USED CAR PRICE PREDICTION

MA 5790

Vaishnavi Jangili

Sahithi Bathini

Introduction:

The project's goal is to create a reliable and accurate predictive model that uses a variety of features and historical data to estimate used car prices. It is now essential for both buyers and sellers to anticipate prices accurately due to the rising demand for used cars. To create a trustworthy model that can accurately estimate used car prices, this study makes use of machine learning techniques that are implemented in R Studio.

To train and assess the predictive models, the project will make use of a variety of linear continuous models and non-linear continuous models. To improve the models' performance and capacity for generalization, cross-validation techniques will be employed to refine and optimize them.

To measure the precision and dependability of the predictions, the models will be evaluated using metrics such as Root Mean Squared Error (RMSE) and R-squared. To add to the interpretability, the most successful model will be chosen and its insights into the significance of the features will be examined.

The dataset contains 8128 records of cars. It contains 13 attributes with the target variable describing the selling price of a particular car. The dataset is an open-source dataset found on Kaggle. Below is the list of the variables present in the dataset.

Features	Data	Description
Name	Categorical	Name of the car
Year	Continuous	Year of manufacturing
Km_driven	Continuous	Total Kms the car was driven
fuel_type	Categorical	Petrol, Diesel, LPG,CNG
seller_type	Categorical	Type of seller. Eg: owner/dealer
transmission	Categorical	Automatic/Manual
owner	Categorical	First or second owner
mileage	Continuous	Mileage offered by car
enginee_cc	Continuous	The displacement volume of the engine in CC.
max_power	Continuous	Maximum power output
seats	Continuous	No. of seats in car
torque	Continuous	Rotational force engine can apply
selling_price	**Continuous**	**Price of the car**

Table:1 Data Description

Preprocessing of the data will involve any necessary predictor removal or transformations. After that, continuous models and both linear and nonlinear classification will be used.

Pre-Processing of data:

To ensure the quality and dependability of the ensuing analyses or model predictions, data preprocessing is an essential step in the pipeline for data analysis and machine learning. The inaccuracies, missing values, anomalies, and variety of formats that are frequently present in raw data can hinder the efficacy of analytical models. Several methods and procedures are used in data preprocessing to convert unprocessed data into a standardized, neat format. This

procedure improves the dataset's quality while also making a substantial contribution to the precision and efficacy of machine-learning models.

Correlation Plot:

A correlation plot of the remaining 30 predictors was created to explore the relationship between predictors. Below is the correlation plot with the blue representing positive correlations between predictors and red representing negative correlations between predictors.

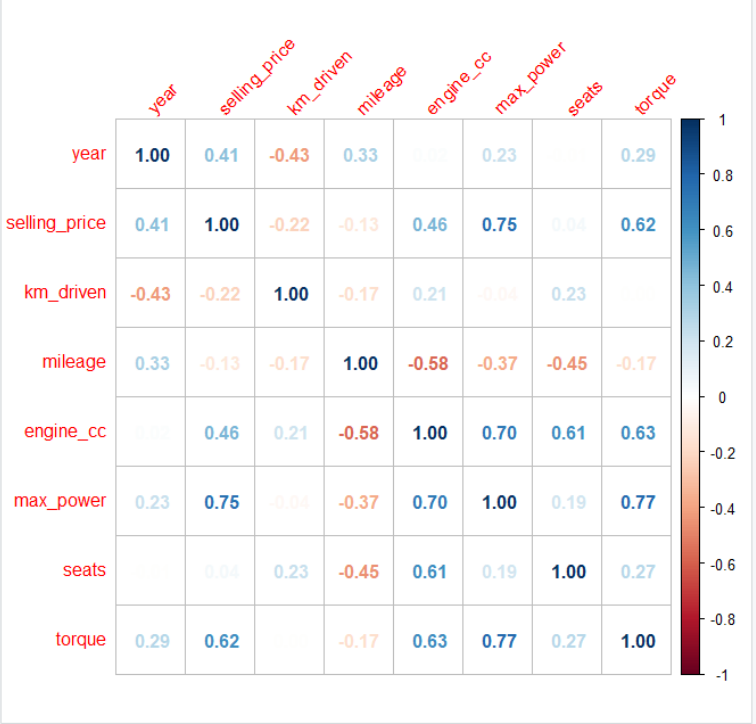


Fig1: Correlation Plot

A. Handling missing data:

The first step involves scanning the entire data frame for cells containing missing values. These missing values can arise due to various reasons, such as errors during data collection, sensor malfunctions, or simply the absence of certain information.

Once the missing values are identified, the na. omit() function acts by removing entire rows from the data frame that contain at least one missing value. This is a practical approach, as it ensures that the integrity of the remaining data is maintained.

B. Transformation:

Analysing the distributions of the continuous variables is a prerequisite before fitting any models to the data. The continuous predictors' box plots and histograms, before any kind of transformation, are shown below. As can be observed, there are outliers and a significant rightward skew in the majority of the predictors. The skewness values for each continuous predictor are displayed in the table 2 below.

VARIABLE	SKEWNESS VALUE	INTERPRETATION
YEAR	-1.071897	Left skewed
KM_DRIVEN	11.16679	Heavily right skewed
MILEAGE	-0.1445934	Left skewed
ENGINE_CC	1.150594	Right skewed
MAX_POWER	1.642627	Right skewed
SEATS	1.996506	Right skewed
TORQUE	1.074661	Right skewed
SELLING_PRICE	4.191986	Right skewed

Table 2 Skewness values

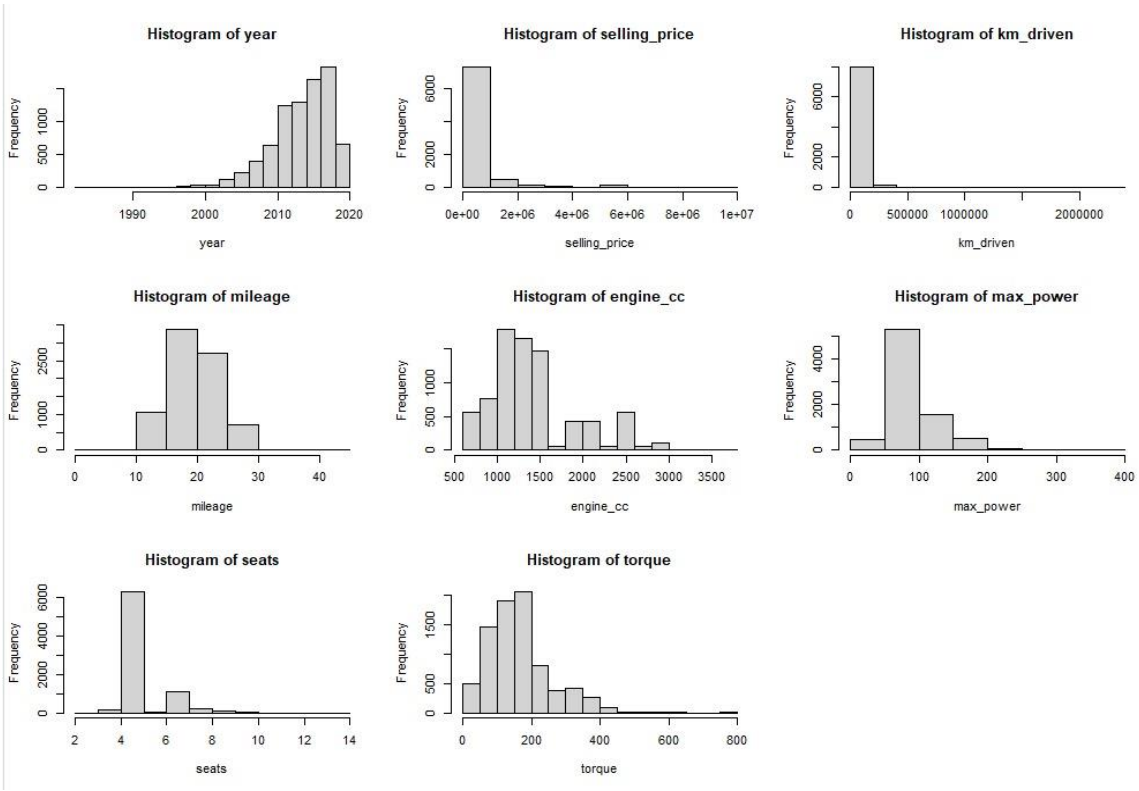


Fig 2: Distribution of Continuous Variables before transformation

The data will first be centered and scaled to account for the continuous variables' skewness, outliers, and varying scales. Then, to account for outliers, a Box-Cox transformation, and a spatial sign transformation will be used. Histograms and boxplots of the data following the

transformations are shown below. It is evident that the data is more symmetrical and has fewer outliers.

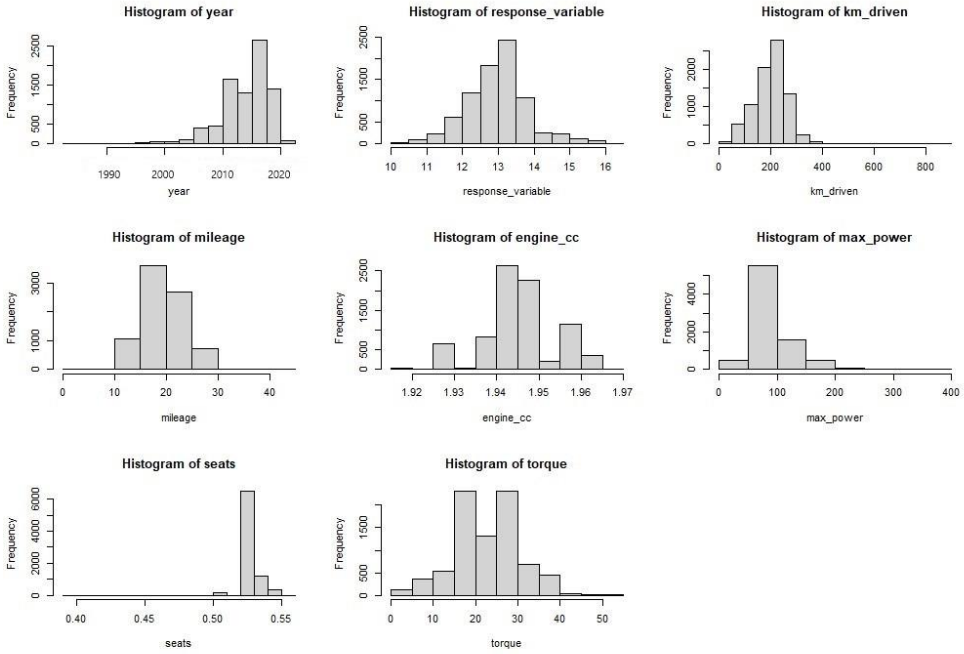


Fig 3: Distribution of Continuous Variables after transformation

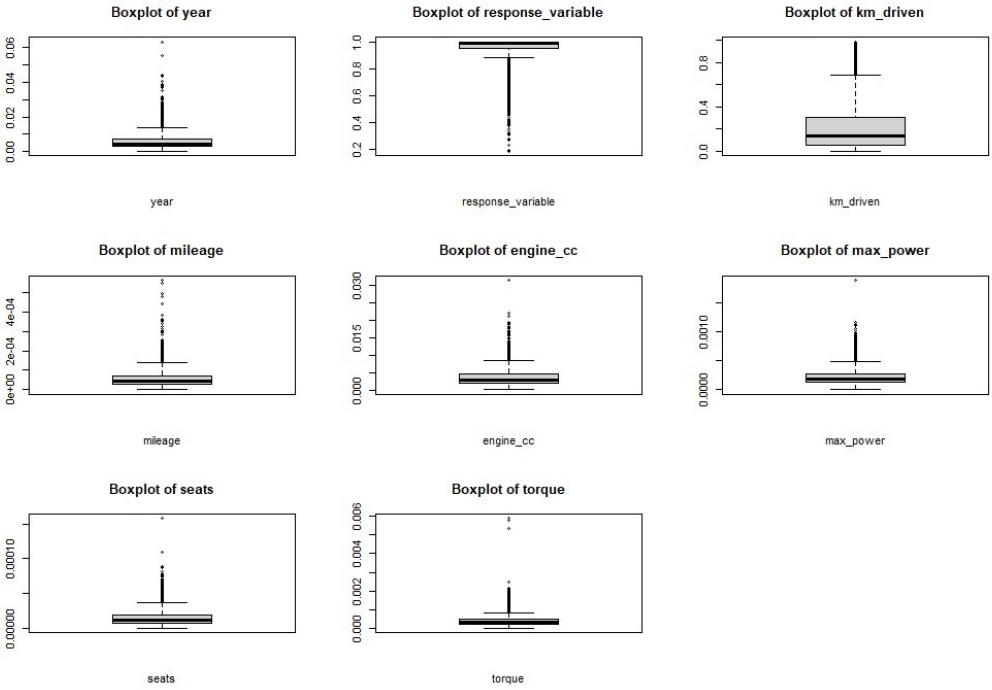


Fig 3: BoxPlot of Continuous Variables after transformation

C. Dummy Variables:

In statistical modeling and machine learning, dummy variables—also referred to as indicator variables or binary variables—are a means of numerically representing categorical data. Using dummy variables makes it easier to incorporate information about categorical variables—variables that can only take on one of a finite number of categories or levels—into models that need numerical input.

The distributions of the categorical predictor variables are displayed in the bar plots below. They have balanced frequencies for the most part.

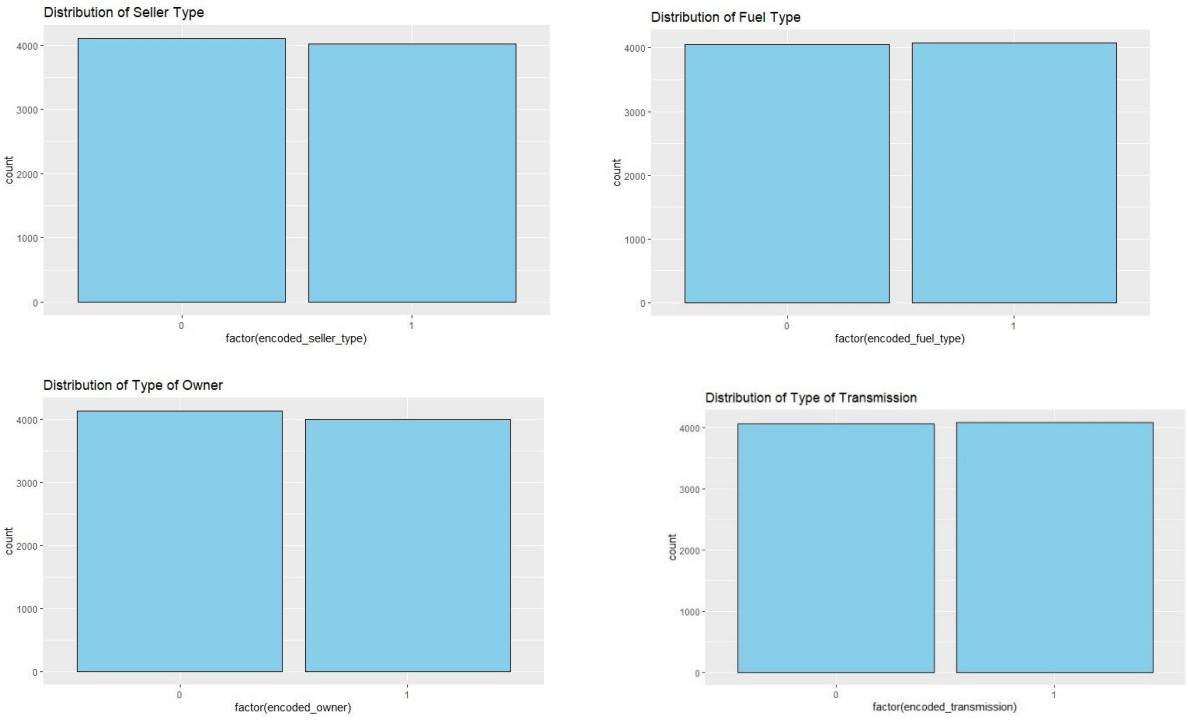


Fig 4: Bar Plots of categorical variables

D. Near Zero Variables:

- fuel_typeCNG and fuel_typeLPG: If most of the vehicles in the dataset use a specific fuel type (e.g., petrol) and only a small proportion use CNG or LPG, these variables might have near-zero variance.
- seller_typeTrustmarkdealer: If the majority of sellers are not trust-marked dealers, and only a small percentage are, this variable may have low variance.
- ownerFour/above and ownerTest: If the majority of cars have owners less than four and only a small fraction have four or more owners or a "Test" owner, these variables may have near-zero variance.
- These near-zero variables will be removed from the data set, sin don't add much valuable information to the model, and their existence could potentially add noise.
- Eliminating these variables can decrease the likelihood of overfitting, increase computational efficiency, and result in a more parsimonious model.

E. Highly Correlated Variables:

Highly correlated variables in the dataset are identified and filtered out to create a new dataset and then checks the dimensions of the filtered dataset. This is a common preprocessing step to address multicollinearity, which can cause issues in certain modeling techniques. Removing highly correlated variables can help improve the stability and interpretability of the model.

After the preprocessing of the dataset, the dataset has 7906 observations and 12 predictors.

Data Splitting:

We are attempting to predict the variable "selling price" which is a continuous variable. From fig5, the distribution is also very imbalanced, and therefore, stratified random sampling has been used when splitting the data into a training and a testing set. 80% of each outcome is used in the training set leaving 20% for testing.

We also utilize the k-cross-validation to set up the training control parameter and the number of folds is 3. Cross-validation is a technique used to assess how well a model will generalize to an independent dataset.

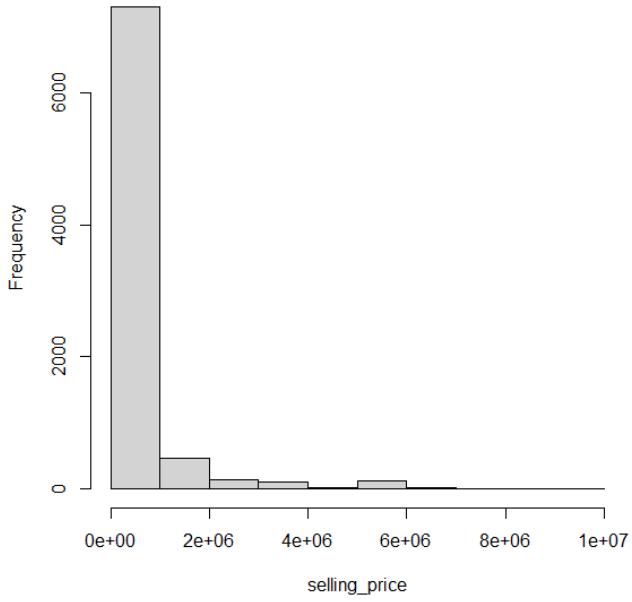


Fig5: Distribution of response variable

Model Fitting:

The data was used to train both linear and non-linear continuous models, and for any model with tuning parameters, additional figures are provided in the first and second appendix. The outcomes of these models' predictions on the training set are shown below. To get the most conservative results, cross-validation produced the R-squared values.

Linear Continuous Models	Best Tuning Parameter	Training R^2
Linear Model	Intercept = True	0.668314
Ridge	Lambda = 0.0071	0.67237
LASSO	alpha=1, lambda=0.1	0.6712203
E NET	fraction = 0.95, lambda = 0.01	0.672509

Table 3 Summary Table For Linear Models

Non-Linear Continuous Model	Best Tuning Parameter	Training R^2
Neural Network	Size=1, Decay=0.1	0.69361
SVM	Sigma = 0.1, C=10	0.81239
KNN	k=4	0.73249
MARS	Nprune = 21, degree=2	0.789187

Table 4 Summary Table For Non-Linear Models

The two top models are the Elastic Net model and the Support Vector Machine based on the R-squared values after predicting the test data set.

Models	R-Squared	RMSE
E Net	0.67710	45915.11
SVM	0.81678	1.64328

Table 5 Top Two Models

Model Performance:

Variable importance in LOESS is often assessed using the R-squared values, where higher R-squared values indicate that a variable has more influence on the response variable. Fig 6 shows the top 5 important variables of the best model "SVM model".

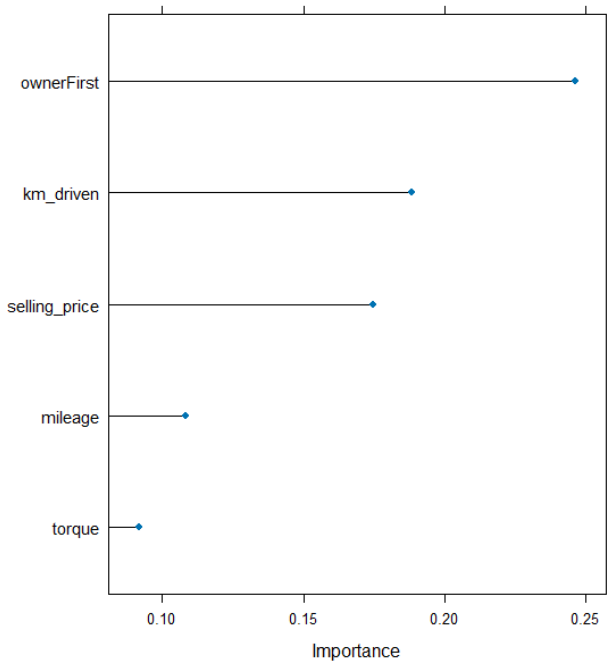


Fig 6: Top 5 Important Variables of SVM model

Conclusion:

In conclusion, our used car price prediction project employs advanced machine learning techniques in R Studio to deliver accurate and reliable price estimates. The user-friendly application enhances transparency in the used car market, aiding buyers and sellers with informed decision-making. This project contributes to the growing field of machine learning in automotive industries.

The best model found during this analysis was the Support Vector Machine with an R-squared value of 0.81678 after predicting the test data set to predict the selling price of the used car.

References:

- https://www.kaggle.com/datasets/avikasliwal/used-cars-price-prediction

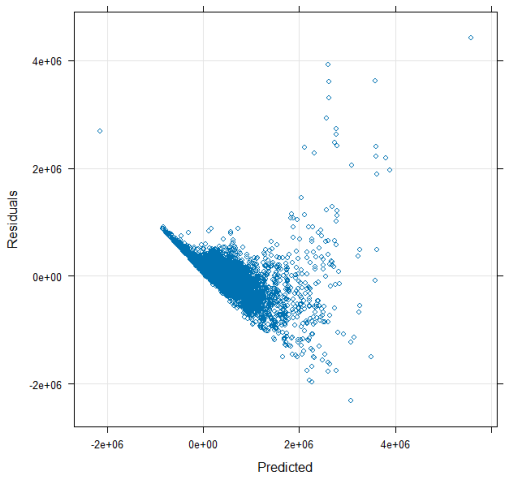
Appendix 1: Linear Continuous Models

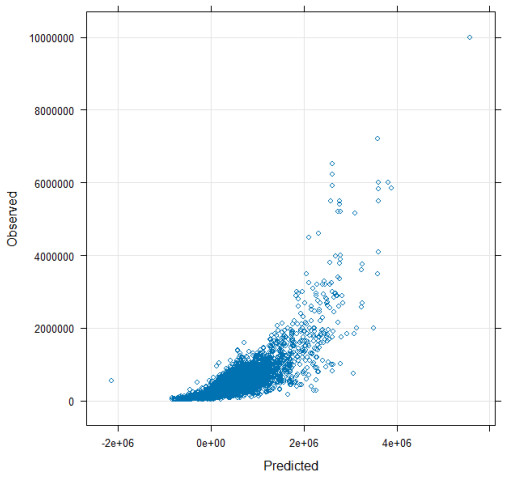
A. Linear Regression

The tuning parameter is “intercept = True”.
6327 samples
12 predictor
No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4218, 4217, 4219
Resampling results:

RMSE	Rsquared	MAE
469908.5	0.668314	278638.7

Tuning parameter 'intercept' was held constant at a value of TRUE





B. Ridge Regression

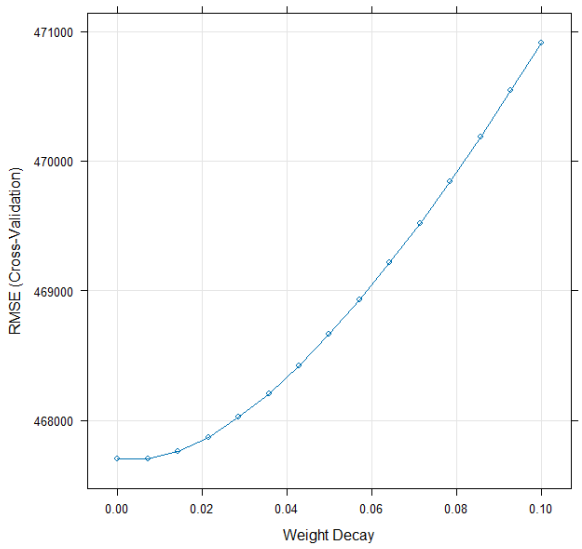
6327 samples
12 predictor

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4218, 4217, 4219
Resampling results across tuning parameters:

lambda	RMSE	Rsquared	MAE
0.000000000	467702.6	0.6723565	277594.5
0.007142857	467700.0	0.6723782	277875.8
0.014285714	467760.6	0.6723301	278231.2
0.021428571	467871.4	0.6722318	278684.6
0.028571429	468023.0	0.6720968	279193.1
0.035714286	468209.0	0.6719351	279746.8
0.042857143	468424.4	0.6717539	280332.7
0.050000000	468665.8	0.6715584	280959.1
0.057142857	468930.3	0.6713525	281610.4

0.064285714	469216.0	0.6711393	282283.6
0.071428571	469521.1	0.6709209	282977.4
0.078571429	469844.4	0.6706992	283702.1
0.085714286	470184.9	0.6704755	284468.9
0.092857143	470541.7	0.6702508	285261.3
0.100000000	470914.1	0.6700259	286076.6

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.007142857.



C. LASSO

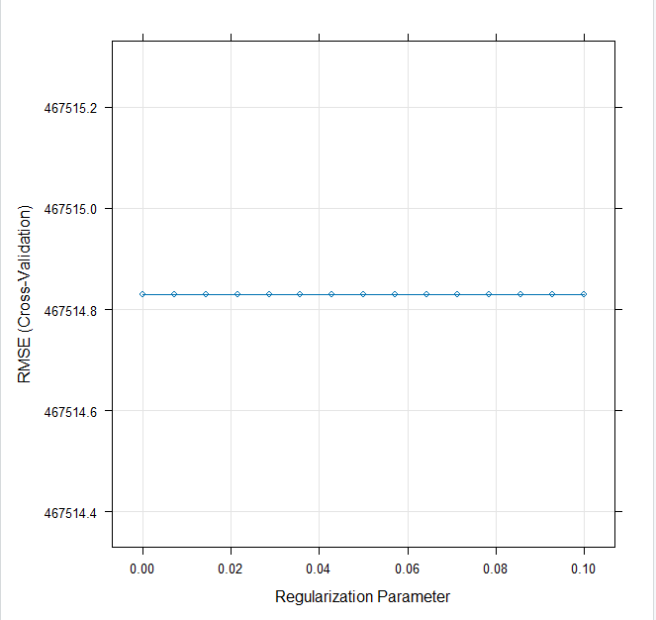
6327 samples
12 predictor

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4218, 4219, 4217
Resampling results across tuning parameters:

lambda	RMSE	Rsquared	MAE
0.000000000	467514.8	0.6712203	276882.5
0.007142857	467514.8	0.6712203	276882.5
0.014285714	467514.8	0.6712203	276882.5
0.021428571	467514.8	0.6712203	276882.5
0.028571429	467514.8	0.6712203	276882.5
0.035714286	467514.8	0.6712203	276882.5
0.042857143	467514.8	0.6712203	276882.5
0.050000000	467514.8	0.6712203	276882.5
0.057142857	467514.8	0.6712203	276882.5
0.064285714	467514.8	0.6712203	276882.5
0.071428571	467514.8	0.6712203	276882.5

0.078571429	467514.8	0.6712203	276882.5
0.085714286	467514.8	0.6712203	276882.5
0.092857143	467514.8	0.6712203	276882.5
0.100000000	467514.8	0.6712203	276882.5

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.1.



D. Elastic Net

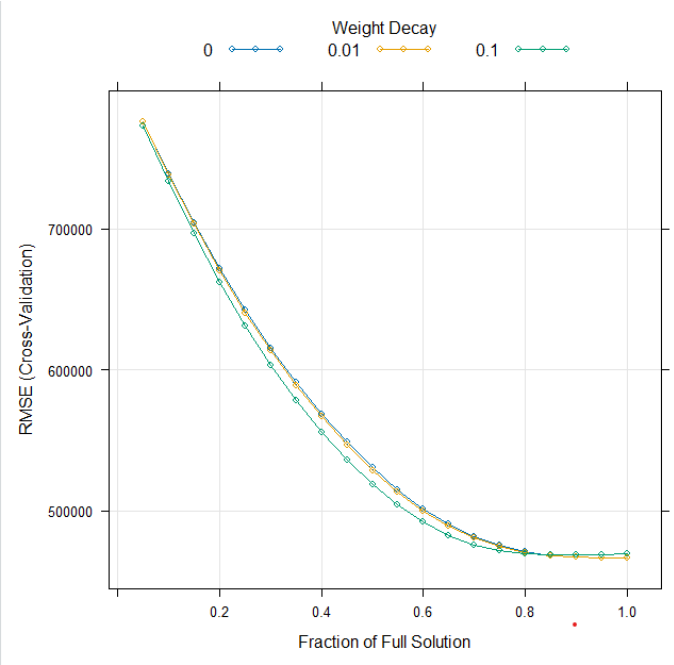
6327 samples
12 predictor
Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4220, 4217, 4217
Resampling results across tuning parameters:

lambda	fraction	RMSE	Rsquared	MAE
0.00	0.05	776131.9	0.5611088	401188.2
0.00	0.10	739307.8	0.5611088	377708.1
0.00	0.15	704473.3	0.5611088	356025.1
0.00	0.20	671937.9	0.5611088	336175.6
0.00	0.25	642051.0	0.5611088	318347.2
0.00	0.30	615143.3	0.5725559	303539.4
0.00	0.35	590641.4	0.5876422	291712.8
0.00	0.40	568629.9	0.6063973	278948.7
0.00	0.45	548491.9	0.6236167	268657.8
0.00	0.50	530491.0	0.6356258	261676.9
0.00	0.55	514943.6	0.6445443	257104.2
0.00	0.60	501509.5	0.6522863	254969.2
0.00	0.65	490494.0	0.6582256	254969.3
0.00	0.70	481888.7	0.6624881	257158.3
0.00	0.75	475468.9	0.6658322	260722.6
0.00	0.80	471051.4	0.6688484	264261.4
0.00	0.85	468272.0	0.6709643	268295.6
0.00	0.90	467093.3	0.6717331	272317.6
0.00	0.95	466493.7	0.6723535	274709.9
0.00	1.00	466418.0	0.6724095	277173.5
0.01	0.05	775695.3	0.5611088	400905.4
0.01	0.10	738478.4	0.5611088	377181.4
0.01	0.15	703305.5	0.5611088	355298.9

0.01	0.20	670498.1	0.5611088	335294.9
0.01	0.25	640420.0	0.5611088	317400.4
0.01	0.30	613361.8	0.5751574	302566.4
0.01	0.35	588785.0	0.5897446	290830.5
0.01	0.40	566683.5	0.6088883	277910.9
0.01	0.45	546517.0	0.6255482	267849.6
0.01	0.50	528583.8	0.6371656	261038.1
0.01	0.55	513125.0	0.6458864	256680.5
0.01	0.60	499882.7	0.6535142	254499.0
0.01	0.65	489075.1	0.6593253	254415.8
0.01	0.70	480722.6	0.6633393	256900.8
0.01	0.75	474589.9	0.6664319	260845.6
0.01	0.80	470492.6	0.6691062	264760.6
0.01	0.85	467962.2	0.6710706	269117.1
0.01	0.90	466943.6	0.6718677	272630.1
0.01	0.95	466387.4	0.6725097	275044.3
0.01	1.00	466461.1	0.6723997	277633.7
0.10	0.05	773084.6	0.5611088	399223.9
0.10	0.10	733530.7	0.5611088	374065.9
0.10	0.15	696361.7	0.5611088	350999.6
0.10	0.20	661979.4	0.5611088	330114.9
0.10	0.25	631021.9	0.5726133	311959.3
0.10	0.30	603071.0	0.5893611	297691.3
0.10	0.35	578102.7	0.6019747	285987.6
0.10	0.40	555731.7	0.6217776	272304.9
0.10	0.45	535897.2	0.6351480	262682.9
0.10	0.50	518747.0	0.6446823	255812.0
0.10	0.55	504023.6	0.6526391	251746.4
0.10	0.60	491863.0	0.6583216	251065.7
0.10	0.65	482504.5	0.6622969	253497.3
0.10	0.70	475819.7	0.6651293	257977.8
0.10	0.75	471678.3	0.6669227	264193.0
0.10	0.80	469834.1	0.6678885	271425.5
0.10	0.85	469004.0	0.6690116	274736.8
0.10	0.90	468567.1	0.6700235	278174.4
0.10	0.95	468860.9	0.6703858	282143.8
0.10	1.00	469745.6	0.6699945	285742.9

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were fraction = 0.95 and lambda = 0.01.



Appendix 2: Non-Linear Continuous Models

A. Neural Network

6324 samples

12 predictor

Pre-processing: centered (12), scaled (12)

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 4215, 4217, 4216

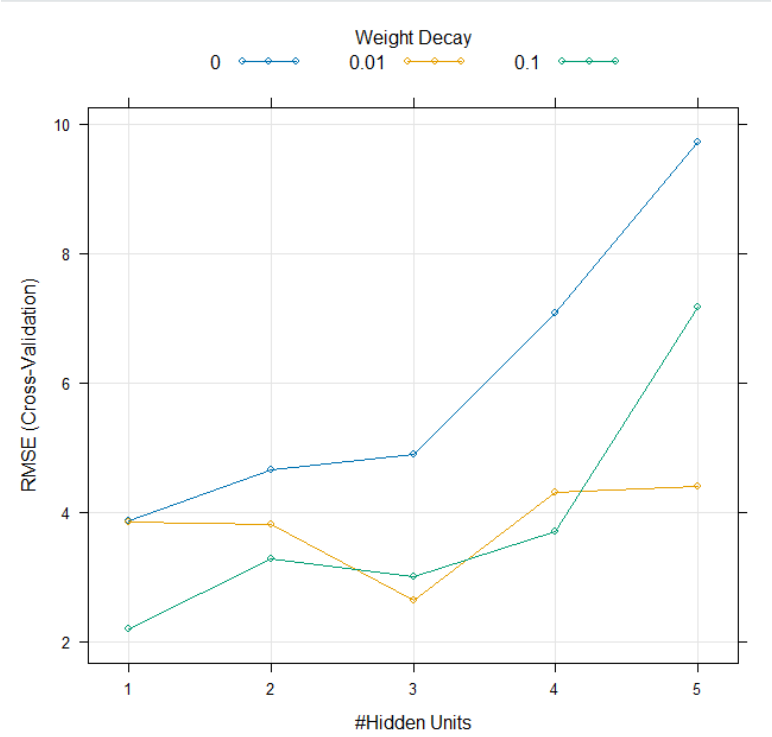
Resampling results across tuning parameters:

decay	size	RMSE	Rsquared	MAE
0.00	1	3.865545	0.27332434	2.842996
0.00	2	4.658231	0.12623854	2.402715
0.00	3	4.893775	0.13336841	2.798459
0.00	4	7.071085	0.02307819	3.145179
0.00	5	9.722237	0.08623998	3.411793
0.01	1	3.851940	0.44222982	1.957639
0.01	2	3.806689	0.33759991	2.031443
0.01	3	2.643523	0.60679700	1.943071
0.01	4	4.300895	0.24709172	2.169208
0.01	5	4.403447	0.45606960	2.129311
0.10	1	2.196803	0.69361172	1.697985
0.10	2	3.279500	0.42812852	1.805088
0.10	3	3.013028	0.53789909	1.686899
0.10	4	3.701389	0.41648122	2.321675
0.10	5	7.170474	0.21231229	2.321623

Tuning parameter 'bag' was held constant at a value of FALSE

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were size = 1, decay = 0.1 and bag = FALSE.



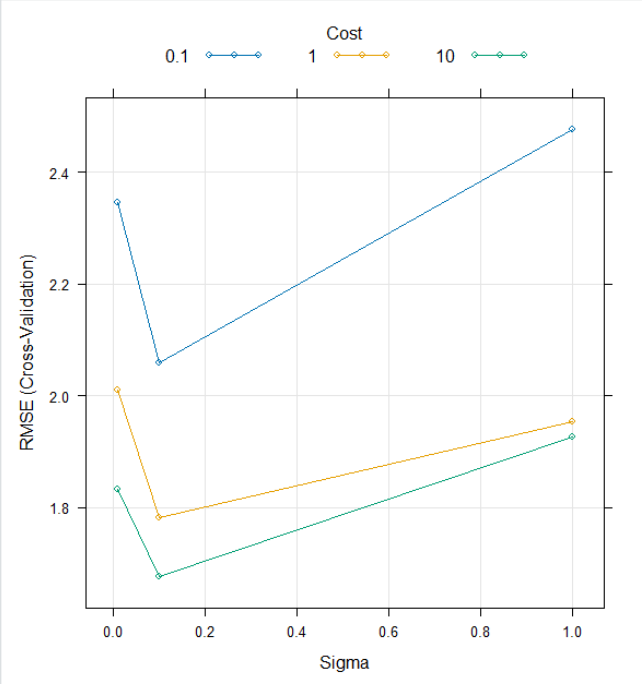
B. Support Vector Machine with Radical Basis Function Kernel

6324 samples
12 predictor

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4216, 4217, 4215
Resampling results across tuning parameters:

sigma	C	RMSE	Rsquared	MAE
0.01	0.1	2.346562	0.6530590	1.668060
0.01	1.0	2.009671	0.7349252	1.411882
0.01	10.0	1.833261	0.7773609	1.293870
0.10	0.1	2.059668	0.7272994	1.429606
0.10	1.0	1.781731	0.7900126	1.247322
0.10	10.0	1.676611	0.8123967	1.170976
1.00	0.1	2.477357	0.6302580	1.688115
1.00	1.0	1.952976	0.7501514	1.306381
1.00	10.0	1.927270	0.7522538	1.287836

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.1 and C = 10.



C. K-Nearest Neighbors

6324 samples

12 predictor

Pre-processing: centered (12), scaled (12)

Resampling: Cross-Validated (3 fold)

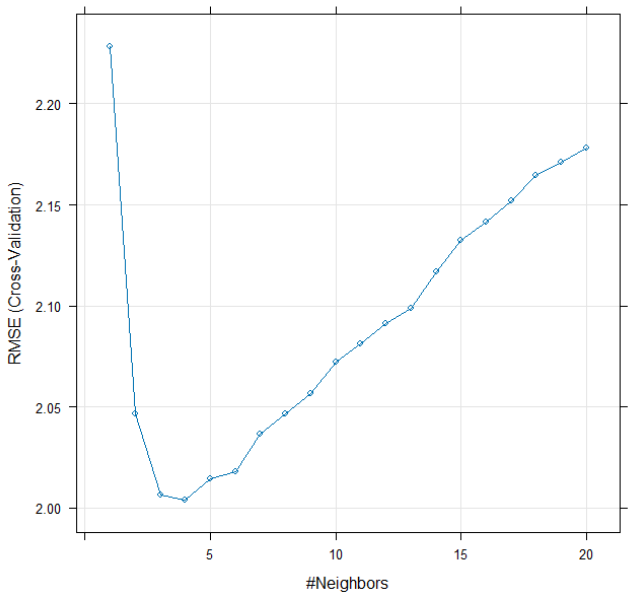
Summary of sample sizes: 4217, 4215, 4216

Resampling results across tuning parameters:

k	RMSE	Rsquared	MAE
1	2.228562	0.6880460	1.403241
2	2.046423	0.7249704	1.374554
3	2.006494	0.7327000	1.368904
4	2.003764	0.7324919	1.369141
5	2.014637	0.7292590	1.383353
6	2.018082	0.7282268	1.397011
7	2.036774	0.7231561	1.412206
8	2.046629	0.7204254	1.422921
9	2.056740	0.7175508	1.432748
10	2.072414	0.7132423	1.445045
11	2.081393	0.7109269	1.454489
12	2.091126	0.7083600	1.462621
13	2.098923	0.7062235	1.472064
14	2.117024	0.7012699	1.483421
15	2.132690	0.6969176	1.495752
16	2.141633	0.6945420	1.505358
17	2.152042	0.6918124	1.515053
18	2.164468	0.6883368	1.525829
19	2.171255	0.6865248	1.536708
20	2.178223	0.6846587	1.542056

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was k = 4.

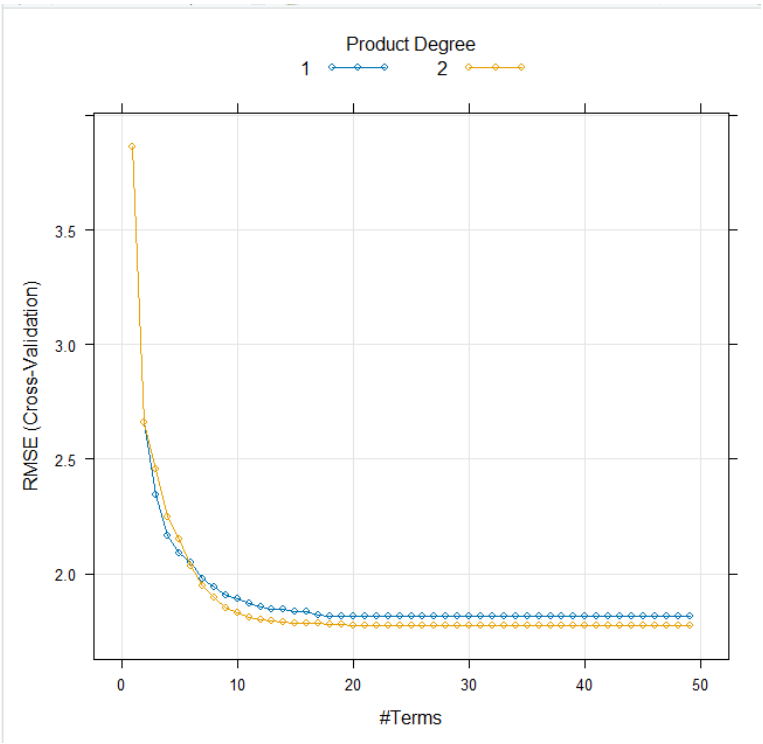


D. Multivariate Adaptive Regression Spline

6324 samples
12 predictor
No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 4215, 4217, 4216
Resampling results across tuning parameters:

degree	nprune	RMSE	Rsquared	MAE
1	1	3.863911	NaN	3.083198
1	2	2.661849	0.5248413	2.103586
1	3	2.346003	0.6292699	1.793385
1	4	2.167459	0.6842370	1.644808
1	5	2.088861	0.7074938	1.571462
1	6	2.048238	0.7183822	1.545083
1	7	1.976409	0.7381426	1.472102
1	8	1.943067	0.7465795	1.444690
1	9	1.907595	0.7557827	1.413604
1	10	1.888578	0.7606871	1.399346
1	11	1.869412	0.7654319	1.370352
1	12	1.855742	0.7688159	1.360951
1	13	1.846486	0.7711014	1.349563
1	14	1.844220	0.7716869	1.348217
1	15	1.834923	0.7740308	1.344149
1	16	1.834969	0.7740762	1.340624
1	17	1.819560	0.7779153	1.332907
1	18	1.816614	0.7784945	1.327643
1	19	1.812884	0.7794705	1.323816
1	20	1.812622	0.7795310	1.324114
1	21	1.812622	0.7795310	1.324114
1	22	1.812622	0.7795310	1.324114
1	23	1.812622	0.7795310	1.324114
1	24	1.812622	0.7795310	1.324114
1	25	1.812622	0.7795310	1.324114
1	26	1.812622	0.7795310	1.324114
1	27	1.812622	0.7795310	1.324114
1	28	1.812622	0.7795310	1.324114
1	29	1.812622	0.7795310	1.324114
1	30	1.812622	0.7795310	1.324114
1	31	1.812622	0.7795310	1.324114
1	32	1.812622	0.7795310	1.324114
1	33	1.812622	0.7795310	1.324114
1	34	1.812622	0.7795310	1.324114
1	35	1.812622	0.7795310	1.324114
1	36	1.812622	0.7795310	1.324114
1	37	1.812622	0.7795310	1.324114
1	38	1.812622	0.7795310	1.324114
1	39	1.812622	0.7795310	1.324114
1	40	1.812622	0.7795310	1.324114
1	41	1.812622	0.7795310	1.324114
1	42	1.812622	0.7795310	1.324114
1	43	1.812622	0.7795310	1.324114
1	44	1.812622	0.7795310	1.324114
1	45	1.812622	0.7795310	1.324114
1	46	1.812622	0.7795310	1.324114
1	47	1.812622	0.7795310	1.324114
1	48	1.812622	0.7795310	1.324114
1	49	1.812622	0.7795310	1.324114
2	1	3.863911	NaN	3.083198
2	2	2.661849	0.5248413	2.103586
2	3	2.456539	0.5950874	1.930645
2	4	2.245254	0.6621195	1.727108
2	5	2.148948	0.6908970	1.599703
2	6	2.031869	0.7230820	1.532090
2	7	1.947471	0.7458853	1.452564
2	8	1.893857	0.7596378	1.410169
2	9	1.849005	0.7708457	1.379064
2	10	1.829555	0.7755328	1.365902
2	11	1.808899	0.7805027	1.343401
2	12	1.800703	0.7824762	1.335741
2	13	1.793827	0.7842153	1.327515
2	14	1.787375	0.7857623	1.321402
2	15	1.784924	0.7863727	1.320967
2	16	1.785236	0.7863315	1.320791
2	17	1.785421	0.7862399	1.317519
2	18	1.780859	0.7873512	1.316026
2	19	1.780096	0.7875293	1.316323
2	20	1.774042	0.7890059	1.313411
2	21	1.773290	0.7891873	1.312983
2	22	1.773290	0.7891873	1.312983
2	23	1.773290	0.7891873	1.312983
2	24	1.773290	0.7891873	1.312983
2	25	1.773290	0.7891873	1.312983
2	26	1.773290	0.7891873	1.312983
2	27	1.773290	0.7891873	1.312983
2	28	1.773290	0.7891873	1.312983
2	29	1.773290	0.7891873	1.312983
2	30	1.773290	0.7891873	1.312983
2	31	1.773290	0.7891873	1.312983
2	32	1.773290	0.7891873	1.312983
2	33	1.773290	0.7891873	1.312983
2	34	1.773290	0.7891873	1.312983
2	35	1.773290	0.7891873	1.312983
2	36	1.773290	0.7891873	1.312983
2	37	1.773290	0.7891873	1.312983
2	38	1.773290	0.7891873	1.312983
2	39	1.773290	0.7891873	1.312983
2	40	1.773290	0.7891873	1.312983
2	41	1.773290	0.7891873	1.312983
2	42	1.773290	0.7891873	1.312983
2	43	1.773290	0.7891873	1.312983
2	44	1.773290	0.7891873	1.312983
2	45	1.773290	0.7891873	1.312983
2	46	1.773290	0.7891873	1.312983
2	47	1.773290	0.7891873	1.312983
2	48	1.773290	0.7891873	1.312983
2	49	1.773290	0.7891873	1.312983

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 21 and degree = 2.


Product Degree
1
2
RMSE (Cross-Validation)
3.5
3.0
2.5
2.0
0
10
20
30
40
50
#Terms

R Code

```
data<-read.csv("C:/Users/vaish/Downloads//car_details.csv")
data
#Dimensions
dim(data)
#Delete car column
data <- data[, !(names(data) %in% c("name"))]
head(data)
#Missing values
data <- na.omit(data)
#Correlation Plot
library(corrplot)
numeric_columns <- sapply(data, is.numeric)
correlation_matrix <- cor(data[, numeric_columns])
corrplot(correlation_matrix, method = "number", tl.col = "red", tl.srt = 45)

###############Exploratory Data Analysis##############
columns_of_interest <- c("year", "selling_price", "km_driven"
                         ,"mileage","engine_cc","max_power","seats","torque")  # Replace with actual
column names
par(mfrow=c(3,3)) #https://www.datamentor.io/r-programming/subplot
for(col in columns_of_interest) {
  if(is.numeric(data[[col]])) {
    hist(data[[col]], main = paste("Histogram of", col), xlab = col)
  }
}
for(col in columns_of_interest) {
  if(is.numeric(data[[col]])) {
    boxplot(data[[col]], main = paste("Boxplot of", col), xlab = col)
  }
}
```

```
trans <- preProcess(data, method = c("BoxCox"))  ## need {caret} package
trans

transformed <- predict(trans, data)

#Visualization for categorical variables.

my_data <- data.frame(
  encoded_fuel_type = sample(0:1, 8128, replace = TRUE)
)
par(mfrow=c(2, 2))
ggplot(my_data, aes(x = factor(encoded_fuel_type))) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Fuel Type")
###############
my_data <- data.frame(
  encoded_seller_type = sample(0:1, 8128, replace = TRUE)
)
ggplot(my_data, aes(x = factor(encoded_seller_type))) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Seller Type")
#################
my_data <- data.frame(
  encoded_owner = sample(0:1, 8128, replace = TRUE)
)
ggplot(my_data, aes(x = factor(encoded_owner))) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Type of Owner")
#############
```

```
my_data <- data.frame(
  encoded_transmission = sample(0:1, 8128, replace = TRUE)
)
ggplot(my_data, aes(x = factor(encoded_transmission))) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Type of Transmission")



#creating dummy variables
encoded_fuel_type <- model.matrix(~ fuel_type - 1, data)
encoded_seller_type <- model.matrix(~ seller_type - 1, data)
encoded_transmission <- model.matrix(~ transmission - 1, data)
encoded_owner <- model.matrix(~ owner - 1, data)

encoded_data <- cbind(data, encoded_fuel_type, encoded_seller_type, encoded_transmission,
encoded_owner)

encoded_data <- subset(encoded_data, select = -c(fuel_type, seller_type, transmission,
owner))
names(encoded_data)
head(encoded_data)

#delete x190 coloumn
encoded_data <- encoded_data[, !(names(encoded_data) %in% c("X190"))]
head(encoded_data)
dim(encoded_data)

#Near zero variables
library(caret)
near_var <- nearZeroVar(encoded_data)
df_final<-encoded_data[, -near_var]
```

```
dim(df_final)

#Highly Correlated
highCor<-findCorrelation(cor(df_final),cutoff = .80)
filtereddf <- df_final[,-highCor]
dim(filtereddf)

hist(selling_price)
selling_price
##############################################################################
#####################################################################
#Splitting data 80% 20%
library(lars)
library(elasticnet)
set.seed(100)

trainIndex <- createDataPartition(filtereddf$selling_price, p = 0.8,
                                  list = FALSE,
                                  times = 1)
train_data <- filtereddf[ trainIndex,]
test_data  <- filtereddf[-trainIndex,]
test_target <- test_data$selling_price
# Define the training control for cross-validation
trainControl <- trainControl(method = "cv", number = 3)

##################LINEAR CONTINUOUS MODELS###############
#linear regression model
lm_model <- train(selling_price ~ ., data = train_data, method = "lm",metric = "RMSE",
                  trControl = trainControl)
lm_model
```

```
xyplot(train_data$selling_price ~ predict(lm_model), type = c("p", "g"), xlab = "Predicted",
ylab = "Observed")

xyplot(resid(lm_model) ~ predict(lm_model), type = c("p", "g"),

      xlab = "Predicted", ylab = "Residuals")


#  Ridge model

ridgeGrid <- data.frame(.lambda = seq(0, .1, length = 15))

ridge_model <- train(selling_price ~ ., data = train_data, method = "ridge",metric = "RMSE",

                     trControl = trainControl, tuneGrid = ridgeGrid,

                     preProc = c("center", "scale"))

ridge_model

plot(ridge_model)


# Lasso model

lassoGrid <- data.frame(alpha = 1, lambda = seq(0.001, 1, length = 100))

lasso_model <- train(selling_price ~ ., data = train_data, method = "glmnet",metric =
"RMSE",

                     trControl = trainControl,

                     tuneGrid = lassoGrid,preProc = c("center", "scale"))

lasso_model

plot(lasso_model)


# Elastic Net model

enetGrid <- expand.grid(.lambda = c(0, 0.01, .1), .fraction = seq(.05, 1, length = 20))

elastic_net_model <- train(selling_price ~ ., data = train_data, method = "enet",

                           trControl = trainControl,

                           tuneGrid = enetGrid, preProc = c("center", "scale"))

elastic_net_model

plot(elastic_net_model)


################Predicting on test data
```

```
pred_enet<-predict(elastic_net_model, newdata = test_data)
pred_df4 <- data.frame(obs = test_target, pred = pred_enet)
defaultSummary(pred_df4)

##############################################################################
####################NON-LINEAR CONTINUOUS MODELS#####################
# Split the data into training and test sets (80%/20%)
library(caret)
trainIndex <- sample(nrow(filtereddf),nrow(filtereddf)*0.8)
train_pred <- filtereddf[ trainIndex, -1]
train_op <- filtereddf[trainIndex, 1]
test_pred  <- filtereddf[-trainIndex, -1]
test_op <- filtereddf[-trainIndex, 1]
dim(filtereddf)
# Define the training control for cross-validation
trainControl <- trainControl(method = "cv", number = 3)

######Neural Network######
nnetGrid <- expand.grid(.decay = c(0, 0.01, 0.1),
                        .size = c(1:5),
                        .bag = FALSE)
set.seed(100)
ctrl <- trainControl(method = "cv", number = 3)
nnetFit <- train(train_pred,train_op,
                 method = "avNNet",
                 tuneGrid = nnetGrid,
                 trControl = ctrl,
                 preProc = c("center", "scale"),
                 linout = TRUE,
```

```
            trace = FALSE,
            MaxNWts = 10 * (ncol(train_pred) + 1) + 10 + 1,
            maxit = 100)
nnetFit
plot(nnetFit)
########SVM#######
svm_Grid <- expand.grid(sigma = c(0.01, 0.1, 1),
                C = c(0.1, 1, 10))

svmRFit <- train(train_pred, train_op,
            method = "svmRadial",
            preProc = c("center", "scale"),
            tuneGrid = svm_Grid,
            trControl = trainControl(method = "cv", number = 3))

svmRFit

plot(svmRFit)
############KNN#########
knnTune <- train(train_pred,train_op,
            method = "knn",
            # Center and scaling will occur for new predictions too
            preProc = c("center", "scale"),
            tuneGrid = data.frame(.k = 1:20),
            trControl = trainControl(method = "cv", number = 3))
knnTune
plot(knnTune)
##########MARS######
mars_Grid <- expand.grid(nprune=seq(2:50),degree=seq(1:2))
```

```
#tune model
MARSModel <- train(train_pred,train_op,
                   method='earth',
                   tuneGrid=mars_Grid,
                   trControl = trainControl(method = "cv", number = 3))
MARSModel
mars_predictions <- predict(MARSModel,test_pred)
plot(MARSModel)

###############Predicting on test data
svm_predictions <- predict(svmRFit,test_pred)
svm_Results <- postResample(pred = svm_predictions, obs = test_op)
svm_Results

#########################################################
################IMPORTANT VARIABLES##############
plsImpSim <- varImp(svmRFit, scale = FALSE)
plsImpSim

plot(plsImpSim, top = 5, scales = list(y = list(cex = .95)))
```