¿ options(width=60)
[entry]nty/global/

Big Data Analysis : Car Evaluation

S M ABDULLAH FERDOUS

May 9, 2020

Contents

List of Figures

1 Introduction

Big Data is changing our lifestyle dramatically and having impact in almost every aspect of our life. Big data is also changing the way business interact with the stakeholders. It helps businesses to understand the target market and optimize the business performance(**Marr**).Big Data is an asset to every size of business nowadays. Therefore, Big data analysis become a very important activity for business and individual personals. According to IBM Big data analysis is basically take a large number of data and try to make sense of the data by analyzing it using different techniques. Then use the results to get some business value out of the data(**Jack**)

For this report the analyst will choose a data set and explore the data. Then analyst will try to make sense of the data through visualization and exploration of different features of the data set. The analyst will then pre-process the data to make the data set suitable for the Machine learning algorithm. The analyst will build a classification model to predict the class label. After building and testing the model the analyst will consider scope of improvement. Finally the Analyst will produce this whole report in a reproducible Document.

2 Data Exploration

In this section, the analyst will explain the reason behind choosing the data set and the technology platform that will be used to conduct the experiments with the data. The analyst will also provide brief description of the data and complete the data pre-processing tasks in this section.

2.1 Data set Choice

For this report the analyst choose the Car Evaluation data set. The data set is available in Kaggle website. The data set file car-evaluation.csv was downloaded from the website on 11.08.2017 for the purpose of this report(**Kaggle**). The data set has some specific information about cars. That information can be used to build a model to conduct the evaluation of the cars(**UCI**). Exploration and discussion about the information in the data set will be conducted later in this chapter.

Like most of the business industry Vehicle buying and selling industry is hugely affected by the Big Data. Buyers using the big data to find the most suitable car for them. The sellers using big data to evaluate the car and find the best price for the targeted customers(**Lloyd**). It is intriguing how many different factors can affect the value of a vehicle.

By analyzing and experimenting the car evaluation data set the analyst will try to understand how some of the features of a vehicle can help to evaluate the vehicle. Specially how comfortless, maintenance, size and other technical details can have impact on the cars evaluation process.The main aim for this report is to build a predictive model to classify a car using other given features. The analyst will use this knowledge efficiently for future personal car purchase. Also one of the analyst's family friend who owns a car trade business is interest in the outcome of this report.

2.2 Technology Platform

To complete the experiment and analysis the analyst will use the programming language r which in embedded in R studio. R studio have numerous packages for analysis and visualization. For visualization R studio package ggplot2 will be widely used in this report. For classification model, the random forest classifier will be considered in this report. The analyst doesn't feel hat use big data platforms such as Hadoop/HPC is necessary for this report. Some of the key influence behind not to use such big data technology are

1. This is a relatively small data set. Hadoop is more suitable for large data set than small ones(**Crayon**).

2. The necessary analysis and experiments can be completed with the resources available in R Studio.

3. Cost efficiency
4. Limited knowledge of Hadoop or other big data analysis platforms
5. Map reduce is not necessary for this report.

2.3 Problem Statement & Data Exploration

The data set contains records of cars features. The cars are divided in different classes based on the given features. Using the data set, the analyst will try to build a model to predict the classification of the cars.First we can check the number of columns and rows in the data set

```
> #putting car_evaluation dataset in a data frame
> df <- read.csv("car_evaluation.csv")
> #counting the number of rows and colums in the dataset
> nr <- nrow(df)
> nc <- ncol(df)
> #displaying the results
> cat ("Car_evaluation Dataset has : ", nr,
+               " Rows", " and ",nc, " Columns")

Car_evaluation Dataset has :  1727  Rows  and  7  Columns
```

we can also check the dimension of the data set to see the number of rows and columns

```
> dim(df)

[1] 1727    7
```

The names of the features was removed from the data set.But for the purpose of this report we will put the features name back on the data set accordingly with the description of the data set in Kaggle and UCI website(**Kaggle**)(**UCI**).

```
> names(df) <- c("buying","maint","doors","persons",
+                          "lug_boot","safety","acceptability")
```

Now we can check the names of the features on the database

```
> names(df)

[1] "buying"        "maint"         "doors"
[4] "persons"       "lug_boot"      "safety"
[7] "acceptability"
```

we can have a quick look at the brief description of the data set

```
> str(df)
```

```
'data.frame':        1727 obs. of  7 variables:
 $ buying       : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ maint        : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ doors        : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...
 $ persons      : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 2 2 ...
 $ lug_boot     : Factor w/ 3 levels "big","med","small": 3 3 2 2 2 1 1 1 3 3 ...
 $ safety       : Factor w/ 3 levels "high","low","med": 3 1 2 3 1 2 3 1 2 3 ...
 $ acceptability: Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...
```

we can check out the first few rows of the data set

```
> head(df)
```

```
  buying maint doors persons lug_boot safety acceptability
1  vhigh vhigh     2       2    small    med         unacc
2  vhigh vhigh     2       2    small   high         unacc
3  vhigh vhigh     2       2      med    low         unacc
4  vhigh vhigh     2       2      med    med         unacc
5  vhigh vhigh     2       2      med   high         unacc
6  vhigh vhigh     2       2      big    low         unacc
```

it is understandable after some exploration of the data set that the features of this data set are divided into different levels accordingly

- Buying: vhigh ,high ,med,low
- Maint: vhigh ,high ,med, low
- Doors:2,3,4,5,more
- Persons:2,4,more
- Lug boot:small,med,big
- Safety:low,med,high

The acceptability attribute is the class label of this data set. The cars are divided onto four class category of unacc(unacceptable),acc(acceptable), good and very good.All the other features of the car have impact on determining the classification of the car.
we can check how many unique class is in the data set for better understanding

```
> length(unique(df$acceptability))
```

```
[1] 4
```

The outcome confirms that there are 4 classes in the class label of this data set as we previously determined.
we can check the names of the unique class label

```
> unique(df$acceptability)
```

```
[1] unacc acc   vgood good
Levels: acc good unacc vgood
```

we can check the class distribution of the data set with some visualization

```
> library(ggplot2)
> ggplot(df, aes(acceptability, fill = acceptability ) ) +
+   geom_bar()
```

Figure 1: Class distribution of the Data set

How safety features impacting the Evaluation of a car? The safety of the car has paramount importance on evaluation of a car. Buyers interest in safety features have long standing history. Many consumers are willing to pay extra for better safety features. Manufacturers were also quick to react to this requirement and focusing on the car safety features more than ever. Car safety rating is something hugely affecting the cars acceptability nowadays(**IIHS**). We can have a quick look at our data set and see how safety having impact on determining the acceptability of a car

```
> library(ggplot2)
> p <- ggplot(df, aes(safety,y=""))
> p +  geom_jitter(aes(colour = acceptability),
+                  width = 0.25,stat="identity")+ facet_wrap(~safety)
>
```

Figure 2: How car safety affecting the acceptability?

Importance of safety is evident from the visualization.we can see All the cars with low safety features was classed unacceptable.The cars with high or medium level of safety have mixture of acceptability.

How Passenger capacity having impact on cars evaluation? Another aspect consumers look before buying car is the passenger capacity. Every consumer has their own requirement's and needs. We can have better understanding about it by visualizing our data.

```
> library(ggplot2)
> p <- ggplot(df, aes(persons, fill = acceptability ) )
> p <- p+facet_wrap(~persons)+geom_bar()
> p
```

Figure 3: How Passenger capacity having impact on cars evaluation?

From the visualized data, it is understandable that cars with low passenger capacity is not very acceptable. perhaps all the cars with 2 passenger capacity classed as unacceptable. the cars which have capacity of 4 or more have higher acceptability.

How buying value having impact on cars evaluation? It is not secret that buying value of the car is one of the most important factor for the evaluation process. We can try to understand the impact by visualizing the data set.

```
> library(ggplot2)
> p <- ggplot(df, aes(buying, fill = acceptability ) )
> p <- p+facet_wrap(~buying)+geom_bar()
> p
```

Figure 4: How buying value having impact on cars evaluation?

It is understandable from the plot that buying price does have same huge impact on acceptability like capacity and safety.But it is noticeable that the cars with high or very high buying value have higher unacceptability level. The cars with medium and low buying value have very similar acceptability level.So cars with very high value more likely to be classes as unacceptable than medium and low valued cars.

2.4 Pre-processing

Before process to classification model it is recommended to check the database for missing value. For this purpose the function called sapply() is very efficient. we are going to use sappy() to check our database for missing values and take appropriate action.

```
> Missing_values <- sapply(df,function(x) sum(is.na(x)))
> Missing_values

      buying        maint        doors      persons
           0            0            0            0
    lug_boot       safety acceptability
           0            0            0
```

We can see that there are no missing values in the data set. So no further action needed and missing value handling methods are not necessary for this report The analyst also considered following facts for pre-processing

1. Given the type and the nature of the data set normalization is not necessary.
2. No new features need to be generated.
3. Irrelevant data features are already been dropped by the source. And there is no feature in the data set which can be dropped to improve the performance
4. Analyst already added the names of features earlier for better visualization and other purposes.

5. The data set is relevantly small.so data reduction or dimension reduction is not necessary

After considering all the above facts the analyst believe that the data set is now ready to be used in classification model.

3 Modelling/Classification

The analyst will use random forest for the classification model in this report. The following facts influence the decision to use the Random forest model

1. High level of accuracy and efficiency
2. Maintain accuracy despite missing data's.
3. Estimate the important variables in the classification
4. Can generate unbiased estimation of the generalization
5. Detect unsupervised clustering and outlier detection
6. Model can be reuse for future data

Some ideas are taken from (**Walker**)

3.1 Test and training subsets

First we will divided the data set into training and testing set.

```
> library(caret)
> #library(lattice)
> #converting class lebels into factors
> df$acceptability <- as.factor(df$acceptability)
> #building testing and training set
> set.seed(10)
> inTrain <- createDataPartition(y=df$acceptability,
+                                p=.7,list=FALSE)
> #this is our training set contains 70% of the original dataset.
> training <- df[inTrain,]
> # this is the testing set containd 30% of the original dataset
> testing <- df [-inTrain,]
```

we can check if any data went missing during the partitioning process

```
> #cheking if any values are missing.
> #it will return true if no data is missing
> nrow(training)+nrow(testing)==nrow(df)

[1] TRUE
```

so we can see no data was lost during the process.

3.2 Building Random Forest Model

Now we will build a basic random forest model with default values for ntree and mtry. we will try to improve this model by using suitable values in them later in improvement chapter.We will use the training data set to train the model.

```
> library(randomForest)
> #setting seed to maintain similiarity of results in every run
> set.seed(122)
> #bulding the model with trainning data set
> RFmodel <- randomForest(acceptability ~., data = training)
> #checking the model
> RFmodel

Call:
 randomForest(formula = acceptability ~ ., data = training)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.88%
Confusion matrix:
      acc good unacc vgood class.error
acc   263    5     0     1  0.02230483
good    9   38     0     2  0.22448980
unacc  18    2   827     0  0.02361275
vgood   9    1     0    36  0.21739130
```

now we can make prediction using the testing data set

```
> #makeing prediction with testing  set
> pred <- predict(RFmodel, testing)
> testing$correctPred <- pred==testing$acceptability
> results <- table(pred,testing$acceptability)
> results

pred    acc good unacc vgood
  acc   114    1     9     1
  good    1   18     0     0
  unacc   0    0   353     0
  vgood   0    1     0    18
```

3.3 Checking the accuracy of the model

we can check the accuracy of the model

```
> # we ca use this alternative manual
> #             calculation to determone the accuaracy too
```

we can visualize the accuracy of our model

```
> library(ggplot2)
> ggplot(testing, aes(acceptability, fill = correctPred ) ) +
+   geom_bar()
```

Figure 5: Visualizing the accuracy of the predictions

```
> #auc <- ((sum of predictions in confusion matrix
> #                 rows systemically)/nrow(testing))*100
> #checking the accuracy of the prediction with testing model
> auc <- (sum(pred ==testing[,7])/nrow(testing))*100
> #displaying the accuracy of the model
> cat("The accuracy of this model is :",auc,"%")

The accuracy of this model is : 97.48062 %
```

it is evident that the accuracy of the model is very high .In the next chapter we will look into ways to improve it even more.

```
> set.seed(25)
> library(randomForest)
> tune.rf <- tuneRF(df[,-7],df[,7], stepFactor=.75)

mtry = 2  OOB error = 3.82%
Searching left ...
mtry = 3         OOB error = 2.49%
0.3484848 0.05
mtry = 4         OOB error = 1.74%
0.3023256 0.05
mtry = 6         OOB error = 1.56%
0.1 0.05
mtry = 8         OOB error = 1.22%
0.2222222 0.05
mtry = 11         OOB error = 1.68%
-0.3809524 0.05
Searching right ...
mtry = 1         OOB error = 18.36%
-14.09524 0.05
```

Figure 6: Fine tune to improve the performance

4 Improving Performance

4.1 Fine tune

There are few ways to improve the performance of the model. One of the widely used technique is to fine tune the parameters of the model(**Tavish**).we will try to fine tune the parameters of our model.

4.1.1 Optimizing mtry

one of the most popular and successful way to tune a random forest model is to optimizing mtry. This is related to splitting the variable for each tree. We can first check which mtry will optimize the model most the apply that number to the model.

we can see that mtry=6 would give us a better efficient.we can apply it to our model and observe the result

```
> library(randomForest)
> set.seed(25)
> RFmodel_tune <- randomForest(acceptability ~.,
+                   data = training,mtry=6)
> #makeing prediction with testing set
> #set.seed(25)
> pred <- predict(RFmodel_tune, testing)
> testing$correctPred <- pred==testing$acceptability
> results <- table(pred,testing$acceptability)
> results
```

```
pred    acc good unacc vgood
  acc   113    0     7     1
  good    0   19     0     0
  unacc   2    0   355     0
  vgood   0    1     0    18
```

```
> #checking the accuracy of the prediction with testing model
> auc <- (sum(pred ==testing[,7])/nrow(testing))*100
> #displaying the accuracy of the model
> cat("The accuracy of this model is :",auc,"%")
```

```
The accuracy of this model is : 97.86822 %
```

we can see some excellent improvement in the accuracy by optimizing mtry.

4.1.2 Optimizing ntrees

Another way to improve the performance of the of the model is to optimize ntree.The idea was taken from the rgu lecture and lab slides.

```
> #converting class lebels into factors
> df$acceptability <- as.factor(df$acceptability)
> #building testing and training set
> set.seed(10)
> inTrain <- createDataPartition(y=df$acceptability,
+                                p=.7,list=FALSE)
> ##this is our training set contains 70% of the original dataset.
> training <- df[inTrain,]
> # this is the testing set containd 30% of the original dataset
> testing <- df [-inTrain,]
> # create dataframe to store accuracies and
> #             corresponding number of trees
> df_result <- data.frame(NTrees=as.numeric(),
```

```
+                  Accuracy=as.numeric())
> ## fit randomForest 10 times, and store the results
> ## vary number of trees in every iteration
> ##number of initial tree
> ntrees <- 100
> for (i in 1:10){
+   set.seed(415)
+   RFModel2 <- randomForest(
+     training[,-7],
+     training[,7],
+     xtest=testing[,-7],
+     ytest=testing[,7],
+     ntree=ntrees,
+     proximity=TRUE
+   )
+ ## Test the RF model for this run
+ preds <- levels(training[,7])[RFModel2$test$predicted]
+ ## compute accuracy
+ auc <- (sum(preds ==testing[,7])/nrow(testing))*100
+ df_result <- rbind(df_result,
+                 data.frame(NTrees=ntrees,Accuracy=auc))
+ ntrees <- ntrees + 200
+ }
> ## end for loop
> df_result

   NTrees Accuracy
1     100 96.89922
2     300 96.70543
3     500 96.70543
4     700 96.51163
5     900 96.51163
6    1100 96.51163
7    1300 96.70543
8    1500 96.89922
9    1700 96.51163
10   1900 96.70543
```

we can see which ntree optimize the model most and use it to improved the performance of the model.

4.2 Optimizing with different partitioning

Another way to improve the model it change the partitioning of the data set and see if a improved result can be achieved.we can change our partitioning to 80 percent for training set and 20 percent to testing set.

```
> library(caret)
> library(randomForest)
> df$acceptability <- as.factor(df$acceptability)
> #creating test and train dataset
> set.seed(11)
> inTrain <- createDataPartition(y=df$acceptability,
+                                p=.8,list=FALSE)
> ##this is our training set contains 80% of the original dataset.
> training1 <- df[inTrain,]
> # this is the testing set containd 20% of the original dataset
> testing1 <- df [-inTrain,]
> #seeting up for reproduceable document
> set.seed(400)
> #building the model
> RFmodel_dp <- randomForest(acceptability ~., data = training1)
> RFmodel_dp

Call:
 randomForest(formula = acceptability ~ ., data = training1)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.03%
Confusion matrix:
      acc good unacc vgood class.error
acc   302    3     1     2  0.01948052
good    4   49     0     3  0.12500000
unacc  21    1   946     0  0.02272727
vgood   7    0     0    45  0.13461538

> #makeing prediction with testing set
> pred <- predict(RFmodel_dp, testing1)
> testing1$correctPred <- pred==testing1$acceptability
> results <- table(pred,testing1$acceptability)
> results

pred    acc good unacc vgood
  acc    75    4     5     2
  good    1    7     0     0
  unacc   0    0   236     0
  vgood   0    2     0    11

> #checking the accuracy of the prediction with testing model
> auc <- (sum(pred ==testing1[,7])/nrow(testing1))*100
> #displaying the accuracy of the model
> cat("The accuracy of this model is :",auc,"%")
```

```
The accuracy of this model is : 95.91837 %
```

we can see the difference in performance and justify if making changes to partition will improve the performance or not.This time it is making a noticeable improvement.

4.3 Cross validation

we can implement cross validation to check if that improve the performance of the model.

```
> library(caret)
> library(randomForest)
> library(e1071)
> df$acceptability <- as.factor(df$acceptability)
> set.seed(10)
> inTrain <- createDataPartition(y=df$acceptability,
+                                p=.7,list=FALSE)
> ##this is our training set contains 70% of the original dataset.
> training <- df[inTrain,]
> # this is the testing set containd 30% of the original dataset
> testing <- df [-inTrain,]
> #cross validation
> # define training control
> train_control<- trainControl(method="cv", number=5)
> # train the model
> set.seed(205)
> RFmodel_cv <- train(acceptability ~., data = training,
+                          trControl=train_control,
+                          method="lda", family=binomial())
> #makeing rediction with testing set
> #set.seed(420)
> pred <- predict(RFmodel_cv, testing)
> testing$correctPred <- pred==testing$acceptability
> results <- table(pred,testing$acceptability)
> results

pred    acc good unacc vgood
  acc   109   12    23     7
  good    2    7     0     1
  unacc   4    0   338     0
  vgood   0    1     1    11

> #checking the accuracy of the prediction with testing model
> auc <- (sum(pred ==testing[,7])/nrow(testing))*100
> #displaying the accuracy of the model
> cat("The accuracy of this model is :",auc,"%")
```

then i plot the data for visualization

```
> library(ggplot2)
> library(scales)
> x <-  ggplot(df_compare, aes(x=models, y=accuracy,
+                              colour=accuracy, fill=accuracy))
> x <-  x+geom_bar(stat="identity")
> x <-  x+ geom_text(aes(label=accuracy),size = 3,
+           position = position_stack(vjust = 1.1))
> x <-  x+ylab("accuracy of the model")
> x
```

Figure 7: Different models comparison

we notice the different accuracy level for different version of random forest models

```
The accuracy of this model is : 90.11628 %
```

we can see that cross validation does not make any improvement to our model.So for better performance this model does not require cross validation.

4.4 Comparing different improvement models with the original model

we witnessed different results for different models.some improve the accuracy of the model and some did not.we can visualize the accuracy of different models to have a better understanding.to do that i first created a database and store the models name and percentage of accuracy

```
> #createing a database with model name and accuracy
> models = c("RFmodel","RFmodel_tune","RFmodel_dp","RFmodel_cv")
> accuracy=c(95.93,98.06,98.25,87.79)
> df_compare=data.frame(models,accuracy)
> df_compare

        models accuracy
1      RFmodel    95.93
2 RFmodel_tune    98.06
3   RFmodel_dp    98.25
4   RFmodel_cv    87.79
```

5 Conclusion

After analyzing and visualizing the data set it is understandable that many features such as size, value and safety have impact on the cars evaluation process. And it is possible to predict the class of a car with the information about its different features. Using random forest to build a classification model for this data set was a very productive approach. The basic random forest model was able to predict more than 95 percent of the classes correctly.

from improvement section of this coursework it is evident that fine tune, cross validation or different partitioning can be useful to find the best performing model. For this data set cross validation did not improve the performance but fine tune and different partitioning did. It is also evident that to follow the right procedure is essential to achieve the finest result. Overall it is understandable that random forest model is very efficient for this type of data set.

6 References and Bibliography