Prudential Life Insurance Assessment Using R

Group 10

Rui Shen

Gayathri Maganti

Kamala Nayana Uppalapati

Purpose

- ✓ Understand data preprocessing steps
- ✓ Understand three analysis techniques
- ✓ Understand the difference between classification and regression techniques

Data Pre-processing

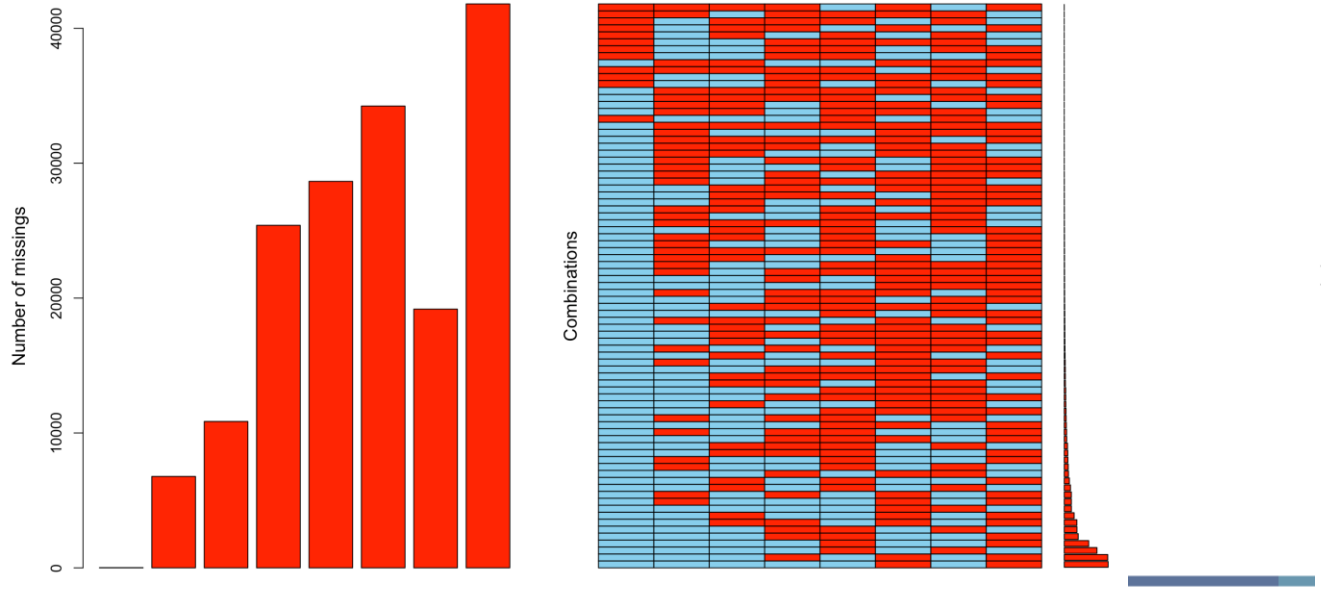
1. Look into our data:

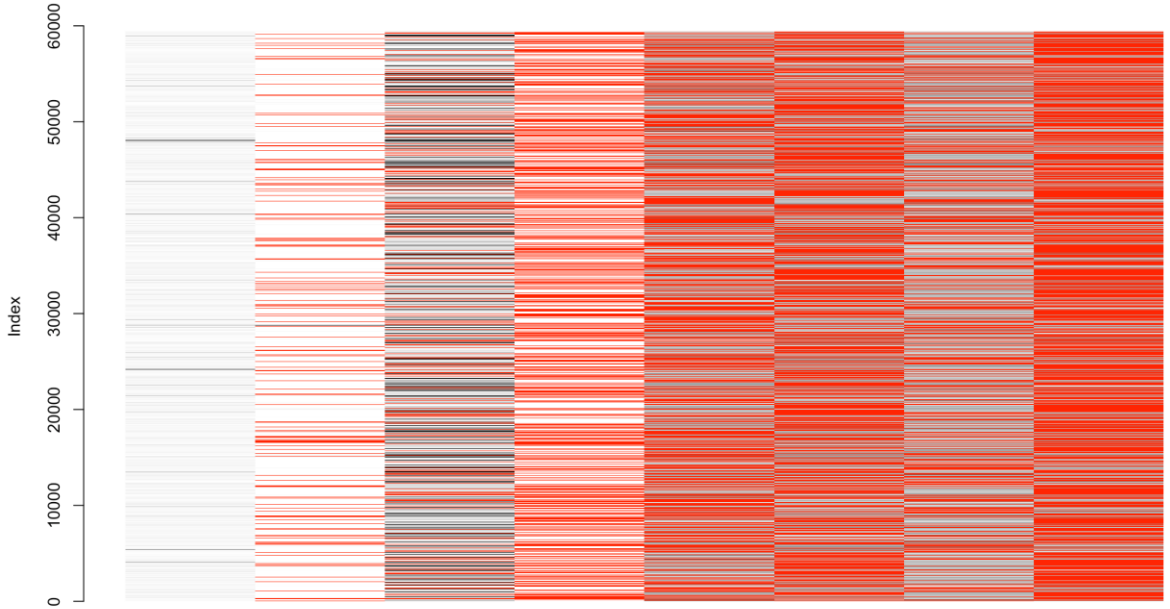
Where are the missing values?

How to fill in missing values?

– mean, median, mice

```
#Detect missing values
sum(is.na(train))/(ncol(train)*nrow(train))
#0.05171885
sum(is.na(train.categ))/(ncol(train.categ)*nrow(train.categ))
#0, no missing value in categorical/nominal variables
sum(is.na(train.cont))/(ncol(train.cont)*nrow(train.cont))
#0.2162567, this is a very large portion
sum(is.na(train.disc))/(ncol(train.disc)*nrow(train.disc))
#0.07186181, only some of the missing values are missing in discrete variables
```





Data Pre-processing

2. Data transformation and Dimension reduction

- Factor columns:
  - Product_Info_2
- Abnormal categorical columns:
  - "Column Product_Info_3 has 34 levels"
  - "Column Employment_Info_2 has 36 levels"
  - "Column InsuredInfo_3 has 11 levels"

```r
#Dimension reduction using linear analysis
fit <- lm(data.appr1$Response~. , data = data.appr1)
tmp.summary <- summary(fit)$coefficients

newcols <- rownames(tmp.summary)[tmp.summary[,4]>=0.05][-1]
data.signi <- cbind(data.appr1[newcols], data.appr1['Response'])
dim(data.signi)
#[1] 59381    42
```

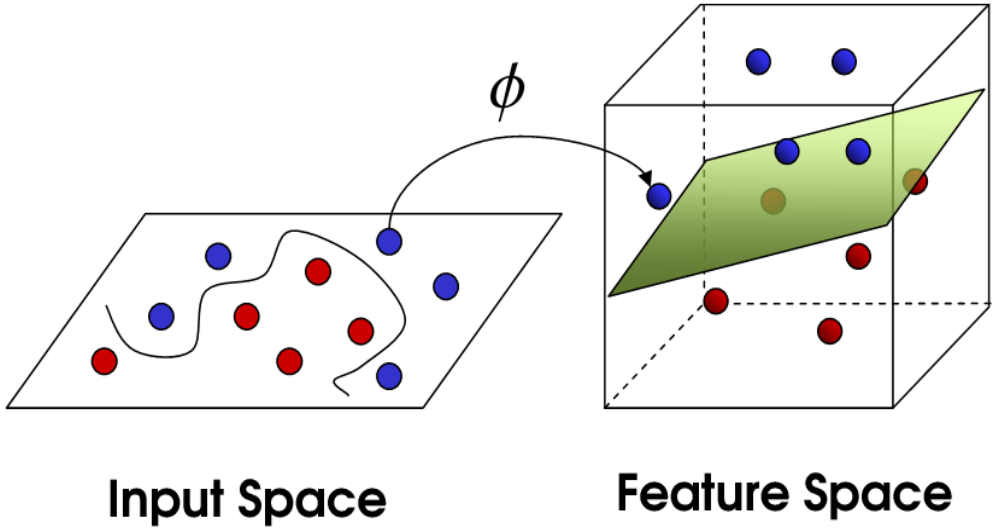
```r
categoricalfunc <- function(colNames, data){
  for(level in unique(data[[colNames]])){
    data[paste(colNames,sep = "_",level)]<- ifelse(data[[colNames]] == level,1,0)
  }
  return(subset(data,select = -get(colNames)))
}
```

SVM Analysis

What is SVM?

Basic steps

1. Build svm model
2. Tune the model by changing parameters
3. Build the svm model using best param




```
svm_tune <- tune(svm, train.x = x, train.y = y, kernel="radial",
                 ranges=list(cost=10^c(-2:2), list(epsilon = seq(0,1,0.1))))

finer_tune <- tune(svm, y ~ x,  data = Data,
                   ranges = list(epsilon = seq(svm_tune$best.model$epsilon-.15,
                                               svm_tune$best.model$epsilon+.15,
                                               0.01),
                                 cost = seq(2^(log2(svm_tune$best.model$cost)-1),
                                            2^(log2(svm_tune$best.model$cost)+1),
                                            length=6)))
```

SVM Analysis

Built 8 svm models

Final svm model:

```r
#Now that my model contains only 98 columns, build SVM regression based on these numbers
sample_index = sample(nrow(data), nrow(data)*0.2)
test_dataset = data[sample_index,]
train_dataset = data[-sample_index,]
x <- subset(train_dataset, select = -Response) # x is the predict variable
y <- train_dataset[,'Response'] #y is factor
```

```r
radial.model <- svm (Response ~ ., data=train_dataset, scale=F)
summary(radial.model)
```

```
Call:
svm(formula = Response ~ ., data = train_dataset, scale = F)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.01030928
    epsilon:  0.1


Number of Support Vectors:  45786
```

SVM Analysis

Prediction using the final model

```
> confusionMatrix(test_dataset$Response, round(tmpp))
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4    5    6    7    8
         1   27   49  109  170  233  286  260   61
         2   11   36   82  162  279  339  293   95
         3    0    4    5   19   23   61   78   21
         4    0    0    4    6   29   70  146   27
         5    5    6   27   81  164  290  411   93
         6    4    8   38  129  338  595  896  254
         7    1    6   23   89  231  469  605  193
         8    0    0   11   36  155  547 2135 1051

Overall Statistics

               Accuracy : 0.2096
                 95% CI : (0.2023, 0.217)
    No Information Rate : 0.4062
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0564
 Mcnemar's Test P-Value : <2e-16
```

```r
#Test performance on training dataset
pred <- predict(radial.model,train_dataset)
min(pred);max(pred)
#[1] -2.066659
#[1] 8.540122

svrPredictionRMSE <- me(y - pred)
svrPredictionRMSE
#[1] 1.727397
```

```r
#Test performance on test dataset
pred_test <- predict(radial.model,test_dataset)
min(pred_test);max(pred_test)
#[1] -1.844811
#[1] 8.249107

svrPredictionRMSE <- me(test_dataset$Response - pred_test)
svrPredictionRMSE
#[1] 1.72519
```

Multiple Linear Regression

Multiple Linear Regression is the most common form of linear regression analysis.

As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable from two or more independent variables

The independent variables can be
1. continuous
2. categorical

Steps involved during building the multiple linear regression model for the prudential life insurance dataset were:

1.) Splitting the cleaned dataset into 80% train data and 20% validation data.

2.) Build the first regression model taking all the columns into consideration and checking R-squared, adjusted R-squared values and p-values.

3.) Reducing Dimension using p-value where p-value is less than 0.05.

4.) During this step the variables (columns) having less or no significance in the model on observing their significant codes are eliminated.

5.) The final linear regression model was selected after the R-squared and adjusted R-squared values started to decrease after eliminating more columns that were later restored.

6.) The final model was used to predict the Response variable for the test dataset predicting values anywhere between 1 to 8.

7.) The predicted output and the actual output was then used to get the mean squared error by applying MSE formula.

Code Snippets & Regression Model

Code Snippet: Data Splitting

```
#Split dataset into train and test data
ind <- sample(2, nrow(train), replace=TRUE, prob = c(0.8, 0.2))
tdata <- train[ind==1,]
vdata <- train[ind==2,]
head(tdata)
head(vdata)
```

First Model with all Columns:

```
#Linear Regression using all the variables together
fit <- lm(train$Response~. , data = train)
summary(fit)
tmp.summary <- summary(fit)$coefficients
```

First Model Summary:

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.988 on 59229 degrees of freedom
Multiple R-squared:  0.3468,    Adjusted R-squared:  0.3451
F-statistic: 208.2 on 151 and 59229 DF,  p-value: < 2.2e-16
```

Final Results

Dimension Reduction using p-value

```
#Dimension reduction with p-value < 0.05
#Binding only those columns with p-value<0.05
newrows <- rownames(tmp.summary)[tmp.summary[,4]<0.05][-1]
train <- cbind(train[newrows], train['Response'])
colnames(train)
dim(train)
```

Code Snippet: Prediction

```
#Predicting the response with vdata or test data
prediction<-predict(fit2, vdata)

summary(prediction)

output<-round(prediction)
head(output)

actual<-vdata$Response
head(actual)
mean(abs(vdata$Response-output)) # 1.831
mean(vdata$Response-output)^2 #MSE= 0.0002

#calculating performance metrics using acccuracy function
performance <- accuracy(prediction, vdata$Response)
performance

#Finding transpose of rows and columns of performance matrix
t(performance)
```

Final model Output:

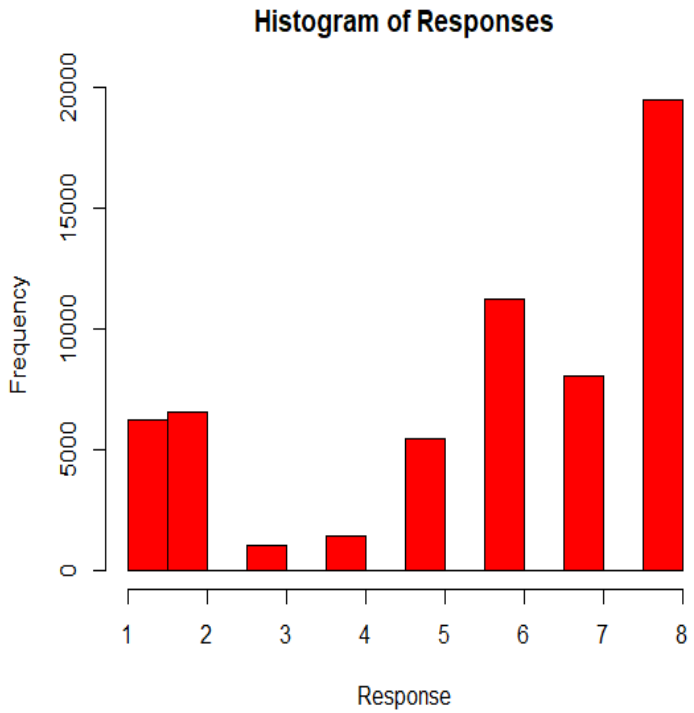
```
>
> #Predicting the response with vdata or test data
> prediction<-predict(fit2, vdata)
Warning message:
In predict.lm(fit2, vdata) :
  prediction from a rank-deficient fit may be misleading
>
> summary(prediction)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.318   5.066   5.982   5.632   6.377  12.183
>
> output<-round(prediction)
> head(output)
 1  9 14 18 26 34
 6  6  6  7  6  4
>
> actual<-vdata$Response
> head(actual)
[1] 8 8 3 7 8 5
> mean(abs(vdata$Response-output)) # 1.831
[1] 1.850896
> mean(vdata$Response-output)^2 #MSE= 0.0002
[1] 8.498145e-05
>
> #calculating performance metrics using acccuracy function
> performance <- accuracy(prediction, vdata$Response)
> performance
                    ME     RMSE      MAE       MPE     MAPE
Test set -0.009102253 2.23842 1.856344 -48.80937 74.00641
>
> #Finding transpose of rows and columns of performance matrix
> t(performance)
          Test set
ME     -0.009102253
RMSE    2.238419512
MAE     1.856344110
MPE   -48.809368590
```

Decision Tree Regression

Decision tree: It is supervised learning method. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation.
Recursive partitioning: Based on the rules applied, the main dataset is split into subsets recursively

Why Decision tree?

- Decision Tree is very simple to understand and interpret the results
- Useful in data exploration
- It works for both numeric and categorical data
- Robust and can handle big data



Basic Steps:

Installed the required packages
Loaded the libraries
Read the pre-processed data
Retrieved column details using str()
Viewed the Prudential Insurance dataset summary

```
#viewing Prudential data summary
summary(Prud_dataset)
nrow(Prud_dataset)
ncol(Prud_dataset)

#Histogram plot to view response
Response <- Prud_dataset$Response
hist(Response, col="red", main="Histogram of Responses")
```

Decision Tree Regression Steps

- Splitting of training dataset in 80:20 training and testing data sets

```
#spliting the dataset in 80:20 training and testing data sets
train_data <- Prud_dataset[1:47505, ]
test_data <- Prud_dataset[47506:59381, ]
```

- Building the model on training data set
- Rpart- Recursive partitioning for classification, regression

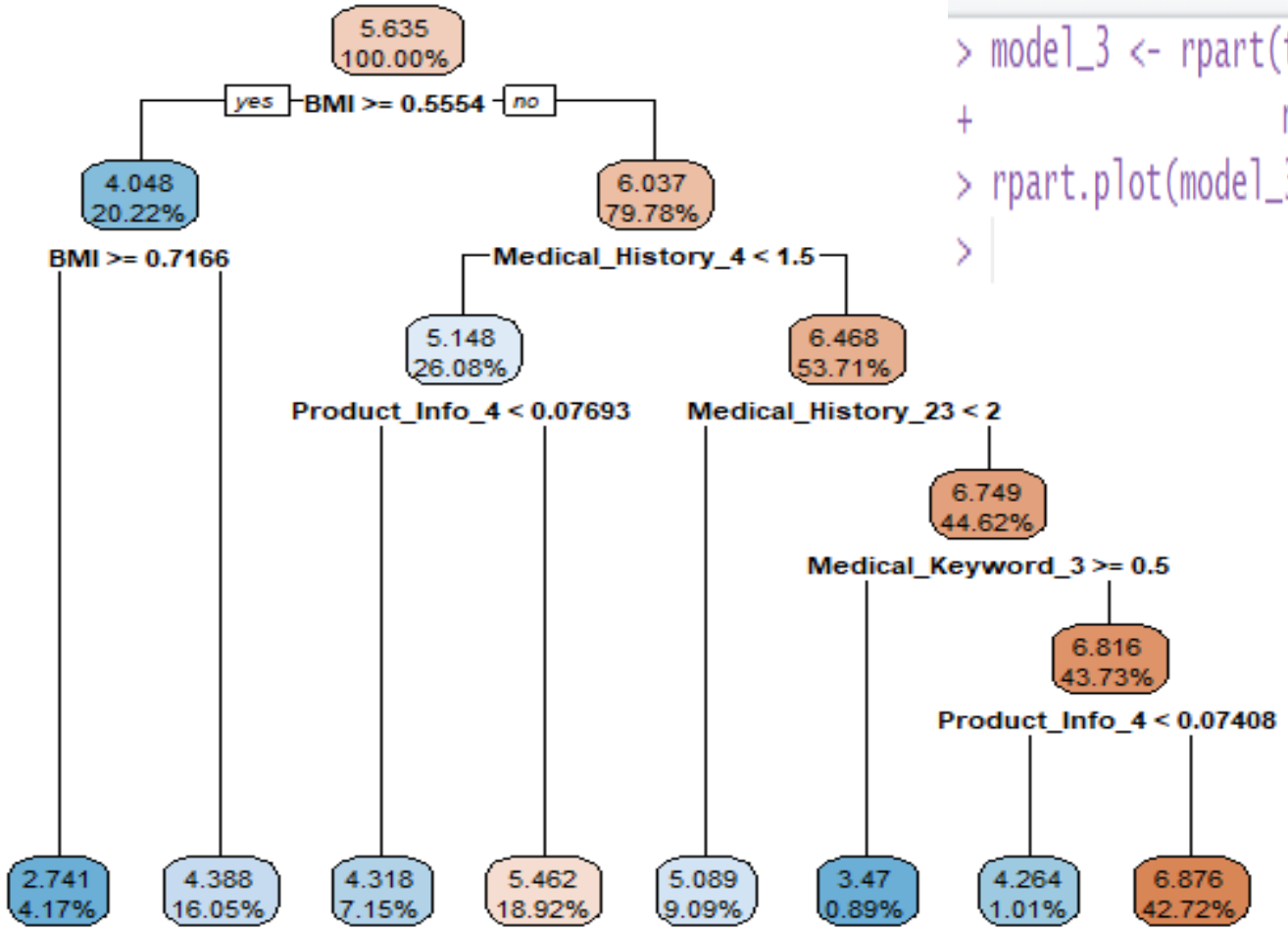
```
model_1 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=35, cp=0.001))
summary(model_1)

model_2 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=65, cp=0.01))
summary(model_2)

model_3 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=100, cp=0.01))
summary(model_3)
```

- Predicting model on to test data
- Measuring the model performance using Mean Absolute Error:
- Improved the performance:
- Rweka- Weka is a collection of machine learning algorithms for data mining tasks, containing tools for classification, regression, clustering. Package 'RWeka' contains the interface code Package: install.packages("Rweka") Library used :library(RWeka)

Plotting the decision tree




```
Console C:/Users/kamal/Desktop/ADS/Final_Midterm_Prudential_Insurance/
> model_3 <- rpart(train_data$Response~., data = train_data,
+                  method = "anova", control=rpart.control(minsplit=100, cp=0.01))
> rpart.plot(model_3, digits=4, box.palette = "BuBn")
>
```

Here minsplit is the minimum number of observations that must exists in a node
cp is complexity parameter we've chosen cp=0.01 to get decent tree

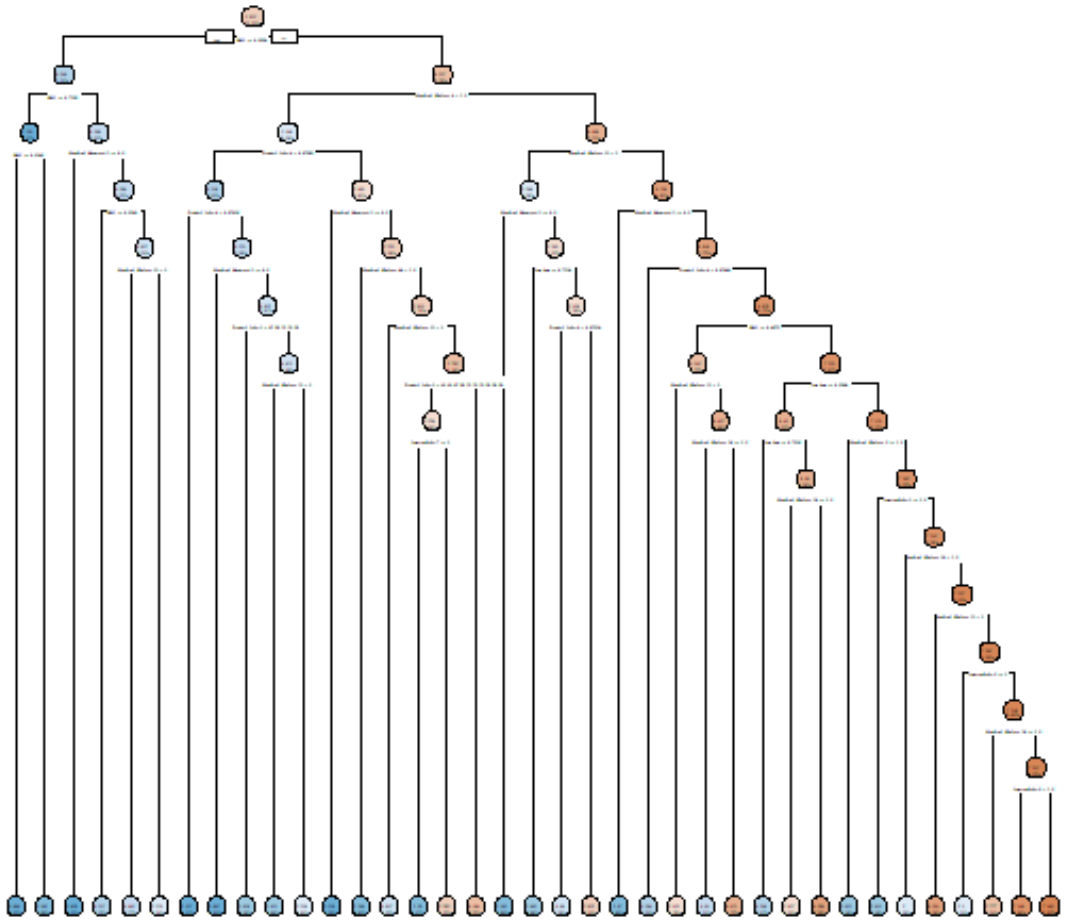
```
#measuring performance using mae (mean absolute error)
mae <- function(actual, prediction) {
  mean(abs(actual - prediction))
}
mae(test_data$Response, round(prediction))
# 1.733833
mean_res <- mean(train_data$Response)
#mean_res= 5.6348
mae(mean_res, test_data$Response)
# 2.04
#mean square error
mse <- mean((test_data$Response-prediction)^2)
#mean square error = 4.539
```

Comparisons

```
model_1 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=35, cp=0.001))
summary(model_1)

model_2 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=65, cp=0.01))
summary(model_2)

model_3 <- rpart(train_data$Response~., data = train_data,
                 method = "anova", control=rpart.control(minsplit=100, cp=0.01))
summary(model_3)
```



Accuracy and Correlation

```r
# Predicting model_3 on testing data
prediction <- predict(model_3, test_data)
#View(round(prediction))
# Confusion matrix
pred_cm <- table(prediction,test_data$Response)
pred_cm

#Mainly we do this steps for classification rather than prediction
#Calculating accuracy
acc <- sum(diag(pred_cm))/sum(pred_cm)
accuracy <- acc * 100
accuracy
# Accuracy is 36.09801%

#Calculating correlation
summary(round(prediction))
summary(test_data$Response)
cor(prediction,test_data$Response)
#correlation is found to be 0.4911
```

```r
prediction_df <- data.frame(test_data[-122],prediction)
write.csv(prediction_df, file = "decision_tree_prediction.csv", row.names = FALSE)

#Plotting Predictions histogram
Prediction_hist <- prediction_df$prediction
hist(Prediction_hist, col="blue", main="Histogram of Predictions")

#Plotting Pie chart of Predictions and it's frequncies
#Prediction_pie <- table(prediction_df$prediction)
#lbls <- paste(names(Prediction_pie), "\n", Prediction_pie, sep="")
#pie(Prediction_pie, labels = lbls,  col=rainbow(length(lbls)),
#    main="Piechart of predictions")
```