

Used Cars Sale Price Prediction Using Machine Learning Algorithms

Submitted by:

Yash Bhardwaj

# **ACKNOWLEDGMENT**

I whole heartedly thank our SME Sapna Verma, flip robo technologies for their support towards me to complete this project.

I also thank my family for supporting me.

Yash Bhardwaj

**Contents**

[**ACKNOWLEDGMENT** 2](#_Toc109852963)

[**INTRODUCTION** 5](#_Toc109852964)

[Business Problem 5](#_Toc109852965)

[Review of Literature 5](#_Toc109852966)

[Motivation for the Problem Undertaken 6](#_Toc109852967)

[Dataset 6](#_Toc109852968)

[**Data Cleaning** 8](#_Toc109852969)

[**Exploratory Data Analysis** 12](#_Toc109852970)

[The response variable – SalePrice 12](#_Toc109852971)

[SalePrice and TransmissionType 12](#_Toc109852972)

[SalePrice and FuelType 13](#_Toc109852973)

[SalePrice and ModelYear 14](#_Toc109852974)

[SalePrice and Location 15](#_Toc109852975)

[SalePrice and KilometersDriven 15](#_Toc109852976)

[**Data Pre-processing** 17](#_Toc109852977)

[Skewness removal 17](#_Toc109852978)

[Outliers 19](#_Toc109852979)

[Converting variables 20](#_Toc109852980)

[Correlation 21](#_Toc109852981)

[**Building Machine Learning Models** 22](#_Toc109852982)

[Best Model? 22](#_Toc109852983)

[K-Fold Cross Validation 22](#_Toc109852984)

[Gradient Boosting 24](#_Toc109852985)

[Feature Importance 24](#_Toc109852986)

[Hyperparameter Tuning 25](#_Toc109852987)

[**Evaluation** 27](#_Toc109852988)

[Mean Squared Error 27](#_Toc109852989)

[Root Mean Squared Error 27](#_Toc109852990)

[Mean Absolute Error 28](#_Toc109852991)

[R Squared score 29](#_Toc109852992)

[Mean Squared Log Error 30](#_Toc109852993)

[Predicted VS Actual Values 31](#_Toc109852994)

[**Conclusion** 32](#_Toc109852995)

[Findings 32](#_Toc109852996)

[Limitations of this work and Scope for Future Work 33](#_Toc109852997)

[Of course there is still a big room for improvement as we in this project did not do much extensive feature engineering. We had taken the more simple way of looking into data, analysing it and making some models. 33](#_Toc109852998)

[**References** 35](#_Toc109852999)

# **INTRODUCTION**

## Business Problem

The value of a car drops right from the moment it is bought and the depreciation continues with each passing year. In fact, in the first year itself, the value of a car decreases by 20 percent of its initial value.

With the covid-19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid-19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

This project is divided in two phases:

* The first one is collecting the used cars data from websites like olx.com, cars24, cardekho.com etc. for various locations in India.
* Building a machine learning models is the other phase.

We have to select the best model in the end after performing tuning and evaluation on the selected model. This might help our clients in understanding the today’s market of used car sale.

## Review of Literature

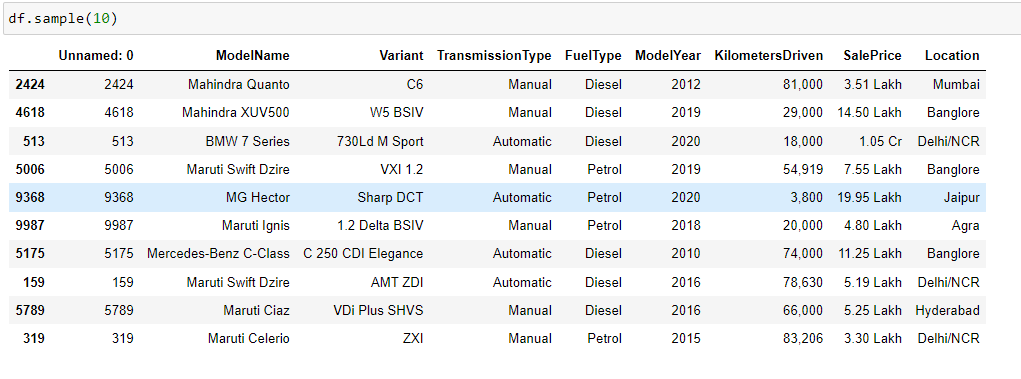
Used cars are generally cheaper because the high depreciation of their early years is already behind them and you may not need as much insurance coverage. First I scraped used cars data from cardekho.com using selenium. Also, I have to predict the reasonable used car sale price with these aspects of the used cars by using the scraped dataset. The scraped dataset contains 9 explanatory variables which are related to major aspects of used cars. In the following steps, I will explore this dataset, do feature engineering, fit some machine learning models to predict the used car sale prices and find which aspects of the used car influence its sale prices mostly. Machine learning is closely related to computational statistics, which focus on using mathematical optimization to deliver methods, theory and application domains to solve medical, industry, social and business problems in the real world. We will try multiple linear regression, regularization, ensemble technique and extreme gradient boosting as well. This model also gives us which aspects have big effects on used cars sale price.

## Motivation for the Problem Undertaken

The objective or motivation is to model the price of used cars with the available (scraped) independent variables. This model will then be used by the clients to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the clients to understand the pricing dynamics of a new market. Moreover, this might give the management team an insight of the real time world.

## Dataset

For this project we had to build a dataset of used cars by web-scraping from websites like olx.com, cars24, cardekho.com and many more. So for this purpose I have used selenium to extract data while most part on an automated chrome window. I scraped data of used cars for different locations like Delhi/NCR, Mumbai, Pune, Bengaluru etc. from cardekho.com. The variables totally depends on the website, so for the dataset I scraped data for 10028 used cars including all types i.e. SUV, Sedans, Coupe, minivan, Hatchback etc. with features as model name, variant, transmission type, fuel type, year of the model, kilometres that car has been driven, city or region and at the last the target variable i.e. sale price. A sample of the dataset:

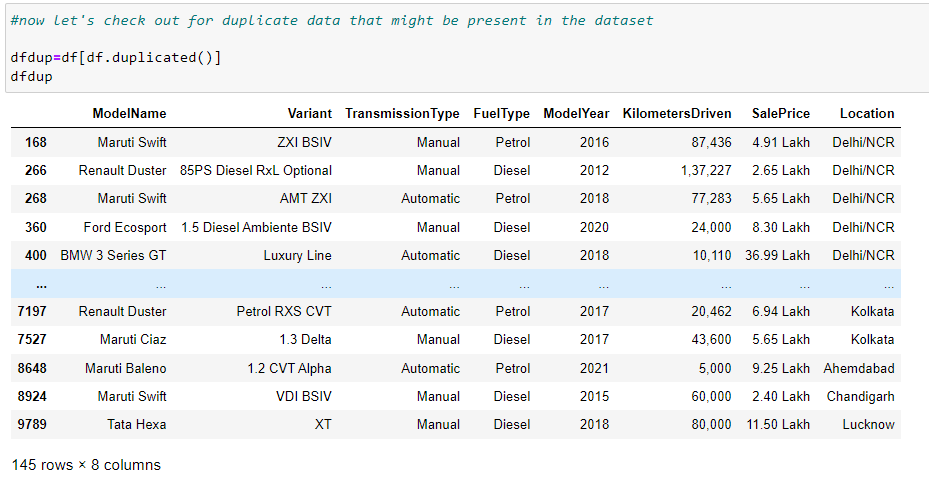


Now, before going for exploratory data analysis the data needs to be clean as some variables might be assigned not preferred data-types. Also, the dataset might have missing values.

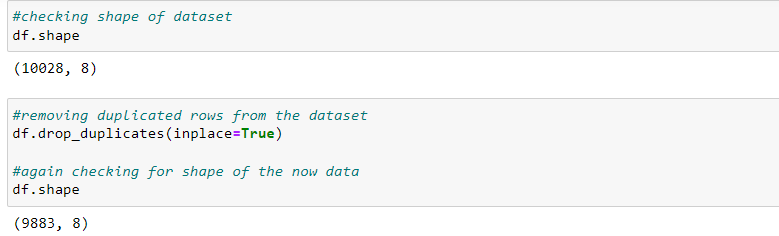
# **Data Cleaning**

Upon checking out the dataset it was seen that the dataset had an unnamed column, which was basically the index only as the dataset was converted into excel format from pandas dataframe after scraping the data. So, firstly this unnamed column was dropped as it didn’t has any importance at all in building the machine learning models.

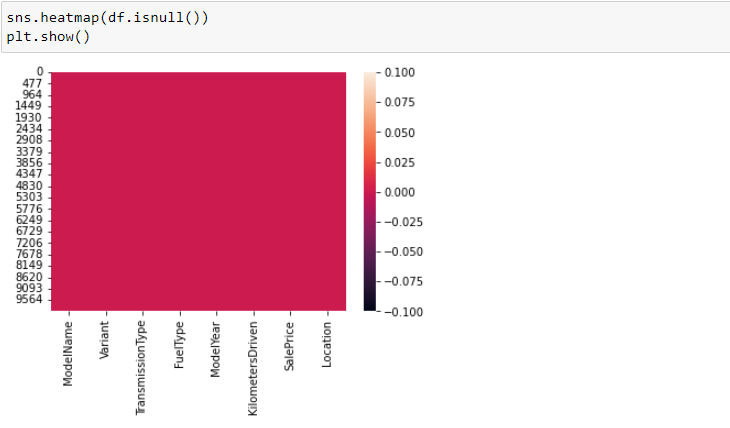
Afterwards, duplicated values (if present) were checked, pandas has an inbuilt function which returns duplicate rows present in the dataframe.



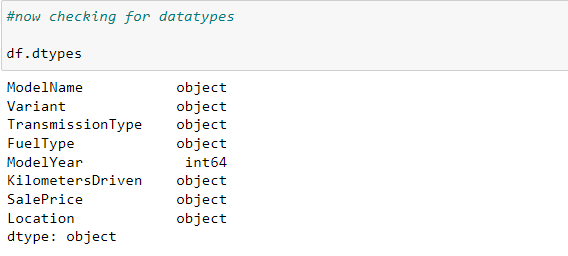
So, it could be seen that the dataset has 145 duplicated rows, it's typically a good idea to remove duplicate data points so the model can better generalize to the full dataset. The duplicated rows can be dropped as:



Afterwards, missing values were checked. It came out that the dataset had no missing values present which can be seen by the below heatmap:

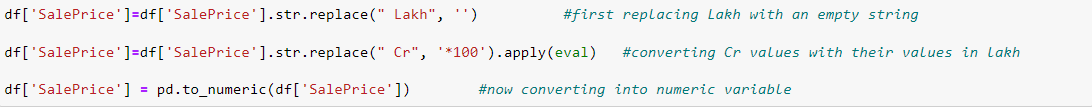


Now, before going for EDA i.e. exploratory data analysis I checked for datatype of the variables, though we are going to convert categorical variables into numerical ones later I wanted to check if the any variable has been assigned certain datatype that it shouldn’t has.

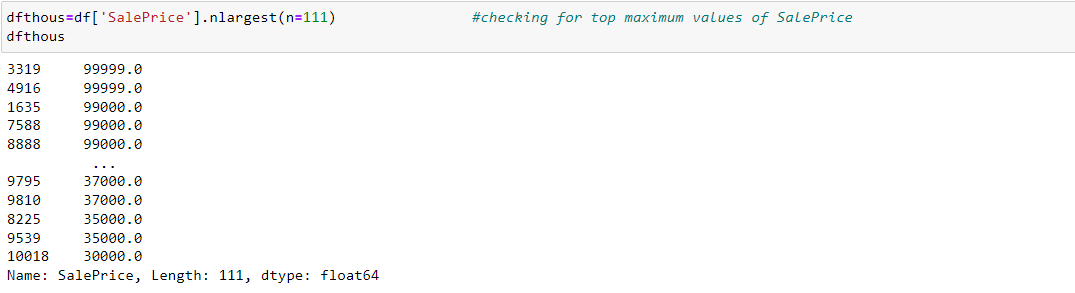


Turns out, kilometres driven and the target variable sale price both of them have been assigned object datatype when they should be of numerical datatype.

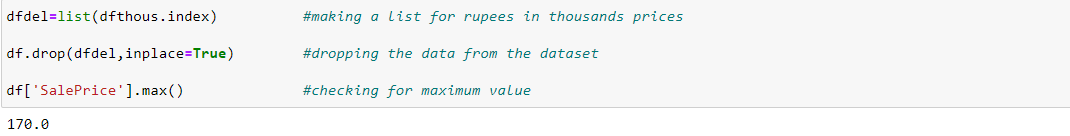
Upon checking out the sale price variable I’d found that most of the prices were in lakh rupees a few in thousands and a few in ten million. So, in order to convert this variable into numeric ones I had to deal with strings present alongside the numbers, I decided to let values present in lakhs only so as the numbers doesn’t look too big. For this, I replaced the ‘lakh’ string with empty strings and multiplied ten lakhs ones with 100.



But as the prices were in thousands as well, obviously the maximum values that were coming in thousands only. There were 111 cars which had prices in thousands, so I decided to drop these values also from the dataset.



Then I dropped the values in thousands and again checked for maximum value for sale prices, this time it was 170 lakhs.



Now for kilometres that the vehicles have been driven the only problem was the period separating ones, tens, hundreds etc. units of numbers. So, I simply replaced those periods with empty strings and converted the variable into numeric variable using pandas only. After that datatypes were checked again.

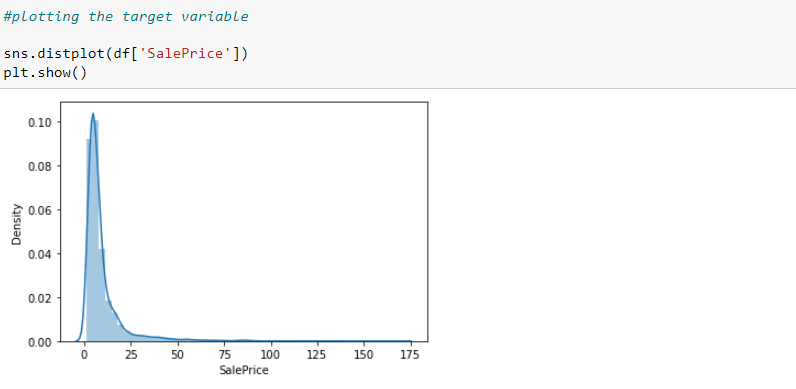


It can be seen that the kilometres driven variable has been changed to ‘int64’ from ‘object’ and sale price variable to ‘float64’.

Now we can go for exploratory data analysis.

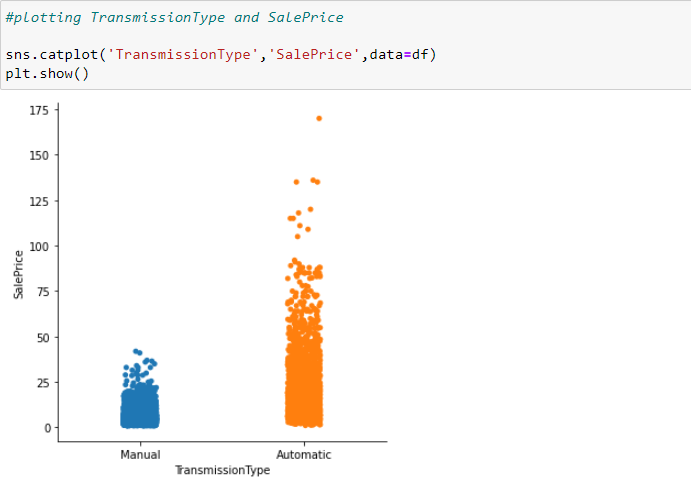
# **Exploratory Data Analysis**

## The response variable – SalePrice



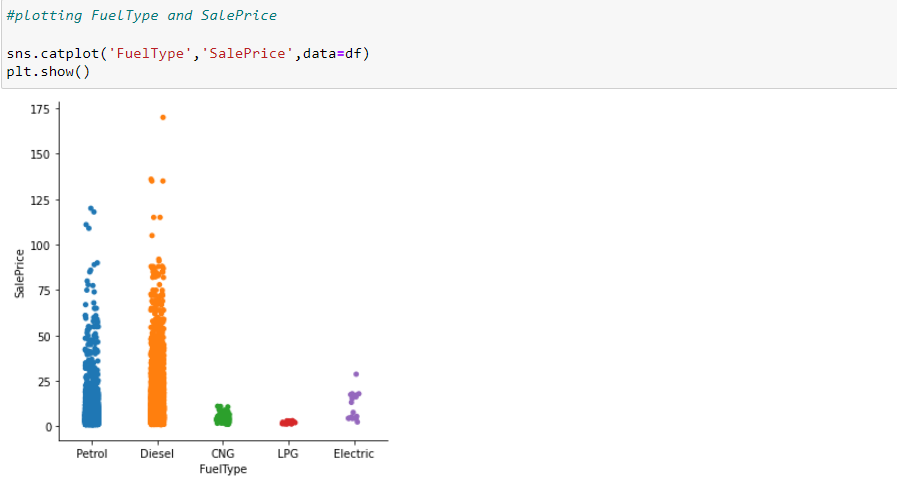
Above figure gives us the distribution plot of the sale price. Most of the prices for used cars is in range 1 lakh to 25 lakhs while maximum is around 170 lakhs. And there seems to be no large outliers in the sale price variable but the data is right skewed.

## SalePrice and TransmissionType



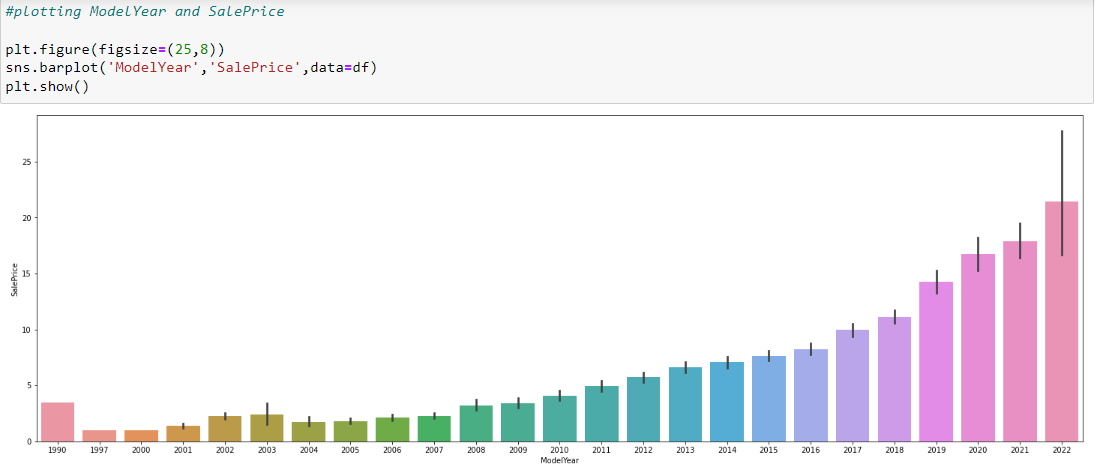
It can be seen that automatic transmission vehicles have higher sale prices than the manual ones, also the used car with maximum price has automatic transmission. Moreover manual transmission vehicles have price in lesser than 40-45 lakhs.

## SalePrice and FuelType



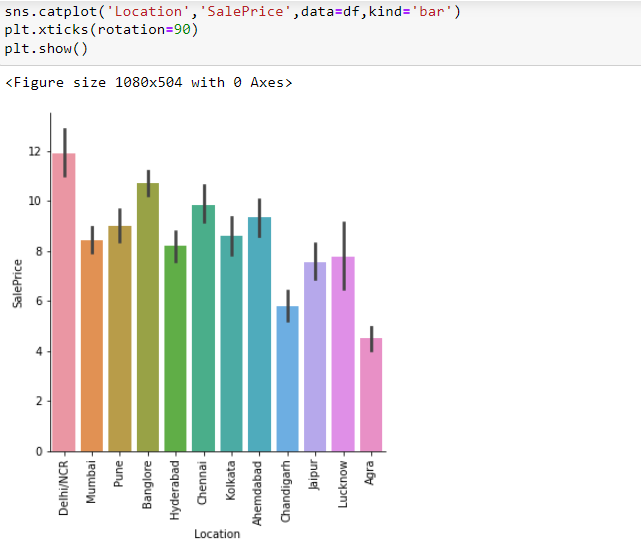
Both petrol and diesel vehicles have good sale price range, the used car with maximum sale price has an automatic transmission and is a diesel vehicle as well. For CNG and LPG vehicles sales price has very short range.

## SalePrice and ModelYear



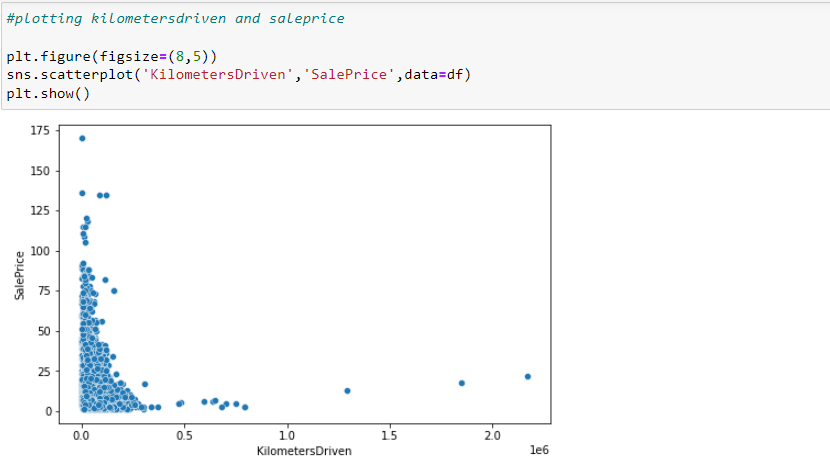
Year the model been made seems to be correlated with the target variable. Model year 2022 certainly has the highest sale prices of used cars. While model year 1997 has the lowest sale prices year 1990 has higher sale prices than 1997 to 2008 model year.

## SalePrice and Location



Delhi/NCR region has the highest sale prices Chandigarh has the lowest also as the sale prices for different locations aren’t varying much, which can means that location variable might not have a good correlation with the target variable sale prices.

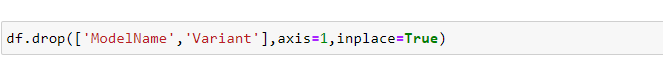
## SalePrice and KilometersDriven



Few the kilometres driven higher the sale prices. Vehicles that have been driven more than even fifty thousand have a very low sale price. From the above plot it can also be seen that data for kilometres driven is highly skewed.

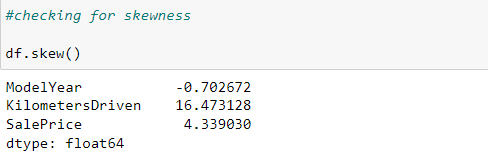
# **Data Pre-processing**

First, we had dropped ‘ModelYear’ and ‘Variant’ from the dataset, because it must not contribute to the prediction of sale prices also both these categorical variables have large number of unique classes present, so it would be a bit complex in converting these categorical variables into numerical ones.

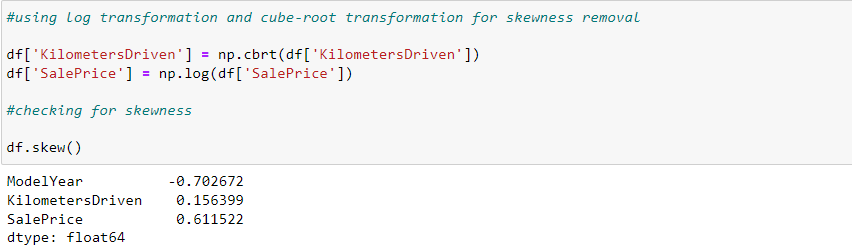


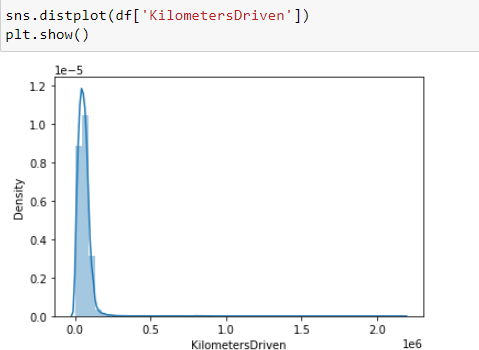
## Skewness removal

Now before going for feature selection and building the models skewness should be checked and reduced or removed if present in high number. So for that, pandas skew function was used to check the skewness. Turns out both the target variable sale price and input variable kilometres driven got a very high skewness.

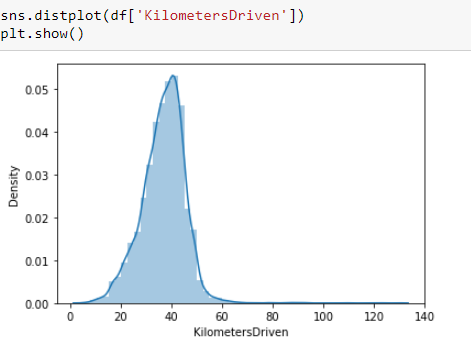


As the distribution is not uniformly distributed for these two variables have a very high skewness present. To reduce the skewness and normalize the data for these two variables I’d used log transformation and cube root transformation to reduce the skewness. Log transformation improves linearity between our dependent and independent variables. It boosts validity of our statistical analyses. In cases of right skewed data log transformation is preferred. After applying log transformation on both those variables I again checked for skewness. Below I am providing the code to do so and two distribution plots for kilometres driven, one is before log transformation and the other one is after that.





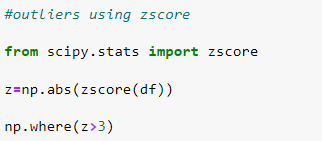
**Before Log transformation**



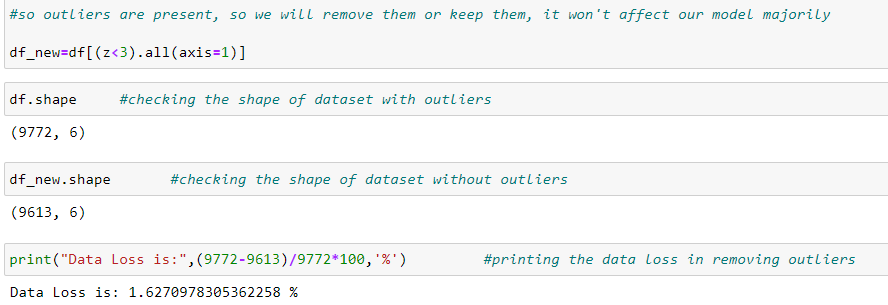
**After log transformation**

## Outliers

Now, that the dataset isn’t skewed I checked for outliers using z present before converting and selecting the features. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean. If the z score of a data point is more than 3or -3, it indicates that the data point is quite different from the other data points. Such a data point can be an outlier.



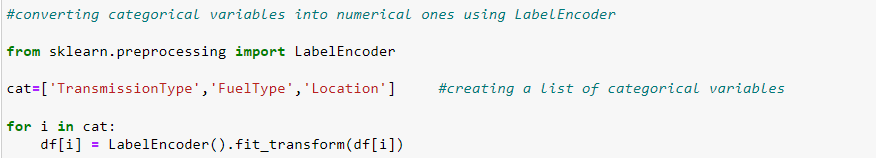
The above code shows locations where the value for z-score is greater than 3. We have ignored negative values by using numpy’s absolute function which returns positive values only. Now, I am going to remove the outliers from the dataset and check for data loss while doing so. If the data loss percent is low we are going to proceed with the dataset without outliers.



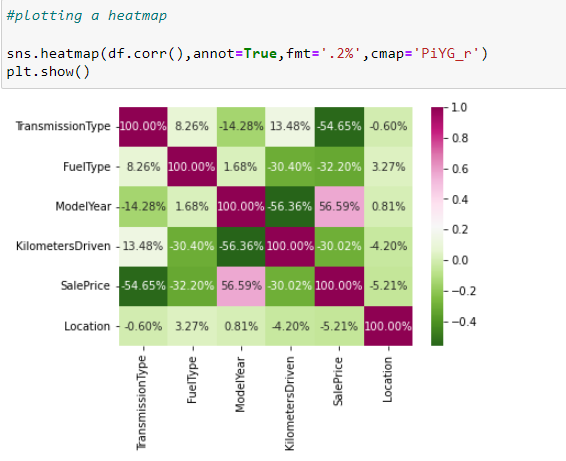
### Converting variables

Before going for the model building phase we **need to convert a lot of features into numeric** ones later on, so that the machine learning algorithms can process them.

For this we had used Label Encoder from sci-kit learn library.  In label encoding, we replace the categorical value with a numeric value between 0 and the number of classes minus 1.



## Correlation



The correlation matrix gives us the correlation value which is in range of -1 to 1. From above figure we can conclude that model year and type of transmission have the highest correlation bond with the target variable following by fuel type and kilometres driven. Location has the weakest correlation bond with the target variable.

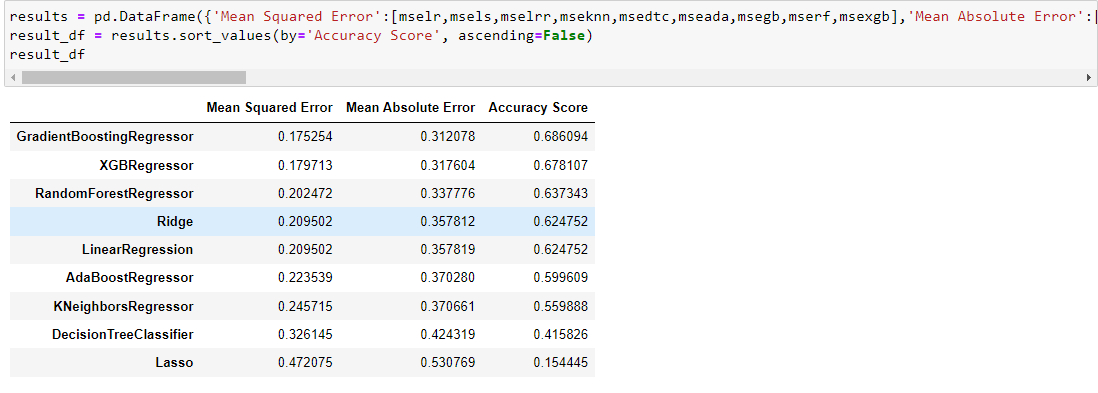
Now, as we have done the preprocessing part, reduced skewness, converted categorical variables into numerical ones, we can go onto building machine learning models using different algorithm.

# **Building Machine Learning Models**

Now we will train several Machine Learning models and compare their results. Later on, we will use cross validation and other evaluation metrics.

So, we had tried in total 9 different algorithms which include, linear regression, 2 regularization methods, k-neighbors, decision tree and 3 ensemble techniques and XGBoost. While trying out different models we had compare the accuracy score and errors for the best model.

## Best Model?

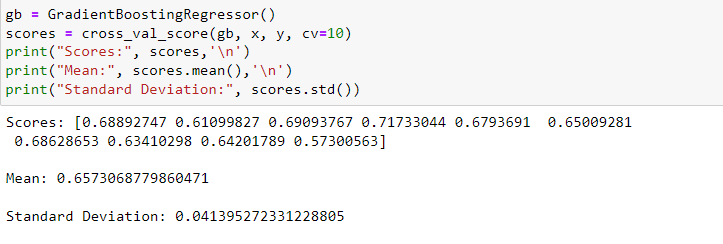


As we can see, the Gradient Boosting Regressor goes on the first place. But first, we must check, how random-forest performs, when we use cross validation.

## K-Fold Cross Validation

K-Fold Cross Validation randomly splits the training data into **K subsets called folds**. Let’s image we would split our data into 4 folds (K = 4). Our random forest model would be trained and evaluated 4 times, using a different fold for evaluation every time, while it would be trained on the remaining 3 folds.

The code below perform K-Fold Cross Validation on our random forest model, using 10 folds (K = 10). Therefore it outputs an array with 10 different scores.  We then need to compute the mean and the standard deviation for these scores.



This looks much more realistic than before. Our model has an average accuracy of 65% with a standard deviation of 4 %. The standard deviation shows us, how precise the estimates are.

This means in our case that the accuracy of our model can differ **+**— 4%.

The accuracy is still really good and since gradient boosting is an easy to use model, we had try to increase its performance even further in the following section.

## Gradient Boosting

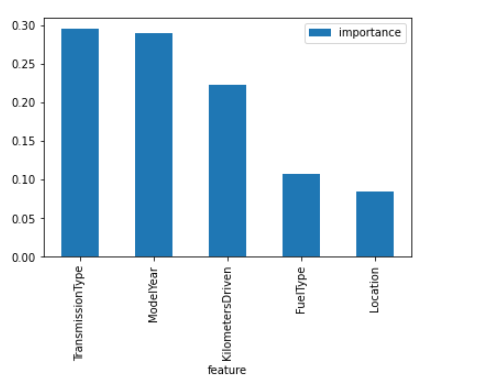
Gradient boosting is one of the most popular machine learning algorithms for tabular datasets. It is powerful enough to find any nonlinear relationship between your model target and features and has great usability that can deal with missing values, outliers, and high cardinality categorical values on your features without any special treatment. While you can build bare bone gradient boosting trees using some popular libraries such as XGBoost or LightGBM without knowing any details of the algorithm, you still want to know how it works when you start tuning hyper-parameters, customizing the loss functions, etc., to get better quality on your model.

Gradient boosting is one of the variants of ensemble methods where you create multiple weak models and combine them to get better performance as a whole.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

## Feature Importance

Another great quality of gradient boosting is that they make it very easy to measure the relative importance of each feature. Sci-kit learn measure a features importance by looking at how much the tree nodes, that use that feature, reduce impurity on average (across all trees in the forest). It computes this score automatically for each feature after training and scales the results so that the sum of all importance is equal to 1.



It came out to be that Location variable doesn’t play a significant role in our gradient boosting regressor prediction process. Because of that we will drop it from the dataset and train the regressor again. We could also remove more or less features, but this would need a more detailed investigation of the features effect on our model.

Also, from the above figure we can conclude that transmission type and model year are highly important features in predicting the sale prices, following by kilometres driven.

Now we can start tuning the hyperameters of random forest.

## Hyperparameter Tuning

Below is the code of the hyperparamter tuning for several parameters of gradient boosting regressor. I have ran the codes separately to find out the best parameter values as it takes a lot of time when ran collectively.



Now that we have optimum values for a few parameters, we are going to test these parameters and now that we have a proper model, we can start evaluating its performance in a more accurate way.

# **Evaluation**

## Mean Squared Error

Mean Squared Error or MSE for short, is a popular error metric for regression problems.

It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here “*least squares*” refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

MSE = 1 / N \* sum for i to N (y\_i – yhat\_i)^2

Where *y\_i* is the i’th expected value in the dataset and *yhat\_i* is the i’th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

The squaring also has the effect of inflating or magnifying large errors. That is, the larger the difference between the predicted and expected values, the larger the resulting squared positive error. This has the effect of “*punishing*” models more for larger errors when MSE is used as a loss function. It also has the effect of “*punishing*” models by inflating the average error score when used as a metric.



## Root Mean Squared Error

The Root Mean Squared Error MSE, is an extension of the mean squared error.

Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units “*dollars*,” then the RMSE error score will also have the unit “*dollars*” and not “*squared dollars*” like the MSE.

As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

The RMSE can be calculated as follows:

RMSE = sqrt(1 / N \* sum for i to N (y\_i – yhat\_i)^2)

Where *y\_i* is the i’th expected value in the dataset, *yhat\_i* is the i’th predicted value, and *sqrt()* is the square root function.

We can restate the RMSE in terms of the MSE as:

RMSE = sqrt(MSE)

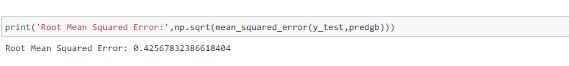
Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an example of Jensen’s Inequality.

You may recall that the square root is the inverse of the square operation. MSE uses the square operation to remove the sign of each error value and to punish large errors. The square root reverses this operation, although it ensures that the result remains positive.

The root mean squared error between your expected and predicted values can be calculated using the mean\_squared\_error function() from the scikit-learn library.

By default, the function calculates the MSE, but we can configure it to calculate the square root of the MSE by setting the “*squared*” argument to *False*.

The function takes a one-dimensional array or list of expected values and predicted values and returns the mean squared error value.



## Mean Absolute Error

Mean Absolute Error or MAE, is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.

That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or *abs()* is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted value may be positive or negative and is forced to be positive when calculating the MAE.

The MAE can be calculated as follows:

* MAE = 1 / N \* sum for i to N abs(y\_i – yhat\_i)

Where *y\_i* is the i’th expected value in the dataset, *yhat\_i* is the i’th predicted value and *abs()* is the absolute function.



## R Squared score

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the R2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

R-squared values range from 0 to 1 and are commonly stated as percentages from 0% to 100%. An R-squared of 100% means that all movements of a security (or another dependent variable) are completely explained by movements in the index (or the independent variable(s) you are interested in).



The Gradient Boosting model is giving us a great r-squared score. 68.5% of R-squared means up to 68.5% of feature variables movements are completely explained by movements of label variable.

## Mean Squared Log Error

Mean squared logarithmic error (MSLE) can be interpreted as a measure of the ratio between the true and predicted values.

Mean squared logarithmic error is, as the name suggests, a variation of the Mean Squared Error.

The introduction of the logarithm makes MSLE only care about the relative difference between the true and the predicted value, or in other words, it only cares about the percent difference between them.  
This means that MSLE will treat small differences between small true and predicted values approximately the same as big differences between large true and predicted values.

The loss is the mean over the seen data of the squared differences between the log-transformed true and predicted values, or writing it as a formula:

L(y, \hat{y}) = \frac{1}{N} \sum\_{i=0}^{N}(\log(y\_i + 1) - \log({\hat{y}}\_i + 1))^2*L*(*y*,*y*^​)=*N*1​*i*=0∑*N*​(log(*yi*​+1)−log(*y*^​*i*​+1))2

where ŷ is the predicted value.

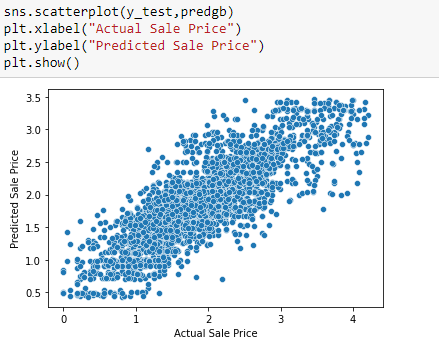
This loss can be interpreted as a measure of the ratio between the true and predicted values, since:

\log(y\_i + 1) - \log({\hat{y}}\_i + 1) = log\Bigg(\frac{y\_i + 1}{{\hat{y}}\_i + 1}\Bigg)log(*yi*​+1)−log(*y*^​*i*​+1)=*log*(*y*^​*i*​+1*yi*​+1​)

The reason ‘1’ is added to both y and ŷ is for mathematical convenience since log(0)is not defined but both y or ŷ can be 0.



## Predicted VS Actual Values



The model that we have created has an accuracy of around 70% which seems realistic, might be low too but the errors are low as well, which means the model is performing really well.

# **Conclusion**

The objective of this project is to fit models to predict the used cars sale price and find some important aspects of the used cars. In order to achieve this goal, I had fit nine different regression models to the dataset: linear regression, lasso regression, ridge regression, k-nearest neighbors, decision tree, ada boost, gradient boosting and random forest and xgboost.

## Findings

As for the first model -linear regression, it doesn’t meet the assumption of equality of the variances. Therefore we can’t use the linear model as the candidate of our final model.

In order to deal with this problem, I try the second and third model -lasso regression and ridge regression but the accuracy score looked not so good.

The fourth and fifth model were k-nearest neighbors and decision tree both these model had a very low accuracy score.

Afterwards, we tried three ensemble algorithms ada boost was the sixth model with a very low accuracy score so we moved ahead. Gradient boosting was the seventh model with a very high accuracy score up to now. Finally, we had tried random forest and xgboost which gave a low accuracy score than gradient boosting. But before, selecting gradient boosting as the model to go on with, we had use cross validation score to check how the gradient boosting model is really performing. Cross validation score of gradient boosting model came out to be as realistic as its accuracy score.

Thus, we had used hyperparameter tuning to find the optimum value of model’s parameters. After re-training our model with tuned parameters we jumped onto evaluation part.

What’s more, from the feature importance plot of the gradient boosting, we could know that the transmission type and model year had the very high feature importance when compared to all the other feature variables. Fuel type and kilometres driven also had reasonable feature importance. We also saw that Location variable had a very low feature importance value.

## Limitations of this work and Scope for Future Work

## Of course there is still a big room for improvement as we in this project did not do much extensive feature engineering. We had taken the more simple way of looking into data, analysing it and making some models.

The limitations of this project is that it lacks extensive feature engineering and intensive hyperparameter tuning.

The scope for future work with this project can be so wide. I had suggest like doing a more extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. Feature engineering also includes creation of new variables using existing feature variables. Extensive feature engineering has very great future scope to further extend the study of this project. Thus, improving all over results.

Before building regression models multicollinearity between feature has to be checked using various methods. One way is using variance inflation factor. A variance inflation factor higher than 4-10, results in multicollinearity. Higher the variance inflation factor higher the multicollinearity. It can be reduced by dropping one of the multicollinear features with lowest correlation and high variance inflation factor. Multicollinearity can be reduced by doing an extensive feature engineering too.

Another thing that can improve the overall result of the model would be a more extensive hyperparameter tuning on several machine learning models. Finding optimum value for several parameters of an algorithm model optimizes the overall performance of the model by some extent.

# **References**

* <https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8>
* <https://www.investopedia.com/terms/r/r-squared.asp#:~:text=R%2Dsquared%20values%20range%20from,)%20you%20are%20interested%20in>
* <https://machinelearningmastery.com/regression-metrics-for-machine-learning/>
* <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-squared-logarithmic-error-(msle)>
* <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>
* <https://www.geeksforgeeks.org/z-score-for-outlier-detection-python/>