, and they need an easy and visual way of explaining it. What model would you recommend, and what are the benefits and disadvantages of your recommended model compared to other models?

The model I would recommend is using the **Random Forest Model**. The *benefits* of using the random forest model are: 1. Random Forest can be used to solve both classification and regression problems. 2. Random Forest works well with both categorical and continuous variables. 3. Random Forest creates many trees on the subset of the data & combines the output of all the trees, which means it reduces the overfitting problem which reduces the variance and improves accuracy. 4. The model output using the varImpPlot() function is a great visual when talking to potential clients and how each variable may affect their charges.

The *disadvantages* of using the random forest model are: 1. Random Forest creates a lot of trees which means it will use a lot of computational power and resources. The complexity of the model is not great. 2. Random Forest require more time to train which means a longer running period.

### 

### 8c. The supervisor from the sales department likes your regression tree model. But she says that the sales people say the numbers in it are way too low and suggests that maybe the numbers on the leaf nodes predicting charges are log transformations of the actual charges. You realize that in step 1.b of this project that you had indeed transformed charges using the log function. And now you realize that you need to reverse the transformation in your final output. The solution you have is to reverse the log transformation of the variables in the regression tree model you created and redisplay the result.

### Follow these steps:

### a. Copy your pruned tree model to a new variable.

copy\_of\_my\_pruned\_tree = prune.tree(tree.insurance, best=6)

### b. In your new variable, find the data.frame named “frame” and reverse the log transformation on the data.frame column yval using the exp() function.

copy\_of\_my\_pruned\_tree$frame$yval <- exp(copy\_of\_my\_pruned\_tree$frame$yval)

### c. After you reverse the log transform on the yval column, then replot the tree with labels.

plot(copy\_of\_my\_pruned\_tree)  
text(copy\_of\_my\_pruned\_tree, pretty=0)

