Cars Selling Price : Selling Price Prediction for Used Cars using Cardekho.com dataset

Arun Kataria

5/31/2021

Introduction : Project

This is an R Markdown document that contains the code & report to showcase the developments in achieving a statistically usable prediction model that predicts the reasonable selling price of used cars based on various factors like age of car, km driven, cost price of new car, engine power, mileage etc.

The statistical viability of this model and various other models developed during the modeling to predict the selling price is evaluated based on the RMSE (Root Mean Square Error) method. The Goal is to reduce the RMSE as much as i can.

This used car selling price prediction model takes in the dataset from kaggle (cardekho.com dataset) The model can easily be used by users to define a reasonable price they can expect for their car. This model can also be used by used car selling site to define a range for the car being sold. This range will help regulate the price of car and hence attract the buyers as buyers can now expect a reasonable and consistent price range for cars based on facts instead of just sellers thoughts.

Method/Analysis :

Overview of DataSet (train_set): The used cars dataset has further been divided into train_set(80%) and test_set(20%) to train and test various models. Hereafter we will be using the train_set for training of models and test_set to test the RMSE of various models.

The train_set dataset contains 7537 records.

Following is the structure of the source Dataset 7537 rows and 19 columns. The columns of the dataset available are id, car_name, brand, model, new_price, min_cost_price, max_cost_price, vehicle_age, km_driven, seller_type, fuel_type, transmission_type, mileage, engine, max_power, seats, selling_price, mcp, avg_cost_price.

The dependent variable or the variable that we are interested to predict here is : **selling price**

The independent variables or the variables that we will be using to predict selling price are: - vehicle_age - km_driven - mileage - max_power - engine - min_cost_price & max_cost_price

```r
head(train_set,10)
```

A Basic exploration of the data in the train_set

```
##     id              car_name         brand       model
##  1:  4         Ford Ecosport          Ford    Ecosport
##  2:  5        Maruti Wagon R        Maruti     Wagon R
##  3:  6           Hyundai i10       Hyundai         i10
##  4:  7        Maruti Wagon R        Maruti     Wagon R
##  5:  8         Hyundai Venue       Hyundai       Venue
##  6: 12          Maruti Swift        Maruti       Swift
##  7: 14         Hyundai Verna       Hyundai       Verna
##  8: 19 Mercedes-Benz C-Class Mercedes-Benz     C-Class
##  9: 22         Maruti Baleno        Maruti      Baleno
## 10: 23    Maruti Swift Dzire        Maruti Swift Dzire
##                                           new_price min_cost_price
##  1: New Car (On-Road Price) : Rs.10.14-13.79 Lakh*        1014000
##  2:   New Car (On-Road Price) : Rs.5.16-6.94 Lakh*         516000
##  3:   New Car (On-Road Price) : Rs.6.54-6.63 Lakh*         654000
##  4:   New Car (On-Road Price) : Rs.5.26-7.01 Lakh*         526000
##  5:  New Car (On-Road Price) : Rs.7.70-13.02 Lakh*         770000
##  6:   New Car (On-Road Price) : Rs.6.35-9.27 Lakh*         635000
##  7: New Car (On-Road Price) : Rs.13.09-18.29 Lakh*        1309000
##  8: New Car (On-Road Price) : Rs.51.30-64.08 Lakh*        5130000
##  9:  New Car (On-Road Price) : Rs.6.81-10.54 Lakh*         681000
## 10:  New Car (On-Road Price) : Rs.6.98-10.40 Lakh*         698000
##     max_cost_price vehicle_age km_driven seller_type fuel_type
##  1:        1379000           6     30000      Dealer    Diesel
##  2:         694000           8     35000  Individual    Petrol
##  3:         663000           8     40000      Dealer    Petrol
##  4:         701000           3     17512      Dealer    Petrol
##  5:        1302000           2     20000  Individual    Petrol
##  6:         927000           4     28321      Dealer    Petrol
##  7:        1829000           8     65278      Dealer    Diesel
##  8:        6408000           7     65000      Dealer    Diesel
##  9:        1054000           6     20000  Individual    Petrol
## 10:        1040000           5     40000  Individual    Petrol
##     transmission_type mileage engine max_power seats selling_price     mcp
##  1:            Manual   22.77   1498     98.59     5        570000 1014000
##  2:            Manual   18.90    998     67.10     5        350000  516000
##  3:            Manual   20.36   1197     78.90     5        315000  654000
##  4:            Manual   20.51    998     67.04     5        410000  526000
##  5:         Automatic   18.15    998    118.35     5       1050000  770000
##  6:            Manual   16.60   1197     85.00     5        511000  635000
##  7:            Manual   22.32   1582    126.32     5        425000 1309000
##  8:         Automatic   19.27   2143    170.00     5       1425000 5130000
##  9:            Manual   21.40   1197     83.10     5        600000  681000
## 10:            Manual   20.85   1197     83.14     5        575000  698000
##     avg_cost_price
##  1:        1196500
##  2:         605000
##  3:         658500
##  4:         613500
##  5:        1036000
##  6:         781000
##  7:        1569000
##  8:        5769000
##  9:         867500
## 10:         869000
```

Lets start analyzing the data more:

Data Analysis & Modeling

Average Selling Price = 8.1426839×10^5

First Model

At first we started with the mean selling price across the complete train_set and tested out the RMSE of this simple model.

```
mu_hat <- mean(train_set$selling_price)
mu_hat
```

```
## [1] 814268.4
```

```
#Validating the RMSE value for simple mean, matching with selling price in test set
naive_rmse <- RMSE(test_set$selling_price, mu_hat)
naive_rmse
```

```
## [1] 881346.8
```

```
# A RMSE results table/dataframe created to store RMSEs of various models as we analyze it
rmse_results <- data_frame(method = "Simple average", RMSE = naive_rmse)
```

```
## Warning: ‘data_frame()‘ is deprecated as of tibble 1.1.0.
## Please use ‘tibble()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```
# Print rmse results for simple average - A baseline RMSE
rmse_results
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Simple average 881347.
```

Lets see if we can improve the RMSE of this model using other independent variables available. We will start using the Machine learning alogrithms that use other independent variables to predict the selling price. Lets start by using the Linear regression model "lm". We will use the caret package, train function to train various models in a similar fashion.

Second Model

```
# Linear regression model to predict the selling price
fit1<- train(selling_price ~ vehicle_age + avg_cost_price + km_driven + max_power
             + mileage, data=train_set, method="lm", na.action = na.omit)

fit1
```

```
## Linear Regression
##
## 7537 samples
##    5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 7537, 7537, 7537, 7537, 7537, 7537, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   472597.3  0.6901581  247845.5
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# check the importance of factors used in model
varImp(fit1)
```

```
## lm variable importance
##
##                Overall
## vehicle_age    100.000
## max_power       87.070
## avg_cost_price  63.889
## km_driven        6.925
## mileage          0.000
```

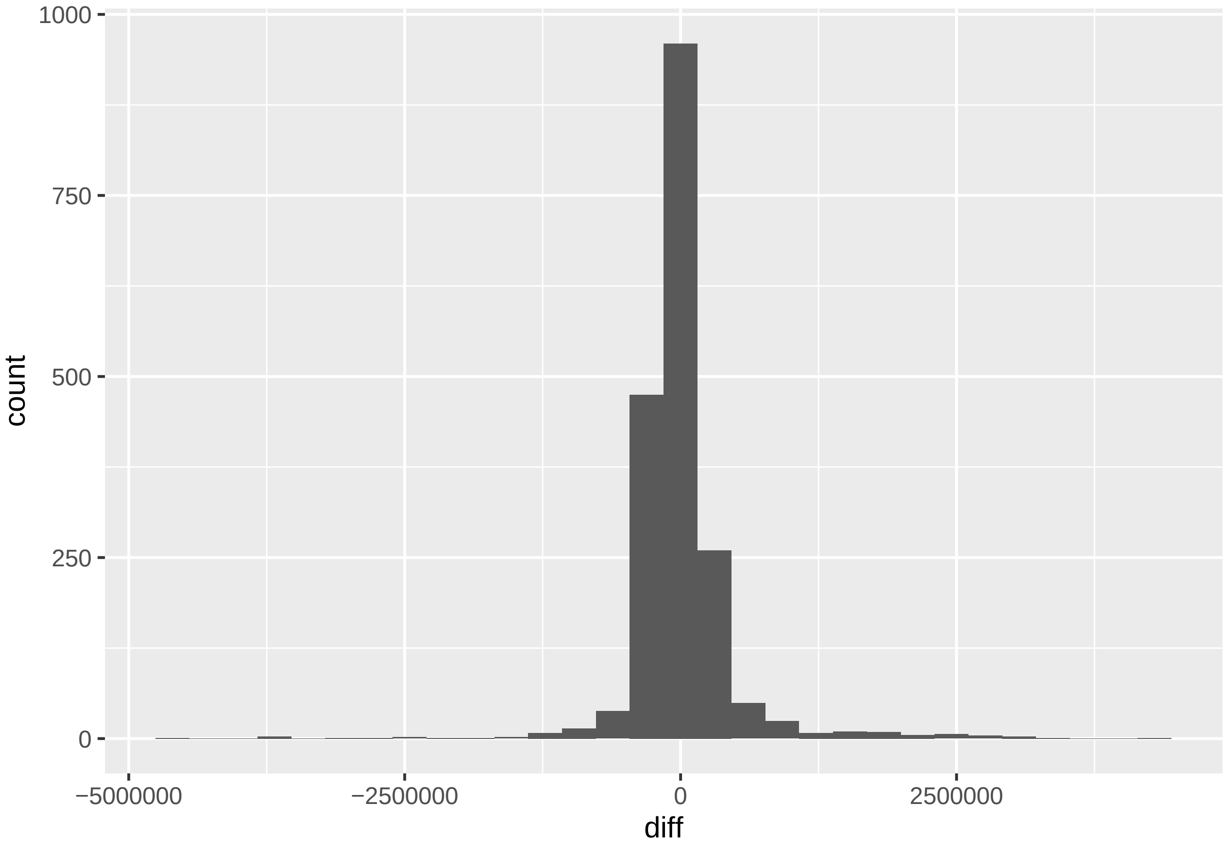
```
# predict the outcome/selling price of cars in test_set using the above model
# (linear regression)
p2<-predict(fit1,newdata=test_set)

# calculate the Root mean square error
lm_rmse<-RMSE(p2,test_set$selling_price)

# Calculate and plot the residual value, histogram
test_set<- test_set %>% mutate(p=p2,diff=selling_price-p2)
test_set %>% arrange(desc(abs(diff))) %>%  select(selling_price, p, diff) %>% head(5)
```

```
##    selling_price       p     diff
## 1:       5200000 9731455 -4531455
## 2:       7985000 3619413  4365587
## 3:       1285000 5077780 -3792780
## 4:        750000 4491420 -3741420
## 5:       1751000 5339233 -3588233
```

```
test_set %>% ggplot(aes(diff)) + geom_histogram()
```



```
# append/add the RMSE value results to rmse_results dataframe
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Linear regression Model",
                                     RMSE = lm_rmse ))

# View the RMSE values just added for Linear regression Model
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	881346.8
Linear regression Model	490799.7

Let us try using other machine learning models and see if we can improve the RMSE values further. We will try out with using rpart or decision tree to predict the selling price.

Third Model

```
# using decision tree - rpart ml algorithm to see if we can get better RMSE values
fit3<-train(selling_price ~ vehicle_age + engine + avg_cost_price + km_driven
            + max_power + mileage, data=train_set, method="rpart",
            na.action = na.omit, tuneGrid = data.frame(cp= seq(0, 0.05, 0.002)))
```

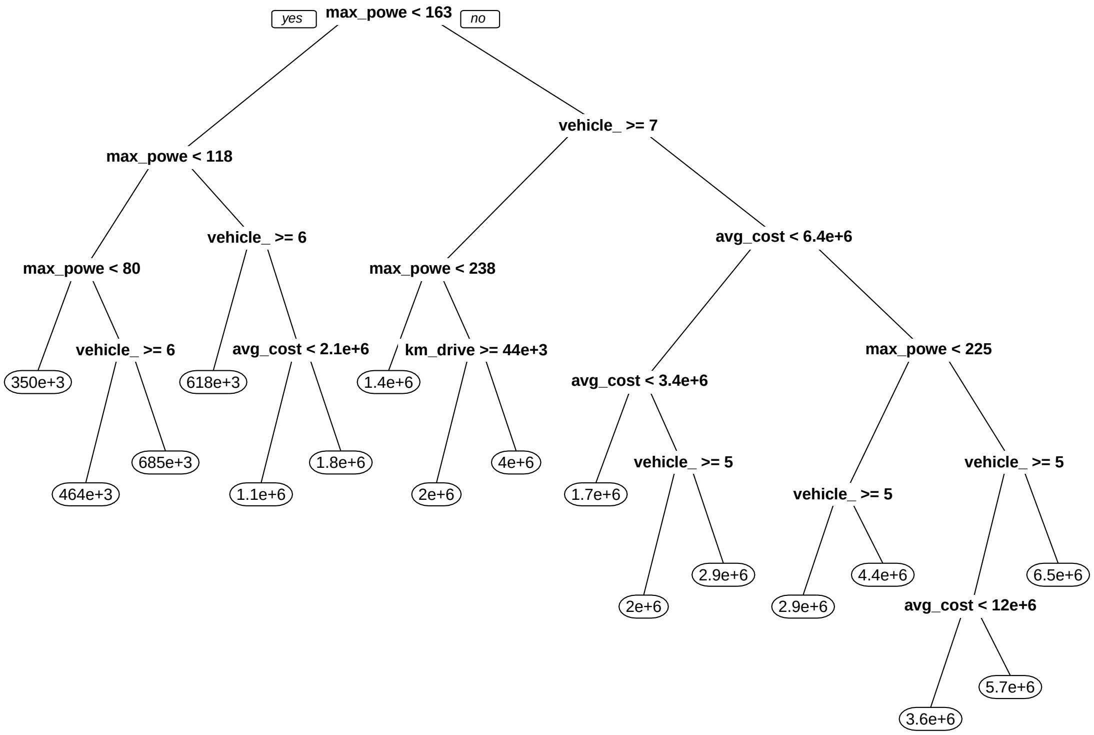
```
# list down all cp's and corresponding R^2 & RMSE values
fit3
```

```
## CART
##
## 7537 samples
##    6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 7537, 7537, 7537, 7537, 7537, 7537, ...
## Resampling results across tuning parameters:
##
##   cp     RMSE      Rsquared   MAE
##   0.000  263033.0  0.9074357  116538.5
##   0.002  307579.4  0.8733840  166031.2
##   0.004  334382.0  0.8501779  182044.2
##   0.006  352829.1  0.8331556  198710.2
##   0.008  367674.2  0.8189613  208513.7
##   0.010  375177.1  0.8118339  212142.9
##   0.012  386812.1  0.7997599  219482.0
##   0.014  394771.3  0.7913750  227640.9
##   0.016  404325.4  0.7811239  236919.0
##   0.018  414771.2  0.7694978  241969.4
##   0.020  427493.0  0.7550026  253949.8
##   0.022  436527.5  0.7446675  260738.6
##   0.024  446052.2  0.7332931  265097.6
##   0.026  453428.8  0.7247927  271517.8
##   0.028  461561.9  0.7149662  279400.2
##   0.030  465795.0  0.7094316  281392.6
##   0.032  468986.8  0.7053923  282611.3
##   0.034  474783.7  0.6980967  285678.2
##   0.036  475848.6  0.6966115  286145.2
##   0.038  482289.6  0.6880645  291429.3
##   0.040  485735.3  0.6835767  293911.7
##   0.042  491865.7  0.6754547  299249.6
##   0.044  495600.6  0.6705104  301358.3
##   0.046  500640.3  0.6643995  304514.7
##   0.048  503308.9  0.6609271  305739.6
##   0.050  503308.9  0.6609271  305739.6
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.
```

```
# final model / best fit from all the predicted ones/ commented as its a long output
# fit3$finalModel

# prune the tree using cp=0.006, making a little easy to apprehend
new_tree<-prune(fit3$finalModel,cp=0.006)

# plot the decision tree
prp(new_tree)
```



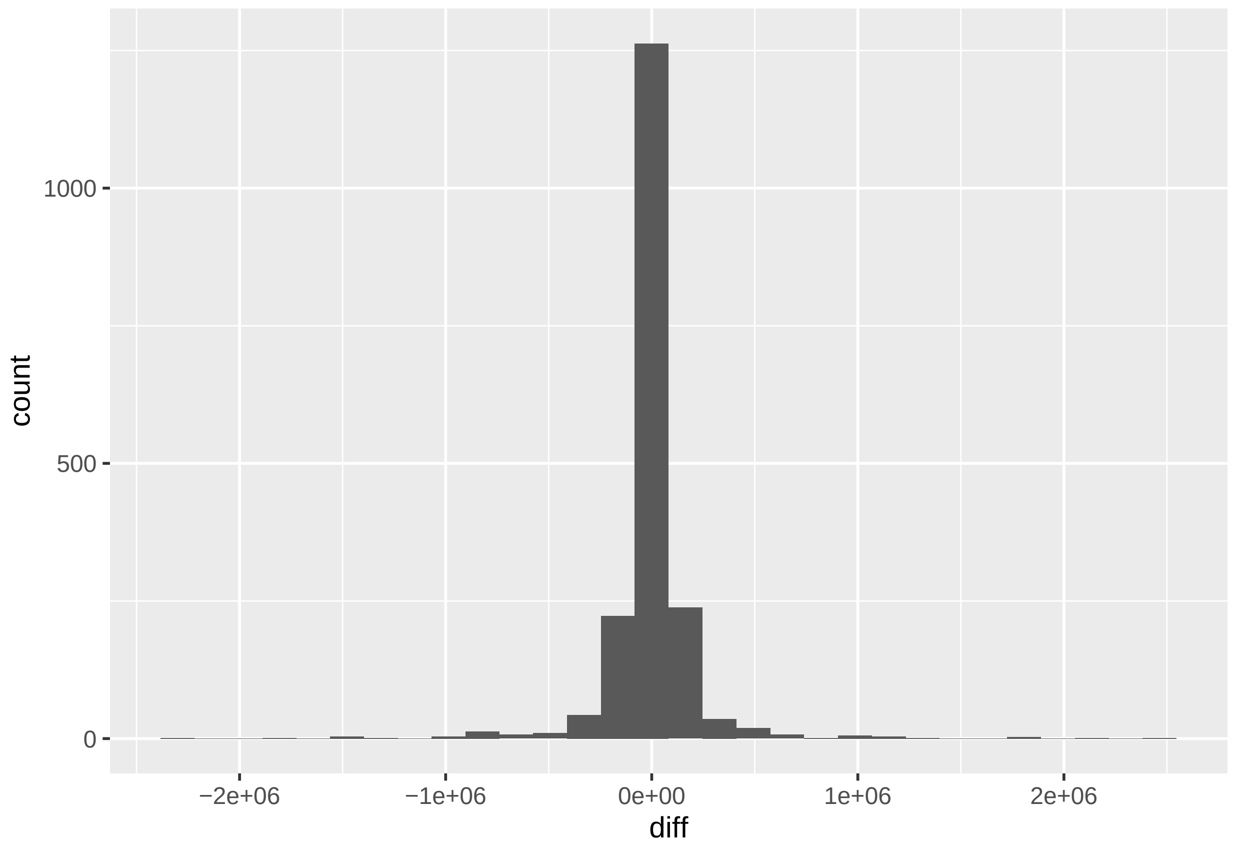
```
# predict the selling price using about decision tree model and test data set
p2<-predict(fit3,newdata=test_set)

# Evaluate the Root mean square error values
rpart_rmse<-RMSE(p2,test_set$selling_price)

# calculate and plot the residual values in histogram
test_set<- test_set %>% mutate(p=p2,diff=selling_price-p2)
test_set %>% arrange(desc(abs(diff))) %>%  select(selling_price, p, diff) %>% head(5)
```

```
##    selling_price       p     diff
## 1:       5375000 2920000  2455000
## 2:       5600000 7909917 -2309917
## 3:       4750000 2662158  2087842
## 4:       4500000 2662158  1837842
## 5:       4000000 2173375  1826625
```

```
test_set %>% ggplot(aes(diff)) + geom_histogram()
```



```
# Test out the RMSE with the latest predicted selling price
# and append these to existing rmse_results dataframe
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Decision Tree Model",
                                     RMSE = rpart_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	881346.8
Linear regression Model	490799.7
Decision Tree Model	226682.9

This has further improved the RMSE value, the model looks good.

Fourth Model : Random Forest

Lets proceed further to see if we can use the random forest ensemble algorithm to predict the values better.

```
# Let us see if we can use random forest algorithm and further improve the RMSE

# Setting up the control parameters and tuning grid parameter for random forest algo
# using repeated control validation, with 3 repeats (random)
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
```

```r
# setting up mtry / no of attributes to try before split as a sequence from 1 to 10
tunegrid <- expand.grid(.mtry = (1:5))

#train_set2<-head(train_set,100)

# Train the random forest ml algo to predict the selling price of used car in train set
fit4<-train(selling_price ~ vehicle_age + avg_cost_price + km_driven + max_power +mileage,
            data=train_set, method="rf", na.action = na.omit, tuneGrid = tunegrid)

# display the importance of attibutes
varImp(fit4)
```

```
## rf variable importance
##
##                Overall
## max_power       100.00
## avg_cost_price   63.85
## vehicle_age      31.00
## km_driven        15.35
## mileage           0.00
```

```r
# check to see what mtry value is the best fit and R^2 values
fit4
```

```
## Random Forest
##
## 7537 samples
##    5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 7537, 7537, 7537, 7537, 7537, 7537, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   1     215610.6  0.9423706  100036.94
##   2     196585.0  0.9488842   92151.56
##   3     196455.0  0.9484169   92362.53
##   4     198387.1  0.9472467   93220.53
##   5     201904.6  0.9452617   94635.19
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 3.
```

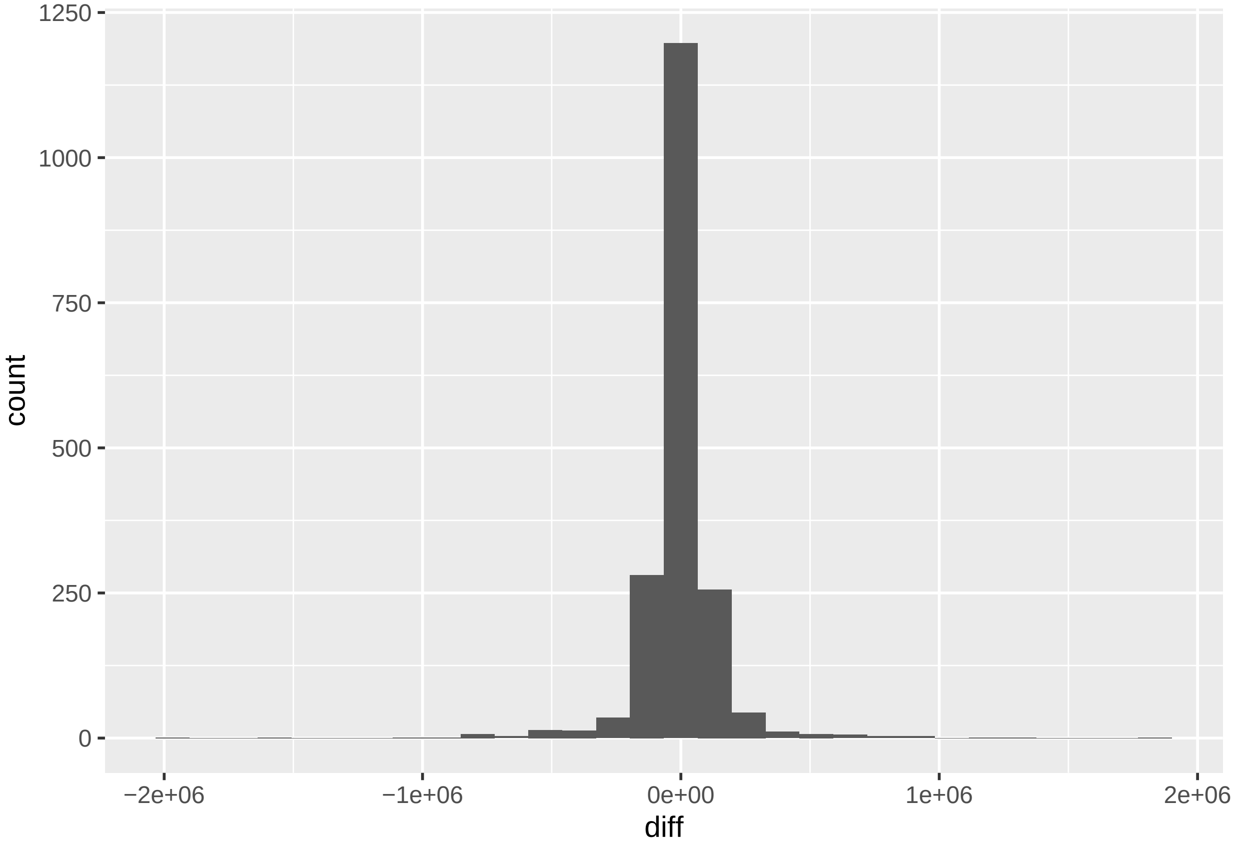
```r
# predict the selling price in test data set using above random forest trained model
p2<-predict(fit4,newdata=test_set)

# calculate the Root Mean square value using test data set/
# comparing predicted selling price with actual selling price
rf_rmse<-RMSE(p2,test_set$selling_price)
```

```
# Calculate the residual value and plot the same in histogram
test_set<- test_set %>% mutate(p=p2,diff=selling_price-p2)
test_set %>% arrange(desc(abs(diff))) %>%  select(selling_price, p, diff) %>% head(5)
```

```
##    selling_price       p     diff
## 1:       2200000 4133149 -1933149
## 2:       4200000 2330604  1869396
## 3:       5100000 6642707 -1542707
## 4:       5950000 4651767  1298233
## 5:       5375000 4151081  1223919
```

```
test_set %>% ggplot(aes(diff)) + geom_histogram()
```



```
# Calcuate the RMSE value of predicted selling price
# and add these to the rmse_results dataframe
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Random Forest Model",
                                     RMSE = rf_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	881346.8
Linear regression Model	490799.7

method	RMSE
Decision Tree Model	226682.9
Random Forest Model	162611.0

From the above modeling and predictions we are able to see that the factors like average cost price of new vehicle, km driven, max_power, mileage, vehicle age etc have a great impact on the selling price of vehicle. An important point to note here is that the brand of vehicle is explicitly not taken into account. we have a tendency to be biased for a particular car maker / brand (based on our liking/previous experience) but instead it is tried here to predict the selling price just based on the vital statistics of the automobile rather than brand name.

Result

Displaying below the final list of models that we tried and also the minimum RMSE values that we achieved using these models.

```r
# Display the final RMSE of the Model
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	881346.8
Linear regression Model	490799.7
Decision Tree Model	226682.9
Random Forest Model	162611.0

Below is the Final model that is created using vehicle_age + avg_cost_price + km_driven + max_power + mileage

The final model is tested out using the test data set and it performs well in predicting selling price.

```r
###########################
# Final Model : Random Forest Model #

Final_Validation_rmse<-rf_rmse
Final_Validation_rmse
```

```
## [1] 162611
```

The Final RMSE using the Validation dataset is 1.6261102×10^5

Conclusion

Here in this project we have learned that with just the tangible few parameters we are able to predict the selling price of used cars. Various models perform differently for different type of problem. The choice of best model depends upon various factors like in this case the data can be transformed into a classification problem (buckets of selling price range) or linear regression problem (continuous). Here we have seen that random forest model works the best for this dataset as the RMSE value, the evaluating criteria is best in this case for random forest. A great piece of learning, on how to use and evaluate different Machine Learning algorithms.

The final RMSE score for this model using validation set is : 1.6261102×10^5

Scope of future enhancements :

There is still a good scope in improving the overall predicted selling price / overall RMSE of the model.

From a users perspective more parameters can be rearched upon and fetched like location, history of insurance, no of owners, no of services, paint color, etc . These parameters can definitely help in improving the RMSE of the model. Although the volume of dataset is not huge, still it takes a good 3+ hours to run on laptop draining almost all of resources. The hardware limitations also restricts from trying out various different things and in turn hinders the tuning efforts.

Another major improvement can be availability of actual price at which the car is sold instead of listed selling price (or seller asking price). Incase the actual sold price is available it can help tune the model further as it will direct/suggest users that per the model their cars actual selling price is x instead of their asking price.

Thank You. Arun