CAR PRICE PREDICTION PROJECT

Submitted by:

Khanin Deka

ACKNOWLEDGMENT

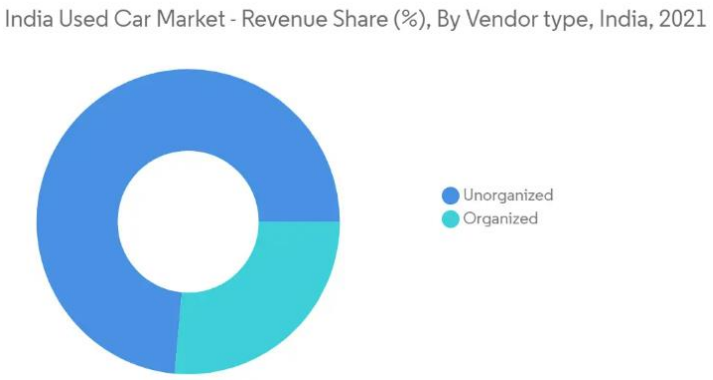
- https://datascience.stackexchange.com/
- https://stackoverflow.com/
- FlipRobo Technologies
- https://scikit-learn.org/
- https://www.cardekho.com/
- https://www.cars24.com/
- https://www.mordorintelligence.com/industry-reports/india-used-car-market
- Irfan Alghani Khalid for his article: https://betterprogramming.pub/faster-web-scraping-in-python-using-multithreading-496da9eaf0c2

INTRODUCTION

- ## Business Problem Framing

  India has a huge market of vehicles. As of 2022, according to the report released by *Organisation Internationale des Constructeurs d'Automobiles* (OICA), India has become the 4th largest vehicle market in the world.

  The Indian used-car market is also growing rapidly. In 2021 the market was valued at USD 32.14 billion. However, the unorganized sector dominates the market, where the scope for consumer protection is on the lower side. Below is a pie-chart of revenue share:



Source: Mordor Intelligence

  There is a huge scope of business for a company willing to invest in the organized sector. To get an idea about the price of any pre-owned car we will attempt to build a machine learning model. This will help to understand the various factors affecting the value depreciation of a car as well as get an approximate idea about its current value.

- ## Conceptual Background of the problem

  Car is a quickly depreciating asset. To get an idea about its current value we have to first analyze how similar cars in the market are doing. Since price is a continuous variable, hence we will use regression algorithms to predict the price based on various features. We have a dataset of 20911 used-cars which we will be using for building our model.

Analytical Problem Framing

- ## Data Sources and their formats

1. The data is collected from www.cardekho.com and www.cars24.com for 11 different cities of India.
2. Most of the data is initially in text format.
3. The following details of a car are collected:
    a) Year on which the car was first purchased.
    b) Brand
    c) Model
    d) Fuel Type
    e) Transmission
    f) Engine Capacity
    g) Kilometers driven.
    h) Number of owners till now.
    i) Location
    j) Price
4. There are some missing values of fuel type, transmission and engine capacity in the dataset.
5. The dataset contains a total of 20911 entries.

- ## Tools and Libraries used

1. This project is built using Jupyter-Notebook of Anaconda Navigator.
2. Python language version 3.9.12 is used.
3. List of libraries include pandas, numpy, sklearn, BeautifulSoup, selenium, requests and xgboost.
4. For data visualization matplotlib and seaborn are used.

- **Data Preprocessing Done**
  - A snapshot of the dataset:

	Unnamed: 0	Year	Brand	Model	Fuel Type	Transmission	Engine Capacity	Kms Driven	No. of Owners	Location	Price
0	0	2020	Jeep	Jeep Compass 1.4 LIMITED PLUS AT	Petrol	AUTOMATIC	NaN	52,301	1st	Pune	1988699.0
1	1	2020	Mercedes-benz	Mercedes-benz C-class Progressive C 200	Petrol	Manual	1950.0	18,000	First	New Delhi	4700000.0
2	2	2018	Hyundai	Hyundai I20 Active 1.2 S	Petrol	Manual	1197.0	53,647	First	Kolkata	590000.0
3	3	2018	Hyundai	Hyundai Creta 1.6 Crdi At Sx Plus	Diesel	Automatic	1582.0	36,000	First	Bangalore	1499000.0
4	4	2012	Maruti	Maruti Wagon R 1.0 VXI	Petrol	MANUAL	NaN	25,674	2nd	Pune	336999.0
...	...	...	...	...	...	...	...	...	...	...	...
20906	20906	2018	Tata	Tata NEXON XZA+ 1.5	Diesel	AUTOMATIC	NaN	13,704	2nd	Pune	1027299.0
20907	20907	2016	Maruti	Maruti Baleno Delta Cvt	Petrol	Manual	1197.0	26,700	First	Gurgaon	675000.0
20908	20908	2009	Toyota	Toyota Corolla Altis G	Petrol	Manual	1798.0	80,000	First	Bangalore	460000.0
20909	20909	2014	Honda	Honda Amaze Ex I-vtech	Petrol	Manual	1198.0	25,000	First	Kolkata	325000.0
20910	20910	2017	Honda	Honda Amaze 1.2 VXMT I VTEC	Petrol	MANUAL	NaN	31,687	2nd	Hyderabad	568899.0

20911 rows × 11 columns

  - *'Unnamed: 0'* column is deleted.
  - 1435 duplicate entries are found and removed.
  - All the values in the 'Transmission' column are mapped to 2 types: A for automatic and M for manual.
  - The entries in 'Kms Driven' column are in text format. They are converted to numeric type.
  - The 'No. of Owners' column contains 9 unique values in text format but they actually refer to 6 different types. The number of owners range from 0 to 5. 0 implies unregistered car and 5 implies 5 owners till now. All these 9 unique values are mapped to 6 values in integer format.
  - Let's look at some characteristics of our continuous features:

	count	mean	std	min	25%	50%	75%	max
Year	19476.0	2.016148e+03	3.438776e+00	1990.0	2014.0	2017.0	2019.0	2022.0
Engine Capacity	14589.0	1.565845e+03	5.631199e+02	0.0	1197.0	1462.0	1956.0	6749.0
Kms Driven	19476.0	5.303756e+04	1.228260e+05	72.0	27000.0	47421.5	70000.0	7400000.0
No. of Owners	19476.0	1.251694e+00	5.157441e-01	0.0	1.0	1.0	1.0	5.0
Price	19476.0	1.034578e+06	1.275901e+06	30000.0	425000.0	645000.0	1093474.0	27100000.0

The minimum value in the 'Engine Capacity' column is 0 which is not possible for a working car. This entry is removed. The maximum value in 'Kms Driven' column is 74lakh. This is not a realistic number. All

entries above 10lakh kms are removed. There were a total of 6 such entries.

- **Data Imputation**

  Let's look at the number of null values column-wise:

```
df.isnull().sum()
```

```
Year                 0
Brand                0
Model                0
Fuel Type          340
Transmission       405
Engine Capacity   4887
Kms Driven           0
No. of Owners        0
Location             0
Price                0
dtype: int64
```

  The null values in the 'Fuel Type' and 'Transmission' columns are filled with mode of the respective column. The null values in 'Engine Capacity' column are filled using KNN (K-nearest neighbor) imputer with the no. of neighbors set to 5. The imputation of 'Engine Capacity' column is done after all the object data-type categorical columns are encoded.

- **Data Encoding**

  'Brand', 'Model', 'Fuel Type', 'Transmission' and 'Location' columns are encoded using *LabelEncoder* of scikit-learn.

- **Data Transformation**

  The skewness of 'Engine Capacity' and 'Kms Driven' column are found to be 1.9 and 4.1 respectively. The data in these columns are transformed using the *PowerTransformer* of scikit-learn. The 'box-cox' power transformation method was used and the skewness is brought down to acceptable level (-0.04 and 0.05 to be precise).

- **Feature Selection**

  Let's look at the absolute values of Pearson's correlation coefficient of various features with the target variable:

```
Price             1.000000
Engine Capacity   0.547056
Year              0.292121
Transmission      0.261942
Fuel Type         0.249993
Kms Driven        0.216244
No. of Owners     0.089949
Location          0.031133
Brand             0.008576
Model             0.003713
Name: Price, dtype: float64
```

  'Brand' and 'Model' columns are dropped as they have a very weak

correlation with the target variable.

- **Feature Scaling**

  The features are scaled using *StandardScaler* of scikit-learn.

- **Checking VIF**

  The variance inflation factor is checked for the features to know about any multicollinearity issues. Below is the screenshot:

	vif	Features
0	1.724030	Year
1	3.316042	Fuel Type
2	3.028016	Transmission
3	1.455945	Engine Capacity
4	1.779712	Kms Driven
5	1.164404	No. of Owners
6	1.007689	Location

  No multicollinearity issues can be seen.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches

Our task is to predict the price of a pre-owned car which is a regression task. For this multiple algorithmic approaches are tried:

1) **Multiple Linear Regression**: In this approach relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data**.**
2) **AdaBoost Regression:** It is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.
3) **Random Forests Regression**: It is an ensemble learning method for regression that operates by constructing a multitude of decision trees at training time. The mean or average prediction of the individual trees is then returned.

4) **XGBoost Regression**: Extreme Gradient Boosting or XGBoost is an ensemble machine learning algorithm which uses an objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e. how far the model results are from the real values.
5) **KNN Regression**: It approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighborhood.

- ## Performance Evaluation

Let's first use the baseline algorithms and evaluate the performance on different train-test splits. Then we do cross-validation of these models for 5-fold to 10-fold to verify the scores.

We will use the coefficient of determination (R^2) to evaluate the models' performance. The R^2 score for a model is a useful statistic in regression analysis, as it often describes how 'good' that model is at making predictions.

The values for R^2 range from 0 to 1. The closer the score is to 1 better is the model. A model can give a negative R^2 score as well, which indicates that the model is arbitrarily worse than the one that always predicts the mean of the target variable.

Let's see the R^2 scores of the 5 different algorithms on 5 different train-test splits:

	Linear Regression	Adaboost Regression	Random-Forests Regression	XGBoost Regression	K-Neighbors Regression
Train-Test Split 1	0.41	0.53	0.90	0.88	0.72
Train-Test Split 2	0.39	0.55	0.89	0.90	0.67
Train-Test Split 3	0.42	0.52	0.91	0.91	0.67
Train-Test Split 4	0.44	0.52	0.90	0.90	0.69
Train-Test Split 5	0.42	0.55	0.91	0.92	0.66

Random-Forests and XGBoost are giving the best performance among all the algorithms.

Let’s see the cross-validation results:

	Linear Regression	Adaboost Regression	Random-Forests Regression	XGBoost Regression	K-Neighbors Regression
5-Fold Cross-Validation	0.42	0.54	0.90	0.91	0.68
6-Fold Cross-Validation	0.42	0.52	0.90	0.91	0.69
7-Fold Cross-Validation	0.42	0.52	0.90	0.90	0.69
8-Fold Cross-Validation	0.42	0.52	0.90	0.91	0.70
9-Fold Cross-Validation	0.42	0.55	0.91	0.91	0.69
10-Fold Cross-Validation	0.42	0.56	0.90	0.90	0.69

From the cross validation results it can be verified that Random-Forests and XGBoost are the best performing algorithms for our dataset.
Let’s tune their hyperparameters using GridSearchCV of scikit-learn and try to improve the scores.

Hyperparameter Tuning of Random-Forests Regressor:

```
# Hyperparameter tuning of Random-Forests model.
from sklearn.model_selection import GridSearchCV

params= {"n_estimators": [100,250,500,1000],
         "max_depth": [15,17,21,23,None],
         "max_features": [0.5,0.75,1.0],
         "max_samples": [0.5,0.75,1.0]
         }

grd= GridSearchCV(rf, param_grid=params,cv=5,verbose=2,n_jobs=12)
grd.fit(x_train,y_train)
print("Best Parameters:",grd.best_params_)
```

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
Best Parameters: {'max_depth': None, 'max_features': 1.0, 'max_samples': 1.0, 'n_estimators': 100}
```

Cross Validation results:

```
# cross-validating the tuned random-forests regression model.
for i in range(5,11):
    cv_score= cross_val_score(rf_tuned,X_scaled,y,cv=i,n_jobs=12).mean()
    print("the cv score for",i,"fold:", round(cv_score,2))
```

```
the cv score for 5 fold: 0.9
the cv score for 6 fold: 0.9
the cv score for 7 fold: 0.9
the cv score for 8 fold: 0.9
the cv score for 9 fold: 0.91
the cv score for 10 fold: 0.9
```

Hyperparameter Tuning of XGBoost Regressor:

```
# Hyperparameter tuning of XGBoost model.
params= {"n_estimators": [100,250,500,1000],
         "max_depth": [None,5,6,7,10],
         "learning_rate": [0.1,0.3,0.5,1,None]
         }

grd= GridSearchCV(xgb_reg, param_grid=params,cv=5,verbose=2,n_jobs=12)
grd.fit(x_train,y_train)
print("Best Parameters:",grd.best_params_)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Parameters: {'learning_rate': 0.1, 'max_depth': None, 'n_estimators': 1000}
```

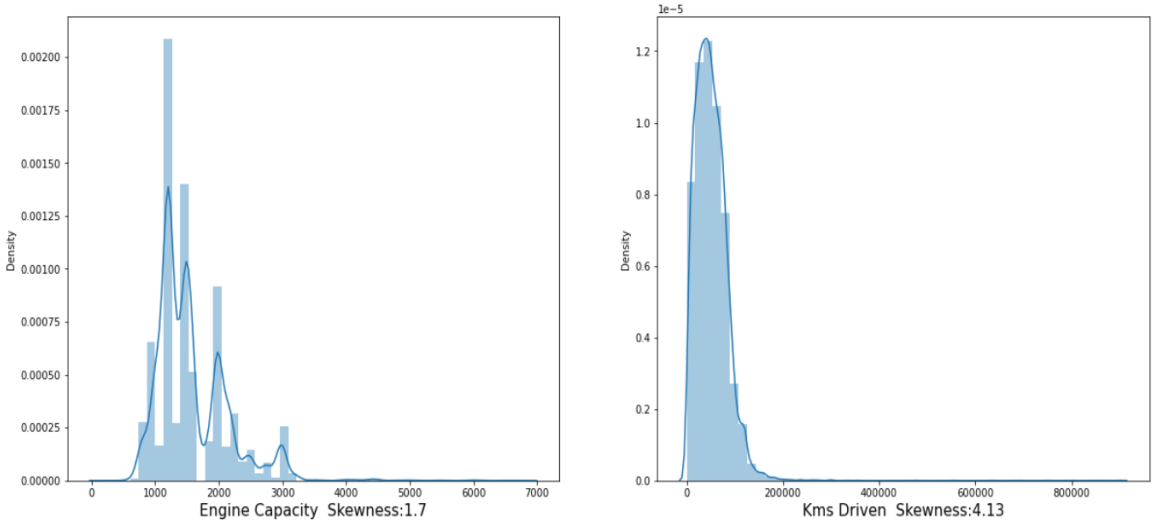
Cross Validation Results:

```
# cross-validating the tuned xgboost regression model.
for i in range(5,11):
    cv_score= cross_val_score(xgb_tuned,X_scaled,y,cv=i,n_jobs=12).mean()
    print("the cv score for",i,"fold:", round(cv_score,2))
```

```
the cv score for 5 fold: 0.91
the cv score for 6 fold: 0.92
the cv score for 7 fold: 0.92
the cv score for 8 fold: 0.92
the cv score for 9 fold: 0.92
the cv score for 10 fold: 0.92
```

We can see that the XGBoost model is giving slightly better results as compared to Random-Forests model.

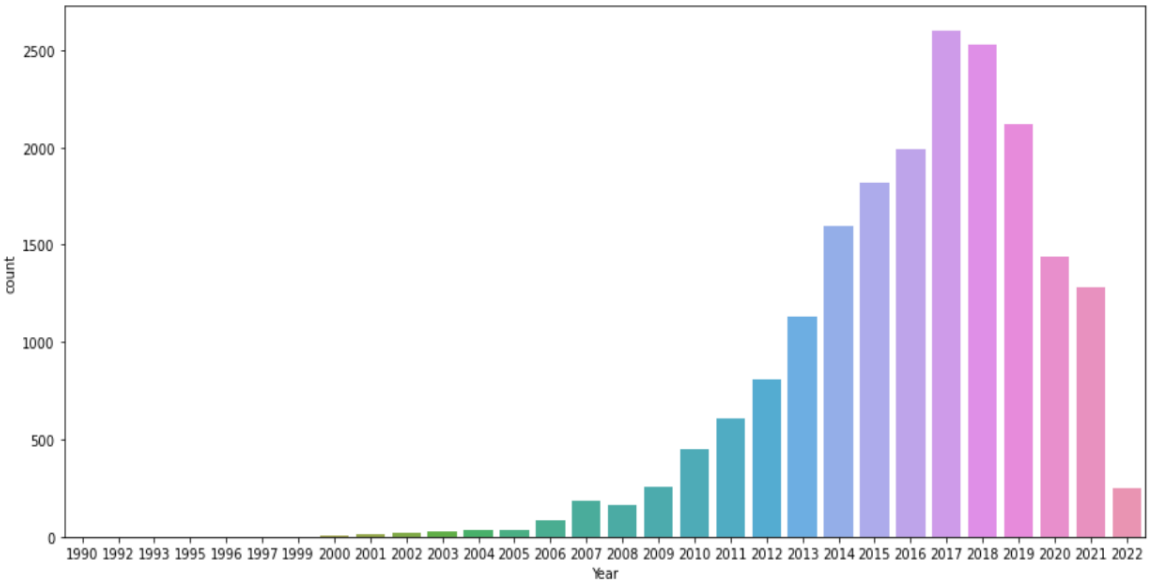
- **Visualizations**
  - ➢ Distribution plots of 'Engine Capacity' and 'Kms Driven' column:



Observations:

1. The engine capacity of most of the cars range between 800 and 3500cc.
2. Most of the cars have not been driven beyond 2lakh kilometers.

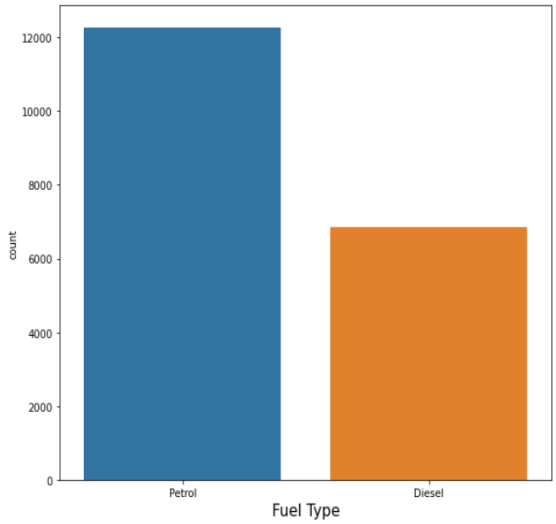
  - ➢ Count plot of 'Year' column:

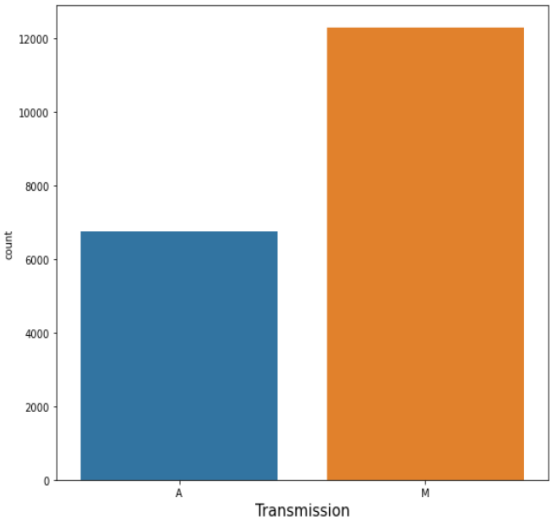


Observations:

1. Majority of the cars are not older than 10 years since first brought.

- Count plots of ‘Fuel Type’ and ‘Transmission’ columns:

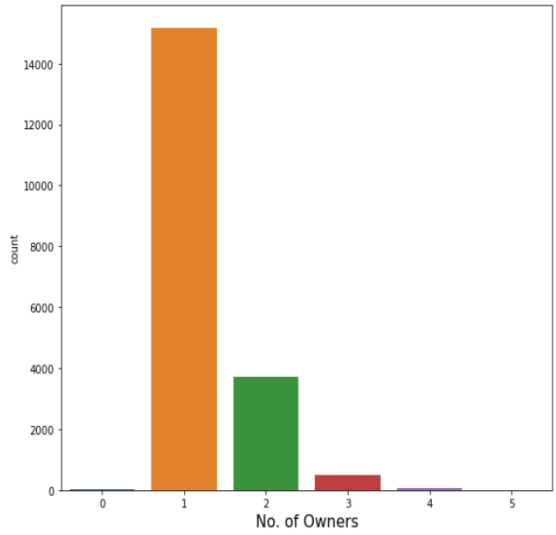


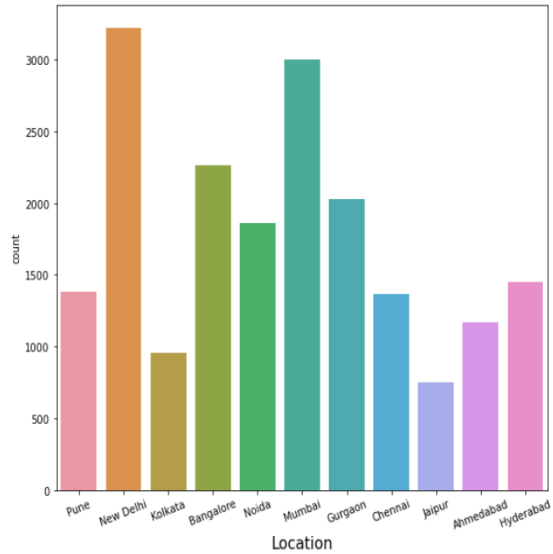


Observations:

1. Majority of the cars in the dataset uses petrol as fuel.
2. Majority of the cars in the dataset have manual transmission.

- Count plots of ‘No. of Owners’ and ‘Location’ columns:

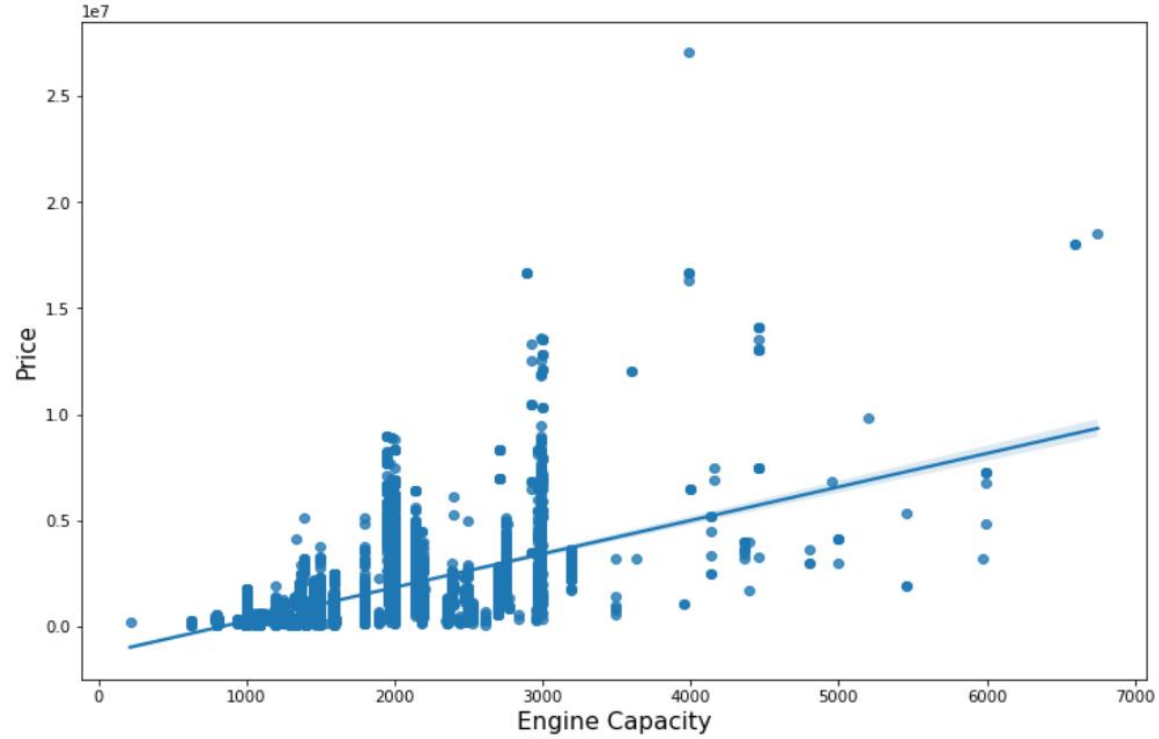




Observations:

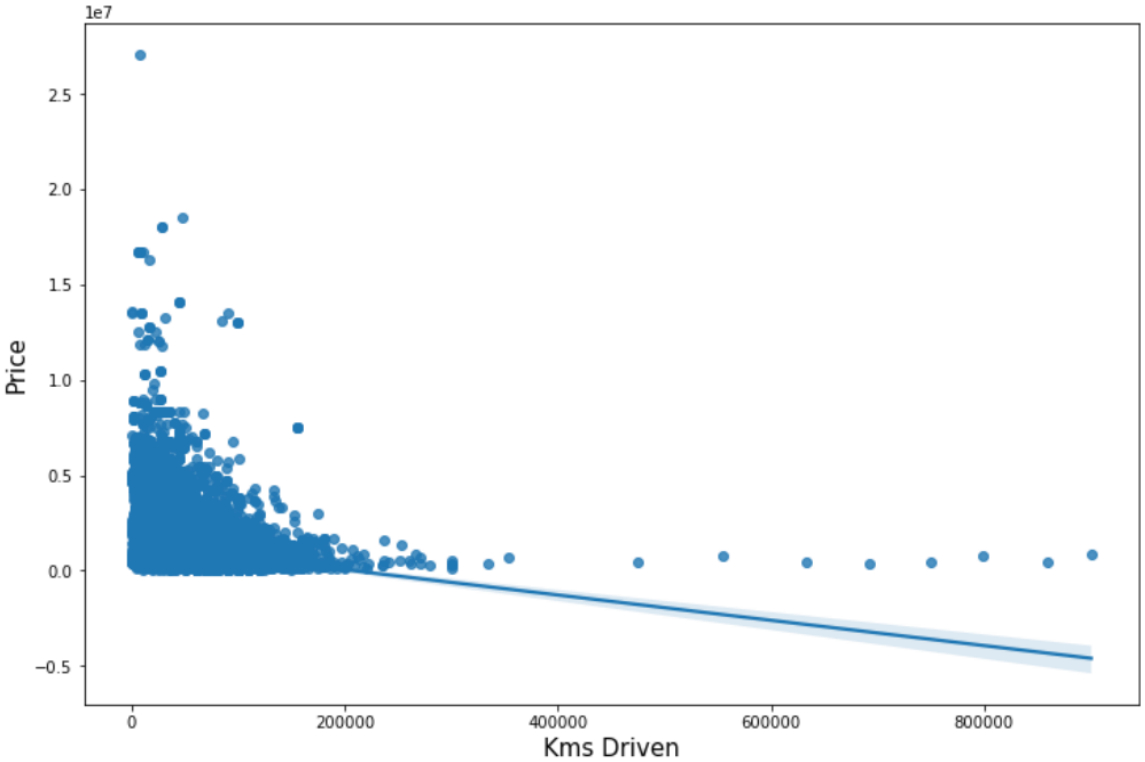
1. Majority of the pre-owned cars have 1 owner till now.

- Regression plot of ‘Engine Capacity’ vs ‘Price’:



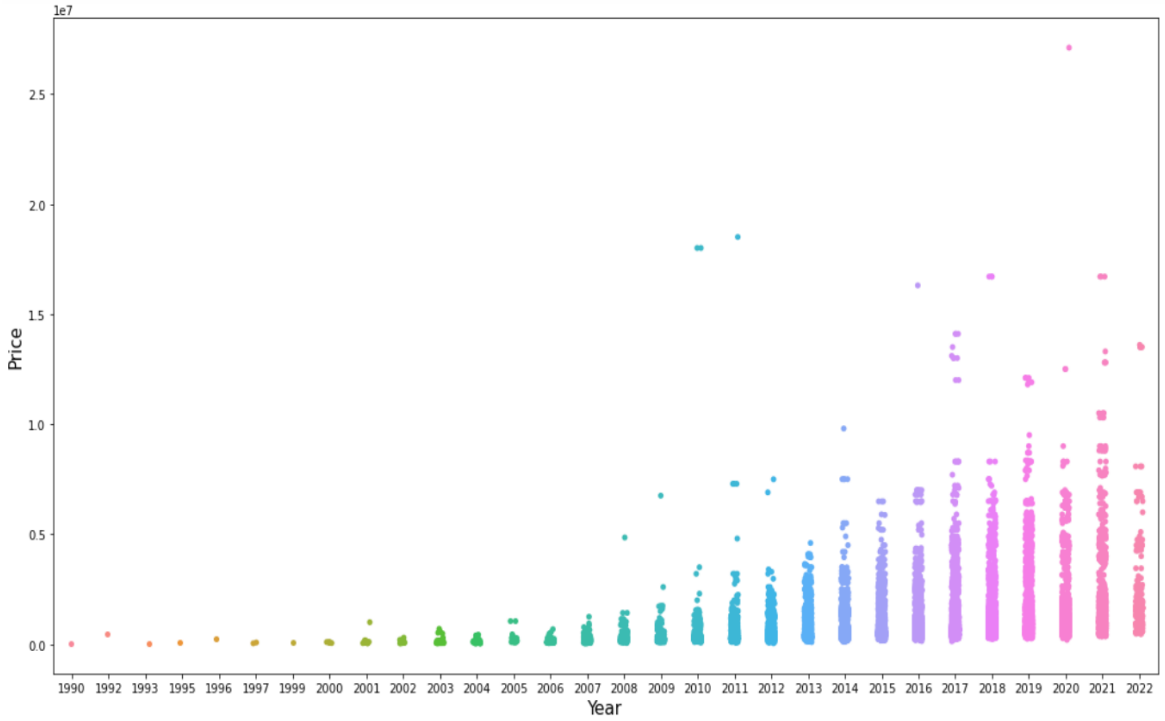
Observation: A positive linear relation can be seen.

- Regression plot of ‘Kms Driven’ vs ‘Price’:



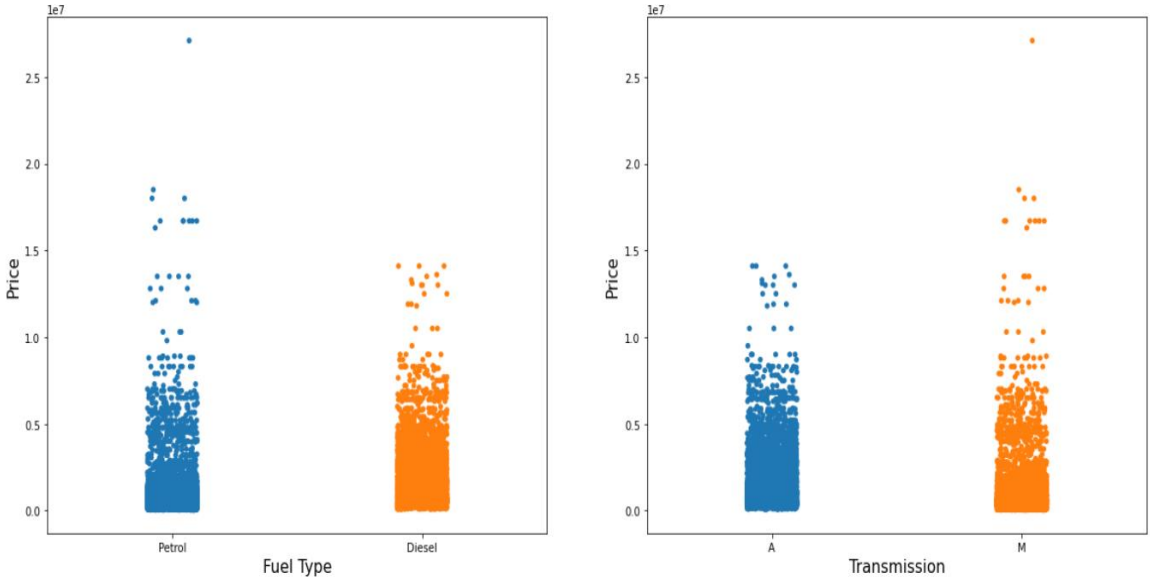
Observation: A negative linear relation can be seen. Cars which have travelled higher distance have lower price in general.

➢ Strip plot of ‘Year’ vs ‘Price’:

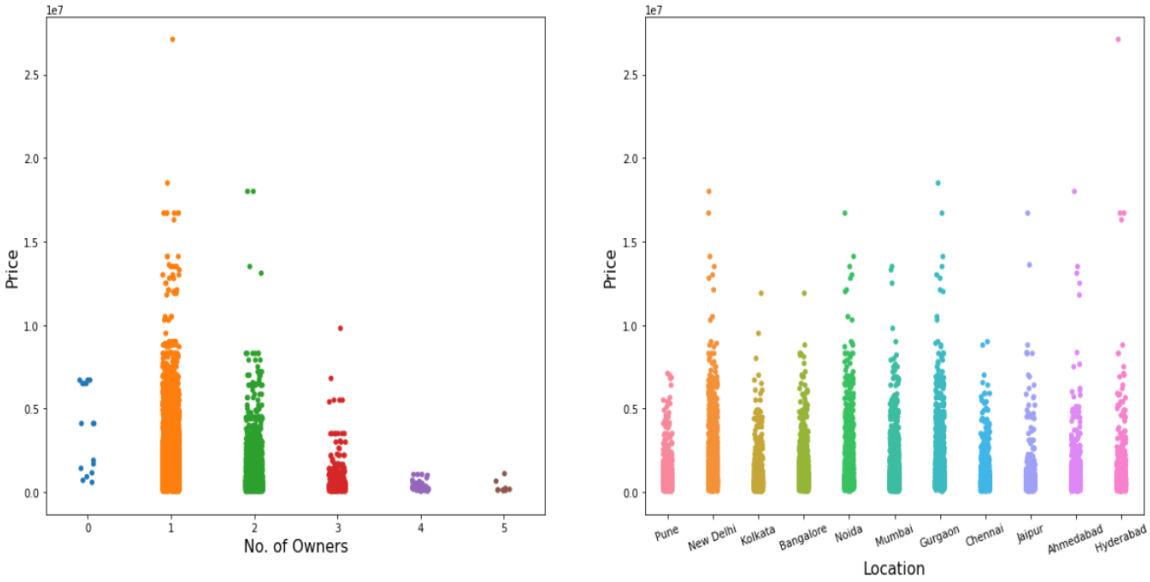


Observation: Older cars have lower value.

➢ Strip plots of ‘Fuel Type’ and ‘Transmission’ vs ‘Price’:

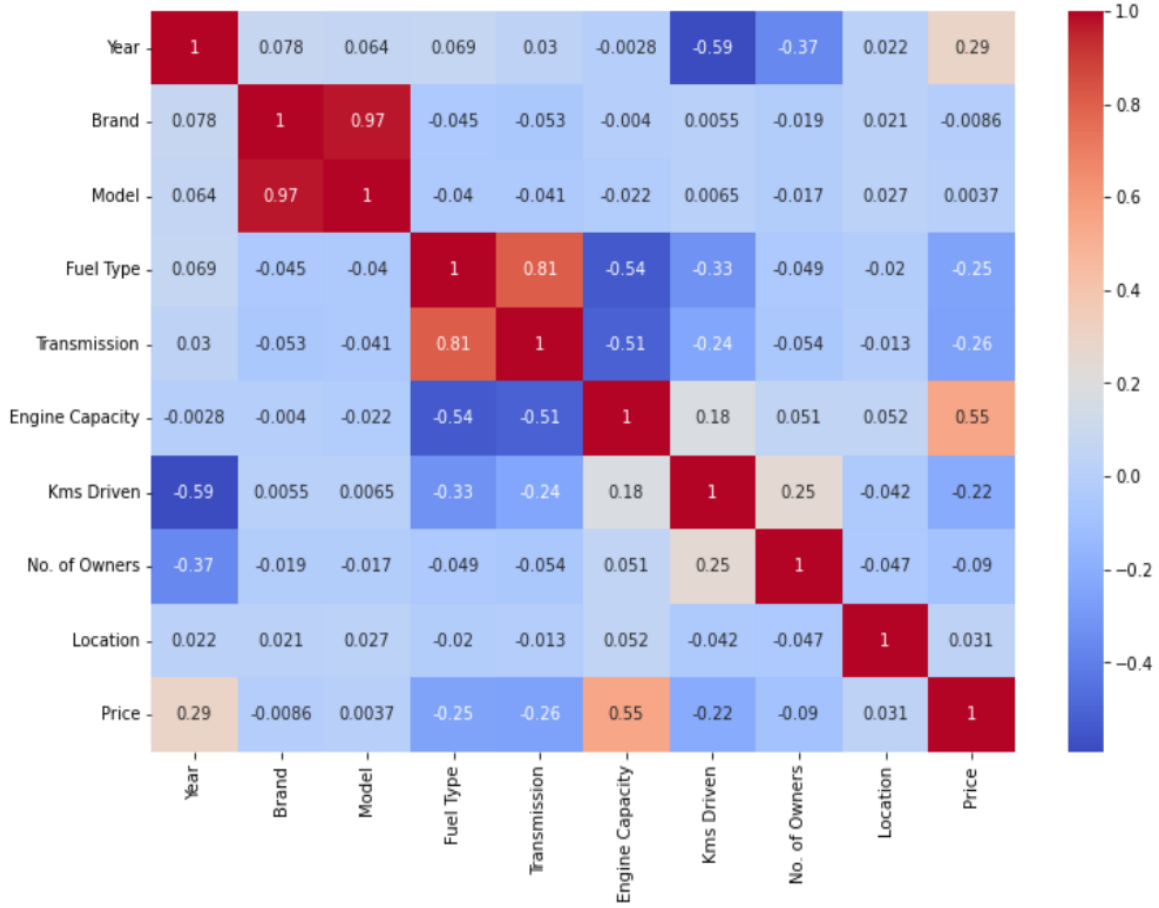


➢ Strip plots of ‘No. of Owners’ and ‘Location’ vs ‘Price’:



Observation: As the number of owners of the car increase the price gets reduced.

➢ Correlation Heat-map:



Observations:

1. ‘Engine Capacity’ has the strongest correlation with price of the vehicle.
2. The ‘Brand’ and ‘Model’ seems to have a very weak relation with price of the vehicle.
3. ‘Brand’ and ‘Model’ have a multicollinearity issue.

CONCLUSION

- ## Key Findings and Conclusions of the Study

1) The list of features affecting the price most, in descending order are:
   a. Engine capacity
   b. Year when the car was first purchased
   c. Transmission type
   d. Fuel type
   e. Distance travelled by the car till now.
   f. Number of owners till now.
2) XGBoost Regression Algorithm is giving the best results for prediction of price.

- ## Limitations of this work and Scope for Future Work

1) The hyper-parameters of our final model can be further tuned.
2) In data collection phase we collected data of 9 features and the price. Some more feature data can also be collected. For example, whether the car has been in any accident, seating capacity, etc.
3) In this project we worked on only 5 algorithms. There are numerous other regression algorithms with which we can try to make a better model.