```
##############################################
#                                            #
# Author:      Calvin Tran                   #
# Date:        08/17/23                      #
# Subject:     Final Project                 #
# Class:       DSCI 512                      #
# Section:     01W                           #
# Instructor:  Nengbing Tao                  #
# File Name:   FinalProject_Tran_Calvin.R    #
#                                            #
##############################################
```

1. Data Preparation

1a. “Load the dataset insurance.csv into memory.”

```
> df_insurance <- read.csv("C:/Users/Cal/Desktop/DSCI 512 Predictive Modelling/Final Project/insurance.csv")
> summary(df_insurance)
      age            sex                 bmi           children        smoker              region             charges
 Min.   :18.00   Length:1338        Min.   :15.96   Min.   :0.000   Length:1338        Length:1338        Min.   : 1
 1st Qu.:27.00   Class :character   1st Qu.:26.30   1st Qu.:0.000   Class :character   Class :character   1st Qu.: 4
 Median :39.00   Mode  :character   Median :30.40   Median :1.000   Mode  :character   Mode  :character   Median : 9
 Mean   :39.21                      Mean   :30.66   Mean   :1.095                                         Mean   :13
 3rd Qu.:51.00                      3rd Qu.:34.69   3rd Qu.:2.000                                         3rd Qu.:16
 Max.   :64.00                      Max.   :53.13   Max.   :5.000                                         Max.   :63
```

1b. “In the data frame, transform the variable charges by setting insurance$charges = log(insurance$charges). Do not transform it outside of the data frame”

```
> df_insurance$charges <- log(df_insurance$charges)
> df_insurance$charges
   [1]  9.734176  7.453302  8.400538  9.998092  8.260197  8.231275  9.016827  8.893093  8.765054 10.272397  7.908873 10.233105  7.510345  9.313864 10.586881  7.51
  [17]  9.287055  7.781210  9.268834 10.514271  9.490155  8.330800  7.036158 10.537465  8.732934  9.546894  9.578577  9.414801  7.928475 10.563879 10.479696  7.69
```

```
> class(df_insurance$charges)
[1] "numeric"
```

1c “Using the data set from 1.b, 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.”

```
> head(df_insurance)
  age    sex    bmi children smoker    region  charges
1  19 female 27.900        0    yes southwest 9.734176
2  18   male 33.770        1     no southeast 7.453302
3  28   male 33.000        3     no southeast 8.400538
4  33   male 22.705        0     no northwest 9.998092
5  32   male 28.880        0     no northwest 8.260197
6  31 female 25.740        0     no southeast 8.231275
> model_insurance <- model.matrix(charges ~ age + sex + bmi + children + smoker + region, data=df_insura
> head(model_insurance) #first column is all ones
  (Intercept) age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthwest
1           1  19       0 27.900        0         1               0               0               1
2           1  18       1 33.770        1         0               0               1               0
3           1  28       1 33.000        3         0               0               1               0
4           1  33       1 22.705        0         0               1               0               0
5           1  32       1 28.880        0         0               1               0               0
6           1  31       0 25.740        0         0               0               1               0
```

- First column is all 1’s

Drop first column

```
> model_insurance <- subset(model_insurance[,-c(1)]) #drop first column by indexing for all other colu
> head(model_insurance) #intercept column of all ones is successfully dropped
  age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthwest
1  19       0 27.900        0         1               0               0               1
2  18       1 33.770        1         0               0               1               0
3  28       1 33.000        3         0               0               1               0
4  33       1 22.705        0         0               1               0               0
5  32       1 28.880        0         0               1               0               0
6  31       0 25.740        0         0               0               1               0
```

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_insurance <- sample(1:nrow(df_insurance), (2*nrow(df_insurance)/3))
```

- check

```
> nrow(df_insurance) #1338 observations
[1] 1338
> length(train_insurance) #892 is 2/3 * 1338, good
[1] 892
```

1e. “Create a training and test data set from the data set created in 1.b using the training and test row indexes created in 1.d. Unless otherwise stated, only use the training and test data sets created in this step.”

```
> df.train_ins <- df_insurance[train_insurance, ]
> df.test_ins <- df_insurance[-train_insurance,]
```

- check

```
> nrow(df.train_ins)
[1] 892
> nrow(df.test_ins)
[1] 446
```

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

- 1c was where I created the model matrix with dummy variables

```
> modelins_train <- model_insurance[train_insurance, ]
> modelins_test <- model_insurance[-train_insurance,]
```

2. 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 data set created in step 1.e to train your model.”**

```
> lm1_ins <- lm(charges ~ age + sex + bmi + children + smoker + region, data = df.train_i
> summary(lm1_ins)

Call:
lm(formula = charges ~ age + sex + bmi + children + smoker + 
    region, data = df.train_ins)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.20936 -0.20274 -0.05386  0.07979  2.08108 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)      7.048746   0.089013  79.188  < 2e-16 ***
age              0.032178   0.001089  29.542  < 2e-16 ***
sexmale         -0.061703   0.030337  -2.034  0.04226 *  
bmi              0.015680   0.002607   6.015 2.64e-09 ***
children         0.095536   0.012434   7.683 4.12e-14 ***
smokeryes        1.505113   0.037106  40.562  < 2e-16 ***
regionnorthwest -0.033518   0.043008  -0.779  0.43598    
regionsoutheast -0.130820   0.042919  -3.048  0.00237 ** 
regionsouthwest -0.092856   0.043673  -2.126  0.03377 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4494 on 883 degrees of freedom
Multiple R-squared:  0.751,	Adjusted R-squared:  0.7487 
F-statistic: 332.9 on 8 and 883 DF,  p-value: < 2.2e-16
```

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

- refer to p-value, "p-value: < 2.2e-16" which is much less than the threshold of 0.05.
- #there is a statistically relevant relationship between the predictors and response.

2c. “Does **sex** have a statistically significant relationship to the response?”:

```
sexmale         -0.061703   0.030337  -2.034  0.04226 *
```

- refer to p value for 'sexmale', p = 0.04226, which is still below the threshold of 0.05 and is statistically significant

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"

```
> library(MASS)
> lm1_ins.bwd <- stepAIC(lm1_ins, direction = 'backward')
Start:  AIC=-1418.15
charges ~ age + sex + bmi + children + smoker + region

           Df Sum of Sq    RSS      AIC
<none>                  178.29 -1418.15
- sex       1      0.84 179.13 -1415.99
- region    3      2.21 180.50 -1413.18
- bmi       1      7.30 185.60 -1384.34
- children  1     11.92 190.21 -1362.43
- age       1    176.22 354.51  -807.07
- smoker    1    332.21 510.51  -481.79
> lm1_ins.bwd

Call:
lm(formula = charges ~ age + sex + bmi + children + smoker +
    region, data = df.train_ins)

Coefficients:
    (Intercept)              age          sexmale              bmi         children           smoker
        7.04875          0.03218         -0.06170          0.01568          0.09554             1.50
regionnorthwest  regionsoutheast  regionsouthwest
       -0.03352         -0.13082         -0.09286
```

- this is all 6 predictors, the best subset is to use all 6.

2e. "Compute the test error of the best model in #3d based on AIC using LOOCV using trainControl() and train() from the caret library. Report the MSE by squaring the reported RMSE."

```
> train_control =trainControl(method='LOOCV')
> cv.lm1_ins <- train(charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, trControl=train_cont
method='lm')
> summary(cv.lm1_ins)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.20936 -0.20274 -0.05386  0.07979  2.08108

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      7.048746   0.089013  79.188  < 2e-16 ***
age              0.032178   0.001089  29.542  < 2e-16 ***
sexmale         -0.061703   0.030337  -2.034  0.04226 *
bmi              0.015680   0.002607   6.015 2.64e-09 ***
children         0.095536   0.012434   7.683 4.12e-14 ***
smokeryes        1.505113   0.037106  40.562  < 2e-16 ***
regionnorthwest -0.033518   0.043008  -0.779  0.43598
regionsoutheast -0.130820   0.042919  -3.048  0.00237 **
regionsouthwest -0.092856   0.043673  -2.126  0.03377 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4494 on 883 degrees of freedom
Multiple R-squared:  0.751,     Adjusted R-squared:  0.7487
F-statistic: 332.9 on 8 and 883 DF,  p-value: < 2.2e-16
```

- RMSE

```
> #str(cv.lm1_ins)
> cv.lm1_ins$results$RMSE
[1] 0.4517924
```

- Square RMSE to get MSE

```
> #get MSE by squaring RMSE
> (cv.lm1_ins$results$RMSE)^2
[1] 0.2041164
```

  - The MSE is 0.2041164

2f. “Calculate the test error of the best model in #3d based on AIC using 10-fold Cross-Validation. Use train and trainControl from the caret library. Refer to model selected in #3d based on AIC. Report the MSE.”

```
> train_control.10f = trainControl(method='cv', number=10)
> #use with model selected by AIC
> cv.10f_lm1_ins <- train(charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, trControl=train_control.10f, method=
> summary(cv.10f_lm1_ins)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.20936 -0.20274 -0.05386  0.07979  2.08108 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)      7.048746   0.089013  79.188  < 2e-16 ***
age              0.032178   0.001089  29.542  < 2e-16 ***
sexmale         -0.061703   0.030337  -2.034  0.04226 *  
bmi              0.015680   0.002607   6.015 2.64e-09 ***
children         0.095536   0.012434   7.683 4.12e-14 ***
smokeryes        1.505113   0.037106  40.562  < 2e-16 ***
regionnorthwest -0.033518   0.043008  -0.779  0.43598    
regionsoutheast -0.130820   0.042919  -3.048  0.00237 ** 
regionsouthwest -0.092856   0.043673  -2.126  0.03377 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4494 on 883 degrees of freedom
Multiple R-squared:  0.751,	Adjusted R-squared:  0.7487 
F-statistic: 332.9 on 8 and 883 DF,  p-value: < 2.2e-16

> cv.10f_lm1_ins
Linear Regression 

892 samples
  6 predictor

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 800, 803, 804, 804, 802, 804, ... 
Resampling results:

  RMSE       Rsquared   MAE     
  0.4486556  0.7503984  0.288448

Tuning parameter 'intercept' was held constant at a value of TRUE
```

- Report MSE

```
  RMSE       Rsquared   MAE     
  0.4486556  0.7503984  0.288448

Tuning parameter 'intercept' was held constant at a value of TRUE
> (cv.10f_lm1_ins$results$RMSE)^2
[1] 0.2012918
```

- MSE is 0.2012918, which is (0.4486556)^2

2f. “Calculate and report the test MSE using the best model from 2.d and the test data set from step 1.e.”

```
> #same models but test set this time
> truepreds.ins <- df.test_ins[,'charges']
> predictions.ins <- predict(lm1_ins.bwd, newdata=df.test_ins)
```

- calculate MSE

```
> #truepreds.ins #view
> #predictions.ins #view
> #predictions.ins <- as.vector(predictions.ins) #vectorize to match typ
> mean((truepreds.ins - predictions.ins)^2) #Calculate MSE, mean of (tru
[1] 0.1944269
> #MSE is 0.1944269
```

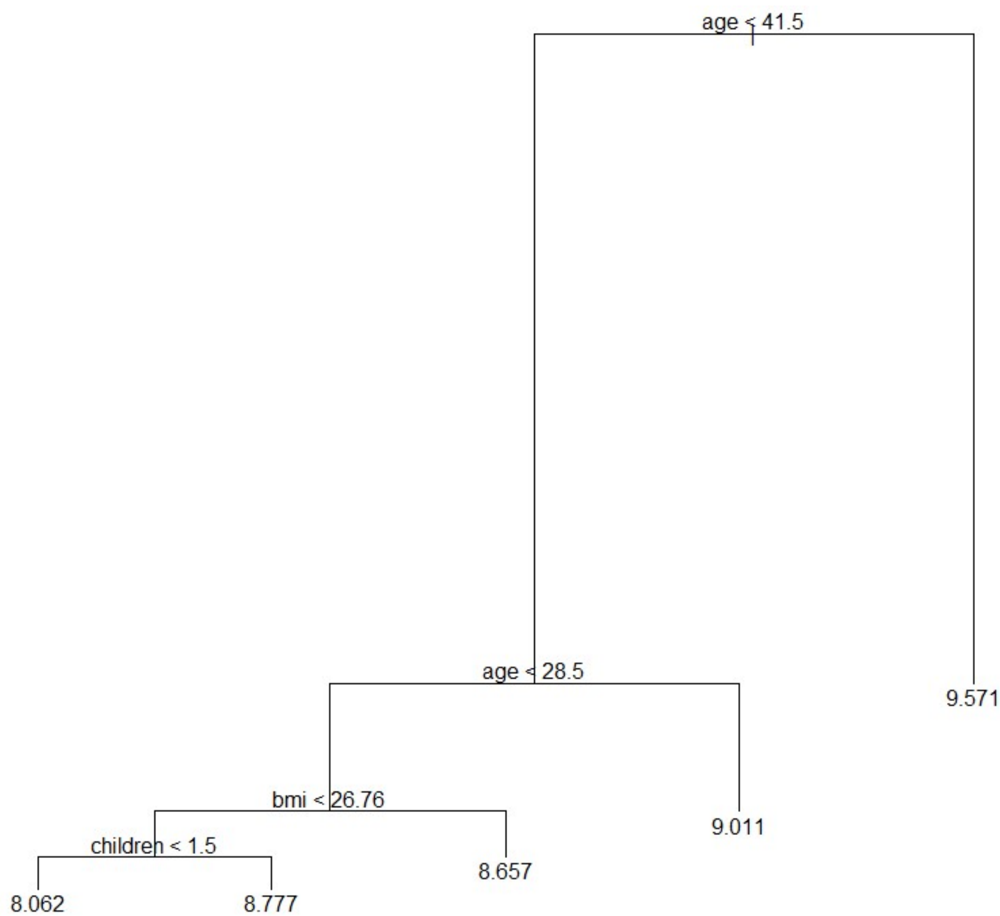
- MSE is 0.1955269

```
> #2f MSE = 0.2012918
> #2g MSE = 0.1944269
> #these are quite close
> #2f MSE = 0.2012918
> #2g MSE = 0.1944269
> #these are quite close, difference of 0.0068
```

3. Build a regression tree model

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.”

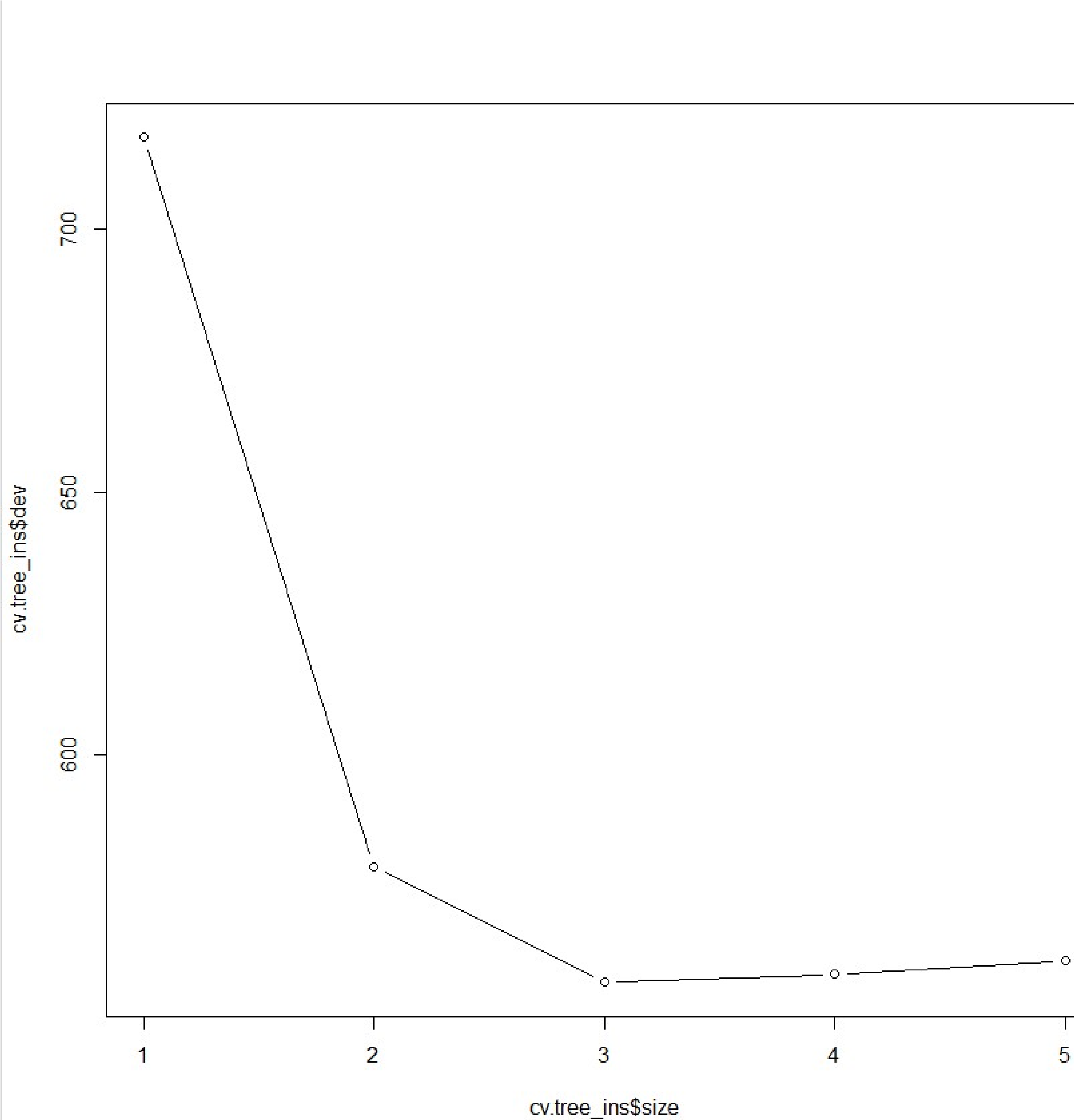
```
> library(tree)
> tree.ins <- tree(charges ~ age + sex + bmi + children + smoker + region, df.train_i
Warning message:
In tree(charges ~ age + sex + bmi + children + smoker + region,   :
  NAs introduced by coercion
> plot(tree.ins)
> text(tree.ins, pretty=0)
```



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

```
> cv.tree_ins <- cv.tree(tree.ins)
There were 32 warnings (use warnings() to see them)
> summary(cv.tree_ins)
       Length Class  Mode
size   5      -none- numeric
dev    5      -none- numeric
k      5      -none- numeric
method 1      -none- character
> plot(cv.tree_ins$size, cv.tree_ins$dev, type='b')
```

Visualize



- The best tree size is 3. It has the lowest deviance & doesn't have too many layers.

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

- Limiting the size of the tree (eg, picking size 2 instead of size 3) reduces the complexity of the tree even in a case where the model fits the training data better,

- we want the model to be able to generalize and be predictive of new data. We don't want the variance too high.

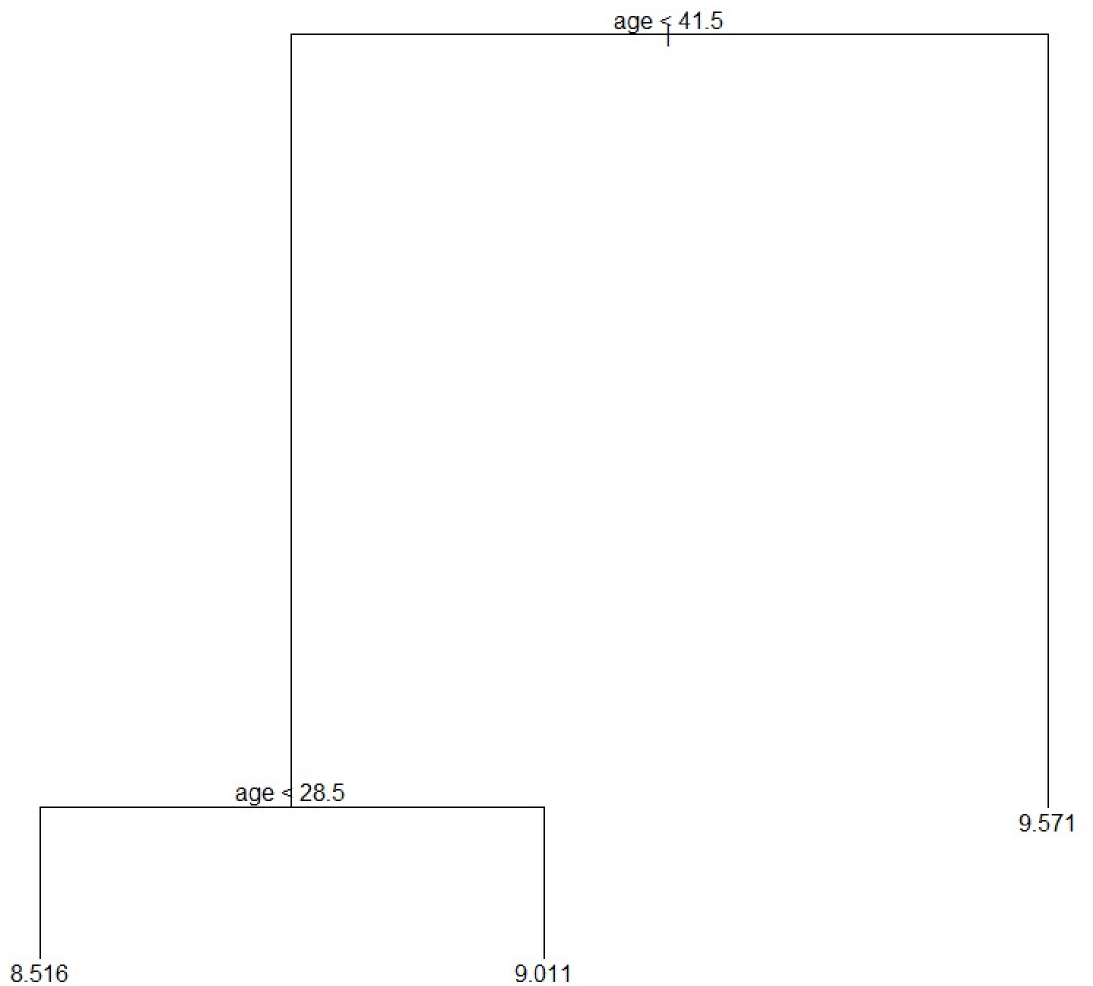
- However if we allow the bias to be too high, and pick only tree size 1 for example, the model will fail to produce accurate results for any data.

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

```
> prune.treeins <- prune.tree(tree.ins, best=3)
```

3e. “Plot the best tree model and give labels.”

```
> plot(prune.treeins)
> text(prune.treeins, pretty=0)
```



3f. “Calculate the test MSE for the best model.”

```
> #truepreds.ins <- df.test_ins[,'charges'] #this is from a previous question, the true values
the same as before
```

```
> pred.besttree_ins <-predict(prune.treeins, newdata = df.test_ins)
Warning message:
In pred1.tree(object, tree.matrix(newdata)) : NAs introduced by coercion
> summary(pred.besttree_ins)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  8.516   8.516   9.011   9.118   9.571   9.571
> mean((truepreds.ins - pred.besttree_ins)^2)
[1] 0.6727747
```

- the MSE is 0.6727747, which is frankly not very good but trees aren't great for regression anyway

4. 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.”

```
> library(randomForest)
> rf.ins <- randomForest(charges~age + sex + bmi + children + smoker + region, data=df.train_ir
mportance=TRUE)
> summary(rf.ins)
                Length Class  Mode
call              4    -none- call
type              1    -none- character
predicted       892    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times       892    -none- numeric
importance       12    -none- numeric
importanceSD      6    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y               892    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

```
> rf.ins

Call:
 randomForest(formula = charges ~ age + sex + bmi + children +      smoker + region, data = df.
n_ins, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.1569236
                    % Var explained: 80.45
```

4b. “Compute the test error using the test data set.”

```
> pred.rf_ins <- predict(rf.ins, newdata = df.test_ins)
> mean((truepreds.ins - pred.rf_ins)^2)
[1] 0.1519717
```

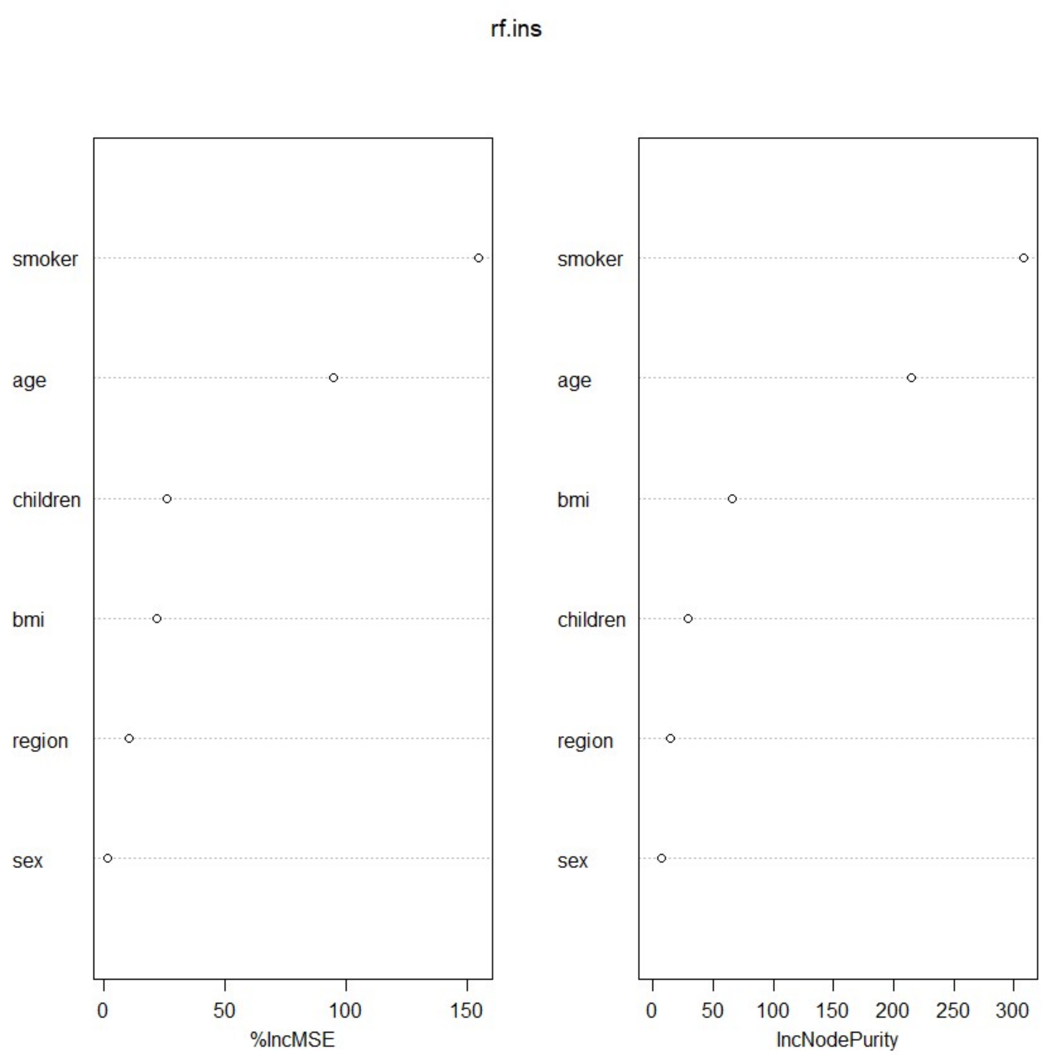
- the Mean Squared Error is 0.1519717

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

```
> importance(rf.ins)
            %IncMSE IncNodePurity
age       94.980778     214.71293
sex        1.836829       6.66082
bmi       21.978981      65.77917
children  26.042879      28.98723
smoker   154.665811     307.58905
region    10.463250      14.79587
```

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

```
> varImpPlot(rf.ins)
```



- The three most (four most) important predictors are smoker, age, and then bmi/children depending on the metric used.

5. 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.ins <- svm(charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, kernel='radial', gamma=5, cost=20)
> svm.ins

Call:
svm(formula = charges ~ age + sex + bmi + children + smoker + region, data = df.train_ins, kernel = "radial", gamma = 5, cost =


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  20
      gamma:  5
    epsilon:  0.1


Number of Support Vectors:  735
```

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", “polynomial",radial " and "sigmoid ". And use the training set created in step 1.e.

```
tune.linear_ins <- tune(svm, charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, kernel='linear', ranges=list(cost=c(1,10,50,100), gamma=c(1,3,5)))
#tune.polynomial_ins <- tune(svm, charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, kernel= 'polynomial', ranges=list(cost=c(1,10,50,100), gamma=c
tune.radial_ins <- tune(svm, charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, kernel='radial', ranges=list(cost=c(1,10,50,100), gamma=c(1,3,5)))
tune.sigmoid_ins <- tune(svm, charges ~ age + sex + bmi + children + smoker + region, data=df.train_ins, kernel='sigmoid', ranges=list(cost=c(1,10,50,100), gamma=c(1,3,5))
```

- Note: the polynomial kernel did not converge in max iterations, it is not a good choice of kernel

Choose model with best performance

```
> summary(tune.linear_ins)$best.performance #0.2218645
[1] 0.2218929
> #summary(tune.polynomial_ins) #did not converge with polynomial kernel
> summary(tune.radial_ins)$best.performance #0.220784
[1] 0.2205502
> summary(tune.sigmoid_ins)$best.performance #2295.375
[1] 2290.769
```

- The best model uses a radial kernel. It has the lowest error.

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

```
> summary(tune.radial_ins)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1     1

- best performance: 0.220784

- Detailed performance results:
   cost gamma     error dispersion
1     1     1 0.2207840 0.03792104
2    10     1 0.2776542 0.05516816
3    50     1 0.3994730 0.08736009
4   100     1 0.4897457 0.11122492
5     1     3 0.3965905 0.05110891
6    10     3 0.4284204 0.07831079
7    50     3 0.5117674 0.18148405
8   100     3 0.5349875 0.24583574
9     1     5 0.4864381 0.06731118
10   10     5 0.4933471 0.07515067
11   50     5 0.5141355 0.11401363
12  100     5 0.5141355 0.11401363
```

```
> summary(tune.radial_ins)$best.model

Call:
best.tune(METHOD = svm, train.x = charges ~ age + sex + bmi + children + smoker + region, data
f.train_ins, ranges = list(cost = c(1, 10, 50, 100), gamma = c(1,
    3, 5)), kernel = "radial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  1
    epsilon:  0.1
```

- The best parameters for this best model are cost = 1 and gamma = 1.

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

```
> preds.svm_ins <- predict(tune.radial_ins$best.model, newdata=df.test_ins
> obs.svm_ins <- df.test_ins$charges
>
> summary(preds.svm_ins)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.298   8.545   9.115   9.020   9.484  10.767
> summary(obs.svm_ins)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.031   8.377   9.085   9.048   9.616  11.044
```

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

```
> mean((obs.svm_ins - preds.svm_ins)^2)
[1] 0.1924279
```

- The MSE is 0.1924279.

6. Perform the K means Cluster Analysis"

6a. Use the training data set created in step 1.f and standardize the inputs using the scale() function.

- View original model matrix

```
> head(modelins_train)
     age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthwes
38    26       1 20.800        0         0               0               0
787   60       1 36.955        0         0               0               0
85    37       0 34.800        2         1               0               0
957   54       1 30.800        1         1               0               1
1110  45       1 20.350        3         0               0               1
270   49       1 25.840        1         0               0               0
```

- scale

```
> scaled.trainins <- scale(modelins_train)
> head(scaled.trainins) #values are between +1/-1, properly scaled
            age    sexmale         bmi    children  smokeryes regionnorthwest regionsoutheast reg
38   -0.9563645  1.0084434 -1.59858513 -0.91354219 -0.5199537      -0.5614983      -0.6150173
787   1.4758890  1.0084434  1.06378226 -0.91354219 -0.5199537      -0.5614983      -0.6150173
85   -0.1694590 -0.9905156  0.70863514  0.73434382  1.9210921      -0.5614983      -0.6150173
957   1.0466678  1.0084434  0.04942935 -0.08959918  1.9210921      -0.5614983       1.6241477
1110  0.4028360  1.0084434 -1.67274578  1.55828683 -0.5199537      -0.5614983       1.6241477
270   0.6889834  1.0084434 -0.76798583 -0.08959918 -0.5199537      -0.5614983      -0.6150173
```

  - Values are between +1/-1, properly scaled

6b. Convert the standardized inputs to a data frame using the as.data.frame() function.

```
> class(modelins_train)
[1] "matrix" "array"
```

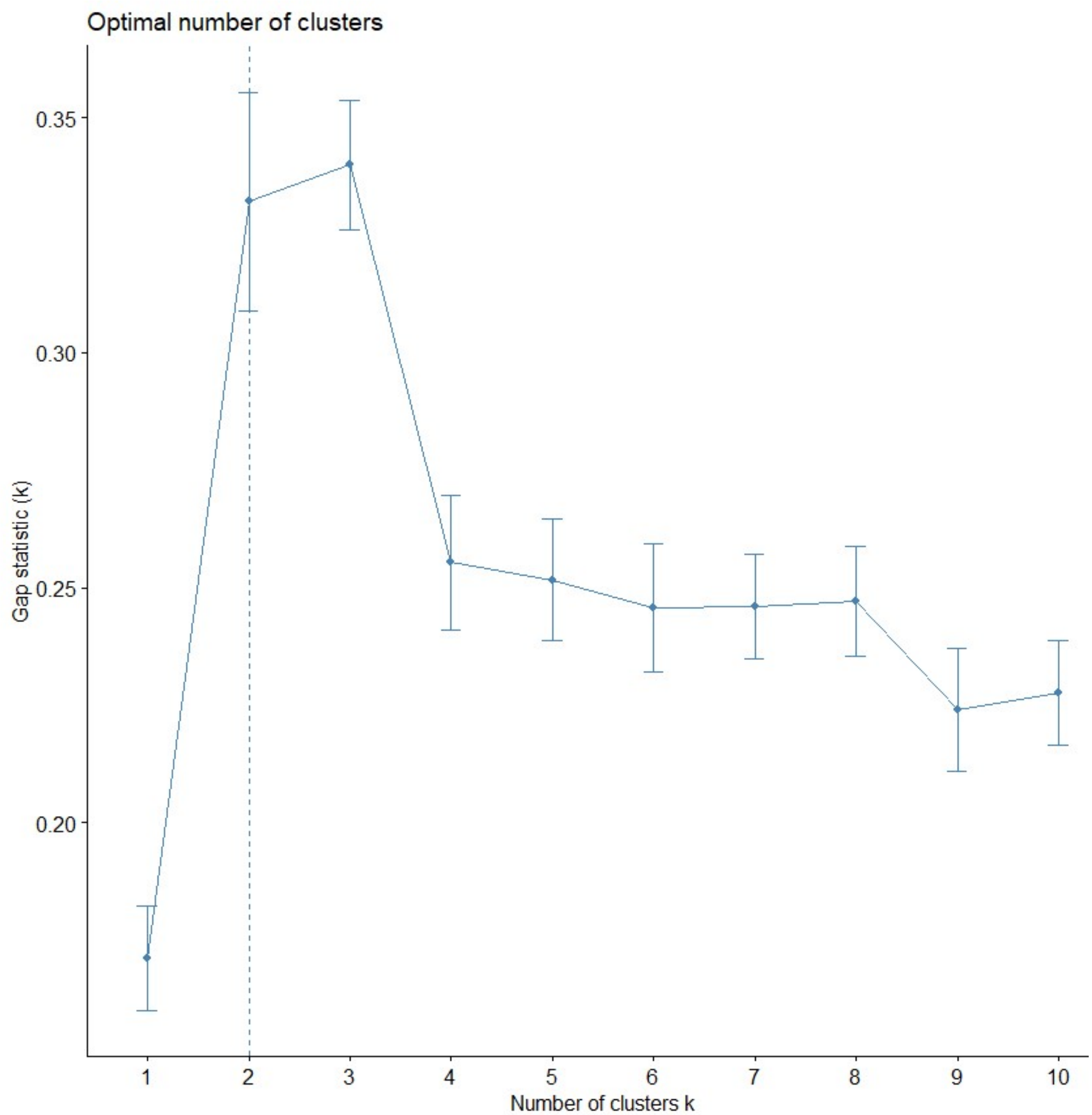
convert

```
> head(df.modelins_train)
     age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthw
38    26       1 20.800        0         0               0               0
787   60       1 36.955        0         0               0               0
85    37       0 34.800        2         1               0               0
957   54       1 30.800        1         1               0               1
1110  45       1 20.350        3         0               0               1
270   49       1 25.840        1         0               0               0
```

```
> class(df.modelins_train)
[1] "data.frame"
```

6c. “Determine the optimal number of clusters, and use the gap_stat method and set iter.max=20. Justify your answer. It may take longer running time since it uses a large dataset.”

```
> fviz_nbclust(df.modelins_train, kmeans, method='gap_stat', iter.max=20)
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 100)  [one "." per sample]:
.................................................. 50
.................................................. 100
>
```



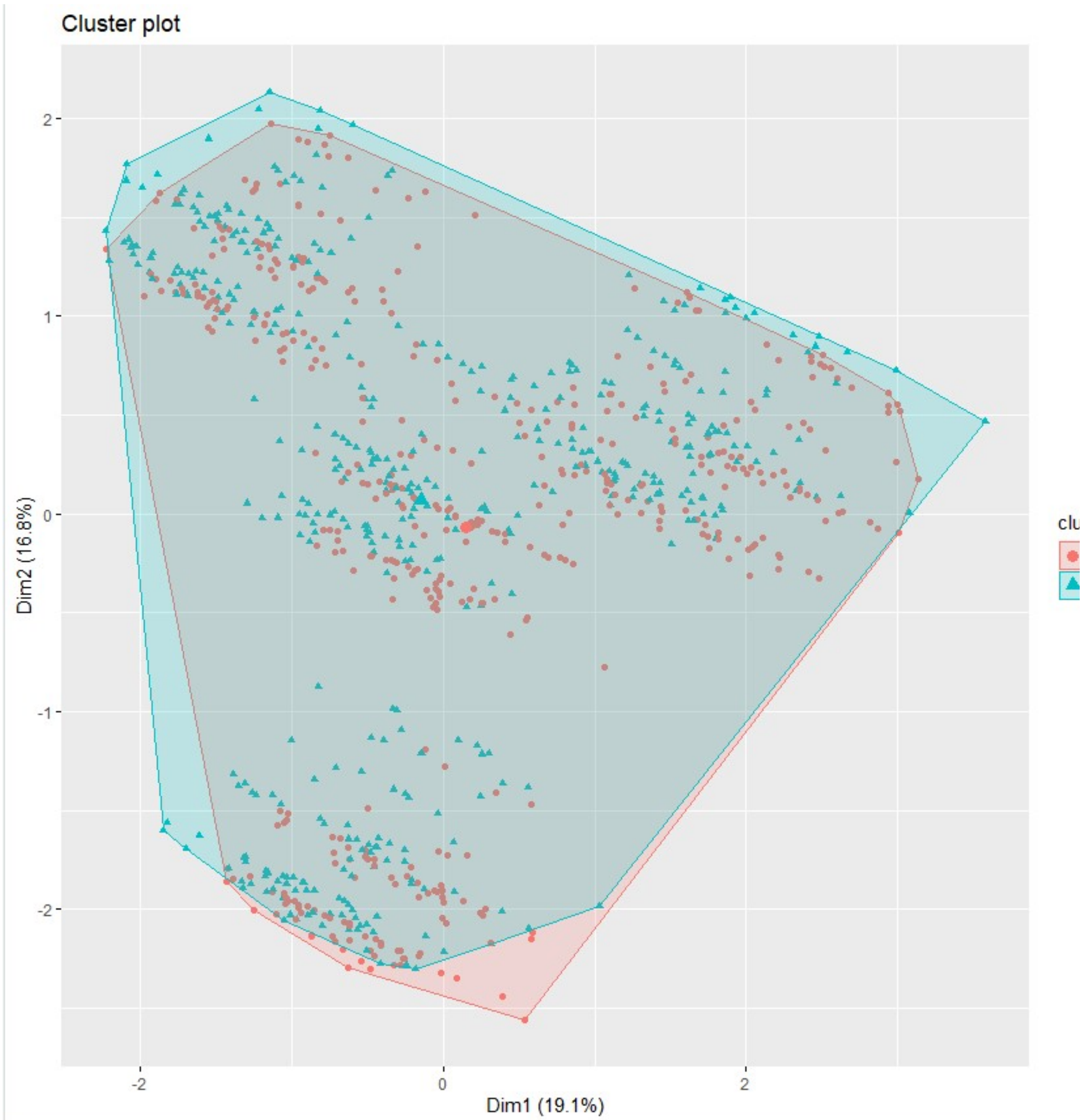
- The optimal number of clusters is 2

6d. Perform k-means clustering using the optimal number of clusters found in step 6.c. Set parameter nstart = 25

```
> km_ins <- kmeans(df.modelins_train, 2, nstart=25)
```

6e. “Visualize the clusters in different colors, setting parameter geom="point"”

```
> fviz_cluster(km_ins, data=df.modelins_train, geom='point')
```



7. “Build a neural networks model”

7a. Using the training data set created in step 1.f, create a neural network model where the response is charges and the predictors are age, sexmale, bmi, children, smokeryes, regionnorthwest, regionsoutheast, and regionsouthwest. Please use 1 hidden layer with 1 neuron. Do not scale the data.

- We dropped charges earlier, regain from original test data

```
> #need charges column from original data as target
> head(df.modelins_train) #charges was dropped earlier, but we want this as a target
     age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthwe
38    26       1 20.800        0         0               0               0
787   60       1 36.955        0         0               0               0
85    37       0 34.800        2         1               0               0
957   54       1 30.800        1         1               0               1
1110  45       1 20.350        3         0               0               1
270   49       1 25.840        1         0               0               0
```

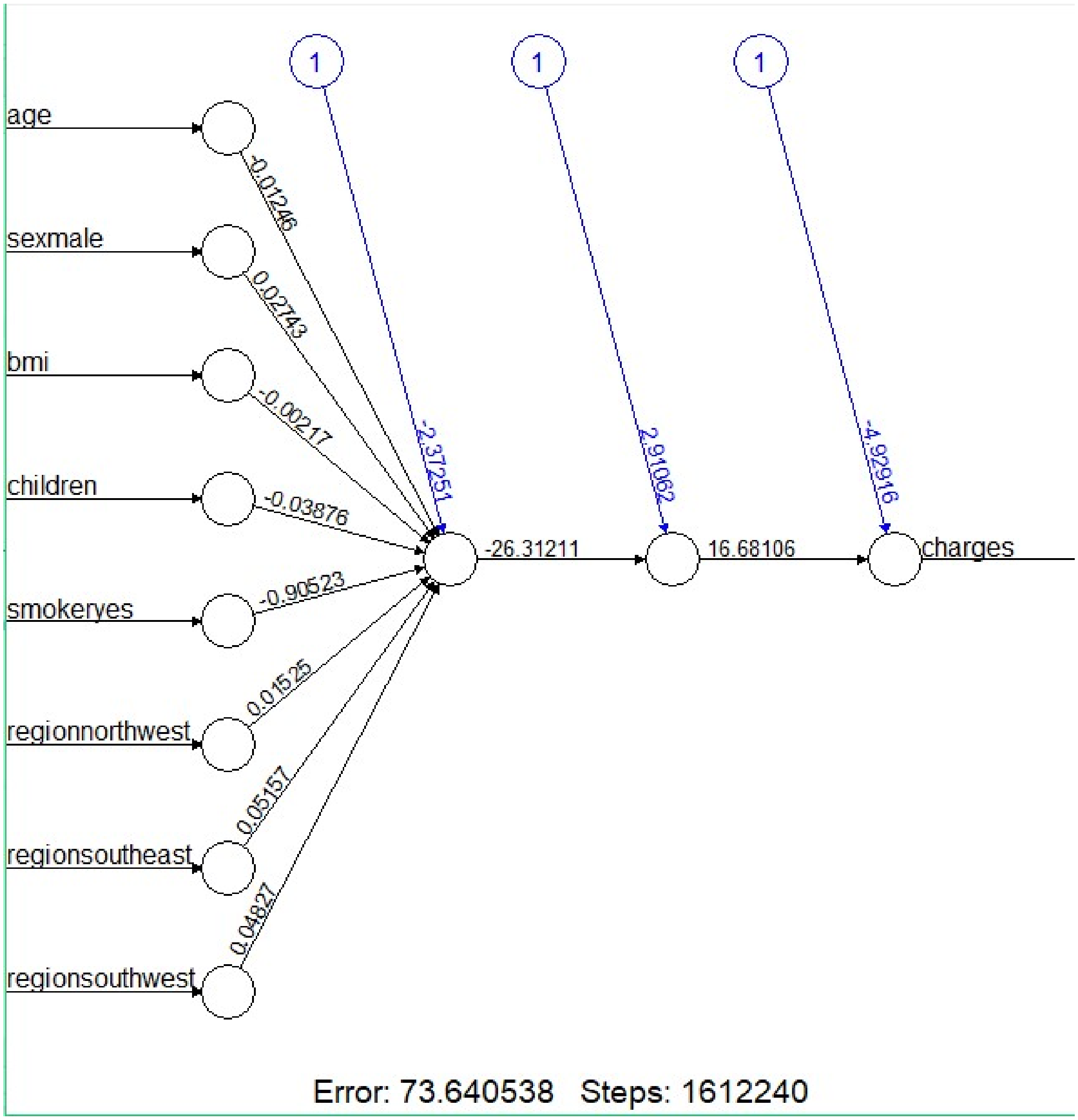
Check that charges are back

```
> df.modelins_train$charges <- df.train_ins$charges
> head(df.modelins_train)
     age sexmale    bmi children smokeryes regionnorthwest regionsoutheast regionsouthwest   cha
s
38    26       1 20.800        0         0               0               0               1  7.74
4
787   60       1 36.955        0         0               0               0               0  9.45
4
85    37       0 34.800        2         1               0               0               1 10.59
9
957   54       1 30.800        1         1               0               1               0 10.64
3
1110  45       1 20.350        3         0               0               1               0  9.06
1
270   49       1 25.840        1         0               0               0               0  9.13
4
```

#Neural Network

```
> nn.ins <- neuralnet(charges ~ age + sexmale + bmi + children + smokeryes + regionnorthwest + r
onsoutheast + regionsouthwest, data=df.modelins_train, hidden=c(1,1), stepmax=1e7)
```

7b. ”Plot the neural network.”



7c.”Forecast the charges in the test dataset”

```
> preds.nn_ins <- compute(nn.ins, df.modelins_test[,c('age', 'sexmale', 'bmi', 'children', 'smokeryes', 'regionnorthwest', 'regionsoutheast', 'regionsouthwe
> obs.nn_ins <- df.test_ins$charges
```

7d.”Compute test error (MSE)”

```
> mean((obs.nn_ins-preds.nn_ins$net.result)^2)
[1] 0.14008
```

- The MSE is 0.14008.

8. 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.d. 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**.

- Saving these values to memory

```
> #2g Multiple linear Regression
> mse.lr <- mean((truepreds.ins - predictions.ins)^2)
> #3f Regression Tree
> mse.tree <- mean((truepreds.ins - pred.besttree_ins)^2)
> #4b Random Forest
> mse.rf <- mean((truepreds.ins - pred.rf_ins)^2)
> #5e Support Vector Machine
> mse.svm <- mean((obs.svm_ins - preds.svm_ins)^2)
> #7d Neural Network
> mse.nn <- mean((obs.nn_ins-preds.nn_ins$net.result)^2)
```

- View values

```
> mse.lr
[1] 0.1944269
> mse.tree
[1] 0.6727747
> mse.rf
[1] 0.1519717
> mse.svm
[1] 0.1924279
> mse.nn
[1] 0.14008
```

- Create dataframe as outlined

```
> selection.df <- data.frame(
+   Model.Type = c( "Multiple Linear Regression", "Regression Tree", "Random Forest",  "Support Vector Machine","Neural Networ
+   Test.MSE = c(mse.lr, mse.tree, mse.rf, mse.svm, mse.nn)
+ )
> selection.df
                  Model.Type  Test.MSE
1 Multiple Linear Regression 0.1944269
2            Regression Tree 0.6727747
3              Random Forest 0.1519717
4     Support Vector Machine 0.1924279
5             Neural Network 0.1400800
> selection.df$Test.MSE <- round(selection.df$Test.MSE, digits = 4)
> selection.df
                  Model.Type Test.MSE
1 Multiple Linear Regression   0.1944
2            Regression Tree   0.6728
3              Random Forest   0.1520
4     Support Vector Machine   0.1924
5             Neural Network   0.1401
```

- This table clearly shows that the Neural Network has the lowest error, and that is the model I recommend to my supervisor.

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 type that he requests, that outlines "kinds" of customers and is easy to read, describes a decision tree.
- The advantage is that it is indeed easy to explain.
- The disadvantages are that they are not very robust and can be prone to error based on setup or variance.
- As shown in this project, the Tree model has the most error out of all the models by a large amount.

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:”

i. “Copy your pruned tree model to a new variable.”

```
> new_tree <- tree.ins
```

ii. “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. (If the copy of your pruned tree model is named copy_of_my_pruned_tree, then the data frame is accessed as copy_of_my_pruned_tree$frame, and it works just like a normal data frame.).”,
find frame of the model

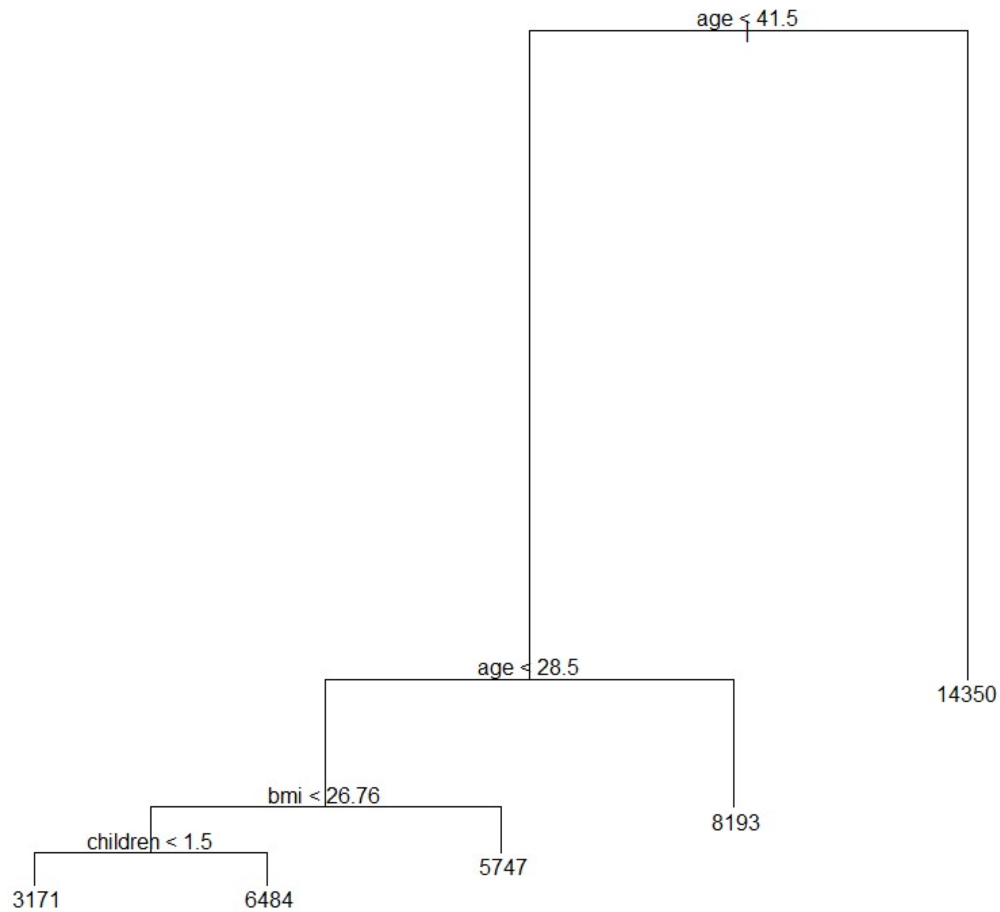
```
> new_tree$frame
        var   n       dev     yval splits.cutleft splits.cutright
1       age 892 716.02595 9.124022          <41.5           >41.5
2       age 484 439.08834 8.746911          <28.5           >28.5
4       bmi 258 302.12262 8.515581         <26.76          >26.76
8  children  84  61.23377 8.223664           <1.5            >1.5
16   <leaf>  65  42.60798 8.061872
17   <leaf>  19  11.10349 8.777161
9    <leaf> 174 230.27508 8.656506
5    <leaf> 226 107.39771 9.010996
3    <leaf> 408 126.45410 9.571380
```

chance ‘yval’ in frame

```
> new_tree$frame$yval
[1] 9.124022 8.746911 8.515581 8.223664 8.061872 8.777161 8.656506 9.010996 9.5713
> new_tree$frame$yval <- exp(new_tree$frame$yval)
> new_tree$frame
        var   n       dev      yval splits.cutleft splits.cutright
1       age 892 716.02595  9173.026          <41.5           >41.5
2       age 484 439.08834  6291.226          <28.5           >28.5
4       bmi 258 302.12262  4991.945         <26.76          >26.76
8  children  84  61.23377  3728.136           <1.5            >1.5
16   <leaf>  65  42.60798  3171.221
17   <leaf>  19  11.10349  6484.441
9    <leaf> 174 230.27508  5747.421
5    <leaf> 226 107.39771  8192.681
3    <leaf> 408 126.45410 14348.202
```

- the values are now much bigger, good

iii “After you reverse the log transform on the yval column, then replot the tree with labels.”


age < 41.5
age < 28.5
14350
bmi < 26.76
8193
children < 1.5
5747
3171
6484

•