FLIP ROBO

CAR PRICE PREDICTION PROJECT

Submitted By :

Amar Kumar

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my trainer at Data Trained as well as my SME at Flip Robo Technology" who gave me the golden opportunity to do this wonderful project on the topic "Car Price Prediction Project", which also helped me in doing a lot of research and exploring so many new things.

I am really thankful to Flip Robo Technologies for giving me this prime opportunity of interning in their organization and enhancing my skills.

Secondly, I want to thank Data Trained Academy for giving me the best possible study notes, online platforms such as www.medium.com, www.stackoverflow.com, www.scikit-learn.org and www.github.com for providing me with the resources when I stumbled and fell along the way.

It helped me increase my knowledge and skills.

THANKS AGAIN TO ALL WHO SUPPORTED.

Introduction

Business Problem

Because of the impact of covid 19 in the growing industries, a lot of uncertainty for its demand have been found in the car market. Few of them show high demand hence costing a high amount and few have no demand making them cost less for their product. Since, there has been a change in the price values of the cars due to the impact of covid 19, price valuation of previous cars needs to be updated with the current scenario in the market hence making the old machine learning models difficult to predict the prices of the cars with respect to current market demands.

Background Domain Knowledge

The prices of cars keep changing in accordance to their demand in the market. The owners of the cars put their used cars on sale on various e-commerce websites such as Olx, cardekho, Cars24 etc. These websites act as a broker/ middleman between the buyer and seller by helping them make a clear deal by direct contacts. These sellers need to understand the current market prices for their used cars so that they could attract more customers. The system can help people in predicting and understanding the range of car prices as per the current market scenario.

Literature Review

It is very surprising to know that tasks on estimated car price prediction is a new work but is also very scattered. Regression models are used to make predictions on such supervised machine learning techniques. The predictions are wholly and solely based on the data of e-commerce websites. Techniques such as Linear Regression are used to predict continuous values of data.

Project Motivation

The prime motivation of this task is to help our client by building a machine learning model that will be used to predict car prices for new data. Since our client was facing problems with their previous car price valuation machine learning models, a machine built on new data will help them resolve their issues and make profits to their business.

Analytical Problem Framing

Analytical Modeling of Problem

The project "Car Price Prediction Project" is accomplished in two phases –

1) Data Collection Phase

2) Model Building Phase.

It holds detailed information on all the important factors required to decide the price of a car. The data collected had missing values in all variables and the information has both categorical and numerical data.

Problem Solving was handled in 5 parts:

1. **Project Domain Research -** The most essential part is to have a clear understanding of the business problem and domain knowledge before starting to have hands on the building phase of the task.
2.
3. **Collecting Data -** Data collection was done by using web scraping on the information provided on the e-commerce sites. All the necessary elements required for prediction were scrapped using tools such as Selenium & BeautifulSoup.

4. **Data Cleaning -** Data contained few extra columns, missing values and redundant information, data cleaning and modifications were made to deal with raw data to convert it to usable format.

5. **Data Analysis -** Visualizing information provided in each variable and understanding their behaviour helps in better understanding of data and clear the unwanted features which do not serve of much importance in making predictions.

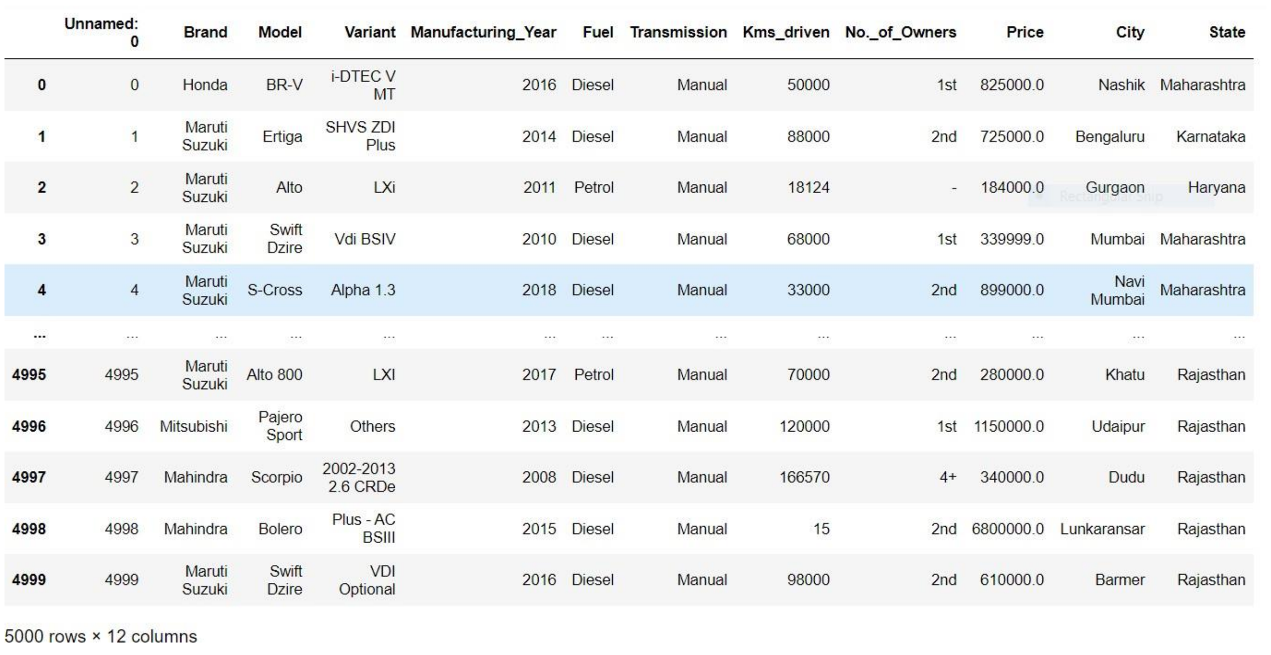
6. **Interpreting Solutions -** Once data is converted into machine understandable format, different Regression algorithms are used to train data, hypertune algorithms and interpreting the best solutions using different testing techniques and evaluation metrics to reduce losses.

Data Sources

The data is collected from an e-commerce website " https://www.olx.in/ " using web scraping methods by making use of tools such as Selenium and Beautifulsoup. The data set consists of different features of cars such as brand, model, variant, manufacturing year, kilometers driven , fuel, number of owners, location and the target variable i.e. Price of the car. The data is saved in the form of an excel sheet.

It contains 5000 records and 12 variables initially. Data set had null values in all columns that were handled and substantial EDA was performed with the redundant data to achieve relationships between input variables and it's target variable "Price".

Representing Data -



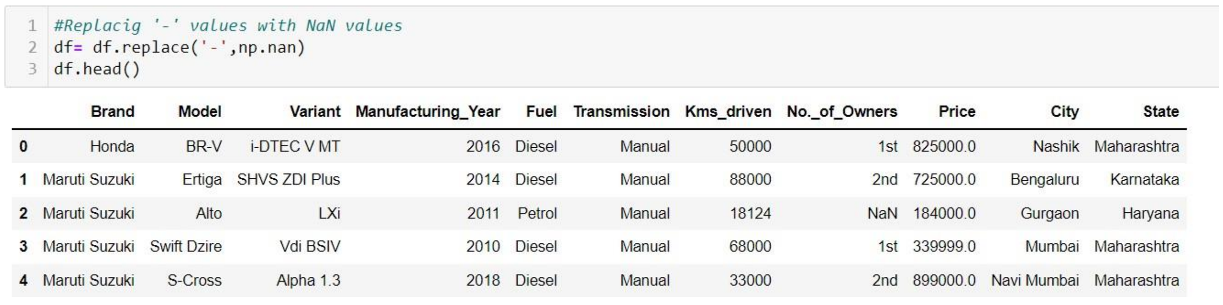
	Unnamed: 0	Brand	Model	Variant	Manufacturing_Year	Fuel	Transmission	Kms_driven	No._of_Owners	Price	City	State
0	0	Honda	BR-V	i-DTEC V MT	2016	Diesel	Manual	50000	1st	825000.0	Nashik	Maharashtra
1	1	Maruti Suzuki	Ertiga	SHVS ZDI Plus	2014	Diesel	Manual	88000	2nd	725000.0	Bengaluru	Karnataka
2	2	Maruti Suzuki	Alto	LXi	2011	Petrol	Manual	18124	-	184000.0	Gurgaon	Haryana
3	3	Maruti Suzuki	Swift Dzire	Vdi BSIV	2010	Diesel	Manual	68000	1st	339999.0	Mumbai	Maharashtra
4	4	Maruti Suzuki	S-Cross	Alpha 1.3	2018	Diesel	Manual	33000	2nd	899000.0	Navi Mumbai	Maharashtra
...	...	...	...	...	...	...	...	...	...	...	...	...
4995	4995	Maruti Suzuki	Alto 800	LXI	2017	Petrol	Manual	70000	2nd	280000.0	Khatu	Rajasthan
4996	4996	Mitsubishi	Pajero Sport	Others	2013	Diesel	Manual	120000	1st	1150000.0	Udaipur	Rajasthan
4997	4997	Mahindra	Scorpio	2002-2013 2.6 CRDe	2008	Diesel	Manual	166570	4+	340000.0	Dudu	Rajasthan
4998	4998	Mahindra	Bolero	Plus - AC BSIII	2015	Diesel	Manual	15	2nd	6800000.0	Lunkaransar	Rajasthan
4999	4999	Maruti Suzuki	Swift Dzire	VDI Optional	2016	Diesel	Manual	98000	2nd	610000.0	Barmer	Rajasthan

5000 rows × 12 columns

Data Pre-processing

A data mining technique used for data cleaning and preparation and therefore transforming raw data into machine understandable format. The approach followed to clean data are :

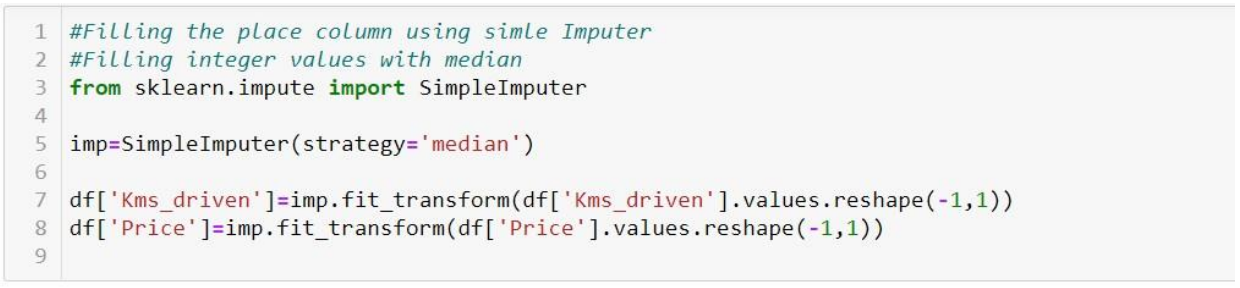
1. An extra existing unwanted variable called 'Unnamed: 0' was removed from the data set.
2. Converting "-" values into NaN values.



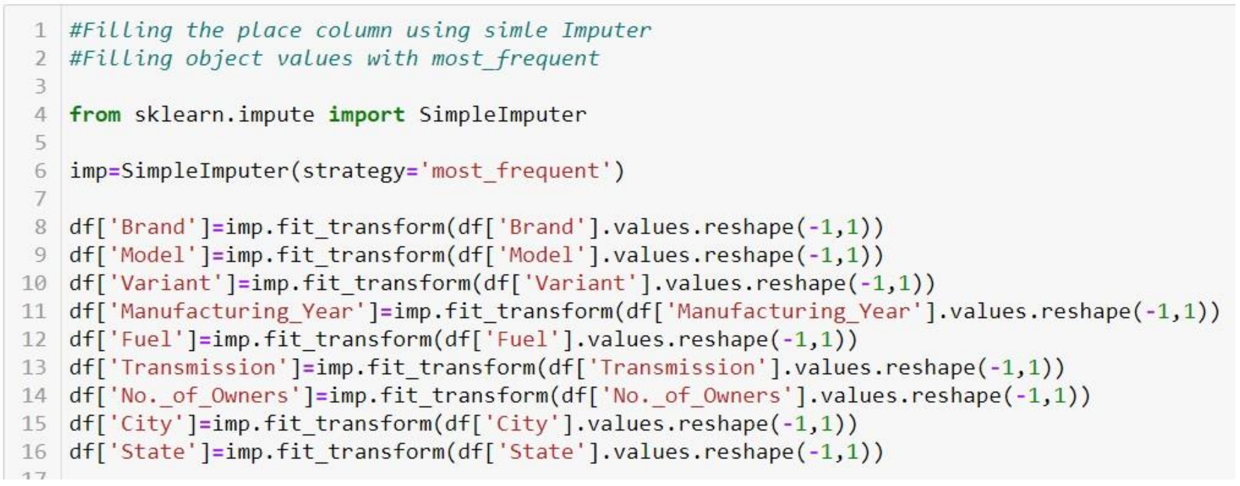
```
#Replacig '-' values with NaN values
df= df.replace('-',np.nan)
df.head()
```

	Brand	Model	Variant	Manufacturing_Year	Fuel	Transmission	Kms_driven	No._of_Owners	Price	City	State
0	Honda	BR-V	i-DTEC V MT	2016	Diesel	Manual	50000	1st	825000.0	Nashik	Maharashtra
1	Maruti Suzuki	Ertiga	SHVS ZDI Plus	2014	Diesel	Manual	88000	2nd	725000.0	Bengaluru	Karnataka
2	Maruti Suzuki	Alto	LXi	2011	Petrol	Manual	18124	NaN	184000.0	Gurgaon	Haryana
3	Maruti Suzuki	Swift Dzire	Vdi BSIV	2010	Diesel	Manual	68000	1st	339999.0	Mumbai	Maharashtra
4	Maruti Suzuki	S-Cross	Alpha 1.3	2018	Diesel	Manual	33000	2nd	899000.0	Navi Mumbai	Maharashtra

3. Modifying null values and replacing them using "Simple Imputer" functions so as to make data fit for training.

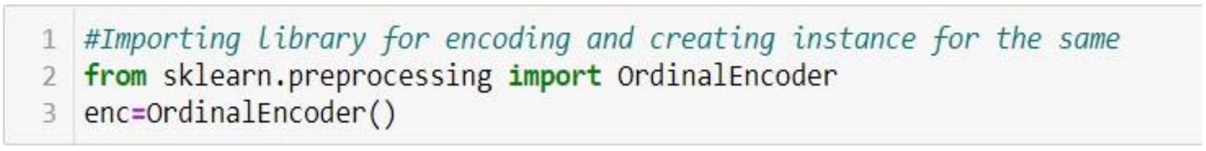


```python
#Filling the place column using simle Imputer
#Filling integer values with median
from sklearn.impute import SimpleImputer

imp=SimpleImputer(strategy='median')

df['Kms_driven']=imp.fit_transform(df['Kms_driven'].values.reshape(-1,1))
df['Price']=imp.fit_transform(df['Price'].values.reshape(-1,1))

```

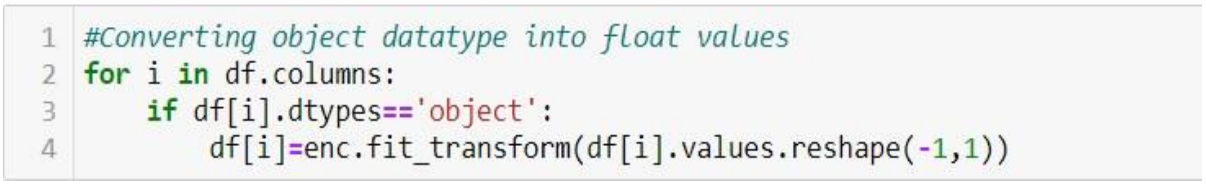


```python
#Filling the place column using simle Imputer
#Filling object values with most_frequent

from sklearn.impute import SimpleImputer

imp=SimpleImputer(strategy='most_frequent')

df['Brand']=imp.fit_transform(df['Brand'].values.reshape(-1,1))
df['Model']=imp.fit_transform(df['Model'].values.reshape(-1,1))
df['Variant']=imp.fit_transform(df['Variant'].values.reshape(-1,1))
df['Manufacturing_Year']=imp.fit_transform(df['Manufacturing_Year'].values.reshape(-1,1))
df['Fuel']=imp.fit_transform(df['Fuel'].values.reshape(-1,1))
df['Transmission']=imp.fit_transform(df['Transmission'].values.reshape(-1,1))
df['No._of_Owners']=imp.fit_transform(df['No._of_Owners'].values.reshape(-1,1))
df['City']=imp.fit_transform(df['City'].values.reshape(-1,1))
df['State']=imp.fit_transform(df['State'].values.reshape(-1,1))
```

4. Visualizing data individually and finding correlations with the target variable, hence clearing data which does not serve any importance in predicting car prices.
5. Use of "Label Encoder" techniques to encode data with object data type transforming them into numerical values. A sample of the code is shown below:



```python
#Importing library for encoding and creating instance for the same
from sklearn.preprocessing import OrdinalEncoder
enc=OrdinalEncoder()
```

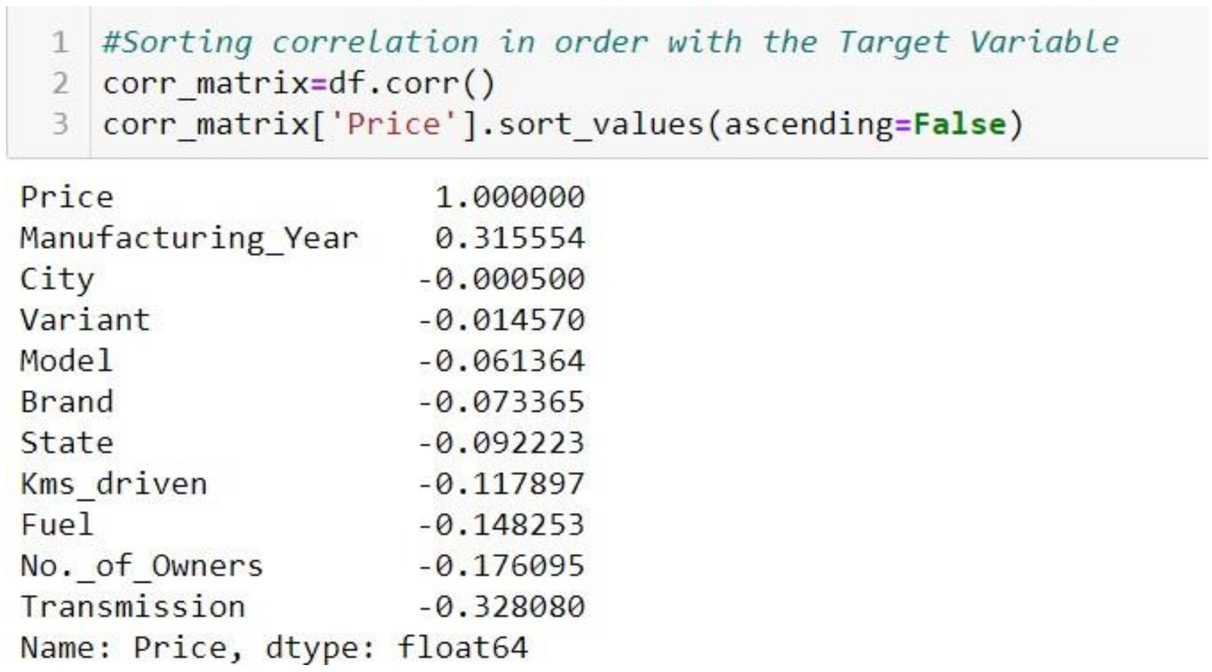


```python
#Converting object datatype into float values
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=enc.fit_transform(df[i].values.reshape(-1,1))
```

6. Outliers and Skewness present in data were not taken into consideration since all the input features are of the Object data type and Price is the Target Variable. Therefore, there is no skewness and outliers present in the data set.

Correlation Between Variables

Correlations between variables allow in identifying which features create a huge impact in making predictions on valuation of car prices and also the features which do not serve much in estimating the car value. Correlations near to 1 are highly correlated, near to 0 hardly have any correlation and variables having negative correlation i.e. near to -1 have high negative correlations. In this case, all features except “Manufacturing_Year” are negatively correlated with the target variable. Variable “City” hardly shows any correlation hence we remove the column from the data set. Correlation of all variables with the target variable are shown below -



```
#Sorting correlation in order with the Target Variable
corr_matrix=df.corr()
corr_matrix['Price'].sort_values(ascending=False)
```

```
Price                  1.000000
Manufacturing_Year     0.315554
City                  -0.000500
Variant               -0.014570
Model                 -0.061364
Brand                 -0.073365
State                 -0.092223
Kms_driven            -0.117897
Fuel                  -0.148253
No._of_Owners         -0.176095
Transmission          -0.328080
Name: Price, dtype: float64
```

Assumptions

Assumptions made during model building were -

- An assumption of the model being over-fitted or under-fitted was made because of the r2 score that was achieved after training and testing of the data set.
- High correlations of input features with the target variable was expected.
- It was assumed to have no skewness and outliers in the data set since the information achieved was in the form of categorical data.

Hardware and Software Requirements

Hardware

1. 16 GB RAM - used for data storage and training models.

2. CPU - used for executing algorithms.

Software

1. Pandas - A fast, powerful and flexible open source tool used for data manipulation and data analysis.

2. Numpy - Numpy is an open source library used for mathematical and computational analysis of data and works best with multidimensional arrays and matrices.

3. Scikit-Learn - Sklearn is a free Machine Learning library used to run various algorithms and consists of scientific libraries like NumPy and SciPy.

4. Seaborn - Seaborn is a library used to plot graphs. It also visualizes random distributions.

5. Matplotlib - It works like MATLAB and helps pyplot function in making some changes to a figure such as creating a figure, creating plotting area, etc.

6. Pickle - Pickle library is used to save models which can be used for both "dumping" and "loading" purposes.The model can be used again to read and write whenever required.

Model Development & Evaluation

Problem Solving Approach

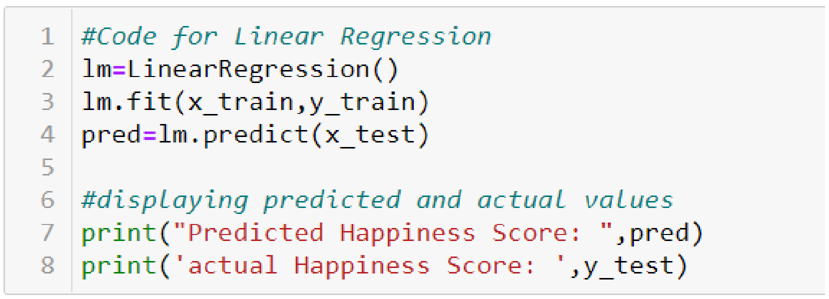
- Since the predicting value is of the continuous type, Regression algorithms are used for data training and testing.
- 7 different algorithms were used and hypertuned to test the performance of the model and the one with the best r2 score was selected.
- Best random state to attain a high performance model.
- Plotting Best Fit line graph to visualize if we are covering all data points or if the model is overfitting or underfitting.
- Different accuracy metrics such as MAE, MSE and RMSE were used apart from r2 score to test the performance accuracy of the model.

Algorithms Used

1. Linear Regression
2. ElasticNet Regression (Regularization Technique)
3. Ridge Regression (Regularization Technique)
4. Lasso Regression (Regularization Technique)
5. AdaBoostRegressor (Ensemble Technique)
6. GradientBoostingRegressor (Ensemble Technique)
7. RandomForestRegressor (Ensemble Technique)

Evaluation of Selected Models

1. **Linear Regression -**
   The training score for Linear Regression is 82.41%. R2 score is also 100%.



```python
#Code for Linear Regression
lm=LinearRegression()
lm.fit(x_train,y_train)
pred=lm.predict(x_test)

#displaying predicted and actual values
print("Predicted Happiness Score: ",pred)
print('actual Happiness Score: ',y_test)
```

```
#training score
lm.score(x_train,y_train)
```

```
1.0
```

```
#Checking r2 Score for the model
print(r2_score(y_test,pred))
```

```
1.0
```

2. ElasticNet Regression -

```
#Using ElasticNet Regression for our Model to solve overfitting and underfitting
from sklearn.linear_model import ElasticNet

parameters={'alpha':[.0001,.001,.01,.1,1,10],'random_state':list(range(0,10)),'selection':['cyclic','random']}
en=ElasticNet()
clf=GridSearchCV(en,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'alpha': 0.0001, 'random_state': 5, 'selection': 'random'}
```

```
#Code for ElasticNet Regression
en=ElasticNet(alpha=1,random_state=3,selection='random')
en.fit(x_train,y_train)
en.score(x_train,y_train)
pred_en=en.predict(x_test)

enn=r2_score(y_test,pred_en)
enn
```

```
0.8887896004178846
```

3. Ridge Regression -

```
#Using Ridge Regression for our Model to solve overfitting and underfitting
from sklearn.linear_model import Ridge

parameters={'alpha':[.0001,.001,.01,.1,1,10],'random_state':list(range(0,10)),'solver':['auto','svd','cholesky']}
rd=Ridge()
clf=GridSearchCV(rd,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

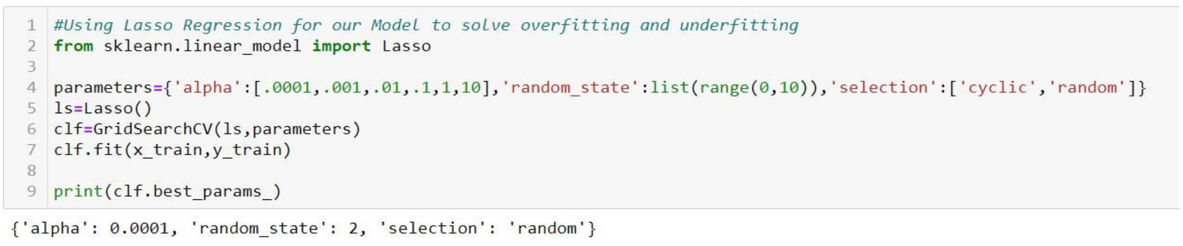
```
{'alpha': 0.0001, 'random_state': 0, 'solver': 'auto'}
```

```
#Code for Ridge Regression
rd=Ridge(alpha=0.0001,random_state=0,solver='auto')
rd.fit(x_train,y_train)
rd.score(x_train,y_train)
pred_rd=rd.predict(x_test)

rdd=r2_score(y_test,pred_rd)
rdd*100
```

```
99.99999999999991
```

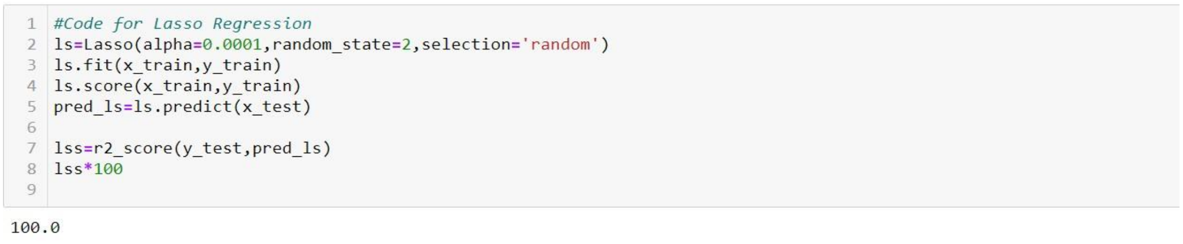
The r2 score for Ridge Regression is 99.99%

4. Lasso Regression -



```
#Using Lasso Regression for our Model to solve overfitting and underfitting
from sklearn.linear_model import Lasso

parameters={'alpha':[.0001,.001,.01,.1,1,10],'random_state':list(range(0,10)),'selection':['cyclic','random']}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'alpha': 0.0001, 'random_state': 2, 'selection': 'random'}
```



```
#Code for Lasso Regression
ls=Lasso(alpha=0.0001,random_state=2,selection='random')
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
pred_ls=ls.predict(x_test)

lss=r2_score(y_test,pred_ls)
lss*100

```

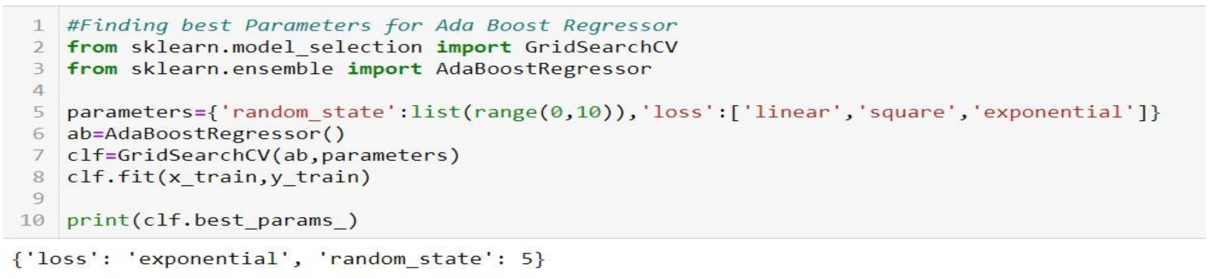
```
100.0
```

The r2 score for Lasso Regression is also 99.99%.

Conclusion Of Comparison between Regularization Techniques

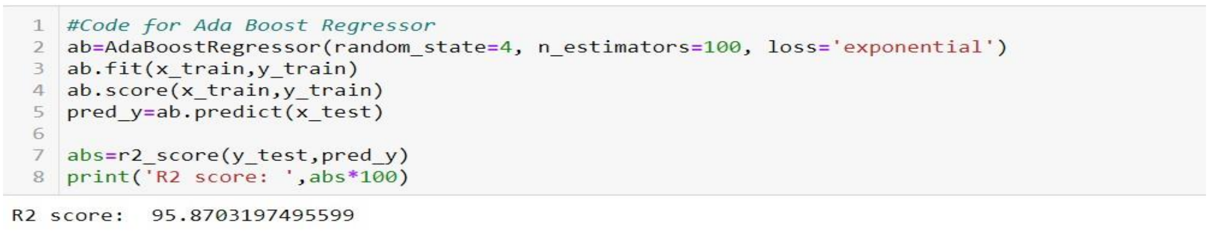
Hence, comparing all the three regularization techniques for regression i.e. ElasticNet, Ridge and Lasso, the model performs best for Lasso Regularization Regression technique at r2 score 100% accuracy.

5. AdaBoostRegressor -



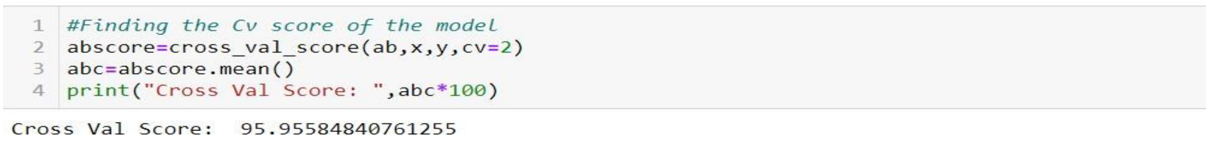
```
#Finding best Parameters for Ada Boost Regressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor

parameters={'random_state':list(range(0,10)),'loss':['linear','square','exponential']}
ab=AdaBoostRegressor()
clf=GridSearchCV(ab,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'loss': 'exponential', 'random_state': 5}
```



```
#Code for Ada Boost Regressor
ab=AdaBoostRegressor(random_state=4, n_estimators=100, loss='exponential')
ab.fit(x_train,y_train)
ab.score(x_train,y_train)
pred_y=ab.predict(x_test)

abs=r2_score(y_test,pred_y)
print('R2 score: ',abs*100)
```

```
R2 score:  95.8703197495599
```

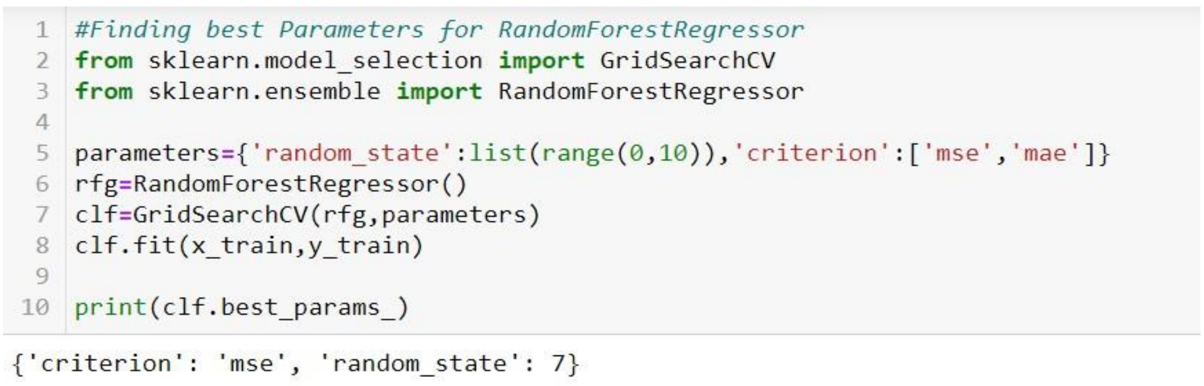


```
#Finding the Cv score of the model
abscore=cross_val_score(ab,x,y,cv=2)
abc=abscore.mean()
print("Cross Val Score: ",abc*100)
```

```
Cross Val Score:  95.95584840761255
```

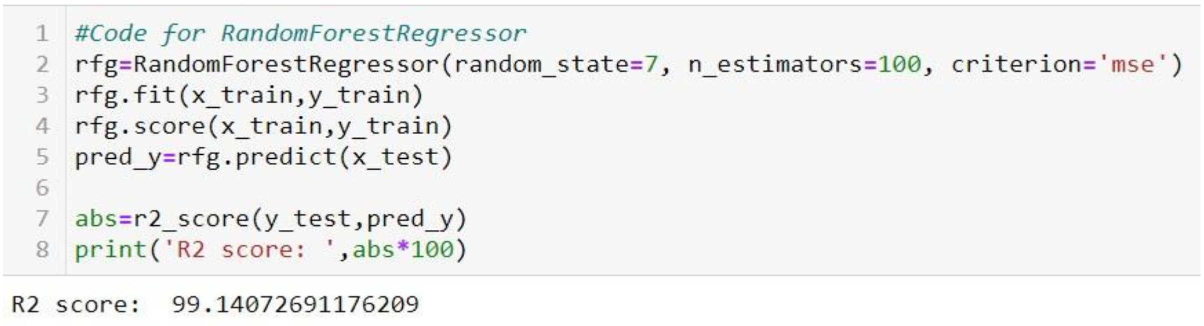
The r2 score for AdaBoost Regressor is 95.87% and that for it's cv score is 95.95%.

6. RandomForestRegressor -



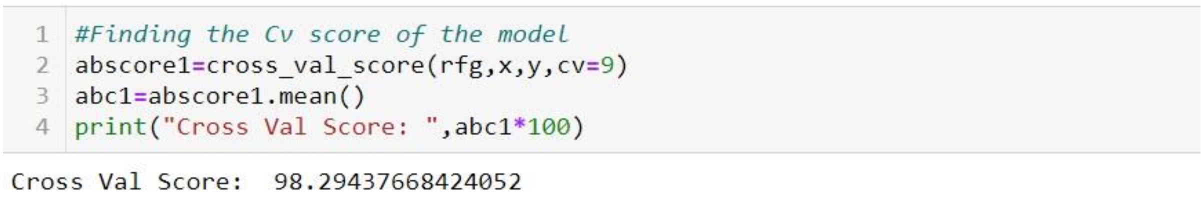
```
#Finding best Parameters for RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameters={'random_state':list(range(0,10)),'criterion':['mse','mae']}
rfg=RandomForestRegressor()
clf=GridSearchCV(rfg,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'criterion': 'mse', 'random_state': 7}
```



```
#Code for RandomForestRegressor
rfg=RandomForestRegressor(random_state=7, n_estimators=100, criterion='mse')
rfg.fit(x_train,y_train)
rfg.score(x_train,y_train)
pred_y=rfg.predict(x_test)

abs=r2_score(y_test,pred_y)
print('R2 score: ',abs*100)
```

```
R2 score:  99.14072691176209
```

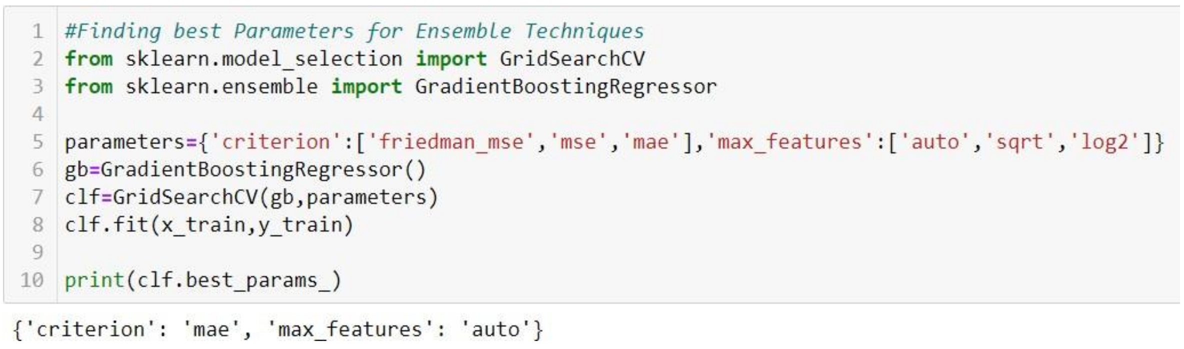


```
#Finding the Cv score of the model
abscore1=cross_val_score(rfg,x,y,cv=9)
abc1=abscore1.mean()
print("Cross Val Score: ",abc1*100)
```

```
Cross Val Score:  98.29437668424052
```

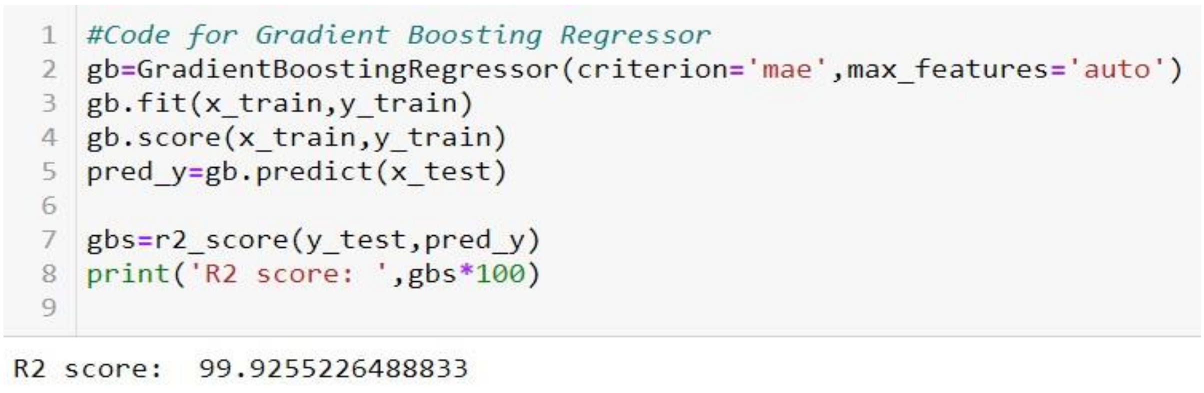
The r2 score for RandomForestRegressor is 99.14% and it's cv score is 98.29%.

7. GradientBoostingRegressor



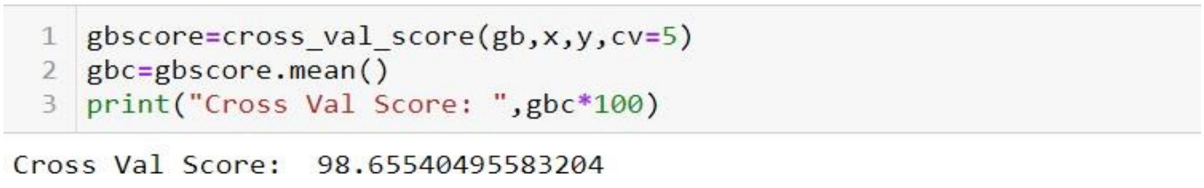
```
#Finding best Parameters for Ensemble Techniques
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor

parameters={'criterion':['friedman_mse','mse','mae'],'max_features':['auto','sqrt','log2']}
gb=GradientBoostingRegressor()
clf=GridSearchCV(gb,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'criterion': 'mae', 'max_features': 'auto'}
```



```python
#Code for Gradient Boosting Regressor
gb=GradientBoostingRegressor(criterion='mae',max_features='auto')
gb.fit(x_train,y_train)
gb.score(x_train,y_train)
pred_y=gb.predict(x_test)

gbs=r2_score(y_test,pred_y)
print('R2 score: ',gbs*100)

```

```
R2 score:  99.9255226488833
```



```python
gbscore=cross_val_score(gb,x,y,cv=5)
gbc=gbscore.mean()
print("Cross Val Score: ",gbc*100)
```

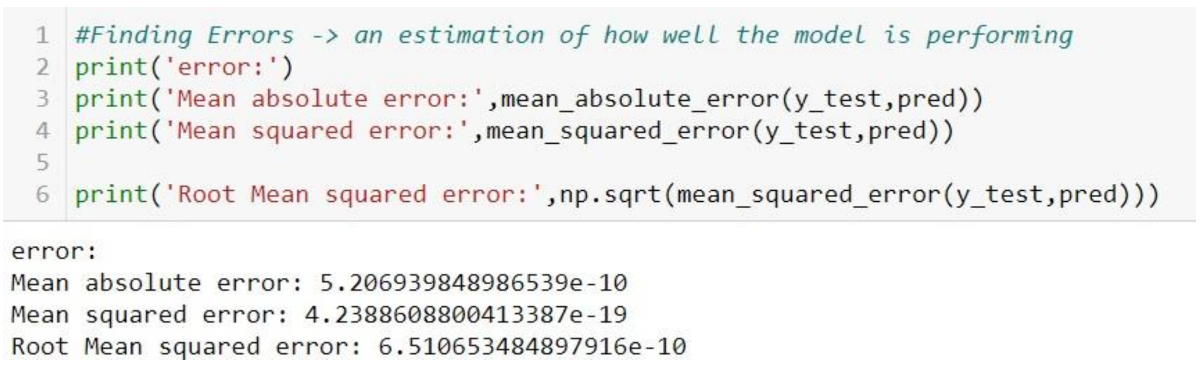
```
Cross Val Score:  98.65540495583204
```

- The r2 score after using Gradient Boosting Regressor is 99.92% which is the best algorithm among the ensemble techniques.
- The CV score after using Gradient Boosting Regressor is 98.65%

Conclusion -

After comparing all the algorithms and hypertuning each of them, we select Lasso Regularization Regression as our best model for this data set. The r2 score for Lasso Regression is 100%.

Evaluation Metrics

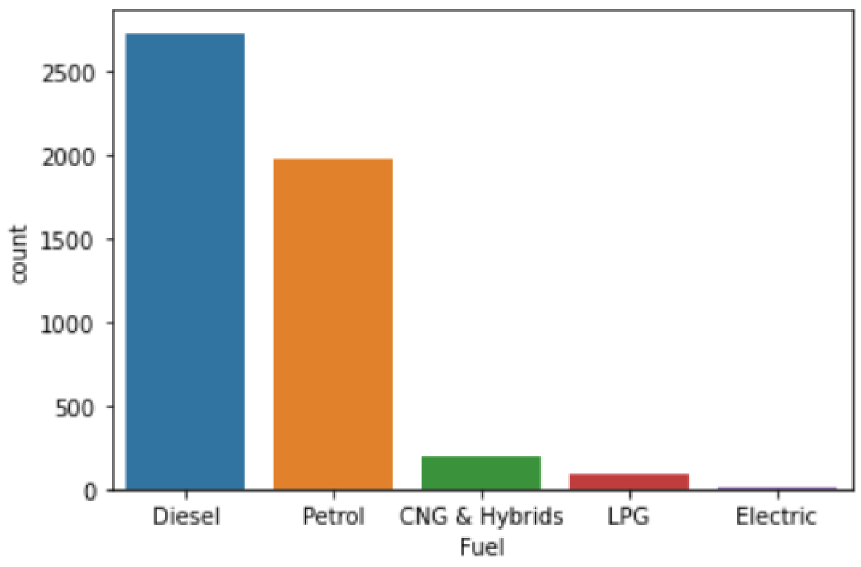


```python
#Finding Errors -> an estimation of how well the model is performing
print('error:')
print('Mean absolute error:',mean_absolute_error(y_test,pred))
print('Mean squared error:',mean_squared_error(y_test,pred))

print('Root Mean squared error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
error:
Mean absolute error: 5.206939848986539e-10
Mean squared error: 4.2388608800413387e-19
Root Mean squared error: 6.510653484897916e-10
```

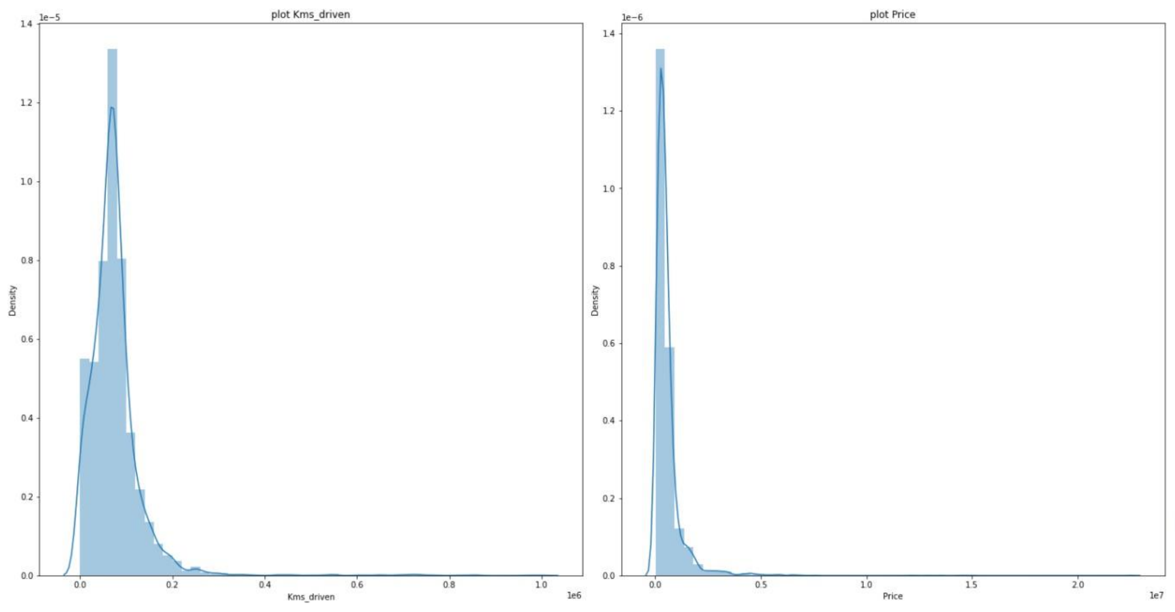
Visualizations

1. Countplot -



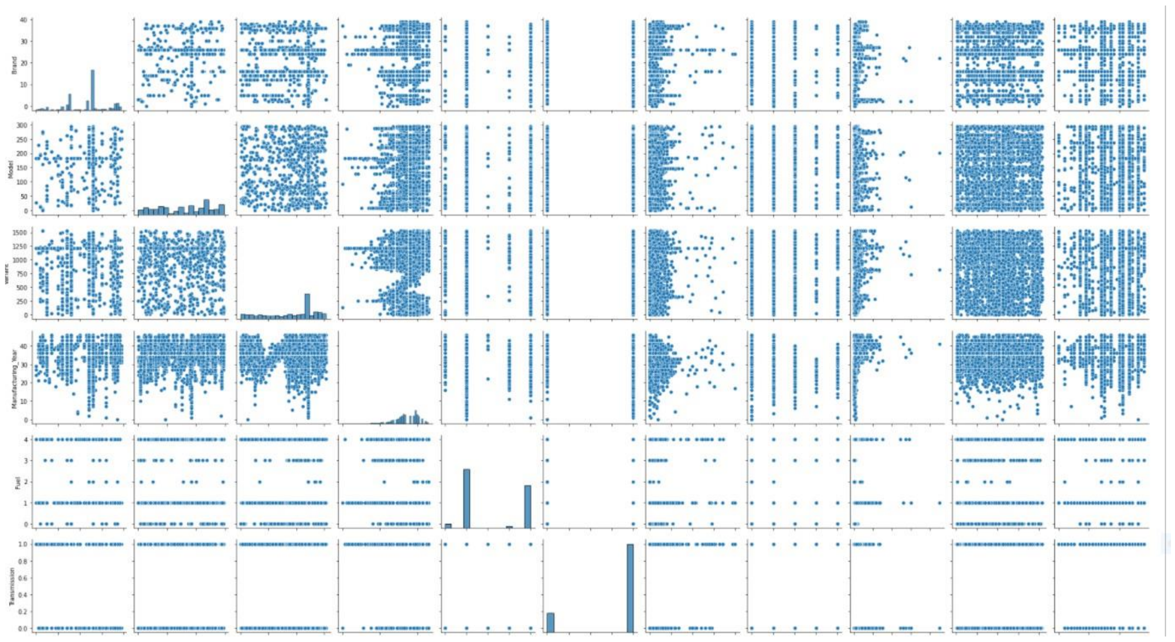
Individually used countplot for each variable to understand the data in detail. The above graph is an example of variable “Fuel” which gives detailed information of how many cars from records belong to a particular category.

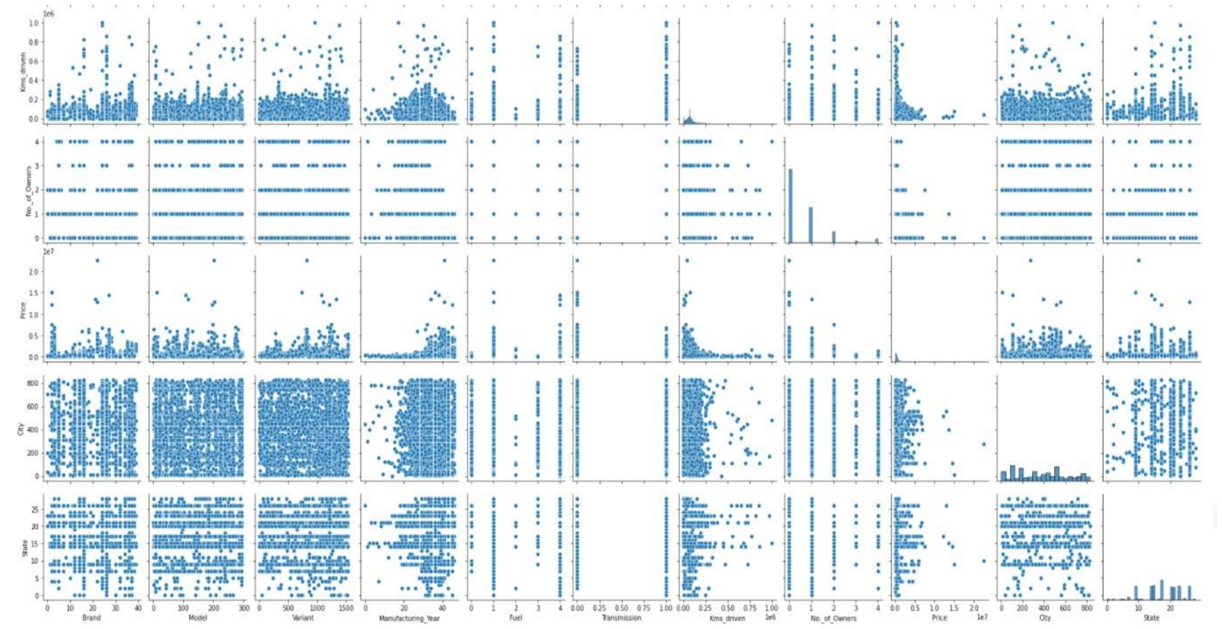
2. Density Graph -



Only two features from the data set belong to numerical type of data i.e. 'Kms_driven' and 'Price' which shows dispersion of values using density graphs.

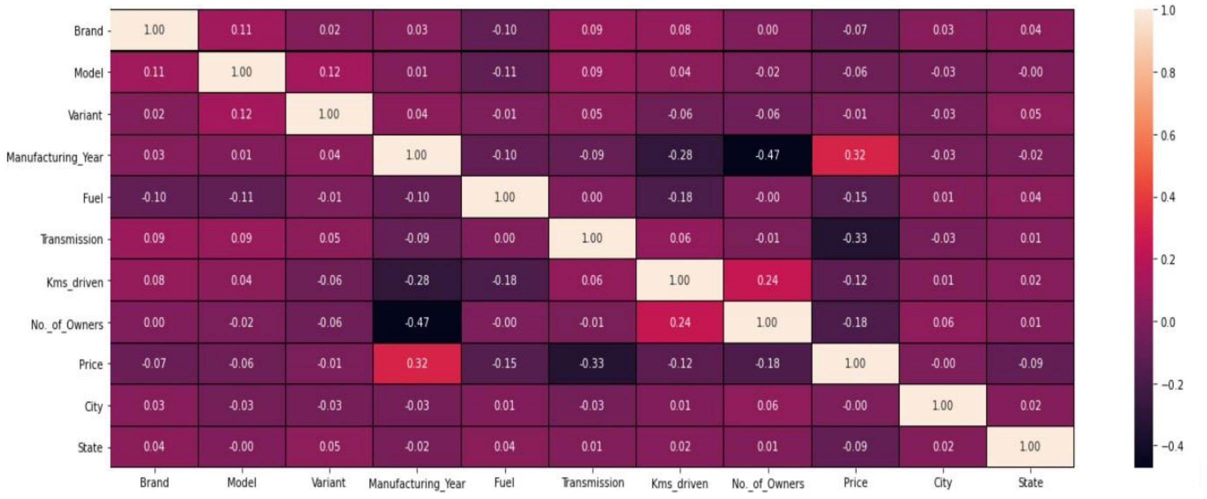
3. Pairplot -





Since most of the data in the dataset is of the object data type, it is difficult to understand the data using this graph. Data is scattered for a few variables.

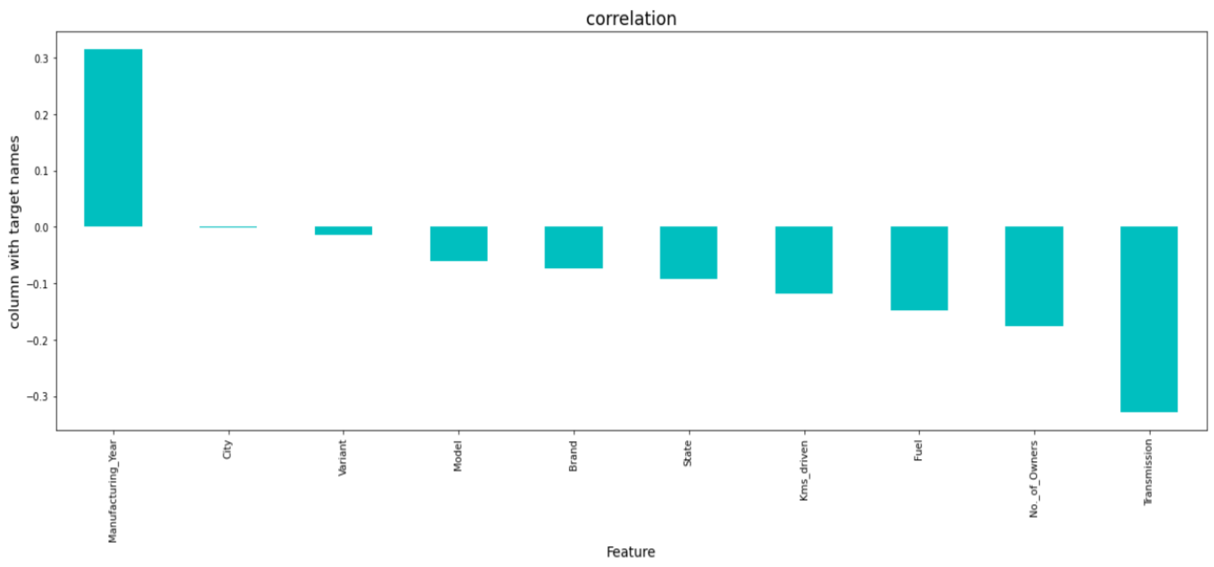
4. Heatmap -



Observations:

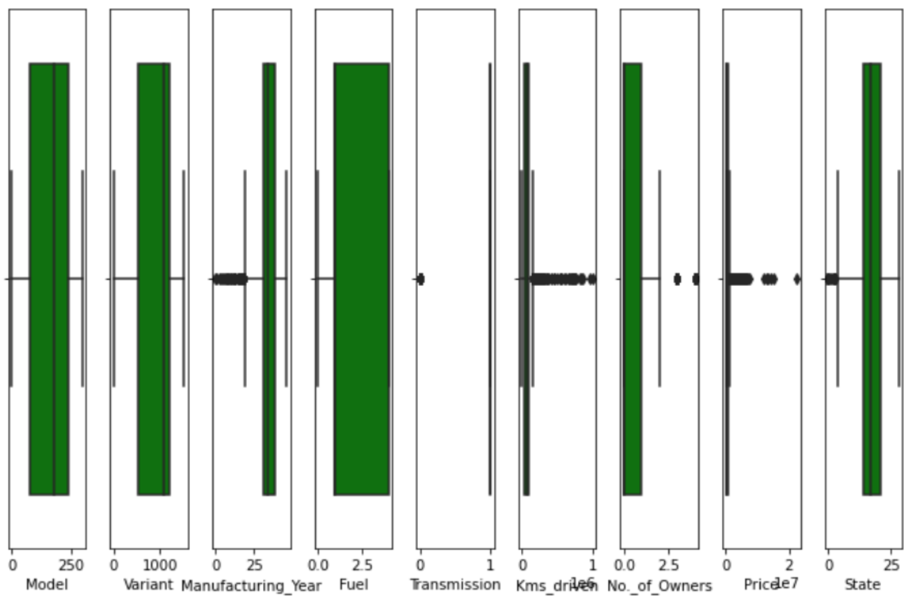
- We don't see any high positive correlation in our data with respect to the target variable.
- The most negatively correlated column is that of the "Transmission".
- The most positively correlated column is the "Manufacturing_Year".

5. Bar Plot -



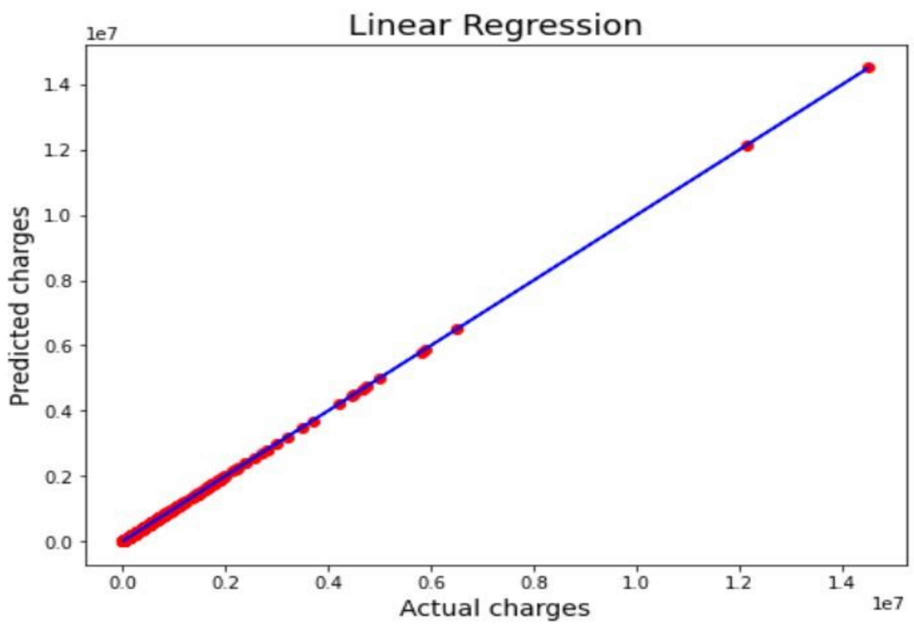
The only variable which is positively correlated to the target variable is "Manufacturing_Year". Rest all are negatively correlated.

6. Box Plot -



Although outliers are seen to be present in data from the above box plot, it is not taken into consideration since all the input features in the data set are of the object data type and “Price” is the target variable. Therefore, there are no outliers present.

7. Scatter Plot -



The Best Fit line is covering almost all the data points which shows a good fit of the model.

Conclusion

- Overcoming difficulties in data collection using web scraping, data cleaning, data analyzing, making comparisons of different Regularization algorithms & Ensemble Techniques and hypertuning one each of them, **Lasso Regularization Regression** concluded to be the best algorithm for the problem. The visualization of Best Fit Line covering all data points depicts that the model is not overfitting or underfitting.

- After using both regularization and ensemble techniques and making comparisons for each, it was observed that "Lasso Regularization Regression" gave the best performance r2 score at 100% accuracy and error rate was proven to be almost zero.

- Data Collection i.e. scraping data from the website took longer than expected for code execution and because of certain issues such as 'low network' or 'website out of reach', the code was required to be executed all over again.

- The limitation of this model is that if the demand in the market changes and if for some reason the prices of the cars are again affected and changed keeping the same features in consideration, the model built in accordance with the current situation cannot be used and made generalized for all times. A new model will have to be built studying the new market conditions and demands. Therefore, a new model will have to be built every time there is a change in the industry demands and conditions.