CAR PRICE PREDICTION

Submitted by:

KONATALA MOHIT

ACKNOWLEDGMENT

I am grateful for the writer, developers and authority of the website pages machinelearningmastry, geekforgeeks, wikipedia and stackoverflow which have helped me to refer for whenever there was need for guidance. Maintenance Engineering and Engineering Economics helped me to get a better understanding of the data to analyse and explore data. I am mostly grateful to DataTrained who have guided me in learning and enhancing my data science skills required to analyse and solve this project.

INTRODUCTION:

In this report I will be discussing about used car price prediction using few of the machine learning models via python and its libraries. Transportation is one greatest invention of man-kind which truly helped connecting the world resources around the world. The invention of a true four-wheel automobile was done by Carl Benz on January 29th 1886. Over the period of 125 year many modifications and improvement have been made to an automobile so that any common person could use an automobile. Automobiles come under the category of perishable goods i.e., the goods that depreciate as they age and automobiles have a high depreciation rate of all the perishable goods due to which a newly purchased automobile for few people becomes a liability. To counter-act the liability issues most people prefer used car.

Used car delivers all necessities required at a much cheaper price and slow depreciating rate but exaggerating the price of a used car is not an easy task as there are factors which influence the pricing of a used car. To predict the pricing must have a good understanding of car maintenance and engineering economics. Maintenance is one of the core parts of engineering which helps in optimizing the product so that its life span and performance be sustained for much longer period. Engineering Economics helps determining strategic measures to be more cost effective and achieve financial competency and also to counter-act depreciation.

Review of Literature:

For the better understanding of the data referred to automobile maintenance & engineering economics. In buying a used car one must have good understanding about automobile parts life and type of technology used by car parts to estimate depreciation value and estimate the final value which can be done by using both maintenance and economics engineering. Maintenance engineering also helps in inspection of the vehicle before purchase to see if the automobile is in the most optimal condition for the given price.

Undertaken Problem:

Objective of the project is to predict the used cars price by analysing various factors that affect the car price outcome. Car price prediction is one the many problems in Data Science as it ticks all the basic fundamentals required to analyse data and creating a machine learning model to predict the necessary outcome.

Mathematical/ Analytical Modelling of the Problem:

Price is the target data in the dataset and the variables present the target data are in continuous form, therefore Regression Machine Learning Models are used in this project. Regression is for estimating continuous form of data by using established

relation between feature and target variables. Descriptive Analysis is used to study and observe the data.

Data Sources and their formats:

The source of the data is collected from cardekho website using selenium webdriver. The data is provided in the excel format. Data contains 10047 entries having 8 variables in the dataset.

Data Pre-processing:

Data pre-processing has two main steps i.e., Data Cleaning and Data Transforming.

Data Cleaning:

Data Cleaning is one of the most important steps creating a machine model. If an uncleaned data is fed to a machine learning model, then the model will perform very poorly.

The first and foremost step in data cleaning is removing and replacing null values in the dataset though our dataset doesn't contain null values. The car name column contains three variations in it i.e., year in which the car model was released, car company and the primary model of the car. To decrease the complexity of the model we are going to split the column into three separate columns and delete the existing car name column. Removing km units for the variables in the Km driven columns as they are not going help in the prediction of our outcome and stops the data from being a numerical quantity. Km driven and price contains commas in-between the digits therefore removing them and converting both columns into numerical datatypes.

Luxury car types are short in count and price of them coincides with that of the sedan so merging luxury car with the sedan type. Same goes for minivan as they are less in numbers and as they have same specification as a MUV merging them with MUV is a proper case to deal with it. Year below 2008 are very less in number therefore clubbing them with 2008 model as they all are considered as old models. And whereas it goes for 2022 model clubbing them with 2021 models. LPG are very minute in number therefore merge them with CNG fuel types. But still as the number for CNG are quite less compared to petrol and diesel therefore dropping the rows containing the CNG as their fuel type. In car company column company that are as followed: Bentley, Isuzu, ICML, Maserati, Ambassador, Premier and Force are quite negligible and are to considered as outlier which can affect the outcome values therefore dropping these values.

After removing few the variables counts the size of the data is (9871,10). Mini car company has less values but it is child organization of BMW car company therefore merging it with BMW.

Four of the columns have categorical data greater than 2 but less than 10 unique categories those columns are Car type, Location, year model, car company. To decrease the complexity of the data we use get dummies method to split the categorical data of those variables into each separate columns to make the data much simpler. After creating dummies, we have to merge those dummy columns with the present and remove those columns from which the dummy columns were created also remove one of the dummies created via the same column to remove multi collinearity created by the dummies. Removing Model column as it has too many categories and can harm the outcome value while training the Machine Learning model

Data Transforming:

After Data Cleaning the data must be transformed into numerical form as one can't feed ordinal data to machine learning model and also the data must be normalized as normalizing the data the machine learning model gives equal importance to all the data.

In this project I am using label encoder to transform the ordinal data present in the data set into numerical form. Using Standard Scalar function to normalize the data.

Note: Before normalize the data separated the target variable from feature variables.

Using VIF to check that value are under 5 as having higher number indicates that the dataset the multi collinearity between the independent variables which must be arrested else we cannot achieve optimal machine learning model.

After standardizing the data, we must split the data into train and test sets. Train Test Split method is to split the data into train and test data

Data Inputs- Logic- Output Relationships:

After Separating the Input (feature) and output (target) data we split them into train and test division one part of the data is used to train the ML model and other part of the data to predict the output. When the train data is fed to machine learning model it generates an algorithm or simply put an equation that is applicable to all the data and when test data is fed to it implements the trained data equation to the current input values to predict the outcome. If the input has no outliers and is clean that there is no over or underfitting in the outcome.

Hardware and Software Requirements and Tools Used:

Hardware Required for Jupyter Notebook Software is as follows:

Memory and disk space required per user: 1GB RAM + 1GB of disk + .5 CPU core.

Server overhead: 2-4GB or 10% system overhead (whatever is larger), .5 CPU cores, 10GB disk space.

Port requirements: Port 8000 plus 5 unique, random ports per notebook

Libraries Used:

Pandas library: To frame raw data, visualize and perform task on it via other libraries.

Numpy library: To perform mathematical functions on the framed data numpy is used. In this project used it to location nan values and replace them with desired value also to find mean and standard values.

Matplotlib library: This library is used to visualize the data. Used to visualize univariate and bivariate analysis (pie plot, count plot and scatter plot) also to visualize outliers via box plot.

Seaborn library: heatmap to see co-relation between feature variable to arrest high collinearity.

Sklearn library: Imported this library to normalize the data, split the data into train and test data, various machine learning model and cross validation techniques.

Warnings library: To ignore filter warning shown while compiling block of codes

Pickle: To save the trained machine learning model.

Algorithm Used:

1. Linear Regression
2. K-Neighbors Regressor
3. XGB Regression
4. Random Forest Regressor
5. Gradient Boosting Regressor
6. Lasso Regression

Model Training and Selection:

Linear Regression:

linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable

```
model1=LinearRegression()
model1.fit(X_train,Y_train)
scores1 = cross_val_score(model1, X_test, Y_test, scoring='r2', cv=5)
print('Mean R2 Score for Linear Regression :',mean(scores1),'\nStandard Deviation is : ',std(scores1))
```

```
Mean R2 Score for Linear Regression : 0.844239967145511
Standard Deviation is :  0.013781218647555283
```

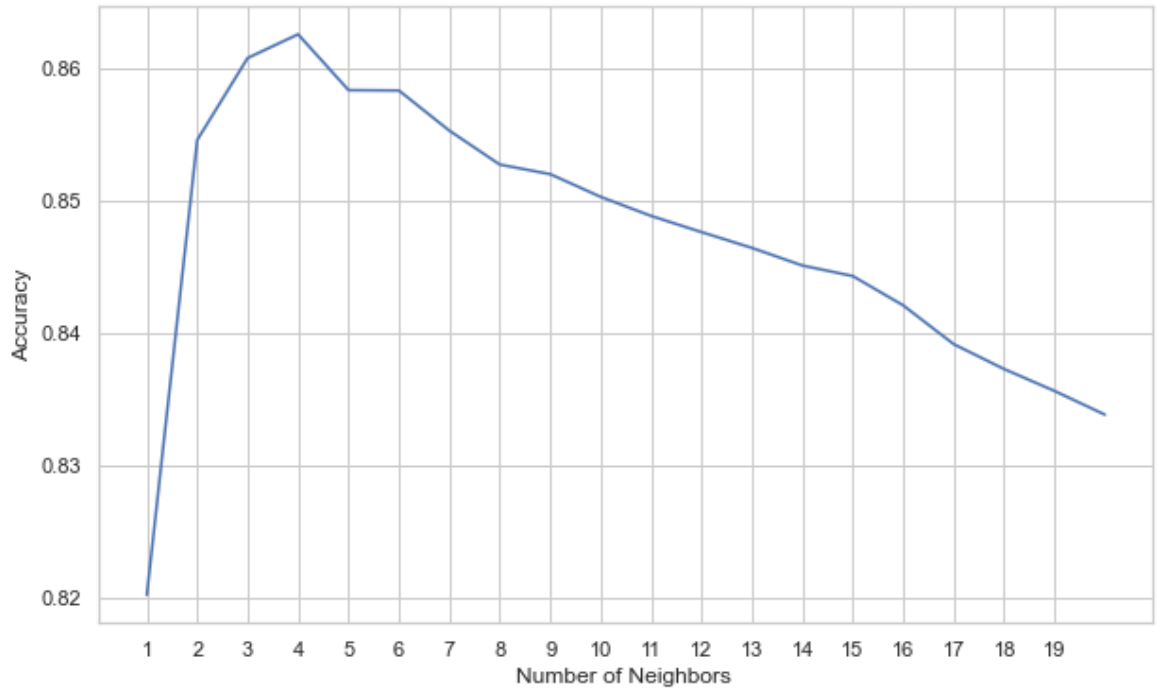
Mean R2 Score for Linear Regression: 0.844239967145511
Standard Deviation is: 0.013781218647555283

K-Neighbors Regressor:

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood.

```
knn_p=KNeighborsRegressor()
mean_acc = np.zeros(20)
for i in range(1,21):
    #Train Model and Predict
    knn = KNeighborsRegressor(n_neighbors = i).fit(X_train,Y_train)
    yhat2= knn.predict(X_test)
    mean_acc[i-1] = metrics.r2_score(Y_test, yhat2)

loc = np.arange(1,20,step=1.0)
plt.figure(figsize = (10, 6))
plt.plot(range(1,21), mean_acc)
plt.xticks(loc)
plt.xlabel('Number of Neighbors ')
plt.ylabel('Accuracy')
plt.show()
```



```
knn=KNeighborsRegressor()
para={
        'n_neighbors':[3,4,5,6],
        'weights':['uniform', 'distance'],
        'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute','auto'],
        'leaf_size':[30,40,50],
        'p':[2],
        'metric':['minkowski']

}
knn_gs= gs(estimator =knn, param_grid=para,cv=5, n_jobs=5)
knn_gs.fit(X_train,Y_train)
```

```
knn_gs.best_score_
```

```
0.8869452076042702
```

```
knn_gs.best_estimator_
```

```
KNeighborsRegressor(n_neighbors=6, weights='distance')
```

```
model2=KNeighborsRegressor(n_neighbors=6, weights='distance')
model2.fit(X_train,Y_train)
```

```
KNeighborsRegressor(n_neighbors=6, weights='distance')
```

```
scores2 = cross_val_score(model2, X_test, Y_test, scoring='r2', cv=5)
print('Mean R2 Score for KNeighbors Regression :',mean(scores2),
      '\nStandard Deviation is : ',std(scores2))
```

```
Mean R2 Score for KNeighbors Regression : 0.7955949647781784
Standard Deviation is :  0.019032900400281556
```

Mean R2 Score for KNN Regression: 0.7955949647781784
Standard Deviation is: 0.019032900400281556

XGB Regressor:

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance

```
xgb=XGBRegressor()
param_grid = {
    'n_estimators': [200,250,300],
    'max_depth': [51],
    'learning_rate':[0.15,0.2],
    'gamma':[0.0,0.1,0.2,0.3],
    'n_jobs':[100,200,300]
}
CV_xgb = gs(xgb,param_grid=param_grid,scoring ='r2',cv=5,verbose=5)
CV_xgb.fit(X_train,Y_train)
```

```
gbr=GradientBoostingRegressor()
parameter_gbr_={
         'n_estimators': [200,250,300,350,400],
         'max_depth': [51],
         'min_samples_split': [2,4,5,6],
         "min_samples_leaf":[1,2,3,4,5],
     'learning_rate':[0.15,0.2],
}
gbr_gs= gs(estimator =gbr, param_grid=parameter_gbr_,cv=5, n_jobs=5)
gbr_gs.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=5,
             param_grid={'learning_rate': [0.15, 0.2], 'max_depth': [51],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [2, 4, 5, 6],
                         'n_estimators': [200, 250, 300, 350, 400]})
```

```
CV_xgb.best_score_
```

```
0.9192629686950908
```

```
CV_xgb.best_estimator_
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.1, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=51, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
model3=XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.1, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=51, min_child_weight=1,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
model3.fit(X_train,Y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.1, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=51, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
scores3 = cross_val_score(model3, X_test, Y_test, scoring='r2', cv=10)
print('Mean R2 Score for XGB Regression :',mean(scores3),
      '\nStandard Deviation is : ',std(scores3))
```

```
Mean R2 Score for XGB Regression : 0.8140135957148293
Standard Deviation is :  0.03825808917601883
```

Mean R2 Score for XGB Regression: 0.8140135957148293
Standard Deviation is: 0.03825808917601883

Random Forest Regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting

```
rfr= RandomForestRegressor()
parameter_rfr={
         'n_estimators': [200,250,300],
         'max_features': ['auto', 'sqrt'],
         'max_depth': [51],
         'min_samples_split': [2, 3, 4],
         'min_samples_leaf': [1, 2,3,4],
         'bootstrap': [True, False]

}
rfr_gs= gs(estimator =rfr, param_grid=parameter_rfr,cv=5,n_jobs=5,verbose=5)
rfr_gs.fit(X_train,Y_train)
```

```
rfr_gs.best_score_
```

```
0.9225683184432867
```

```
rfr_gs.best_estimator_
```

```
RandomForestRegressor(bootstrap=False, max_depth=51, max_features='sqrt',
                      n_estimators=300)
```

```
model4=RandomForestRegressor(bootstrap=False, max_depth=51,
                             max_features='sqrt', n_estimators=300)
model4.fit(X_train,Y_train)
```

```
RandomForestRegressor(bootstrap=False, max_depth=51, max_features='sqrt',
                      n_estimators=300)
```

```
scores4 = cross_val_score(model4, X_test, Y_test, scoring='r2', cv=10)
print('Mean R2 Score for Random Forest Regression :',mean(scores4)
      ,'\nStandard Deviation is : ',std(scores4))
```

```
Mean R2 Score for Random Forest Regression : 0.8517433797226925
Standard Deviation is :  0.030635818273004264
```

Mean R2 Score for Random Forest Regression: 0.8517433797226925
Standard Deviation is: 0.030635818273004264

Gradient Boosting Regressor:

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

```
gbr=GradientBoostingRegressor()
parameter_gbr_={
         'n_estimators': [200,250,300,350,400],
         'max_depth': [51],
         'min_samples_split': [2,4,5,6],
         "min_samples_leaf":[1,2,3,4,5],
     'learning_rate':[0.15,0.2],
}
gbr_gs= gs(estimator =gbr, param_grid=parameter_gbr_,cv=5, n_jobs=5)
gbr_gs.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=5,
             param_grid={'learning_rate': [0.15, 0.2], 'max_depth': [51],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [2, 4, 5, 6],
                         'n_estimators': [200, 250, 300, 350, 400]})
```

```
gbr_gs.best_score_
```

```
0.9067247304324221
```

```
gbr_gs.best_estimator_
```

```
GradientBoostingRegressor(learning_rate=0.15, max_depth=51, min_samples_leaf=5,
                          min_samples_split=4, n_estimators=200)
```

```
model5=GradientBoostingRegressor(learning_rate=0.15, max_depth=51, min_samples_leaf=5,
                         min_samples_split=4, n_estimators=200)
model5.fit(X_train,Y_train)
```

```
GradientBoostingRegressor(learning_rate=0.15, max_depth=51, min_samples_leaf=5,
                          min_samples_split=4, n_estimators=200)
```

```
scores5 = cross_val_score(model5, X_test, Y_test, scoring='r2', cv=10)
print('Mean R2 Score for GradientBoosting Regression :',mean(scores5),
      '\nStandard Deviation is : ',std(scores5))
```

```
Mean R2 Score for GradientBoosting Regression : 0.8239103840044283
Standard Deviation is :  0.047654193002608015
```

Mean R2 Score for Gradient Boosting Regression: 0.8239103840044283
Standard Deviation is: 0.047654193002608015

Lasso Regression:

Lasso regression is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
lasso_alphas = np.linspace(0, 0.2, 21)
lasso = Lasso()
grid = dict()
grid['alpha'] = lasso_alphas
gscv = gs( \
    lasso, grid, scoring='r2', \
    cv=cv, n_jobs=-1)
results = gscv.fit(X_train, Y_train)
print('R2: %.5f' % results.best_score_)
print('Config: %s' % results.best_params_)
```

```
R2: 0.85340
Config: {'alpha': 0.2}
```

```
model6=Lasso(alpha=0.2, max_iter=250, tol=0.1)
model6.fit(X_train,Y_train)
```

```
Lasso(alpha=0.2, max_iter=250, tol=0.1)
```

```
scores6 = cross_val_score(model6, X_test, Y_test, scoring='r2', cv=10)
print('Mean R2 Score for Lasso Regression :',mean(scores6),
      '\nStandard Deviation is : ',std(scores6))
```

```
Mean R2 Score for Lasso Regression : 0.8246470127449568
Standard Deviation is :  0.025501909108410624
```

Mean R2 Score for Lasso Regression: 0.8246470127449568
Standard Deviation is: 0.025501909108410624

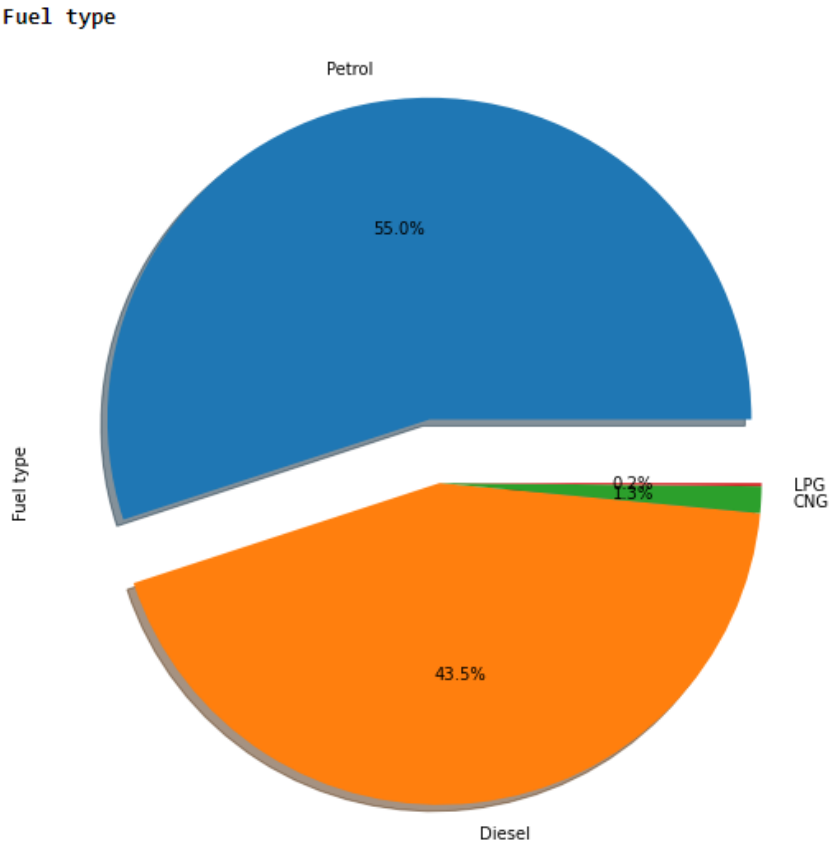
Metrics Used in the project:

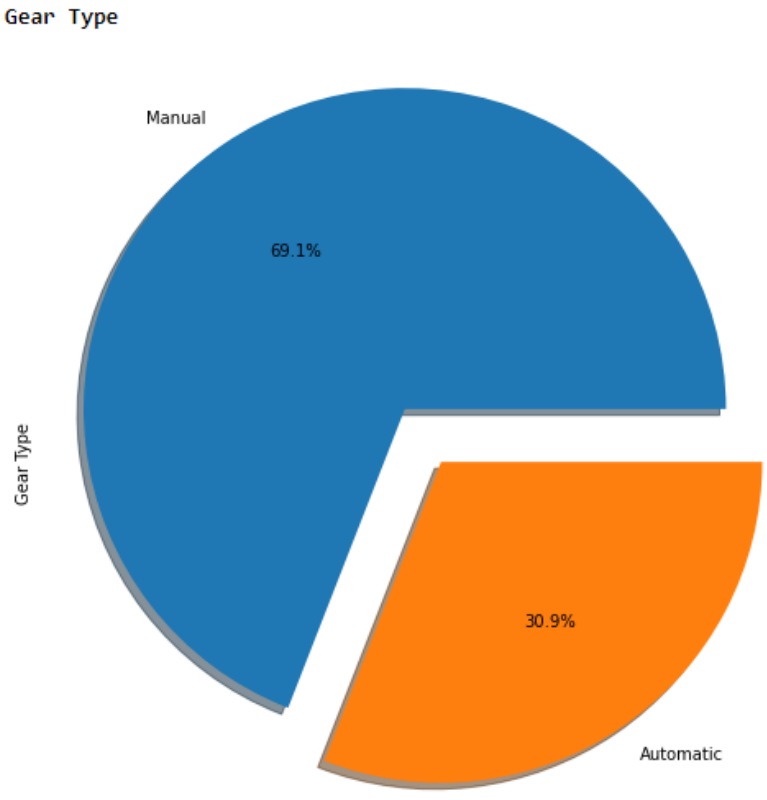
cross_val_score: Used cross_val_score to evaluate the model and observe (r2 value) it perform for particular number of folds to determine which model performs better

mean: To get the mean value of cross validation score and determine the model accuracy.

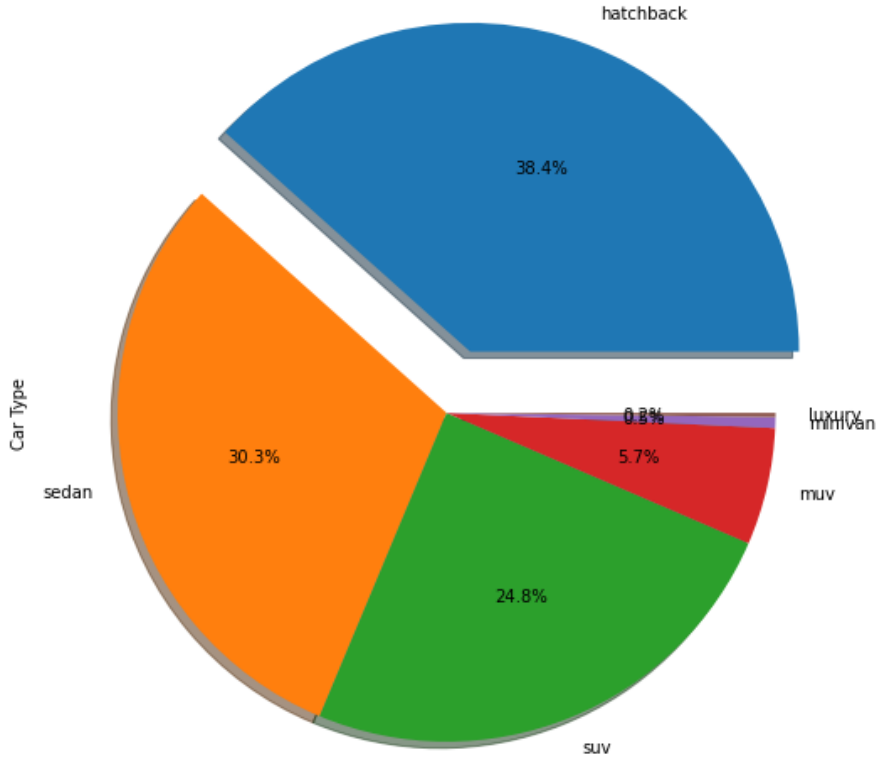
Standard deviation: To determine the average deviation for all the fold observed in cross validation scores

Visualization:

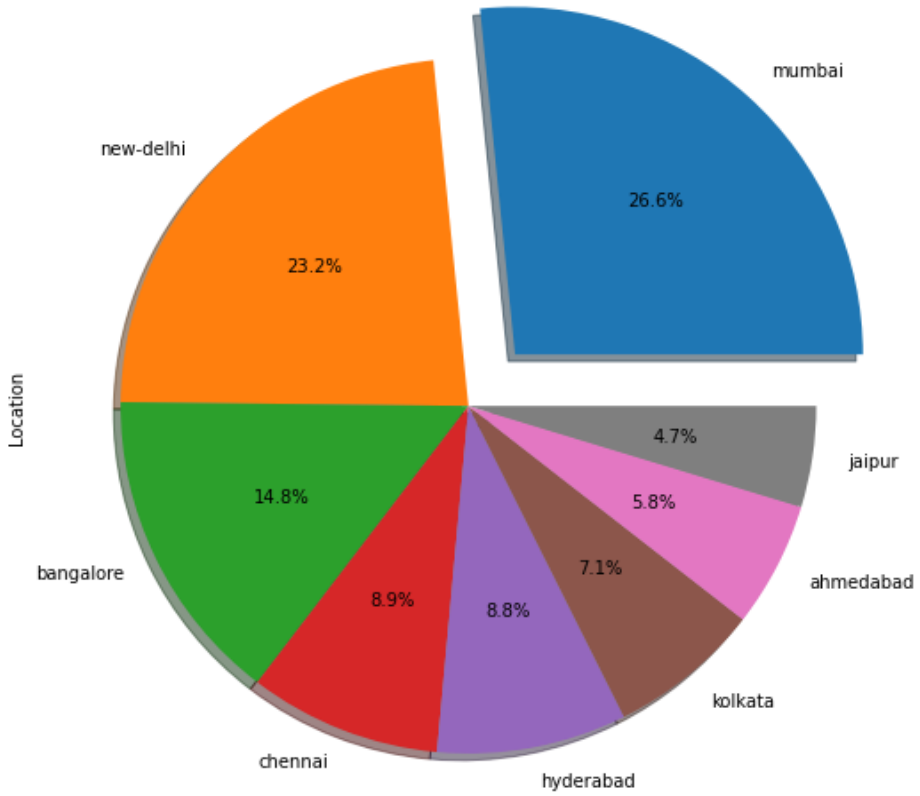




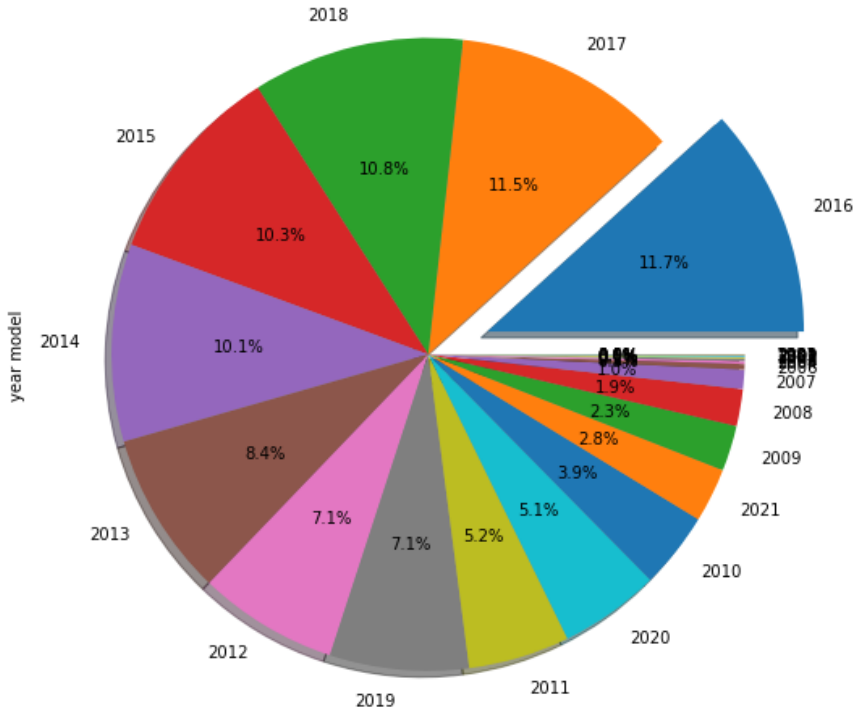
Car Type



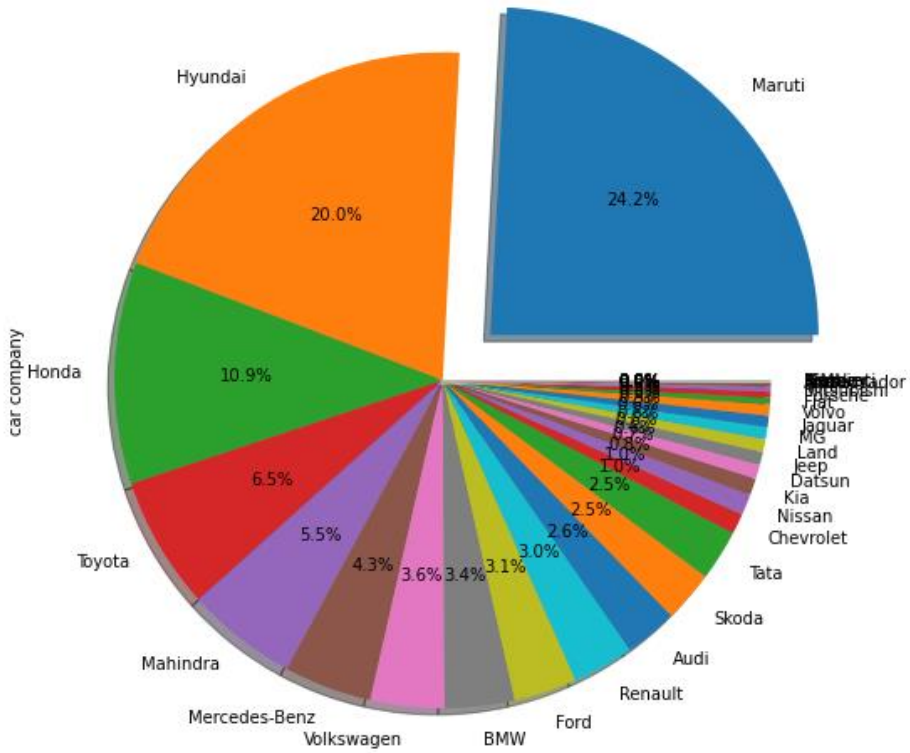
Location

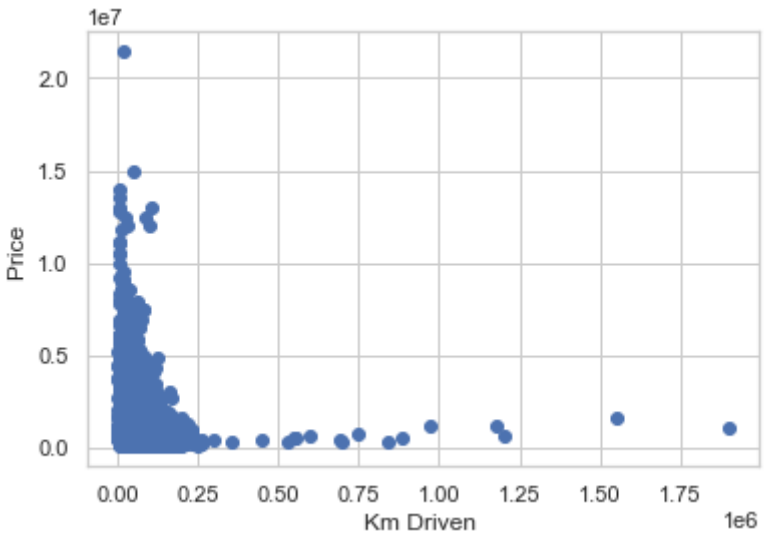


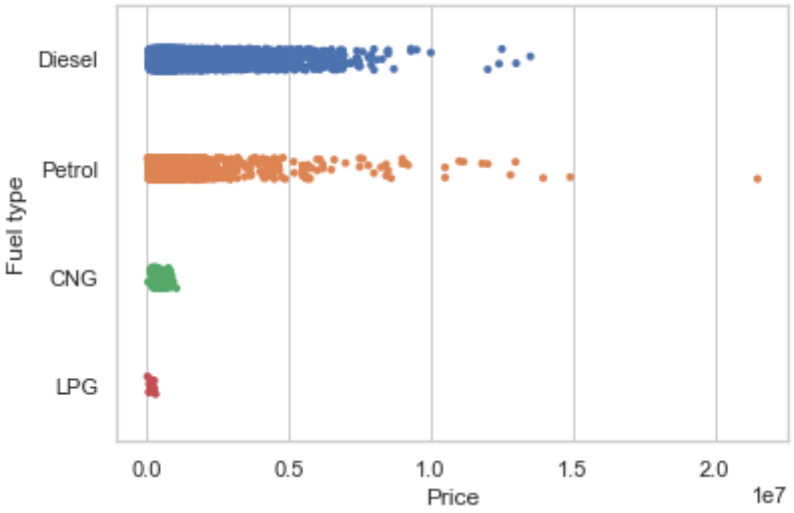
year model

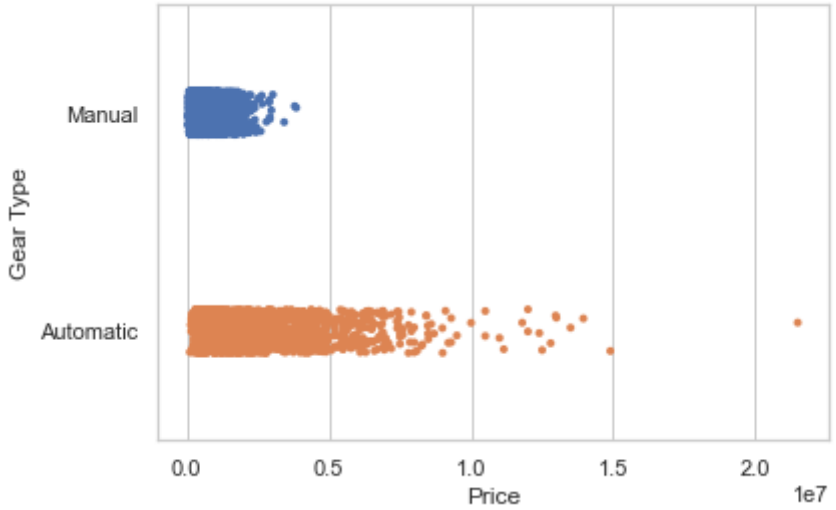


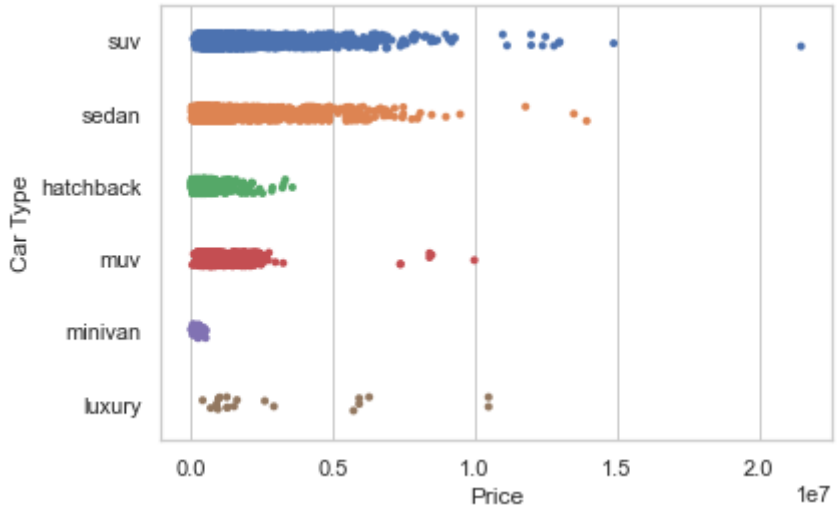
car company

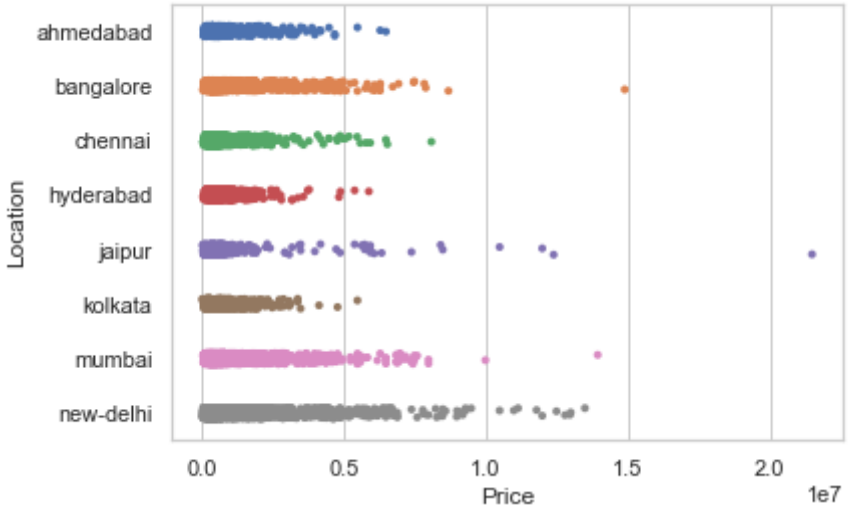


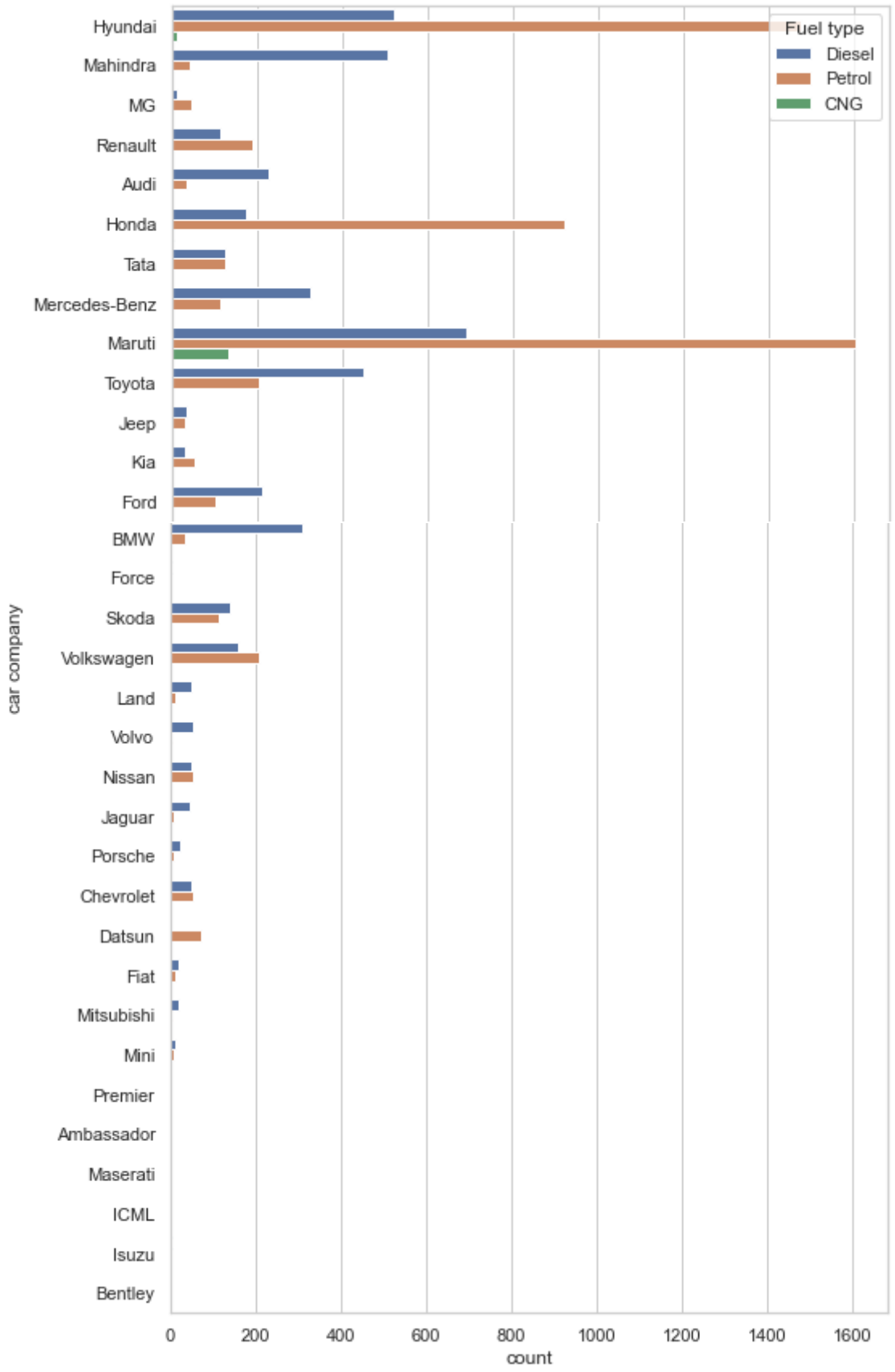

1e7
2.0
1.5
1.0
0.5
0.0
Price
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Km Driven
1e6

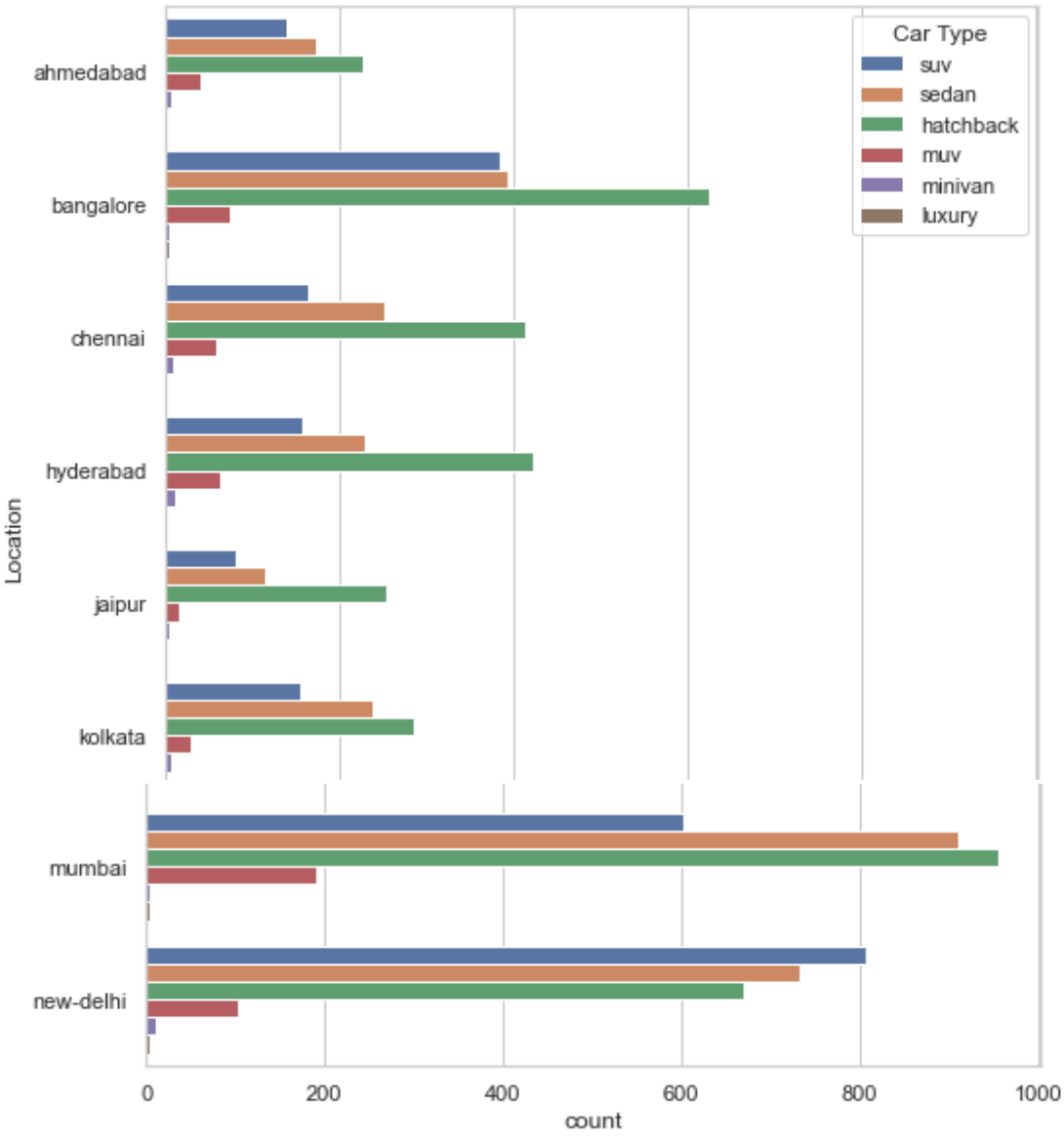

Diesel
Petrol
CNG
LPG
Fuel type
0.0
0.5
1.0
1.5
2.0
Price
1e7

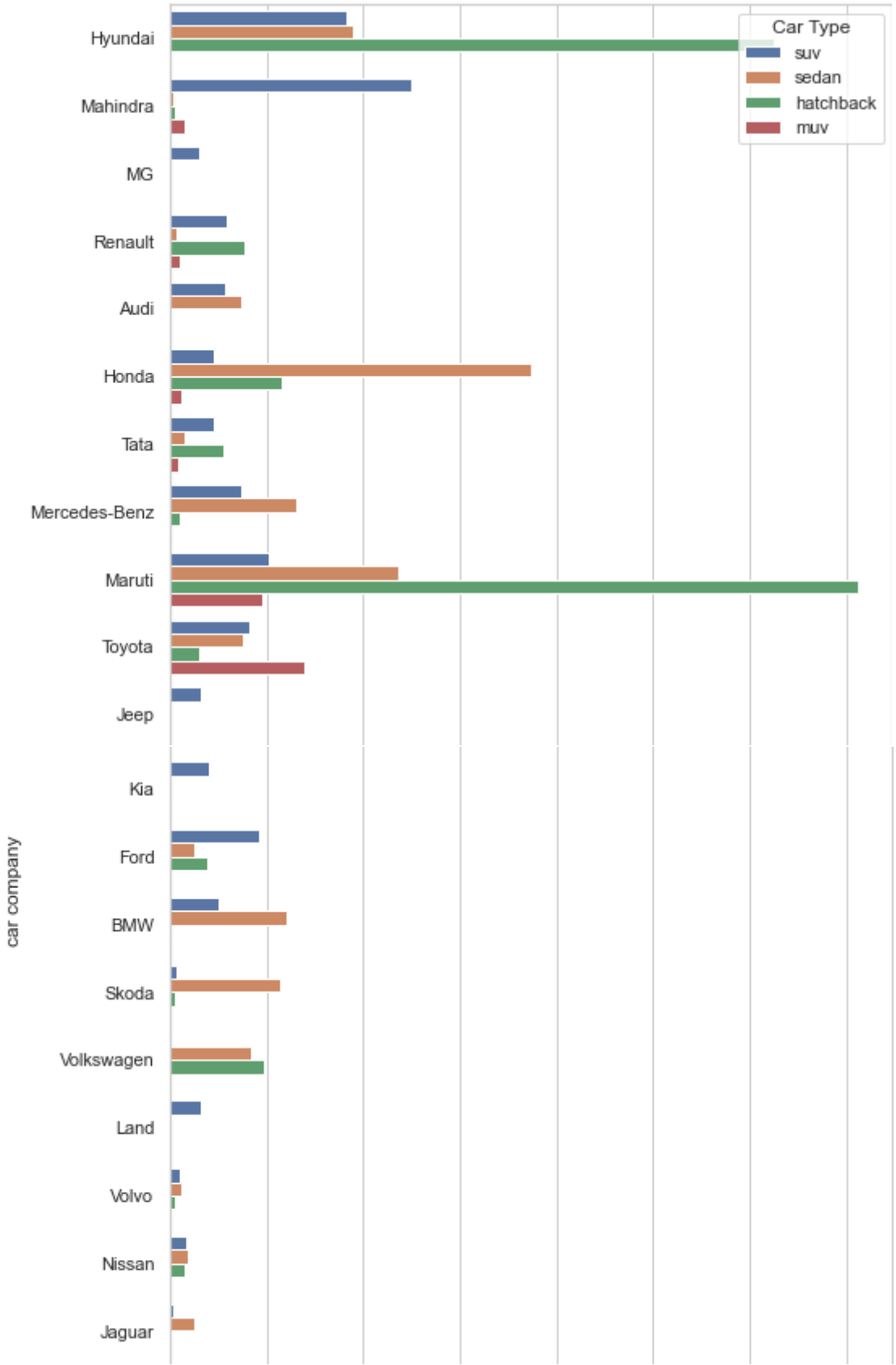

Manual
Automatic
Gear Type
0.0
0.5
1.0
1.5
2.0
Price
1e7

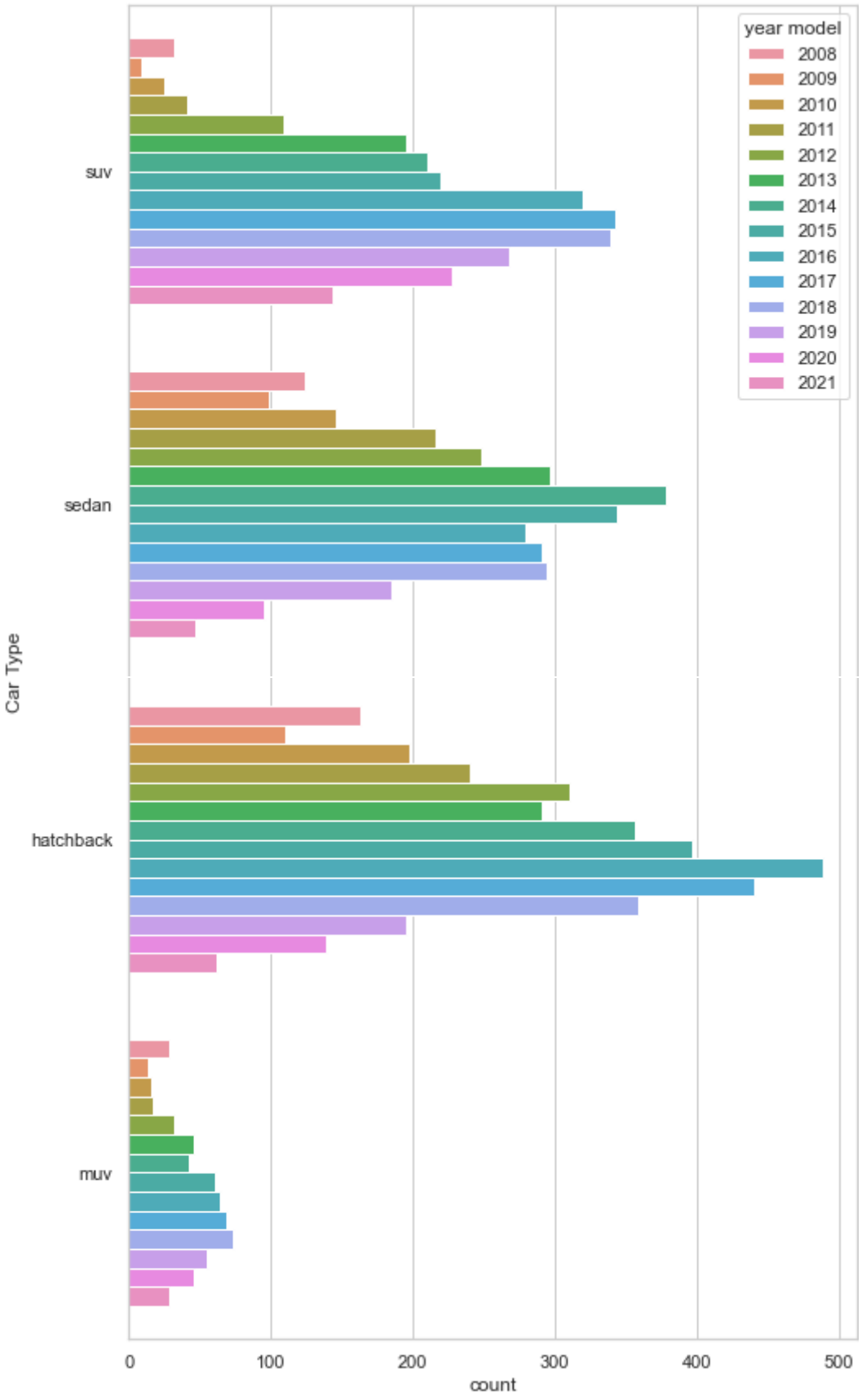

suv
sedan
hatchback
muv
minivan
luxury
Car Type
0.0
0.5
1.0
1.5
2.0
Price
1e7

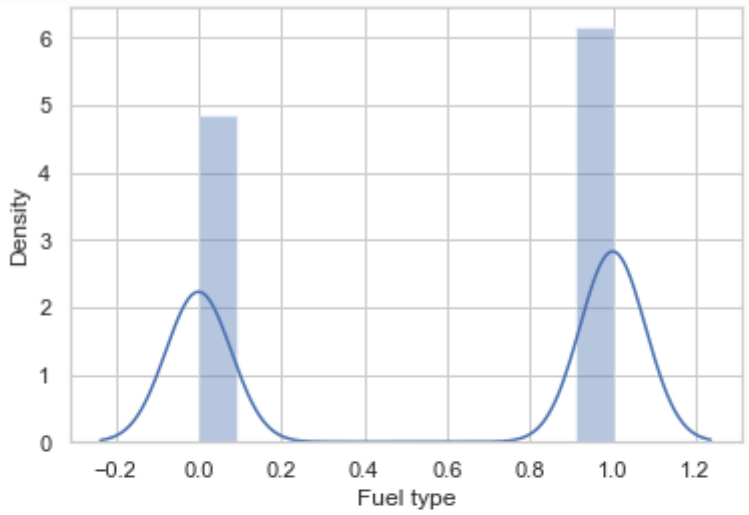

ahmedabad
bangalore
chennai
hyderabad
jaipur
kolkata
mumbai
new-delhi
Location
0.0
0.5
1.0
1.5
2.0
Price
1e7

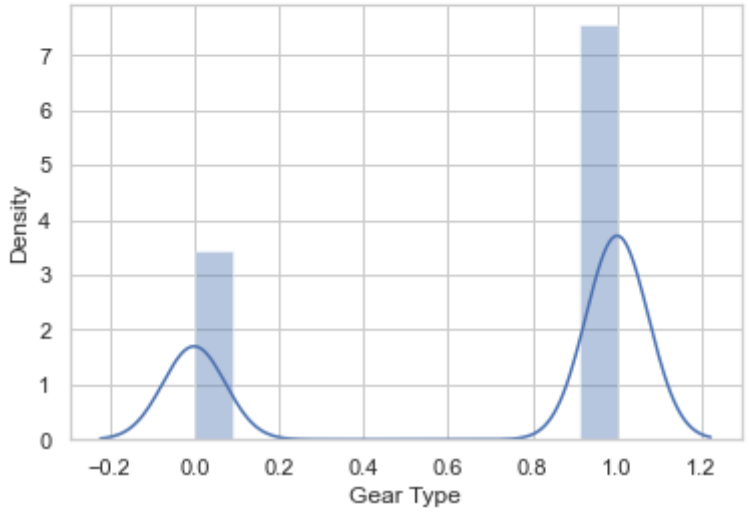

Fuel type
Diesel
Petrol
CNG
car company
Hyundai
Mahindra
MG
Renault
Audi
Honda
Tata
Mercedes-Benz
Maruti
Toyota
Jeep
Kia
Ford
BMW
Force
Skoda
Volkswagen
Land
Volvo
Nissan
Jaguar
Porsche
Chevrolet
Datsun
Fiat
Mitsubishi
Mini
Premier
Ambassador
Maserati
ICML
Isuzu
Bentley
0
200
400
600
800
1000
1200
1400
1600
count

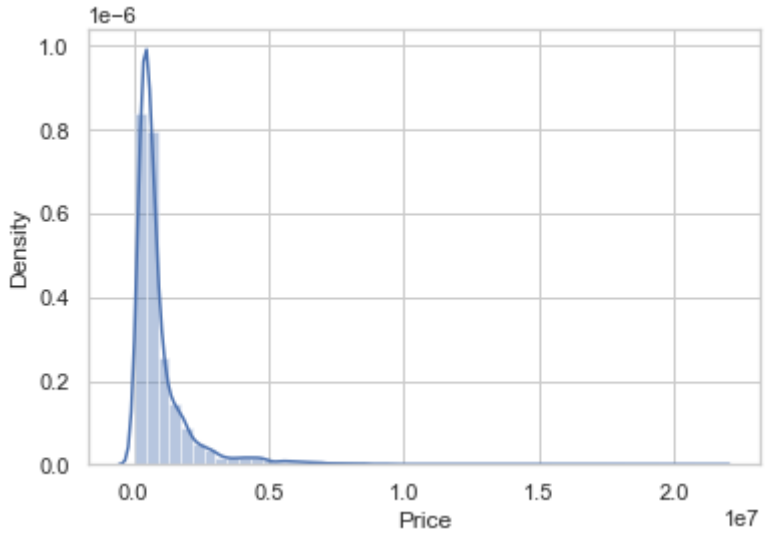

Car Type
suv
sedan
hatchback
muv
minivan
luxury
ahmedabad
bangalore
chennai
hyderabad
jaipur
kolkata
mumbai
new-delhi
Location
0
200
400
600
800
1000
count

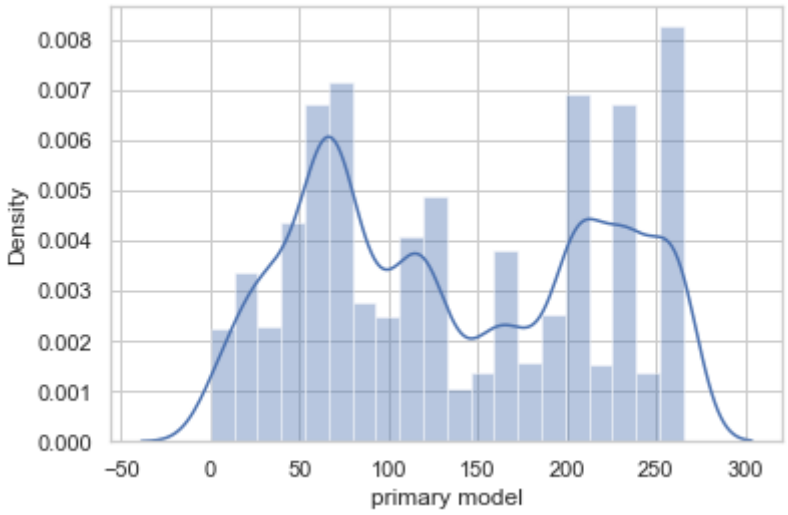

Car Type
suv
sedan
hatchback
muv
Hyundai
Mahindra
MG
Renault
Audi
Honda
Tata
Mercedes-Benz
Maruti
Toyota
Jeep
Kia
Ford
BMW
Skoda
Volkswagen
Land
Volvo
Nissan
Jaguar
car company

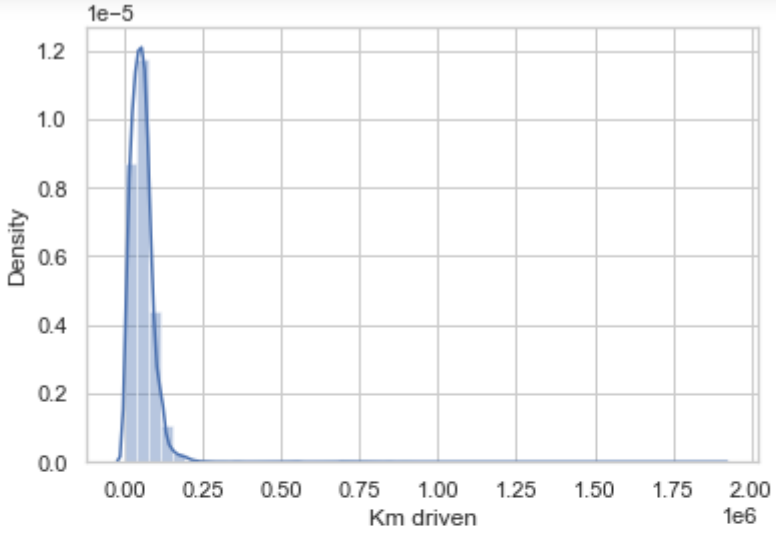

year model
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
suv
sedan
hatchback
muv
Car Type
0
100
200
300
400
500
count

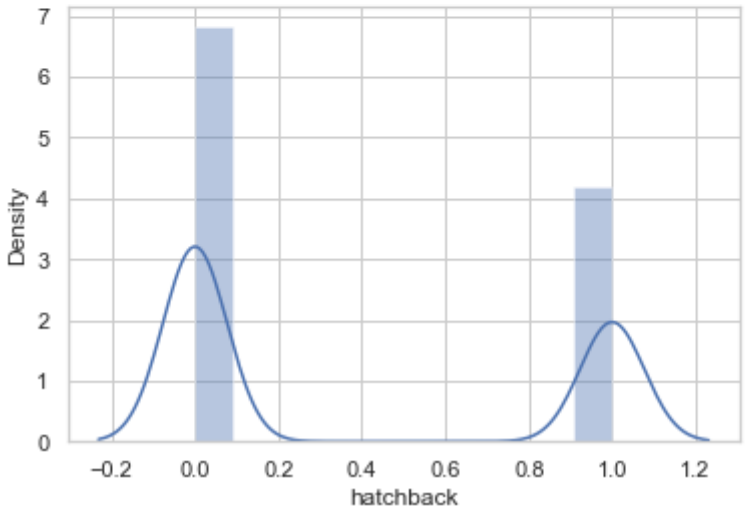

Density
Fuel type

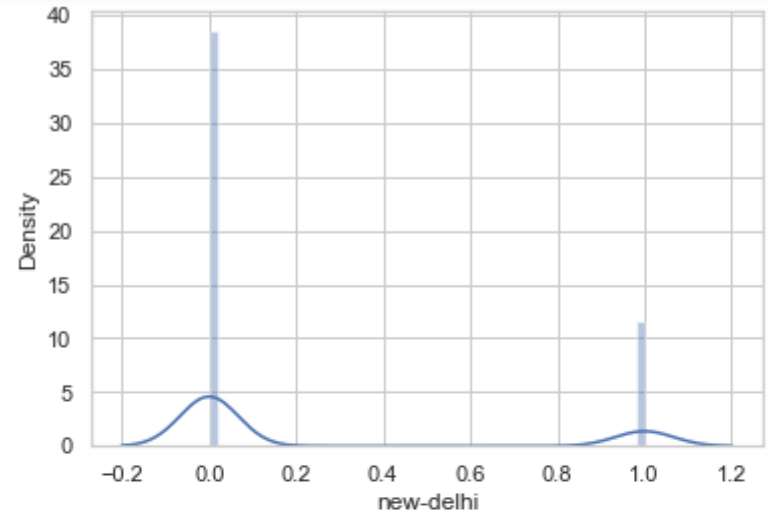

Density
Gear Type

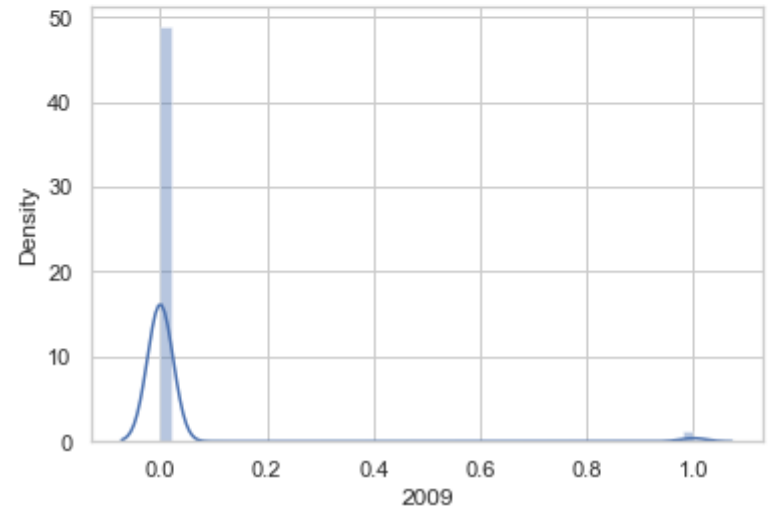

1e−6
Density
Price
1e7

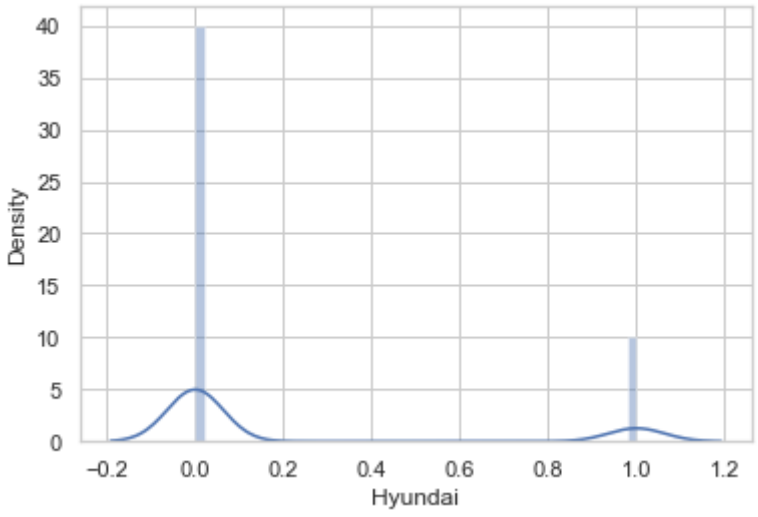

0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
−50
0
50
100
150
200
250
300
primary model

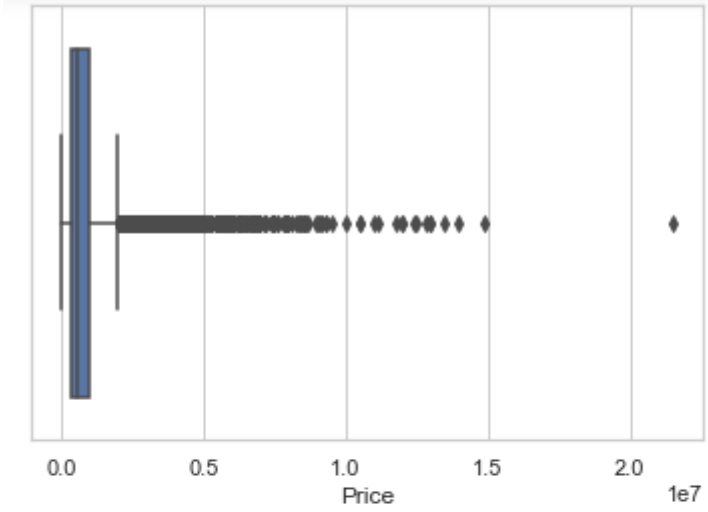

1e−5
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Km driven
1e6

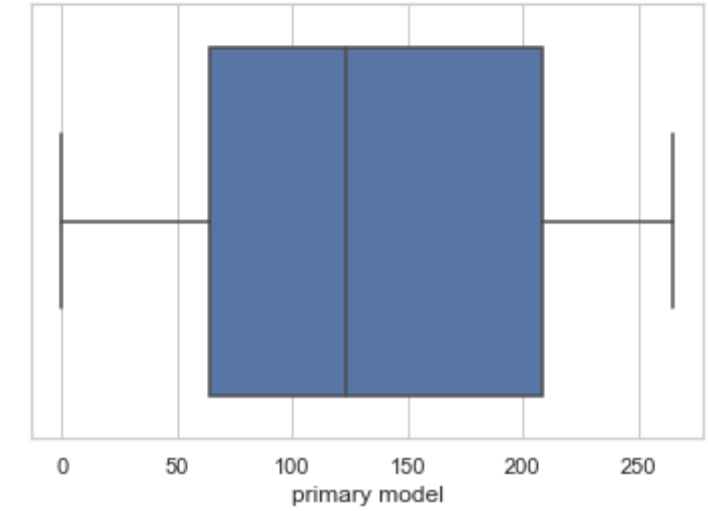

7
6
5
4
3
2
1
0
Density
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
hatchback

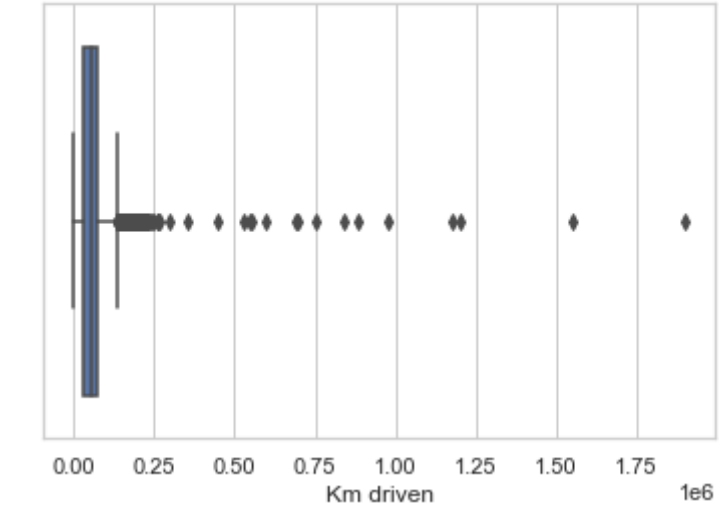

Density
40
35
30
25
20
15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
new-delhi


Density
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
2009


Density
40
35
30
25
20
15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Hyundai

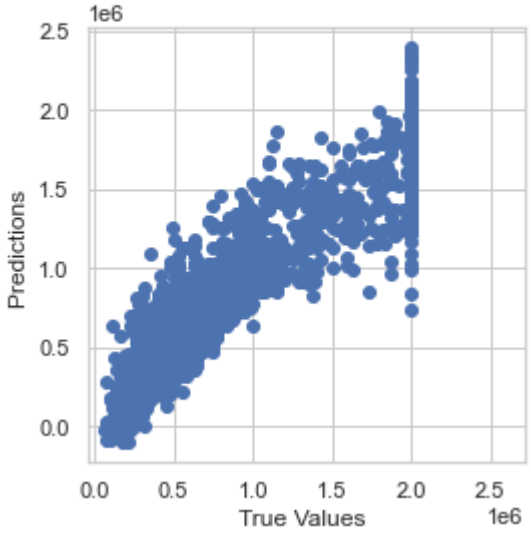

0.0
0.5
1.0
1.5
2.0
Price
1e7


0
50
100
150
200
250
primary model


0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Km driven
1e6

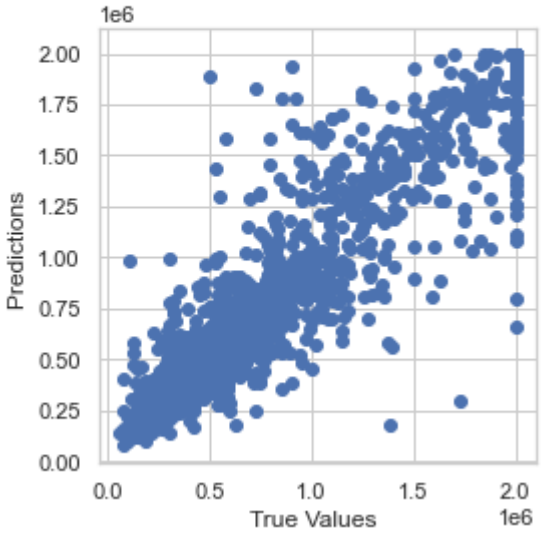
Linear Regression

```
(-38575.0, 2710958.720523153, -226160.0166018006, 2523373.7039213525)
```



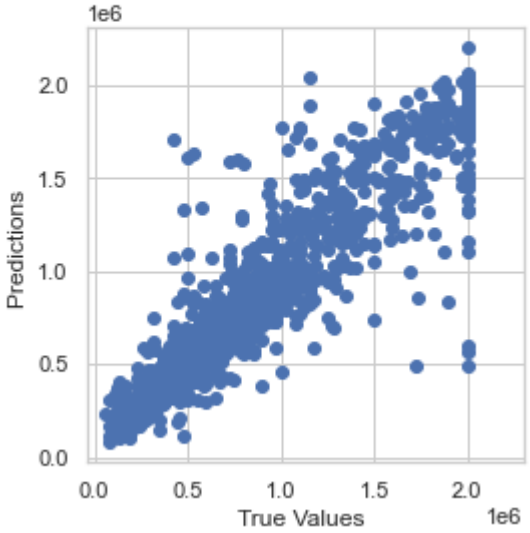
KNN Regression

```
(-38575.0, 2097075.0, -10820.896138642143, 2124829.103861358)
```



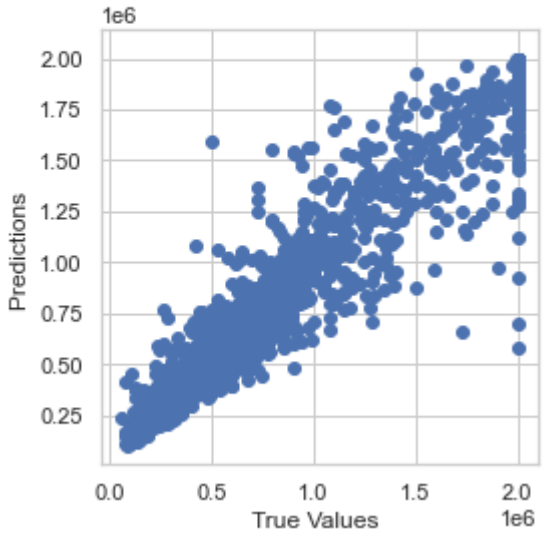
XGB Regression

```
(-38575.0, 2300980.27265625, -28825.85117187501, 2310729.421484375)
```



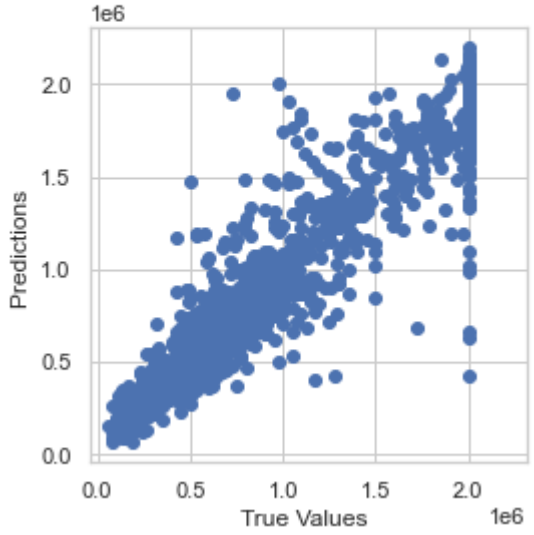
Random Forest Regression

```
(-38575.0, 2097075.0, 9016.599999999991, 2144666.6)
```



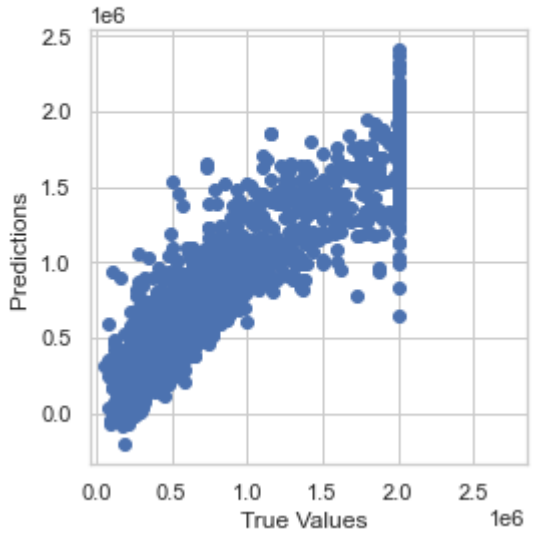
Gradient Boosting Regression

```
(-38575.0, 2310721.7774999626, -41957.544540896764, 2307339.232959066)
```



Lasso Regression

```
(-38575.0, 2850144.9013828007, -341139.6129242332, 2547580.2884585676)
```



Visualization Observation:

1. Petrol is most used fuel by the automobile followed by diesel. LPG and CNG make only tiny fraction.

2. Manual gear type the most used gear type in automobiles.

3. Hatchback, sedan and suv occupy 1/3 data each and there are barely luxury cars and minivans.

4. Mumbai, New Delhi and Bangalore has the most used automobile for sale.

5. 2014-2018 car model are cars which are most sold.

6. Maruti followed by Hyundai and Honda are the major car companies which are up for sale.

7. Car price is inversely proportional to Km driven.

8. Price of diesel vehicles is more compared to petrol driven cars.

9. Manual gear type cars are cheaper compared to Automatic transmission cars.

10. SUV and Sedan are the most expensive car type.

11. Delhi is city the where most expensive cars are sold, budget-oriented cars are available in Hyderabad and Jaipur.

12. All cities except for Delhi primarily sells hatchback cars whereas for Delhi it is SUV.

13. Maruti and Hyundai are major car companies which are on sales.

14. Most number of SUV and Hatchback are of 2017 model.

15. Most number of Sedan are of 2014 model.

16. Most number of MUV are of 2018 model.

Results:

Out of all the Regression models Linear Regression has performed well, XGB Regressor has some over fitting compared all the other model i.e., Random Forest and Gradient Boosting too have performed good but the standard deviation of linear regression is better compared to all the other models. Also, the predicted values and true value have good fit in case of Linear Regression.

Conclusion:

In the whole dataset Gear type column has the most influence on the price of the automobile. Car companies, km driven and model of the car are the other factors which have good influence on the price prediction.

Outcome of the Study:

Visualizing data helped to negotiate few outliers and biased data. Data Cleaning helps in minimizing the overfitting created during model training and improves the model performance. Random Forest and Gradient Boosting can neglect outliers even when the data is fed with outliers to the machine learning model. XGB Regression is not worth the use in this type of problem as it required ample amount of time and results are almost similar to Linear Regression. Simplifying the data was the most challenge part in this project but it overcome if minimal domain knowledge is exploited into this project.

Limitations of this work and Scope for Future Work:

The type price predicted are only limited to particular city and the given models. When we try to predict price of different city and models the machine learning model will fail. Also, for used car most often the data for variety models is limited, as most of the data available are data of the best-selling cars therefore prediction of unique model is always constrained. To improve the model efficiency, we can introduce various other automobile factors such as engine power, mileage, interior quality, battery warranty, tyre quality and insurance of the vehicle etc., inclusion of these can improve the model quality.