Build Predictive Models to Predict Insurance Charges

Rakesh Karki

INTRODUCTION:
In this project, we will analyze data to build various predictive models to predict the insurance change of an individual based on 'Age', 'Sex', 'BMI', 'Number of Children', 'Smoker status', ' Region of residence', 'Current Insurance Charges' as features/predictors.
We will randomly split the data into training (2/3rd of the data) and testing (1/3rd of the data) data. We will then create multiple "linear regression model", "regression tree model", "random forest model", "support vector machine (SVM) model", "K-means cluster analysis" and "neural network model" using the training data. We will then calculate the Mean Square Error (MSE) value of each of the models using out of sample (testing) data and select the model with the lowest MSE as best model to predict the insurance charge.
This is a relatively small size data for easy demonstration, but the process can be applied to a real life data with millions of rows.

Analysis Flow:
STEP 1. DATA PREPARATION
STEP 2. BUILD A MULTIPLE LINEAR REGRESSION MODEL
STEP 3. BUILD A REGRESSION TREE MODEL
STEP 4. BUILD A RANDOM FOREST MODEL
STEP 5. BUILD A SUPPORT VECTOR MACHINE (SVM) MODEL
STEP 6. PERFORM THE K-MEANS CLUSTER ANALYSIS
STEP 7. BUILD THE NEURAL NETWORK MODEL
STEP 8. PUTTING IT ALL TOGETHER

```
knitr::opts_chunk$set(include=TRUE, error=FALSE, message=FALSE,echo = TRUE)
```

STEP 1. DATA PREPARATION

Step 1-a. We will start by loading the DataFrame insurance.csv into R memory and then preview the data to understand the structure.

```
library(readr)
insurance <- read_csv("insurance.csv")
dim(insurance)       # check the data size: rowsXcolumns
```

```
## [1] 1338    7
```

```
head(insurance)      # preview first few rows
```

```
## # A tibble: 6 x 7
##     age sex      bmi children smoker region    charges
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
```

```
## 1    19 female  27.9        0 yes    southwest  16885.
## 2    18 male    33.8        1 no     southeast   1726.
## 3    28 male    33          3 no     southeast   4449.
## 4    33 male    22.7        0 no     northwest  21984.
## 5    32 male    28.9        0 no     northwest   3867.
## 6    31 female  25.7        0 no     southeast   3757.
```

Step 1-b. Now, we will log transform the label variable 'charges' which helps the analysis by converting the skewed data towards normal distribution.

```
insurance$charges = log(insurance$charges)
```

Let's check how the data looks after the log transformation.

```
head(insurance)
```

```
## # A tibble: 6 x 7
##     age sex      bmi children smoker region    charges
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
## 1    19 female  27.9        0 yes    southwest    9.73
## 2    18 male    33.8        1 no     southeast    7.45
## 3    28 male    33          3 no     southeast    8.40
## 4    33 male    22.7        0 no     northwest   10.0
## 5    32 male    28.9        0 no     northwest    8.26
## 6    31 female  25.7        0 no     southeast    8.23
```

Step 1-c. In the steps below, use model.matrix() function to transform the data from 1-b above to create another data set that uses dummy variables in place of categorical variables. This transformation is needed to build the K-means cluster and neural network model later. We will then verify that the first column has only ones (1) as values, and then discard the first column as it does not have any meaning on future analysis.

Convert the data to matrix using model.matrix() function.

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

Verify the conversion.

```
head(insurance.matrix)
```

```
##   (Intercept) age sexmale    bmi children smokeryes regionnorthwest
## 1           1  19       0 27.900        0         1               0
## 2           1  18       1 33.770        1         0               0
```

```
## 3           1  28       1 33.000        3         0               0
## 4           1  33       1 22.705        0         0               1
## 5           1  32       1 28.880        0         0               1
## 6           1  31       0 25.740        0         0               0
##   regionsoutheast regionsouthwest  charges
## 1               0               1 9.734176
## 2               1               0 7.453302
## 3               1               0 8.400538
## 4               0               0 9.998092
## 5               0               0 8.260197
## 6               1               0 8.231275
```

Remove the first column from the matrix and check if the first column has been removed from the DataFrame.

```
insurance.matrix<- insurance.matrix[, -1]
head(insurance.matrix)
```

```
##   age sexmale    bmi children smokeryes regionnorthwest regionsoutheast
## 1  19       0 27.900        0         1               0               0
## 2  18       1 33.770        1         0               0               1
## 3  28       1 33.000        3         0               0               1
## 4  33       1 22.705        0         0               1               0
## 5  32       1 28.880        0         0               1               0
## 6  31       0 25.740        0         0               0               1
##   regionsouthwest  charges
## 1               1 9.734176
## 2               0 7.453302
## 3               0 8.400538
## 4               0 9.998092
## 5               0 8.260197
## 6               0 8.231275
```

Step 1-d. Use the sample() function with set.seed = 1 to generate row indexes for the training and tests sets, with 2/3 of the row indexes for the training set and 1/3 for the test set.

Set the seed for result reproducibility.

```
set.seed(1)
```

Randomly sample the 2/3rd of the data rows as training set.

```
training.rows<- sample(1:nrow(insurance), nrow(insurance)*2/3)
```

Step 1-e. Create a training and test data from the data created in 1.b using the training and test row indexes created at 1-d above.

Training data.

```
training.data<- insurance[training.rows, ]
```

Preview the training data.

```
dim(training.data)
```

```
## [1] 892   7
```

```
head(training.data)
```

```
## # A tibble: 6 x 7
##     age sex      bmi children smoker region    charges
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
## 1    19 female  24.6        1 no     northwest    7.90
## 2    56 male    36.1        3 no     southwest    9.42
## 3    32 female  17.8        2 yes    northwest   10.4
## 4    41 male    34.2        1 no     southeast    8.75
## 5    27 male    32.7        0 no     southeast    7.82
## 6    31 male    34.4        3 yes    northwest   10.6
```

Testing data.

```
test.data<- insurance[-training.rows, ]
```

Preview the testing data.

```
dim(test.data)
```

```
## [1] 446   7
```

```
head(test.data)
```

```
## # A tibble: 6 x 7
##     age sex      bmi children smoker region    charges
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
## 1    19 female  27.9        0 yes    southwest    9.73
## 2    18 male    33.8        1 no     southeast    7.45
## 3    33 male    22.7        0 no     northwest   10.0
## 4    37 male    29.8        2 no     northeast    8.77
## 5    60 female  25.8        0 no     northwest   10.3
## 6    62 female  26.3        0 yes    southeast   10.2
```

Step 1-f. Create training and test DataFrames from matrix DataFrame created at 1-c using the training and test row indexes created at 1-d

Training data - matrix.

```
training.matrix<- insurance.matrix[training.rows, ]
```

Preview the training data

```
dim(training.matrix)
```

```
## [1] 892   9
```

```
head(training.matrix)
```

```
##      age sexmale    bmi children smokeryes regionnorthwest regionsoutheast
## 1017  19       0 24.605        1         0               1               0
## 679   56       1 36.100        3         0               0               0
## 129   32       0 17.765        2         1               1               0
## 930   41       1 34.210        1         0               0               1
## 471   27       1 32.670        0         0               0               1
## 299   31       1 34.390        3         1               1               0
##      regionsouthwest   charges
## 1017               0  7.904425
## 679                1  9.422508
## 129                0 10.396175
## 930                0  8.746677
## 471                0  7.822861
## 299                0 10.564792
```

Testing data - matrix.

```
testing.matrix<- insurance.matrix[-training.rows, ]
```

Preview the testing data.

```
dim(testing.matrix)
```

```
## [1] 446   9
```

```
head(testing.matrix)
```

```
##    age sexmale    bmi children smokeryes regionnorthwest regionsoutheast
## 1   19       0 27.900        0         1               0               0
## 2   18       1 33.770        1         0               0               1
## 4   33       1 22.705        0         0               1               0
## 9   37       1 29.830        2         0               0               0
## 10  60       0 25.840        0         0               1               0
## 12  62       0 26.290        0         1               0               1
##    regionsouthwest   charges
## 1                1  9.734176
## 2                0  7.453302
## 4                0  9.998092
## 9                0  8.765054
## 10               0 10.272397
## 12               0 10.233105
```

STEP 2. BUILD A MULTIPLE LINEAR REGRESSION MODEL

Step 2-a. Here, we will build a multiple linear regression model with 'charges' as the response/label and 'age', 'sex', 'bmi', 'children', 'smoker', and 'region' as predictor/feature variables. We will then print out the result using the summary() function.

First let's convert the categorical variables ('sex', 'smoker' and 'region') to factor.

```
insurance$sex<-factor(insurance$sex)
insurance$smoker<-factor(insurance$smoker)
insurance$region<-factor(insurance$region)
```

Now, let's build a multiple linear regression model with predictors and response as defined above using the training data and print out the result using the summary() function.

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

```
##
## Call:
## lm(formula = charges ~ age + sex + bmi + children + smoker +
##     region, data = training.data)
##
## 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
```

Step 2-b. Let's check the relationship between the predictor variables and the response (find the coefficients).

Coefficients of variables of the multiple linear regression model.

```
lm$coefficients
```

```
##     (Intercept)             age         sexmale             bmi        children
##      7.04207637      0.03491480     -0.06322846      0.01180498      0.10015175
##       smokeryes regionnorthwest regionsoutheast regionsouthwest
##      1.53356109     -0.08239774     -0.14002478     -0.10771800
```

As the coefficient of each variables are non zero, each predictors has effect on the response variable. The p values above (with 1 or more * sign) also show that effect of each of predictor variable is statistically significant.

Step 2-c. Now, let's perform best subset selection using the stepAIC() function from the "MASS" library and choose best model based on AIC. For the "direction" parameter in the stepAIC() method, we will set direction = "backward"

Load package "MASS" into R memory.

```
library(MASS)
```

Build the multiple linear regression model using all predictor variables.

```
full.lm = lm(charges ~ age + sex + bmi+ children +smoker + region, data = training.data)
```

Run backward feature selection calling the function stepAIC.

```
lm.bwd = stepAIC(full.lm, 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
```

Print the result.

```
summary(lm.bwd)
```

```
##
## Call:
## lm(formula = charges ~ age + sex + bmi + children + smoker +
##     region, data = training.data)
##
## 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
```

As all of the variables are statistically significant based on P value above (confirmed again), the model should include all feature variables: 'age', 'sex', 'bmi', 'children', 'smoker' and 'region' as the predictors.

Step 2-d. Here, we will compute the MSE of the best model built in Step 2-c above based on AIC using Leave-One-Out Cross-Validation (LOOCV) by trainControl() and train() function of the library 'caret'. We will then calcuate the Mean Square Error (MSE) by squaring the reported RMSE.

Load package 'caret' into R memory.

```
library(caret)
```

```
## Warning in file(con, "r"): cannot open file '/var/db/timezone/zoneinfo/
## +VERSION': No such file or directory
```

Create the trainControl with Leave-One-Out Cross-Validation (LOOCV) method.

```
train.control.lm1 <- trainControl(method="LOOCV")
```

Identify the best linear regression model using the Leave-One-Out Cross-Validation (LOOCV) method.

```
model.lm1 <- train(charges ~ age + sex + bmi+ children +smoker + region, data = training.data,
               trControl=train.control.lm1, method="lm")
print(model.lm1)
```

```
## 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
```

Here, we will calculate MSE as square of RMSE calculated above.

```
MSE.lm1<- 0.4282153^2
print(MSE.lm1)
```

```
## [1] 0.1833683
```

Step 2-e.Now, let's calculate the MSE of the best model based on AIC using 10-fold Cross-Validation. Then calcualte the MSE.

Create the trainControl method with 10-fold Cross-Validation.

```
train.control.lm2 <- trainControl(method="CV", number = 10)
```

Indentify the best linear regression model with 10-fold Cross-Validation.

```
model.lm2 <- train(charges ~ age + sex + bmi+ children +smoker + region, data = training.data,
                   trControl=train.control.lm2, method="lm")
```

Print the result.

```
print(model.lm2)
```

```
## 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
```

Let's calcucalte the MSE.

```
MSE.lm2<- 0.4260563^2
print(MSE.lm2)
```

```
## [1] 0.181524
```

tep 2-f. Calculate and report the test MSE using the best model from Step 2-c and the test data set from step 1.e.

Predict the insurance charges using the test data.

```
yhat.insurance.charges.lm <- predict(lm.bwd, newdata = test.data)
```

Identify the actual insurance charges from the test data.

```
test.insurance.charges <- insurance[-training.rows, "charges"]
```

Calculate the mean square error (MSE) of the model using out of sample (test) data.

```
MSE.Test.lm<-mean((test.insurance.charges$charges-yhat.insurance.charges.lm)^2)
print(MSE.Test.lm)
```

```
## [1] 0.231291
```

The calculated MSE with test data is 0.231291 which is larger than the calculated MSE using training data (as expected - as model tends to perform better with training data than Out of Sample - test data).

STEP 3. BUILD A REGRESSION TREE MODEL

Step 3-a. Now, Let's build a regression tree model using function tree(), with 'charges' as the response variables and the 'age', 'sex', 'bmi', 'children', 'smoker', and 'region' as predictors.

Start by loading the package "tree" into R memory.

```
library(tree)
```

Now, build the regression tree model using the training data and view the model.

```
tree.train<-tree(charges ~ age + sex + bmi+ children + smoker + region, insurance,
        subset = training.rows)
print(tree.train)
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 892 741.000  9.086
```

```
##    2) smoker: no 707 378.700  8.773
##      4) age < 38.5 346 147.500  8.246
##        8) age < 22.5 114  47.040  7.756
##         16) children < 1.5 100  29.600  7.644 *
##         17) children > 1.5 14   7.134  8.560 *
##        9) age > 22.5 232  59.680  8.487
##         18) children < 0.5 76  16.170  8.192 *
##         19) children > 0.5 156  33.650  8.631 *
##      5) age > 38.5 361  43.230  9.278
##       10) age < 51.5 181  16.220  9.074 *
##       11) age > 51.5 180  11.910  9.483 *
##    3) smoker: yes 185  28.120 10.280
##      6) bmi < 30.1 93   4.994  9.943 *
##      7) bmi > 30.1 92   1.577 10.630 *
```

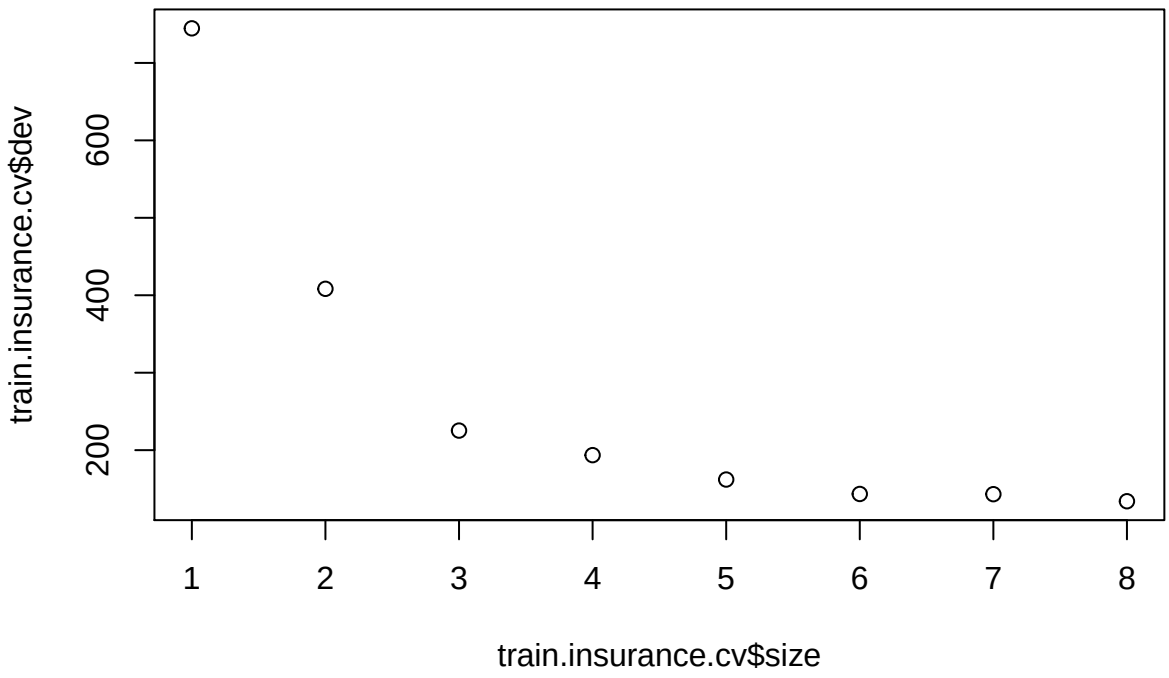
Step 3-b. Now, we will find the optimal tree using cross-validation and display the result graphically. We will then determine the best tree size.

Perform the cross-validation to choose the optimal tree.

```
train.insurance.cv <- cv.tree(tree.train)
```

Plot the model results from Step 3-b and determine the best optimal tree size.

```
plot(train.insurance.cv$size, train.insurance.cv$dev, ttpe = "b")
```



Looking at the graph, it seems that the optimal tree size is 4. Best tree is the tree with the lowest test error (y value of the dev vs size graph). Even though the tree of size 8 has the lowest test error, the test error is not much better than the tree of size 4. Model with tree size 8 will have low bias as it captures variation on most of the training data while doing so it will over fit the model and the model will also be more complex than tree of size 4. Because of these reasons, we will select the tree size of 4 as the optimal model.

Step 3-c. Now, we will prune the tree using the optinal size (4) determined above.

```
prune.train.insurance<- prune.tree(tree.train, best = 4)
```

Step 3-d. Now, let's visually display the best tree model with labels.

As the variable 'charges' was log transformed at the beginning of the analysis, we will have to convert it back by using the exp() to make the predicted output meaningful.

i. Copy the pruned tree model to a new variable.

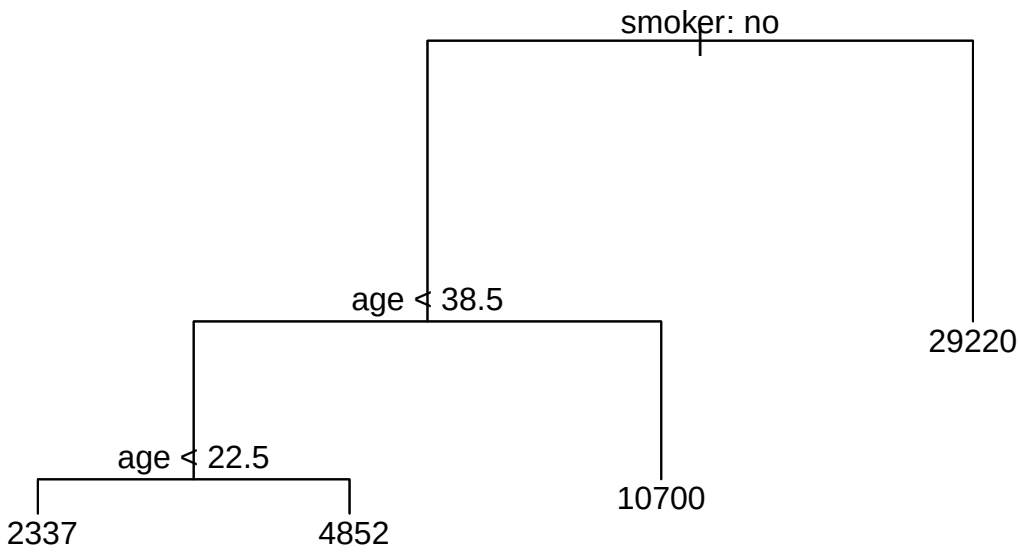
```
copy.prune.train<-prune.train.insurance
```

ii. Reverse the log transformation of column yval using the exp() function.

```
copy.prune.train$frame$yval<-exp(copy.prune.train$frame$yval)
```

iii. Now, let's show our optimal regression tree model.

```
library(tree)
plot(copy.prune.train)
text(copy.prune.train, pretty = 0)
```



Step 3-e. Calculate the test MSE for the best model.

Let's predict the charges using the test data.

```
yhat.insurance.charges.tree <- predict(prune.train.insurance, newdata = insurance[-training.rows, ])
```

Let's find the actual charges value from the test data.

```
test.insurance.charges <- insurance[-training.rows, "charges"]
```

Now, let's calculate the mean square error (MSE) with the test data.

```
MSE.Test.tree <-mean((test.insurance.charges$charges-yhat.insurance.charges.tree)^2)
print(MSE.Test.tree)
```

```
## [1] 0.2421874
```

STEP 4. BUILD A RANDOM FOREST MODEL

Step 4-a. Here, we will build a random forest model using function randomForest(), with 'charges' as the response variables and the 'age', 'sex', 'bmi', 'children', 'smoker', and 'region' as predictors.

Let's begin by loading the package "randomForest" into R memory.

```
library(randomForest)
```

Here, we will build the random forest model.

```
rf.model <- randomForest(charges ~ age + sex + bmi+ children + smoker + region,
            data = insurance, subset = training.rows, importance = TRUE)
```

Step 4-b. Now, let's calculate the MSE using the test data.

Predict insurance charge based using the test data.

```
yhat.insurance.charges.rf <- predict(rf.model, newdata = insurance[-training.rows, ])
```

Find the actual insurance charge from the test data.

```
test.insurance.charges <- insurance[-training.rows, "charges"]
```

Now, let's calculate the mean square error (MSE) with test data.

```
MSE.Test.rf<- mean((test.insurance.charges$charges-yhat.insurance.charges.rf)^2)
print(MSE.Test.rf)
```

```
## [1] 0.1786931
```

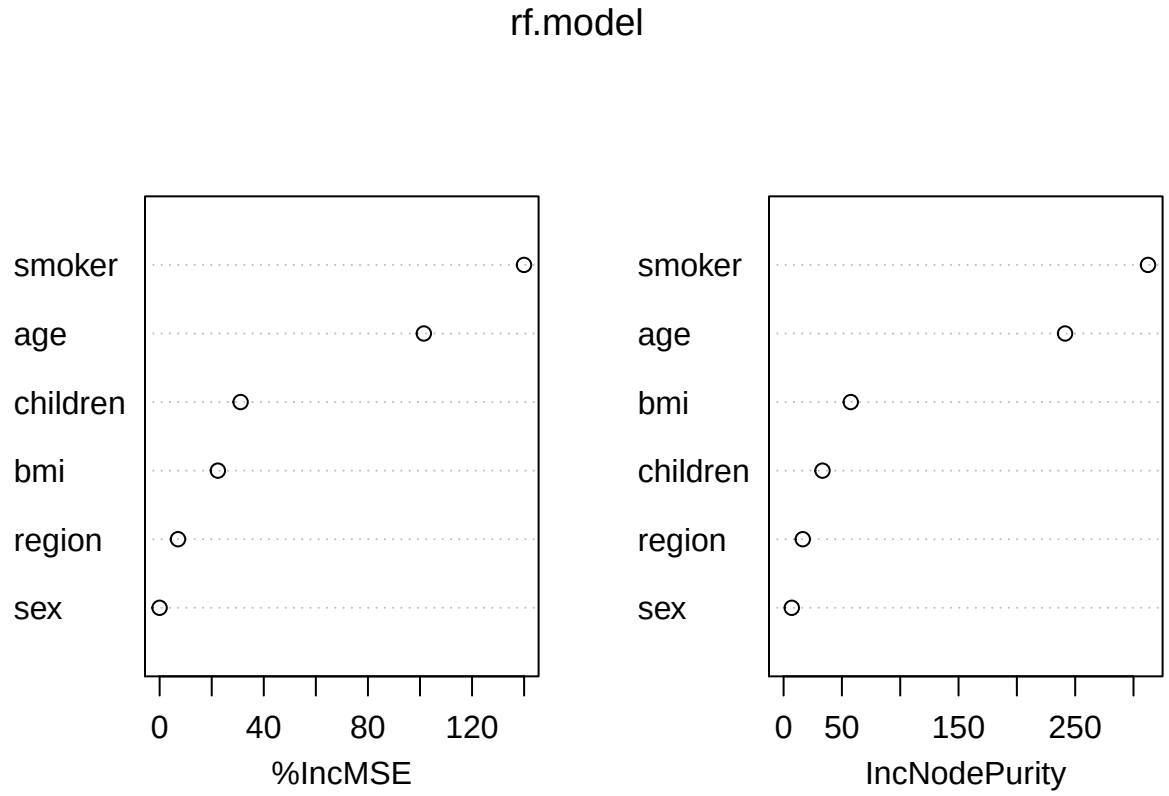
Step 4-c. Now, Let's extract variable importance measure using the importance() function.

```
importance(rf.model)
```

```
##               %IncMSE IncNodePurity
## age      101.47587921    241.349930
## sex       -0.02726974      7.036124
## bmi       22.37085750     57.591647
## children  31.08976277     33.375714
## smoker   139.96007384    312.453784
## region     7.05497624     16.454965
```

Step 4-d. Here, we will plot the variable importance and identify the top 3 important predictors in this model.

```
varImpPlot(rf.model)
```



Top three most important predictors of insurance charges are smoker, age and children (Based on %IncMSE) or bmi(Based on IncNodePurity).

STEP 5. BUILD A SUPPORT VECTOR MACHINE (SVM) MODEL

Step 5-a. Let's build the SVM with 'charges' as the response variables and the 'age', 'sex', 'bmi', 'children', 'smoker', and 'region' as predictors. We will use the svm() function with radial kernel and gamma = 5 and cost = 50.

Let's begin by loading the package "e1071" into R memory.

```
library(e1071)
```

Here, we will build the SVM model and print the model summary.

```
SVM.fit<- svm(charges ~ age + sex + bmi+ children + smoker + region, data = training.data,
              kernel= "radial", cost=50, gamma=5)
summary(SVM.fit)
```

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

Step 5-b. Now, let's 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".

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

Step 5-c. Let's print out the model results and identify the best model parameters.

```
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     1
##
## - best performance: 0.1952968
##
## - Detailed performance results:
##    cost gamma     error dispersion
## 1     1     1 0.1952968 0.05607578
## 2    10     1 0.1953342 0.05570885
## 3    50     1 0.1953874 0.05572157
## 4   100     1 0.1953997 0.05571223
## 5     1     3 0.1952968 0.05607578
## 6    10     3 0.1953342 0.05570885
## 7    50     3 0.1953874 0.05572157
## 8   100     3 0.1953997 0.05571223
## 9     1     5 0.1952968 0.05607578
## 10   10     5 0.1953342 0.05570885
## 11   50     5 0.1953874 0.05572157
## 12  100     5 0.1953997 0.05571223
```

As shown on the output, the best model is the model with cost value 1 and gamma value 1.

Step 5-d. Now, we will calcuate the MSE using the test data.

Predict the charges using the test data.

```
Predicted.charges.svm<- predict(tune.out$best.model, newdata = test.data)
```

Find the actual charges from the test data.

```
test.insurance.charges<- insurance[-training.rows, "charges"]
```

Now, let's calcualte MSE with test data.

```
MSE.Test.svm<-mean((test.insurance.charges$charges-Predicted.charges.svm)^2)
print(MSE.Test.svm)
```

```
## [1] 0.2567148
```

STEP 6. PERFORM THE K-MEANS CLUSTER ANALYSIS

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

Standarizing the data using scale() function.

```
scale.training.matrix<-scale(training.matrix)
```

Preview the data.

```
head(scale.training.matrix)
```

```
##             age    sexmale        bmi    children  smokeryes regionnorthwest
## 1017 -1.4456614 -0.9838742 -0.9681800 -0.07087773 -0.5112491       1.8412723
## 679   1.1944785  1.0152506  0.8982433  1.59288376 -0.5112491      -0.5424939
## 129  -0.5180447 -0.9838742 -2.0787790  0.76100302  1.9538007       1.8412723
## 930   0.1241515  1.0152506  0.5913673 -0.07087773 -0.5112491      -0.5424939
## 471  -0.8748203  1.0152506  0.3413201 -0.90275848 -0.5112491      -0.5424939
## 299  -0.5893998  1.0152506  0.6205936  1.59288376  1.9538007       1.8412723
##      regionsoutheast regionsouthwest    charges
## 1017      -0.6271432      -0.5476814 -1.2957888
## 679       -0.6271432       1.8238322  0.3688572
## 129       -0.6271432      -0.5476814  1.4365275
## 930        1.5927446      -0.5476814 -0.3722211
## 471        1.5927446      -0.5476814 -1.3852277
## 299       -0.6271432      -0.5476814  1.6214233
```

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

```
scale.training.matrix<-as.data.frame(scale.training.matrix)
```

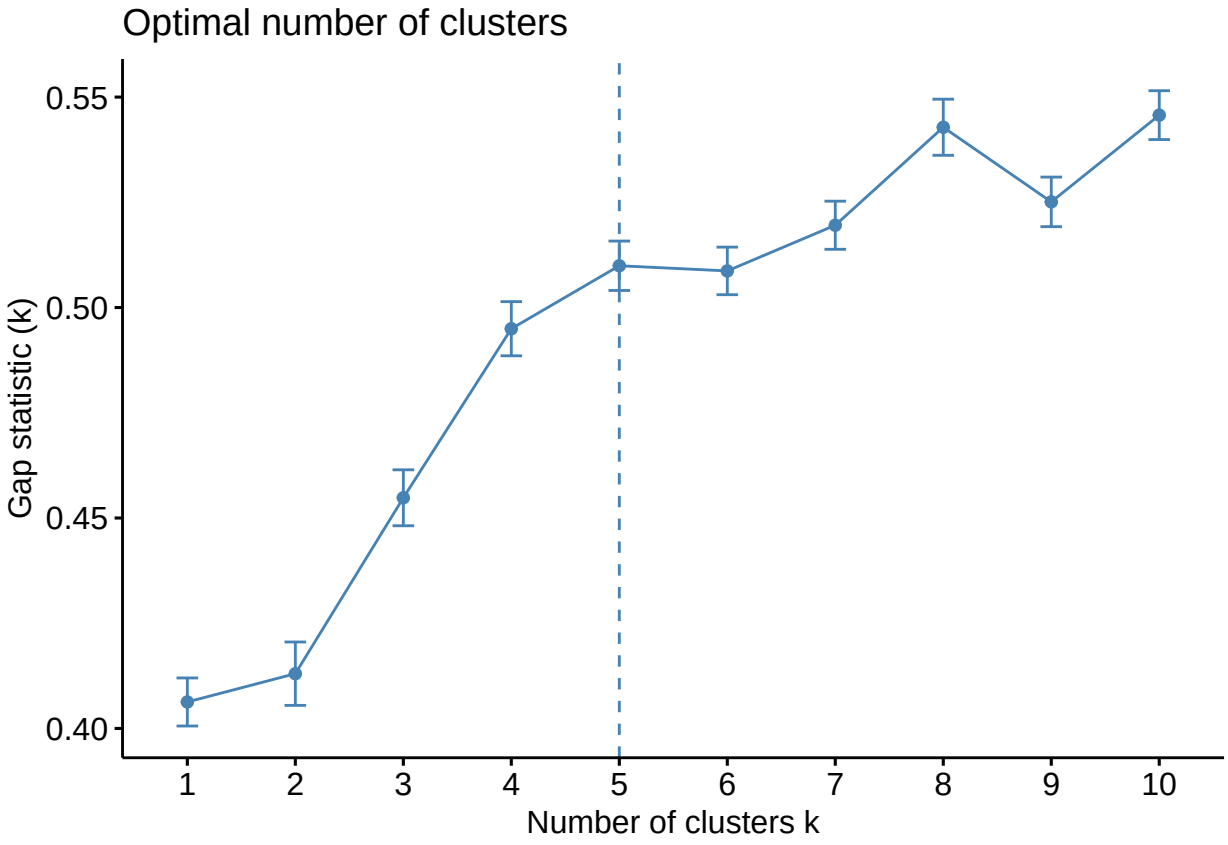
Step 6-c. Determine the optimal number of clusters.

Let's begin by loading the package "cluster" and "Factoextra" into R memory.

```
library(cluster)
library(factoextra)
```

Now, we will determine the optimal number of clusters. Use the gap_stat method and set iter.max=20.

```
fviz_nbclust(scale.training.matrix, kmeans, method = "gap_stat", iter.max=20)
```



As shown on the graph above, optimal level of cluster is 5. This is based on the change in the gap statistic vs the number of clusters. Increasing the number of clusters beyond 5 did not increase the Gap statistic significantly.

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

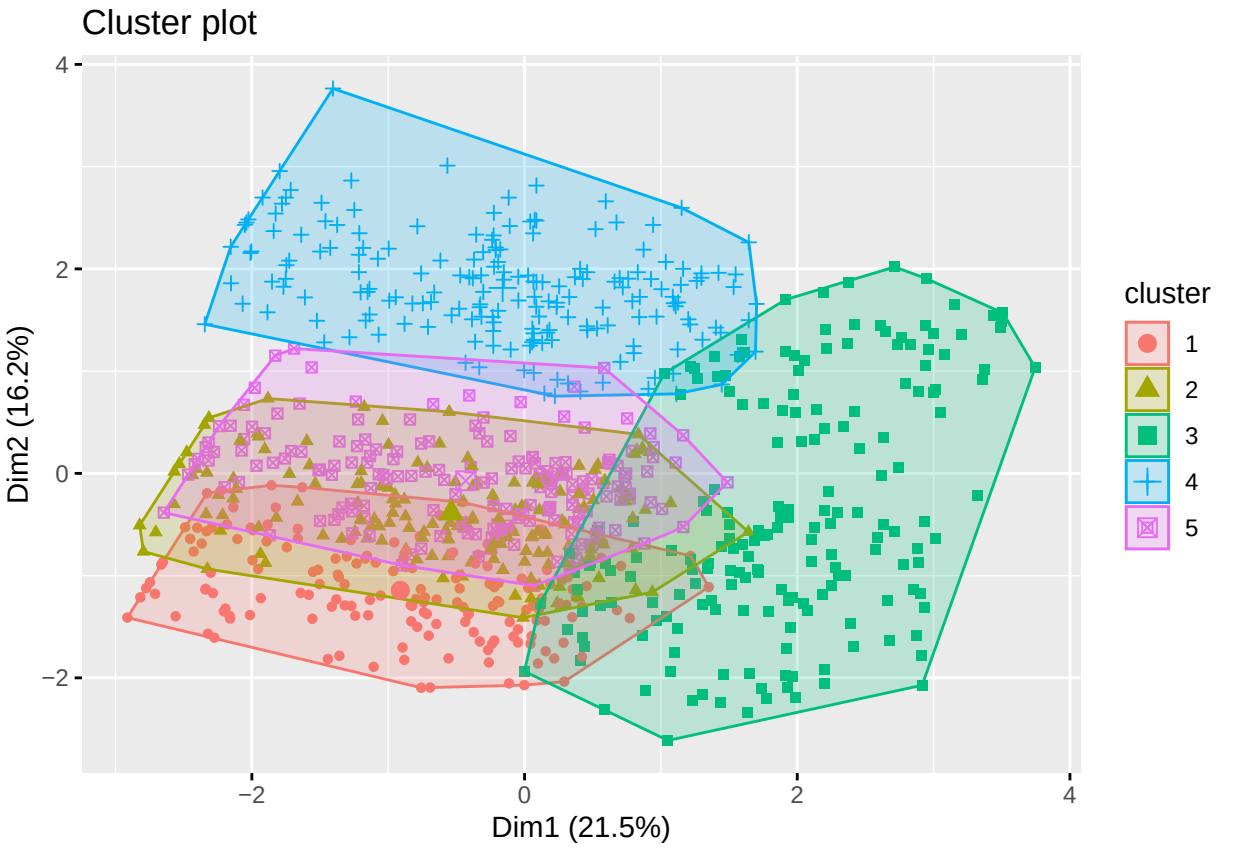
Build K-means clustering.

```
km.res<-kmeans(scale.training.matrix, 5, nstart = 25)
```

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

Let's plot the clusters.

```
fviz_cluster(km.res, data = scale.training.matrix, geom = "point")
```



STEP 7. BUILD THE NEURAL NETWORK MODEL

Step 7-a. Using the training data set created in step 1.f (matrix tranformed), we will create a neural network model where the response is 'charges' and the predictors are 'age', 'sex', 'bmi', 'children', 'smoker', 'region'. We will use 1 hidden layer with 1 neuron.

Let's begin by loading package "neuralnet" in to R memory.

```
library(neuralnet)
```

Next, build the neural network model.

```
nn.model<- neuralnet(charges~ age+sexmale+bmi+children+smokeryes+regionnorthwest+
    regionsoutheast+regionsouthwest, data = training.matrix, hidden=1)
```

Step 7-b. Let's plot the neural network.

```
plot(nn.model)
```

Step 7-c. Calculate the MSE using the test data.

Predict the charges in the test dataset.

```
Predicted.charges.nm<-compute(nn.model, testing.matrix[ ,c("age","sexmale", "bmi", "children",
    "smokeryes", "regionnorthwest", "regionsoutheast", "regionsouthwest")])
```

Preview the predicted charges (log tranformed)

```
head(Predicted.charges.nm$net.result)
```

```
##        [,1]
## 1  9.086121
## 2  9.086121
## 4  9.086121
## 9  9.086121
## 10 9.086121
## 12 9.086121
```

Actual charges value (log tranformed) from test data.

```
observ.test<- testing.matrix[ ,"charges"]
head(observ.test)
```

```
##         1         2         4         9        10        12
##  9.734176  7.453302  9.998092  8.765054 10.272397 10.233105
```

Now, let's compute the Mean Square Error (MSE) using the test data.

```
MSE.Test.nm<-mean((observ.test-Predicted.charges.nm$net.result)^2)
print(MSE.Test.nm)
```

```
## [1] 0.8737058
```

STEP 8. PUTTING IT ALL TOGETHER

Step 8-a. Let's compare the best models among multiple regression, regression tree, random forest, support vector machine, and neural network models using MSE of the models calcualted using test data as the performance indicator (lower is better).

Summarize the model performance in a DataFrame.

Model type

```
Model.Type<-c("Multiple Linear Regression", "Regression Tree", "Random Forest",
              "Support Vector Machine","Neural Network")
```

MSE value of corresponding models above

```
Test.MSE<- c(MSE.Test.lm, MSE.Test.tree, MSE.Test.rf, MSE.Test.svm, MSE.Test.nm)
```

Let's create a FataFrame with model type and MSE (round the MSE values to 4 decimal places).

```
Test.result<-data.frame(Model.Type, Test.MSE)
print(Test.result, digits = 4)
```

```
##                    Model.Type Test.MSE
## 1 Multiple Linear Regression   0.2313
## 2            Regression Tree   0.2422
## 3              Random Forest   0.1787
## 4     Support Vector Machine   0.2567
## 5              Neural Network   0.8737
```

As the Random Forest model has the lowest MSE (calculated using the test data), this model is the best model among the models above.

But, I would recommend Regression tree model (Step 3-d) as the best model to use to explain the label values based on features as it is straight forward and simple to grasp. Disadvantage of this model over other models is that it is not the best model in terms of accuracy of prediction as shown above with the calculated test error (MSE).

Let's predict the insurance charges of new individuals.

```
age <- as.numeric(c(60, 20))
sex<- as.character(c("female", "male"))
bmi<- as.numeric(c(35.8, 28.02))
children <- as.numeric(c(0, 1))
smoker<- as.character(c("no", "yes"))
region <- as.character(c("northeast","northwest"))
predict_df <- data.frame(age, sex, bmi, children, smoker, region)
head(predict_df)
```

```
##   age    sex   bmi children smoker    region
## 1  60 female 35.80        0     no northeast
## 2  20   male 28.02        1    yes northwest
```

Here, we will convert the predictor variables "sex", "smoker" and "region" to factor.

```
predict_df$sex<-factor(predict_df$sex)
predict_df$smoker<-factor(predict_df$smoker)
predict_df$region<-factor(predict_df$region)
```

Lets match the factor labels of predict_df factors with the respective variable in training data. Random forest may show error if the factor level does not match.

```
levels(predict_df$sex) <- rf.model$forest$xlevels$sex
levels(predict_df$smoker) <- rf.model$forest$xlevels$smoker
levels(predict_df$region) <- rf.model$forest$xlevels$region
```

Now, let's predict the insurance charges based on feature variables. We will need to convert the log tranformed charges values back to actual charges by using exp() to get the actual predicted insurance charge.

```
predict_insurance_charge <- exp(predict(rf.model, newdata = predict_df))
predict_insurance_charge
```

```
##        1        2
## 12568.32 17354.93
```