DSCI 512 Final Project

Lauren Schmiedeler

2022-12-11

1. Data Preparation

1. Open an rmd and load the `insurance.csv` dataset.

```
library(tidyverse)

# read in insurance.csv
insurance <- read.csv("insurance.csv")
# convert sex, smoker, and region to factors
insurance <- mutate(insurance, sex = factor(sex), smoker = factor(smoker), region = factor(regio
n))

head(insurance)
```

```
##   age    sex    bmi children smoker    region   charges
## 1  19 female 27.900        0    yes southwest 16884.924
## 2  18   male 33.770        1     no southeast  1725.552
## 3  28   male 33.000        3     no southeast  4449.462
## 4  33   male 22.705        0     no northwest 21984.471
## 5  32   male 28.880        0     no northwest  3866.855
## 6  31 female 25.740        0     no southeast  3756.622
```

2. In the data frame, log transform the variable `charges` and name it `log_charges` .

```
insurance$log_charges <- log(insurance$charges)
head(insurance)
```

```
##   age    sex    bmi children smoker    region   charges log_charges
## 1  19 female 27.900        0    yes southwest 16884.924    9.734176
## 2  18   male 33.770        1     no southeast  1725.552    7.453302
## 3  28   male 33.000        3     no southeast  4449.462    8.400538
## 4  33   male 22.705        0     no northwest 21984.471    9.998092
## 5  32   male 28.880        0     no northwest  3866.855    8.260197
## 6  31 female 25.740        0     no southeast  3756.622    8.231275
```

3. Use the `sample ()` function with `set.seed` equal to 1 to generate row indexes for your training and tests sets, with 70% of the row indexes for your training set and 30% for your test set. Do not use any method other than the `sample()` function for splitting your data.

```
set.seed(1)
train_i <- sample(1:nrow(insurance), as.integer(0.7 * nrow(insurance)))
train <- insurance[train_i,]
test <- insurance[-train_i,]
```

2. Build a Multiple Regression Model

1. Perform multiple linear regression with `log_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 you created in 1 above.

```
mod_lm <- lm(log_charges ~ age + sex + bmi + children + smoker + region, data = train)
summary_mod_lm <- summary(mod_lm)
summary_mod_lm
```

```
##
## Call:
## lm(formula = log_charges ~ age + sex + bmi + children + smoker +
##     region, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03524 -0.17125 -0.03964  0.06275  2.11914
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.052139   0.081384  86.653  < 2e-16 ***
## age               0.034948   0.001005  34.785  < 2e-16 ***
## sexmale          -0.057167   0.027939  -2.046 0.041022 *
## bmi               0.011503   0.002394   4.804 1.81e-06 ***
## children          0.099796   0.011625   8.585  < 2e-16 ***
## smokeryes         1.540724   0.034743  44.347  < 2e-16 ***
## regionnorthwest -0.090603   0.039661  -2.284 0.022571 *
## regionsoutheast -0.140780   0.039344  -3.578 0.000364 ***
## regionsouthwest -0.112012   0.040208  -2.786 0.005449 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4246 on 927 degrees of freedom
## Multiple R-squared:  0.7841, Adjusted R-squared:  0.7822
## F-statistic: 420.8 on 8 and 927 DF,  p-value: < 2.2e-16
```

2. Is there a relationship between the predictors and the response? Yes, there is a relationship between the predictors and the response. The R^2 value for this model indicates that almost 80% of the variation in `log_charges` is explained by the predictors.

```
summary_mod_lm$r.squared
```

```
## [1] 0.784103
```

3. Does `sex` have a statistically significant relationship to the response? Yes, based on the p-value associated with the coefficient for `sexmale` (which is about 0.041), we reject the null hypothesis that there is no relationship between `log_charges` and `sex` at a 5% significance level.

```
summary_mod_lm$coefficients
```

```
##                     Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept)      7.05213916 0.081383689 86.652980  0.000000e+00
## age              0.03494761 0.001004671 34.785137 2.617911e-170
## sexmale         -0.05716683 0.027938540 -2.046164  4.102195e-02
## bmi              0.01150287 0.002394271  4.804330  1.810194e-06
## children         0.09979558 0.011624527  8.584915  3.797082e-17
## smokeryes        1.54072378 0.034742529 44.346909 2.308209e-231
## regionnorthwest -0.09060285 0.039660900 -2.284438  2.257110e-02
## regionsoutheast -0.14077985 0.039343737 -3.578202  3.638866e-04
## regionsouthwest -0.11201211 0.040208231 -2.785800  5.448507e-03
```

4. Compute the test error of the model in 2a. Report the RMSE.

```
# create a function that calculates the RMSE
rmse <- function(observed, predicted) {
  sqrt(mean((observed - predicted)**2))
}

predict_mod_lm <- predict(mod_lm, newdata = test)
rmse_mod_lm <- rmse(test$log_charges, predict_mod_lm)
rmse_mod_lm
```

```
## [1] 0.4890933
```

3. Build a Regression Tree Model

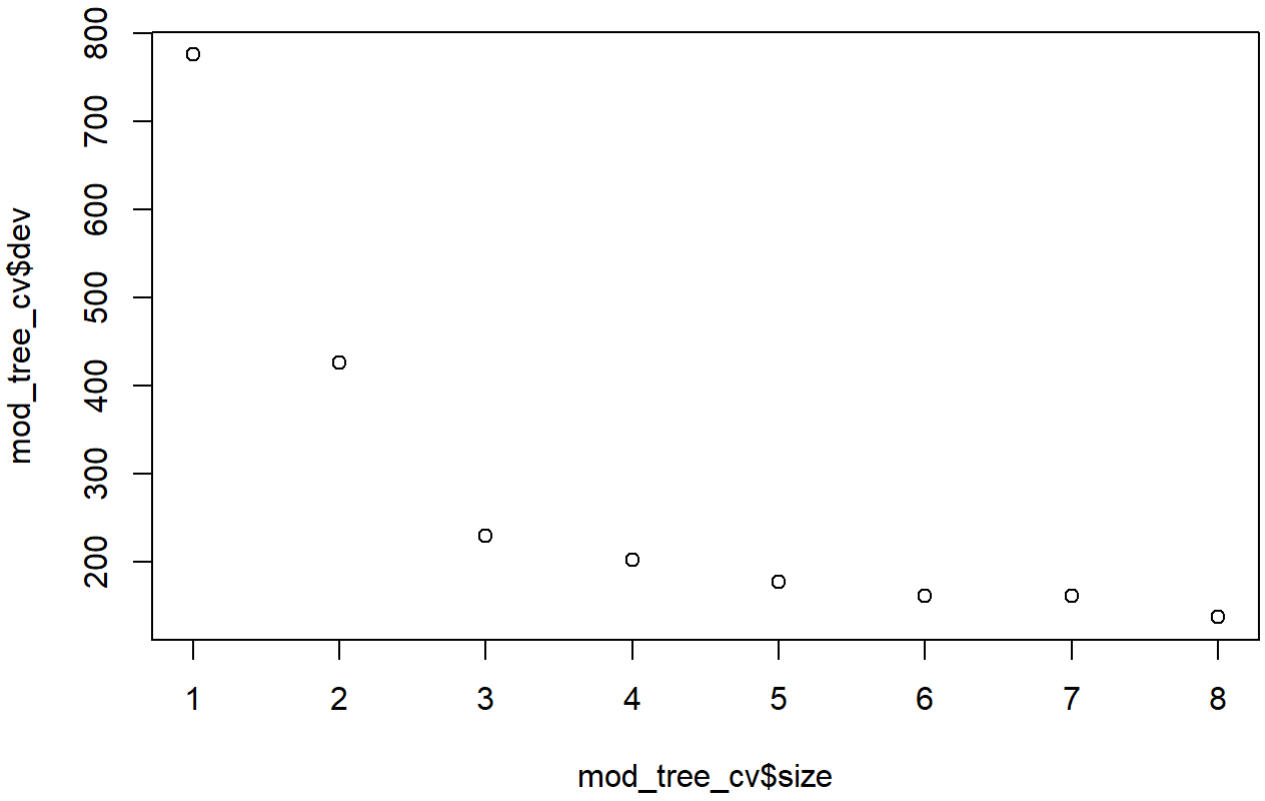
1. Build a regression tree model using function `tree()`, where `log_charges` is the response and the predictors are `age`, `sex`, `bmi`, `children`, `smoker`, and `region`.

```
library(tree)

mod_tree <- tree(log_charges ~ age + sex + bmi + children + smoker + region, data = train)
summary(mod_tree)
```

```
##
## Regression tree:
## tree(formula = log_charges ~ age + sex + bmi + children + smoker +
##     region, data = train)
## Variables actually used in tree construction:
## [1] "smoker"   "age"      "children" "bmi"
## Number of terminal nodes:  8
## Residual mean deviance:  0.136 = 126.2 / 928
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.84430 -0.19100 -0.04892  0.00000  0.08654  2.39900
```

2. Find the optimal tree and display the results in a graphic. Report the best size. The best size is 3.

```
mod_tree_cv <- cv.tree(mod_tree)
plot(mod_tree_cv$size, mod_tree_cv$dev)
```



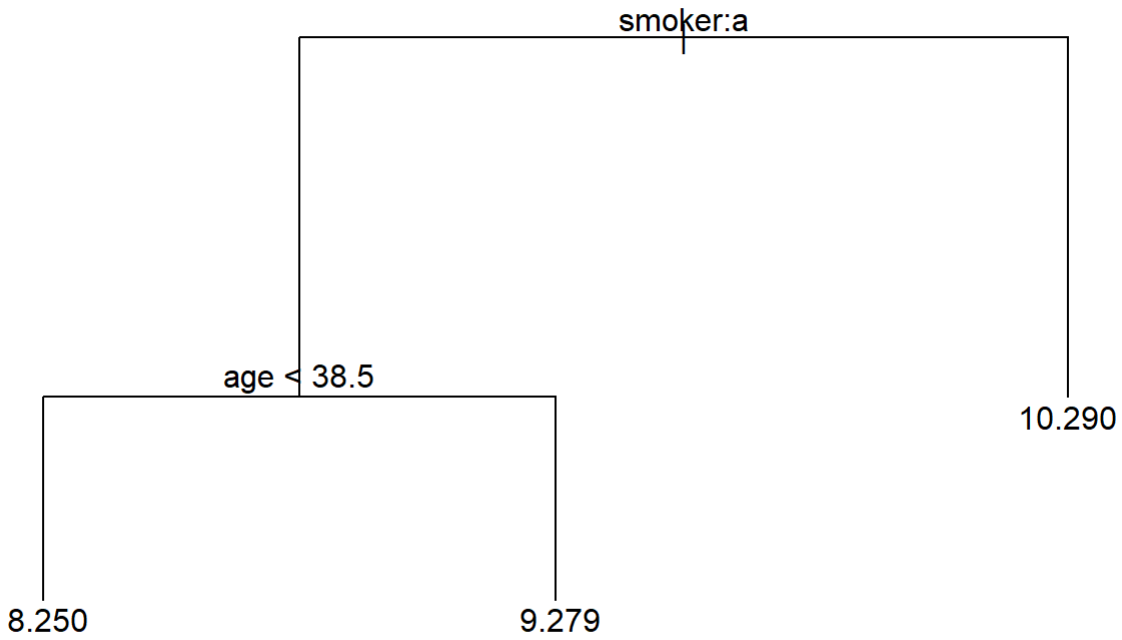
3. Justify the number you picked for the optimal tree with regard to the principle of the variance-bias trade-off. With a more complex model (a tree with a larger size), the variance will increase. With a less complex model (a tree with a smaller size), the bias (or accuracy) will increase.

4. Prune the tree using the optimal size found in 3b.

```
mod_tree_prune <- prune.tree(mod_tree, best = 3)
```

5. Plot the best tree model and give labels.

```
plot(mod_tree_prune)
text(mod_tree_prune)
```



6. Calculate the RMSE for the best model.

```
predict_mod_tree <- predict(mod_tree_prune, newdata = test)
rmse_mod_tree <- rmse(test$log_charges, predict_mod_tree)
rmse_mod_tree
```

```
## [1] 0.5462604
```

4. Build a Random Forest Model

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

```
library(randomForest)

mod_rand_forest <- randomForest(log_charges ~ age + sex + bmi + children + smoker + region, data
= train)
mod_rand_forest
```

```
##
## Call:
##  randomForest(formula = log_charges ~ age + sex + bmi + children +      smoker + region, data
= train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.132413
##                     % Var explained: 83.99
```

2. Compute the test error (using the test data set).

```
predict_mod_rand_forest <- predict(mod_rand_forest, newdata = test)
rmse_mod_rand_forest <- rmse(test$log_charges, predict_mod_rand_forest)
rmse_mod_rand_forest
```

```
## [1] 0.4307213
```

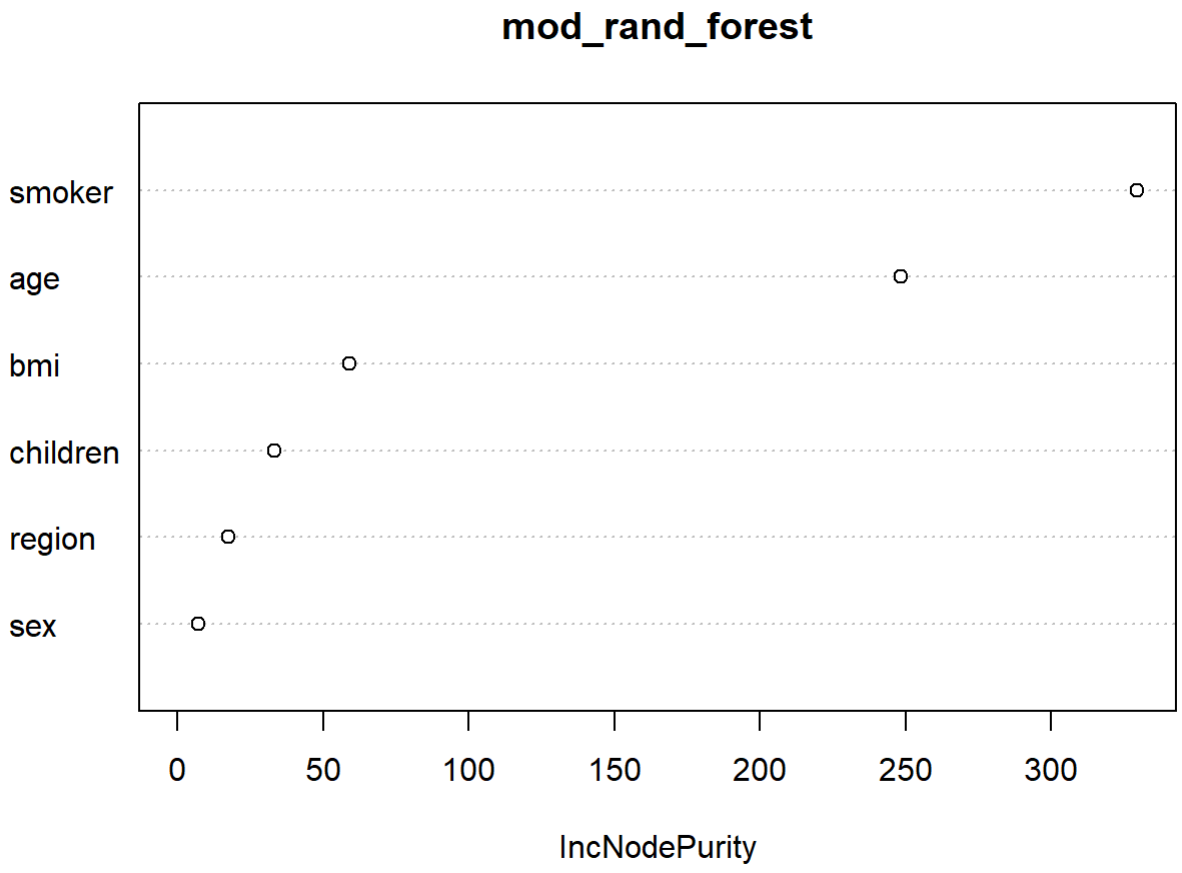
3. Extract variable importance measure using the `importance()` function.

```
importance(mod_rand_forest)
```

```
##          IncNodePurity
## age         248.375183
## sex           7.108785
## bmi          58.815431
## children     33.191026
## smoker      329.652711
## region       17.358646
```

4. Plot the variable importance using the function, `varImpPlot()`. Which are the top 3 important predictors in this model? The top 3 important predictors in this model are `smoker`, `age`, and `bmi`.

```
varImpPlot(mod_rand_forest)
```



5. Build a Support Vector Machine Model

1. 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)

mod_svm <- svm(charges ~ age + sex + bmi + children + smoker + region, data = train, kernel = "r
adial", gamma = 5, cost = 50)
summary(mod_svm)
```

```
##
## Call:
## svm(formula = charges ~ age + sex + bmi + children + smoker + region,
##     data = train, 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:  790
```

2. 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".

```
mod_svm_tune <- tune(svm, charges ~ age + sex + bmi + children + smoker + region, data = train,
ranges = list(kernel = c("linear", "radial", "sigmoid"), gamma = c(1, 3, 5), cost = c(1, 10, 50,
100)))
```

3. Print out the model results. What are the best model parameters? The best model parameters are `kernel` = radial, `gamma` = 1, and `cost` = 1.

```
mod_svm_tune$best.parameters
```

```
##   kernel gamma cost
## 2 radial     1    1
```

4. Forecast `charges` using the test dataset and the best model found in 3.

```
predict_mod_svm <- predict(mod_svm_tune$best.model, newdata = test)
```

5. Compute the RMSE (Root Mean Squared Error) on the test data.

```
rmse(test$charges, predict_mod_svm)
```

```
## [1] 7070.447
```

6. Perform k-means Cluster Analysis

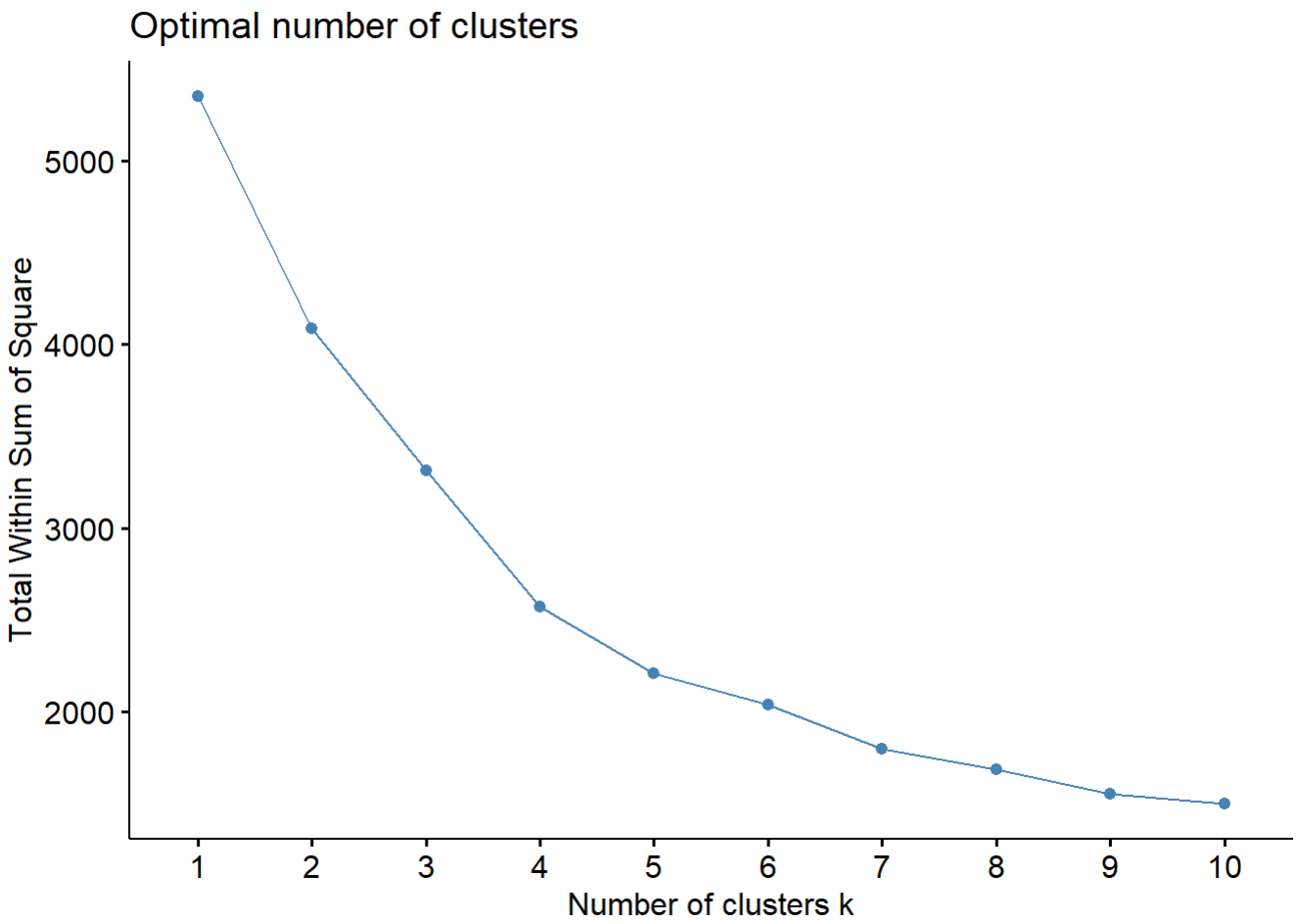
1. Remove the `sex`, `smoker`, and `region`, since they are not numerical values.

```
# remove sex, smoker, and region (non-numeric features)
insurance_numeric <- select(insurance, -sex, -smoker, -region, -log_charges)
# scale the remaining features and convert to a data frame
insurance_numeric_scaled_mod_kmeans <- as.data.frame(scale(insurance_numeric))

head(insurance_numeric_scaled_mod_kmeans)
```

```
##          age        bmi    children    charges
## 1 -1.4382265 -0.4531506 -0.90827406  0.2984722
## 2 -1.5094011  0.5094306 -0.07873775 -0.9533327
## 3 -0.7976553  0.3831636  1.58033487 -0.7284023
## 4 -0.4417824 -1.3050431 -0.90827406  0.7195739
## 5 -0.5129570 -0.2924471 -0.90827406 -0.7765118
## 6 -0.5841316 -0.8073542 -0.90827406 -0.7856145
```

2. Determine the optimal number of clusters. Justify your answer. The optimal number of clusters is 4. Before k = 4, the cost decreases significantly between each value of k. After k = 4, the cost is almost flat between each value of k.

```
library(factoextra)

fviz_nbclust(insurance_numeric_scaled_mod_kmeans, kmeans, method = "wss")
```

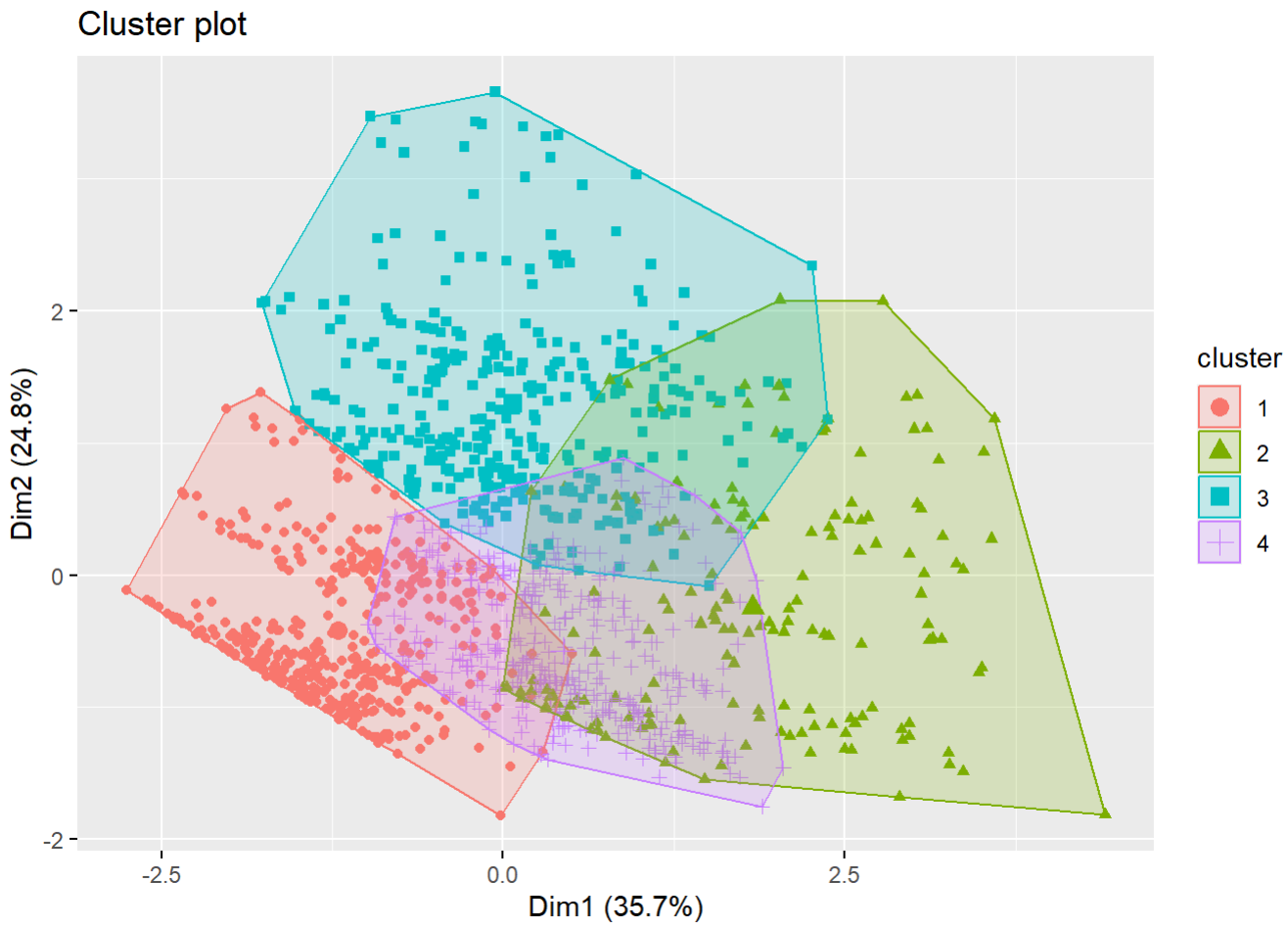


3. Perform k-means clustering using the optimal number of clusters from 5b.

```
mod_kmeans <- kmeans(insurance_numeric_scaled_mod_kmeans, centers = 4, nstart = 25)
```

4. Visualize the clusters in different colors.

```
fviz_cluster(mod_kmeans, data = insurance_numeric_scaled_mod_kmeans, labelsize = 0)
```



7. Build a Neural Network Model

1. Remove the `sex`, `smoker`, and `region`, since they are not numerical values.

```
head(insurance_numeric)
```

```
##   age    bmi children   charges
## 1  19 27.900        0 16884.924
## 2  18 33.770        1  1725.552
## 3  28 33.000        3  4449.462
## 4  33 22.705        0 21984.471
## 5  32 28.880        0  3866.855
## 6  31 25.740        0  3756.622
```

2. Standardize the inputs using the `scale()` function.

```
insurance_numeric_scaled_mod_nn <- cbind(as.data.frame(scale(select(insurance_numeric, -charge
s))), insurance_numeric$charges)
names(insurance_numeric_scaled_mod_nn) <- c(names(insurance_numeric_scaled_mod_nn)[1:(ncol(insur
ance_numeric_scaled_mod_nn) - 1)], "charges")
head(insurance_numeric_scaled_mod_nn)
```

```
##          age        bmi    children   charges
## 1 -1.4382265 -0.4531506 -0.90827406 16884.924
## 2 -1.5094011  0.5094306 -0.07873775  1725.552
## 3 -0.7976553  0.3831636  1.58033487  4449.462
## 4 -0.4417824 -1.3050431 -0.90827406 21984.471
## 5 -0.5129570 -0.2924471 -0.90827406  3866.855
## 6 -0.5841316 -0.8073542 -0.90827406  3756.622
```

3. Convert the standardized inputs to a data frame using the `as.data.frame()` function.

```
head(insurance_numeric_scaled_mod_nn)
```

```
##          age        bmi    children   charges
## 1 -1.4382265 -0.4531506 -0.90827406 16884.924
## 2 -1.5094011  0.5094306 -0.07873775  1725.552
## 3 -0.7976553  0.3831636  1.58033487  4449.462
## 4 -0.4417824 -1.3050431 -0.90827406 21984.471
## 5 -0.5129570 -0.2924471 -0.90827406  3866.855
## 6 -0.5841316 -0.8073542 -0.90827406  3756.622
```

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

```
train_i <- sample(1:nrow(insurance_numeric_scaled_mod_nn), as.integer(0.8 * nrow(insurance_numer
ic_scaled_mod_nn)))
train <- insurance_numeric_scaled_mod_nn[train_i,]
test <- insurance_numeric_scaled_mod_nn[-train_i,]
```

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

```
library(neuralnet)

mod_nn <- neuralnet(charges ~ age + bmi + children, data = train, hidden = c(1))
```

6. Plot the neural networks.

```
plot(mod_nn)
```

7. Forecast the `charges` in the test dataset.

```
predict_mod_nn <- predict(mod_nn, newdata = test)
```

8. Get the observed `charges` of the test dataset.

```
observed <- test$charges
```

9. Compute test error (RMSE).

```
rmse_mod_nn <- rmse(observed, predict_mod_nn)
rmse_mod_nn
```

```
## [1] 12214.21
```

Putting it all Together

1. a. For predicting insurance charges, your supervisor asks you to choose the best model among the multiple regression, regression tree, and random forest. Compare the test RMSEs of the models generated above. Display the names for these models (Multiple Linear Regression, Regression Tree, and Random Forest) and their corresponding test RMSEs in a `data.frame` . Label the column in your data frame with the labels as `Model.Type` , and label the column with the test RMSEs as `Test.RMSE` 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. b. 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 random forest model is the best according to RMSE because its test RMSE is better than that of both the multiple linear regression model and the regression tree model. However, for part (b), I would recommend the multiple linear regression model because it is easy to explain using its coefficients, which are obtained using `summary(mod_lm)` . The disadvantage of this simpler model compared to the random forest model is that it has a higher test RMSE. Note that the regression tree model is also easy to explain, but its test RMSE is higher than that of the multiple linear regression model.

```
data.frame(Model.Type = c("Multiple Linear Regression", "Regression Tree", "Random Forest"), Tes
t.RMSE = round(c(rmse_mod_lm, rmse_mod_tree, rmse_mod_rand_forest), 4))
```

```
##                   Model.Type Test.RMSE
## 1 Multiple Linear Regression    0.4891
## 2            Regression Tree    0.5463
## 3              Random Forest    0.4307
```

2. The supervisor from the sales department likes your regression tree model. But she says that the salespeople 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 1b 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.

```
# Copy your pruned tree model to a new variable name.
pruned_tree <- mod_tree_prune

# 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.
pruned_tree$frame$yval <- exp(pruned_tree$frame$yval)

# After you reverse the log transform on the yval column, then replot the tree with labels.
plot(pruned_tree)
text(pruned_tree)
```