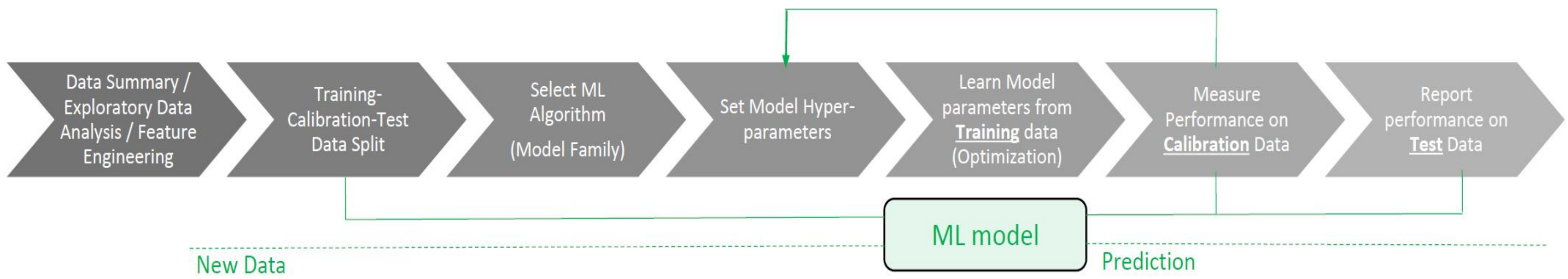
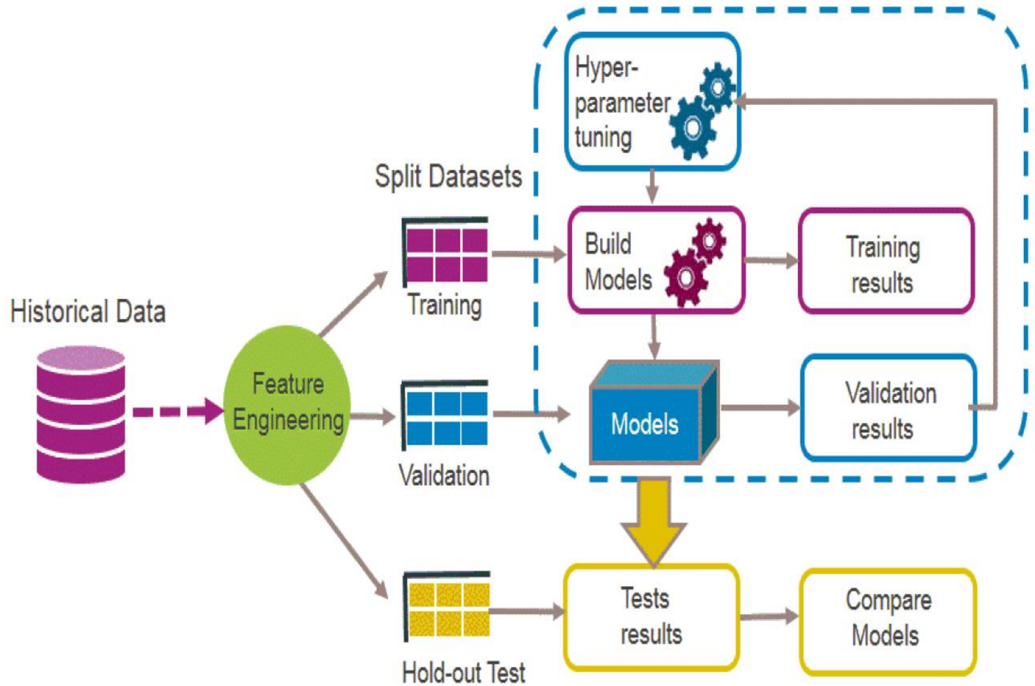
Predictive Modelling to determine Vehicle Prices

Regression



Data Summary / Exploratory Data Analysis / Feature Engineering
Training-Calibration-Test Data Split
Select ML Algorithm
(Model Family)
Set Model Hyper-parameters
Learn Model parameters from Training data (Optimization)
Measure Performance on Calibration Data
Report performance on Test Data
ML model
New Data
Prediction



Historical Data
Feature Engineering
Split Datasets
Training
Validation
Hold-out Test
Hyper-parameter tuning
Build Models
Training results
Models
Validation results
Tests results
Compare Models

Problem Statement

Given the historical data, the aim is to build a predictive model to determine the price of used car vehicles in India.

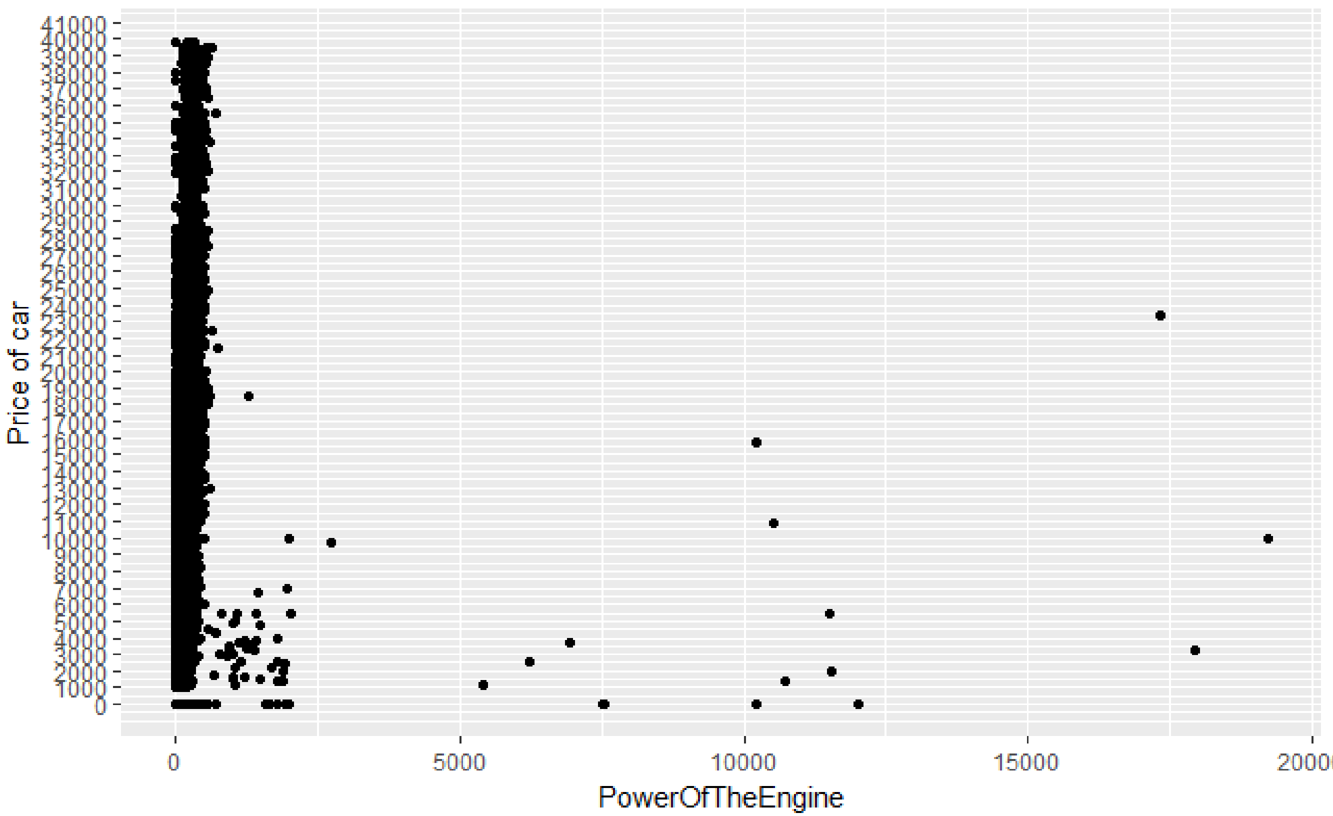
- From the problem statement it is evident that it is a regression problem.
- Segments benefitted : Car manufacturers, car dealers, banks, individuals

Data Preprocessing

- Changed the data types to numeric, factors, and Strings as per given data description.
- Removed the Data columns
- Removed the NameOfTheVehicle String column
- Removed the columns exhibiting NearZero Variance
- Removed the ID and Zipcode columns

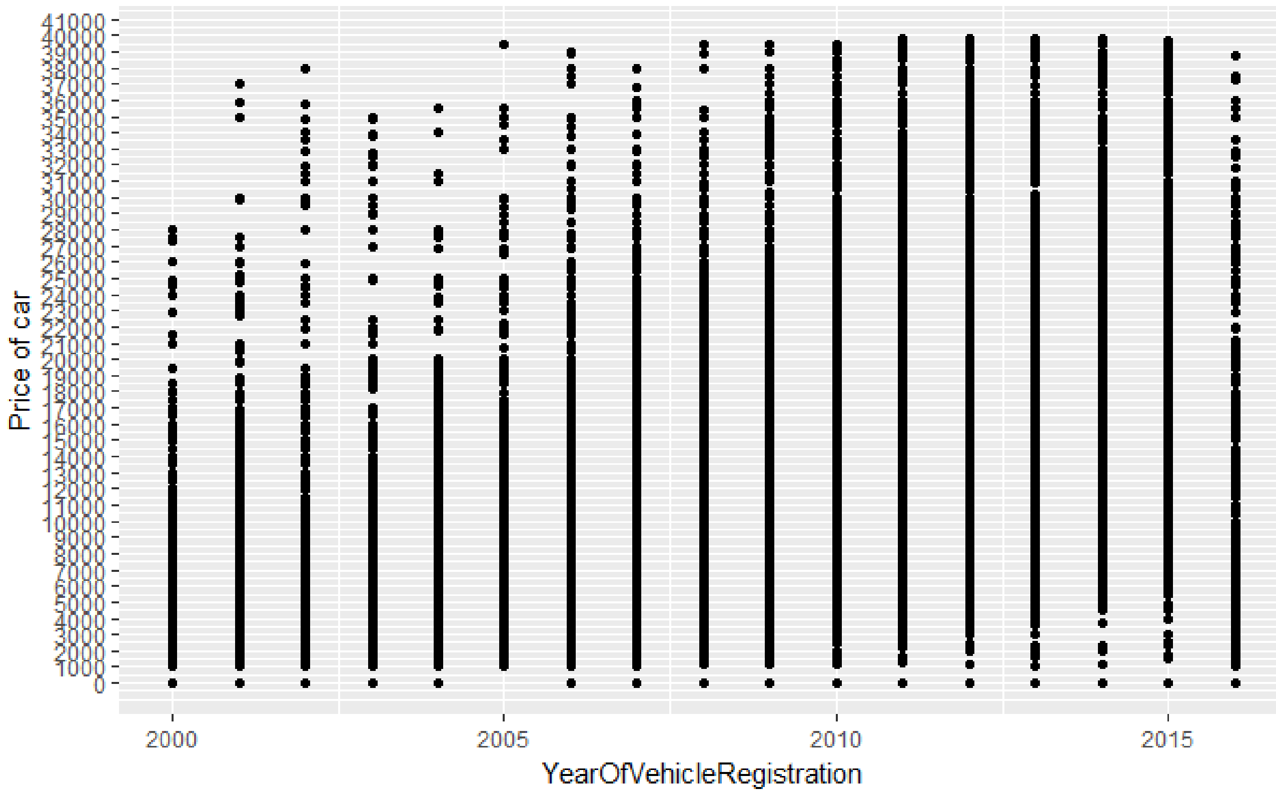
Data Exploration

- Most of the Engine Powers associated with higher power don't have higher price range. ➔ Outliers



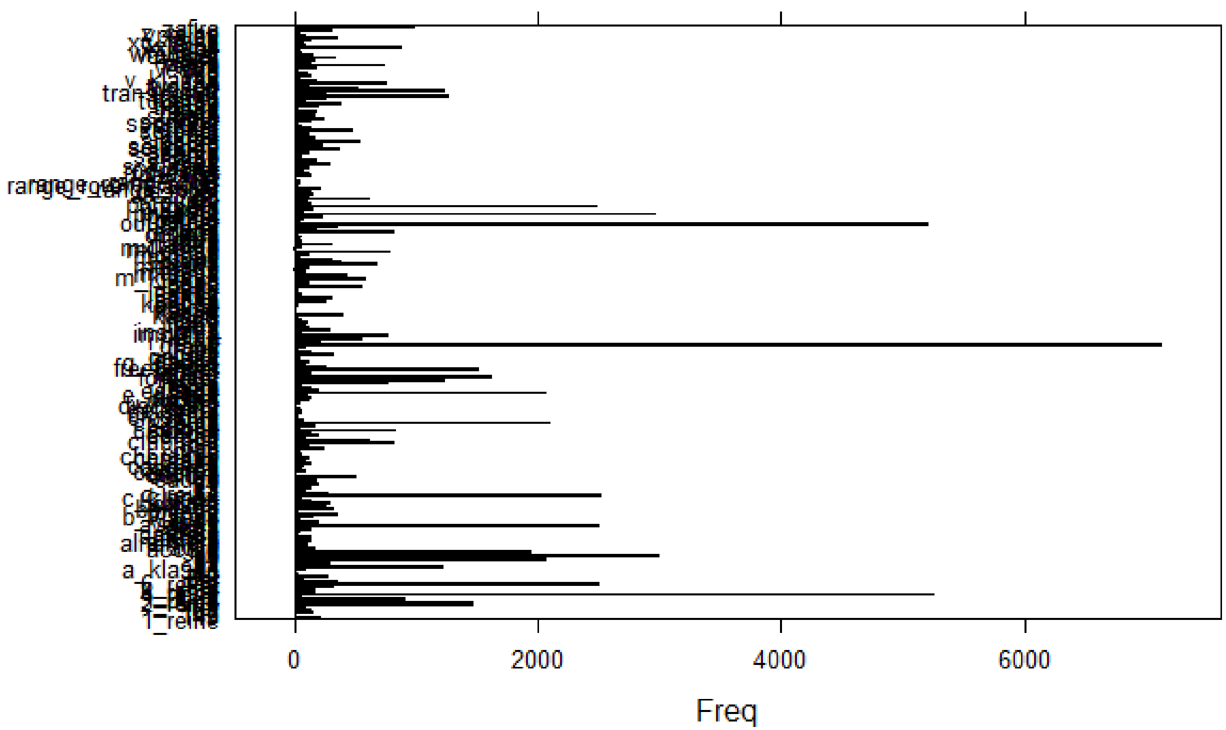
Data Exploration

- Older the Vehicle Registration, lower the price



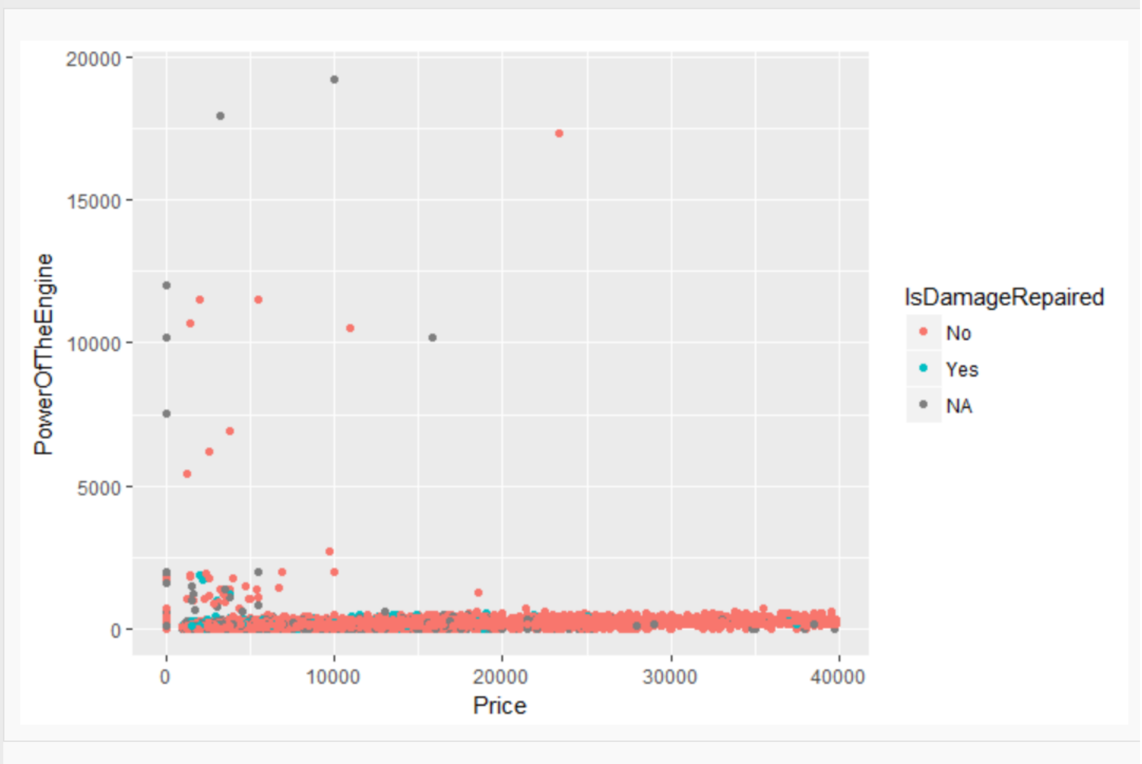
Data Exploration

- ModelOfTheVehicle Frequency distribution. => Decided to make factors, and combine those with frequency < 3% to reduce the levels.



Data Exploration

- Very few observations with IsDamageRepaired as “Yes” => But decided to keep this attribute



Data Cleaning

- Dataset (train + test) split into numeric and categorical separately.
- Checked for numeric predictors having high correlations (>0.80)
- No missing value was found in numerical attributes
- Created a new factor for missing categorical attributes
- Combine categorical variables which have several levels with low frequencies (<=3%) and created new factor "Other"
- Standardized the numeric data

Preparation of Train/Test Data

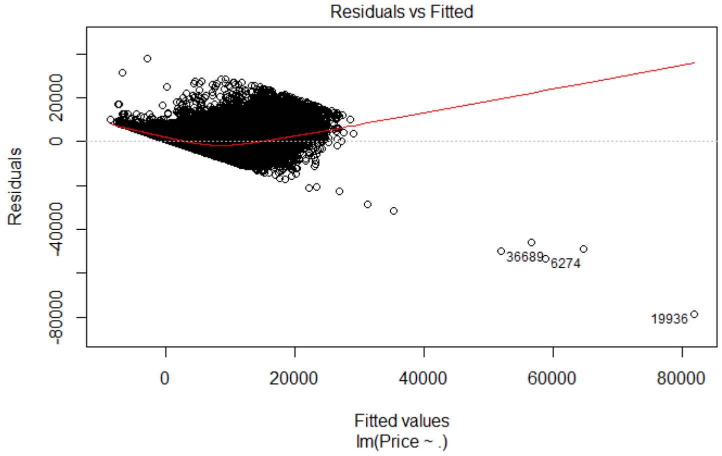
- Test data : New Test data
- Train/Valid Data : Combined the old test data and original training data, and randomly split the data into ratio 70-30 using library caTools
- Created entire data into numeric format, using model.matrix
- Final Dimesions : 44

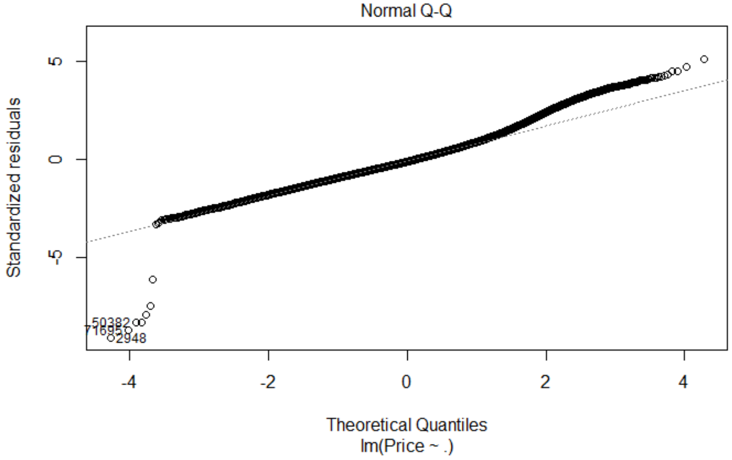
Model Intuition

- I decided to adopt a workflow testing several different models and check the respective error metrics.
  - Models Built :
    - Linear Regression
    - CART
    - Random Forest
    - XGBoost
- After that I used stacking to check if model MAPE improves.

Linear Regression

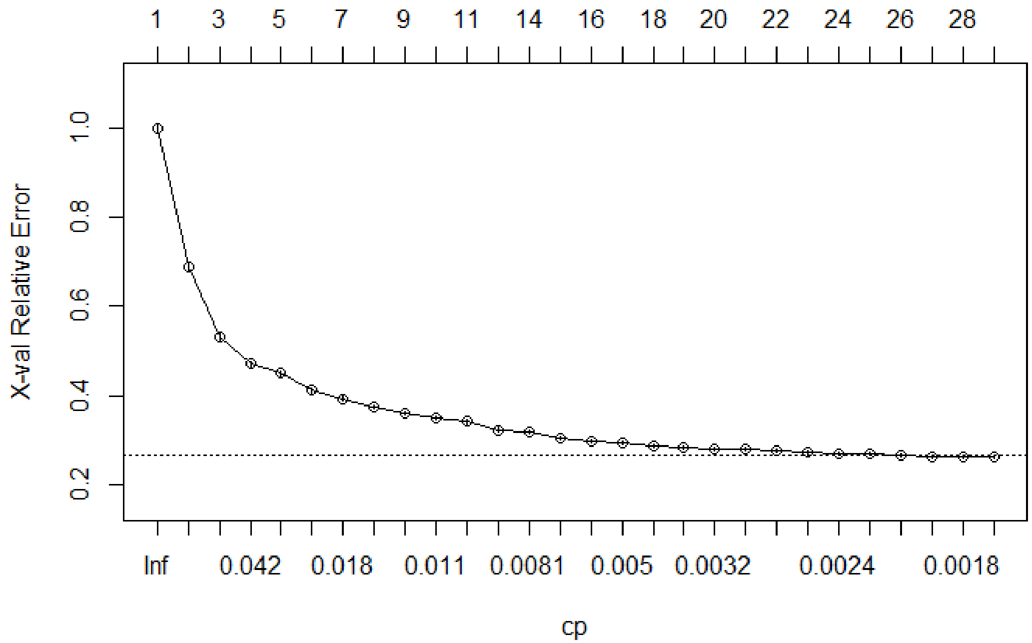
- The model was firstly built and then updated by removing observations based on cooks distance.
- The linearity assumptions didn't hold valid.





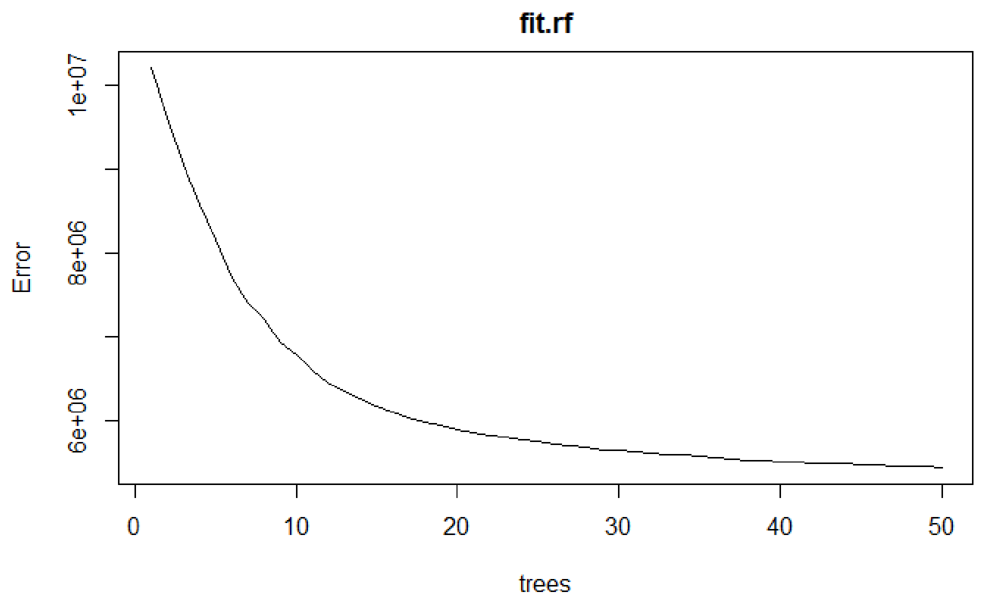
CART

- Built and tuned the cp parameter to prune the tree



Random Forest

- Tuned the ntree and mtry parameters to obtain optimal metrics.
- % Var explained: 88.09



Extreme Gradient Boosting using DT

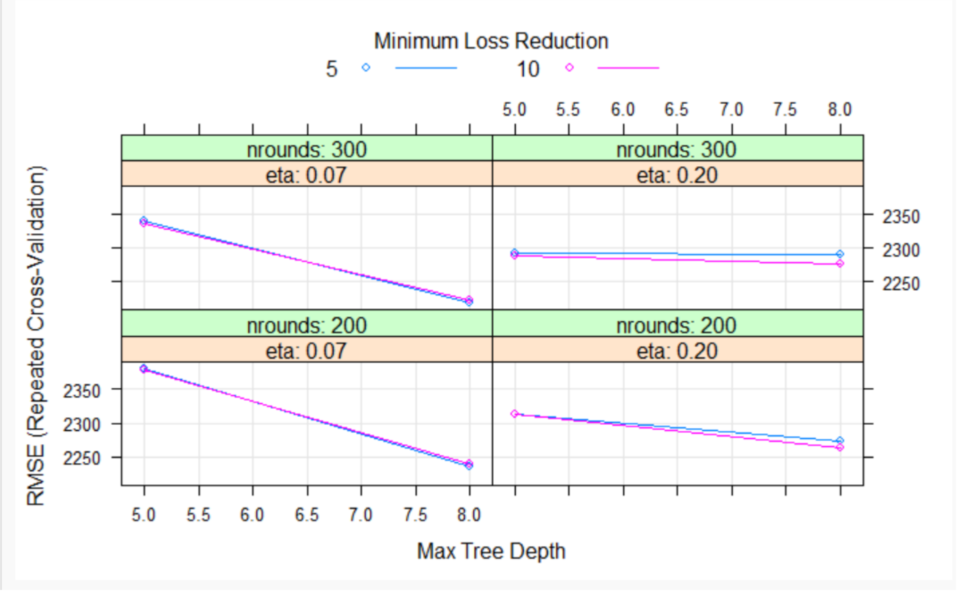
- Firstly did a random grid search, and then manually gave range of hyperparameters for model to find best combination.
- The later gave better results.

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Extreme Gradient Boosting using DT

- Hyper parameter Tuning



- Best Tune

	nrounds <dbl>	max_depth <dbl>	eta <dbl>	gamma <dbl>	colsample_bytree <dbl>	min_child_weight <dbl>	subsample <dbl>
6	300	8	0.07	5	0.8	1	0.8

Stacking

- After that I employed Stacking to combine the results.
- The correlations in stacked predictors were quite high. So I did not expect a significant improvement after stacking.
- However I gave it a try, and see if error metrics improve. Models tried as meta learners : RF, Linear Regression and CART

Stacking

- Correlations

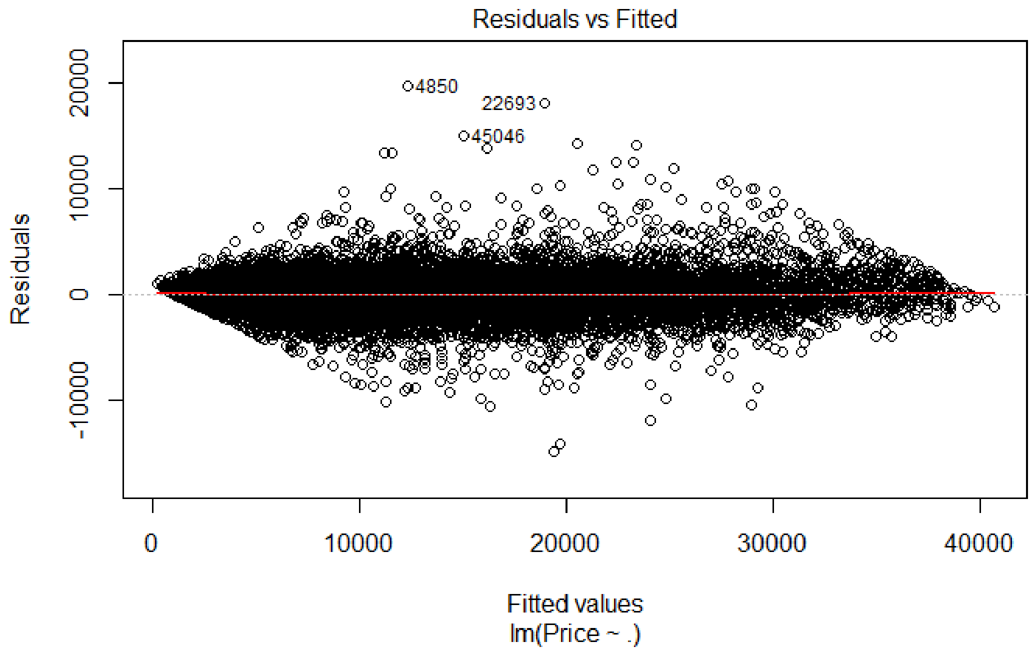
	RF	XGBTREE	CART	Price
RF	1.0000000	0.9897701	0.8983955	0.9854124
XGBTREE	0.9897701	1.0000000	0.9030494	0.9701990
CART	0.8983955	0.9030494	1.0000000	0.8610358
Price	0.9854124	0.9701990	0.8610358	1.0000000

- Head of data

	RF <dbl>	XGBTREE <dbl>	CART <dbl>	Price <dbl>
1	3774.896	4063.092	2719.588	3850
3	6568.484	8208.192	6195.689	5990
4	4095.842	4552.498	6195.689	4000
5	11980.811	9968.633	12068.061	12950
6	3213.363	2623.744	2719.588	3600
7	3095.410	3322.637	2719.588	4450

6 rows

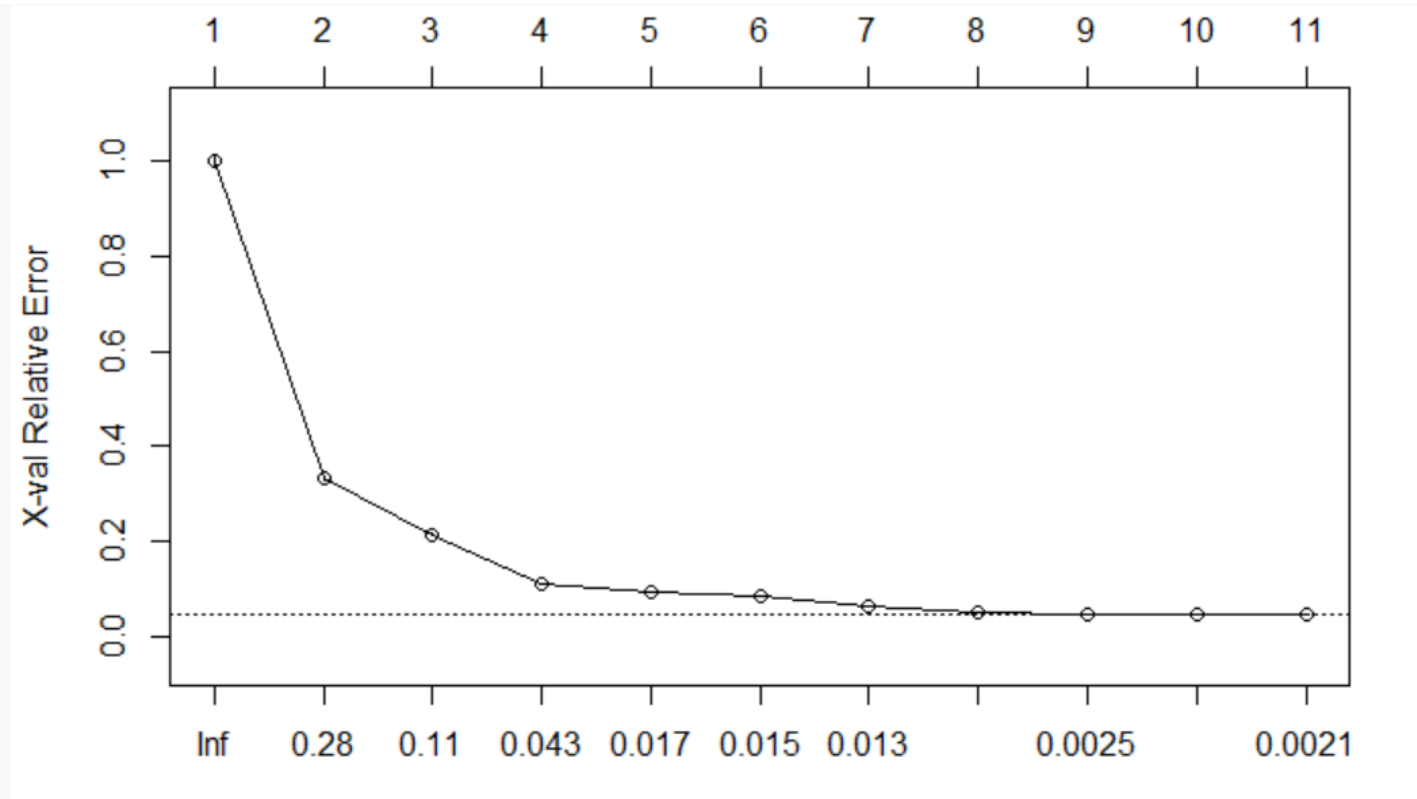
Stacking using Linear Regression




```
Adjusted R-squared: 0.9747
```

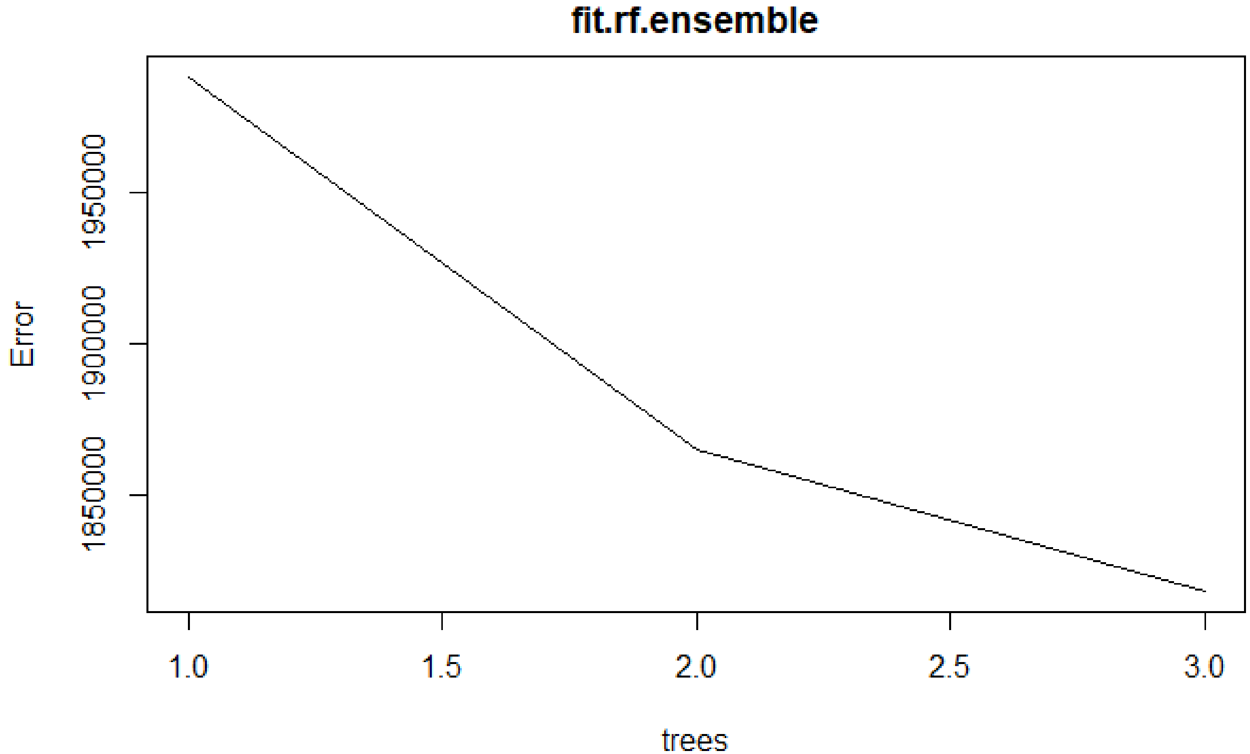
Stacking using CART

- Cp tuning



Stacking using Random Forest

- % Var explained: 96.02



Comparison of standalone models

MAPE	Train Error	Validation Error
Linear Regression	1.521916e+00	1.962097e+00
CART	4.424521e-01	4.469816e-01
Random Forests	1.264031e-01	2.476127e-01
XGBoost (Random Grid Search)	2.054038e-01	2.397260e-01
XGBoost (Manual Grid Search)	2.612451e-01	2.657679e-01
Stacking using CART	1.991522e-01	2.854830e-01
Stacking using Linear Regression	1.140418e-01	2.535492e-01
Stacking using RF	1.060275e-01	2.532537e-01

Results

- I got best results using Random Forest standalone model.
- Stacking improved the train error, but test error didn't decrease
- Least Test Data MAPE came approximately 39%

Your answer passed the tests! Your score is 2.58%
Congratulations!! Your model surpassed the baseline model with MAPE of 38.8233%
However your score will be scaled post submission deadline.

Further Improvements

- I would like to later work on this data to improve the model. Some of the ideas I have are
  - Clean the “NameOfTheVehicle” attribute using string regex techniques and add into model
  - Use the date columns in the model