Car Price Prediction Project

Submitted by:

Moncy Kurien

ACKNOWLEDGMENT

INTRODUCTION

- Business Problem Framing

  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 carprice valuation model.

- Conceptual Background of the Domain Problem

  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.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  The data would contains both numeric and text data. The test data will be encoded and used along with numerical data to feed into the model.

- Data Sources and their formats

  The data is scraped from OLX and Cardekho sites. There are 10 columns in the dataset. The description of each of the column is given below:

  - "brand": Brand of the car.
  - "model": Model of the car.
  - "variant": Variant of the car.
  - "manufactured_year": The manufactured year of the car.
  - "age_of_car": Derived from the manufactured_year of the car
  - "kilometers_driven": The distance for which the car has been driven since it was bought.
  - "fuel_type": The fuel type of the car.
  - "owners" : The number of owners of the car.
  - "location": The location where the car is being sold.
  - "price": The selling price of the car.

Screen-Shot:

	brand	model	variant	manufactured_year	kilometers_driven	fuel_type	owners	location	price
0	Maruti	Eeco	5 Seater AC BSIV	2016	45,347 Kms	Petrol	1st Owner	Ahmedabad	3,74,500
1	Maruti	Eeco	5 Seater AC	2020	17,116 Kms	Petrol	1st Owner	Ahmedabad	4,74,500
2	Maruti	Swift Dzire	VXI 1.2	2019	11,935 Kms	Petrol	1st Owner	Ahmedabad	6,91,000
3	Maruti	Wagon R	VXI	2014	131,125 Kms	Petrol	1st Owner	Ahmedabad	3,09,500
4	Maruti	Wagon R	VXI	2014	42,381 Kms	Petrol	1st Owner	Ahmedabad	3,61,000

- Data Preprocessing Done
  - Imputed missing values.
  - Cleaned the data to remove unwanted symbols like currency symbols.
  - Derived the age of the car from manufactured_year.
  - Converted the price to show in the same range.
  - Removed ‘km’/’kms’ from kilometers_driven.
  - Used 1,2,3 and 4 in owners feature. 4 represents 4 or 4+ owners.
  - Corrected data in fuel_type to be grouped under the main categories.

- Data Inputs- Logic- Output Relationships

  The input data consists of int values which are the encoded text values using get_dummies() from Pandas. The input data also contains float values.

  The model approximates the function between the input and the output.

- Hardware and Software Requirements and Tools Used
  1. Google Colab
  2. SKLEARN
  3. MATPLOTLIB
  4. PANDAS
  5. NUMPY
  7. glob

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  - Clean the dataset.
  - Encoding the data to get numerical input data.
  - Compare different models and identify the suitable model.
  - R2 score is used as the primary evaluation metric.
  - MSE and RMSE are used as secondary metrics.

- Testing of Identified Approaches (Algorithms)
  - DecisionTreeRegressor
  - LinearRegression
  - Ridge
  - SVR
  - RandomForestRegressor
  - AdaBoostRegressor
  - GradientBoostingRegressor

- Run and Evaluate selected models

<u>Comparing basic models using metrics:</u>

Code:

```
models = [LinearRegression(),Ridge(), DecisionTreeRegressor(), SVR()]

for model in models:

    model.fit(X_train, y_train)

    y_train_pred = model.predict(X_train)

    y_test_pred = model.predict(X_test)

    print(model.__class__.__name__)

    print("Training scores:")

    print("MSE: ",mean_squared_error(y_train, y_train_pred))
```

```
print("RMSE: ",mean_squared_error(y_train, y_train_pred, squared=False))
print("R2 score:",r2_score(y_train, y_train_pred))
print("-"*20)
print("Testing scores:")
print("MSE: ",mean_squared_error(y_test, y_test_pred))
print("RMSE: ",mean_squared_error(y_test, y_test_pred, squared=False))
print("R2 score:",r2_score(y_test, y_test_pred))
print("="*20)
print("\n")
```

Results:

```
LinearRegression
Training scores:
MSE:  2.721764856699407
RMSE:  1.6497772142624005
R2 score: 0.9805746507820545
--------------------
Testing scores:
MSE:  5.038447790436159e+20
RMSE:  22446487008.964584
R2 score: -4.176647606093721e+18
====================
Ridge
Training scores:
MSE:  7.337371911344532
RMSE:  2.7087583707936247
R2 score: 0.9476328708672305
--------------------
Testing scores:
MSE:  23.659619866024766
RMSE:  4.8641155276190515
R2 score: 0.8038723456416614
```

```
DecisionTreeRegressor
Training scores:
MSE:  0.0014301610852055955
RMSE:  0.037817470634689404
R2 score: 0.9999897928807298
--------------------
Testing scores:
MSE:  26.884004134126943
RMSE:  5.184978701414977
R2 score: 0.7771436438774829
====================
SVR
Training scores:
MSE:  122.70799442965733
RMSE:  11.077364056022413
```

```
R2 score: 0.12422792962343776
--------------------
Testing scores:
MSE:  100.60737152760484
RMSE:  10.030322603366496
R2 score: 0.16600994011733705
====================
```

Comparing Ensemble techniques using metrics:

Code:

```
models = [RandomForestRegressor(),AdaBoostRegressor(), GradientBoostingRegressor()]
for model in models:
  model.fit(X_train, y_train)
  y_train_pred = model.predict(X_train)
  y_test_pred = model.predict(X_test)
  print(model.__class__.__name__)
  print("Training scores:")
  print("MSE: ",mean_squared_error(y_train, y_train_pred))
  print("RMSE: ",mean_squared_error(y_train, y_train_pred, squared=False))
  print("R2 score:",r2_score(y_train, y_train_pred))
  print("-"*20)
  print("Testing scores:")
  print("MSE: ",mean_squared_error(y_test, y_test_pred))
  print("RMSE: ",mean_squared_error(y_test, y_test_pred, squared=False))
  print("R2 score:",r2_score(y_test, y_test_pred))
  print("="*20)
  print("\n")
```

Results:

```
RandomForestRegressor
Training scores:
MSE:  4.010541105120959
RMSE:  2.0026335423938546
R2 score: 0.9713766009844165
--------------------
Testing scores:
MSE:  15.811796186680368
RMSE:  3.9764049324333617
R2 score: 0.8689272898361748
====================
AdaBoostRegressor
Training scores:
MSE:  163.57763395377202
RMSE:  12.78974722008891
R2 score: -0.16746039099446497
--------------------
Testing scores:
MSE:  168.18655479608762
RMSE:  12.968675907589319
R2 score: -0.39419122849622856
====================
GradientBoostingRegressor
Training scores:
MSE:  22.09186579971225
```

```
RMSE:  4.700198485139989
R2 score: 0.8423294330591791
--------------------
Testing scores:
MSE:  29.225528852368054
RMSE:  5.406064081415245
R2 score: 0.7577334524538175
====================
```

Observations:

1. Random Forest is giving a very good score.
2. The second best is the Ridge model.

Comparing the top 2 models:

Ridge

Code:

```
alpha = np.arange(0,1,0.01)
ridge_grid = {'alpha':alpha}
rdg = Ridge()
rdg_randomSearch = RandomizedSearchCV(estimator = rdg, param_distributions =
ridge_grid, scoring = 'r2', n_iter = 500, cv = 5, verbose = 10, random_state = 42, n_jobs = -1)
rdg_randomSearch.fit(X_train, y_train)
rdg_randomSearch.best_score_
```

Results:

```
R2 score: 0.6858481404992058
```

RandomForestRegressor

code:

```
rf_random = RandomForestRegressor()
###Hyperparameters
import numpy as np
#nuber of tress to be used in the Random Forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
#Number of features to consider at every split
max_features = ['auto','sqrt']
#Maximum number of levels in the tree
max_depth = [int(x) for x in np.linspace(5,30, num = 6)]
#Maximum number of observations/samples required to do the split
min_samples_split = [2,5,10,15,100]
#The minimum number of samples required to be at a leaf node.
min_samples_leaf = [1,2,5,10]
```

```
random_grid = {'n_estimators':n_estimators,
               'max_features': max_features,
               'max_depth':max_depth,
               'min_samples_split':min_samples_split,
               'min_samples_leaf':min_samples_leaf}
print(random_grid)
rf_randomSearch = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, scoring = 'r2', n_iter = 500, cv = 5, verbose = 10, random_state = 42, n_jobs = -
1)
rf_randomSearch.fit(X_train, y_train)
rf_randomSearch.best_score_
```

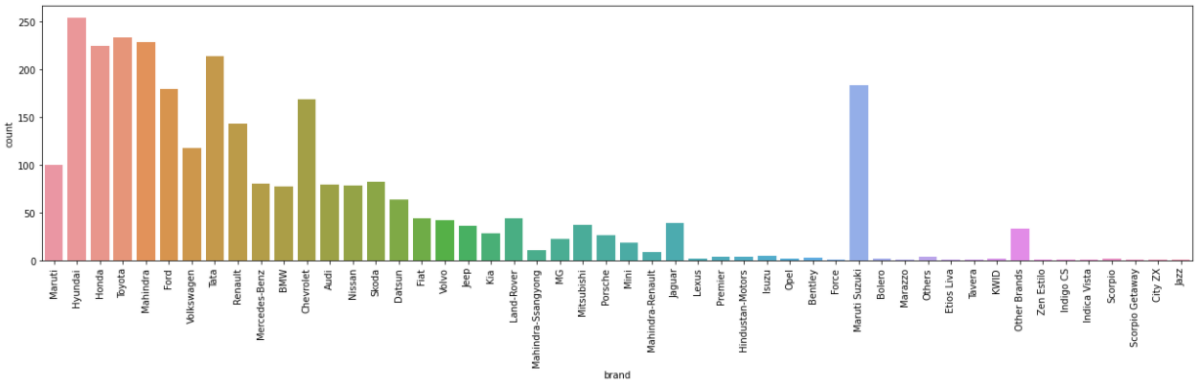
Results:

```
R2 Score: 0.6784482396453451
```

Observations:

- Ridge model shows a better performance than RandomForestRegressor.

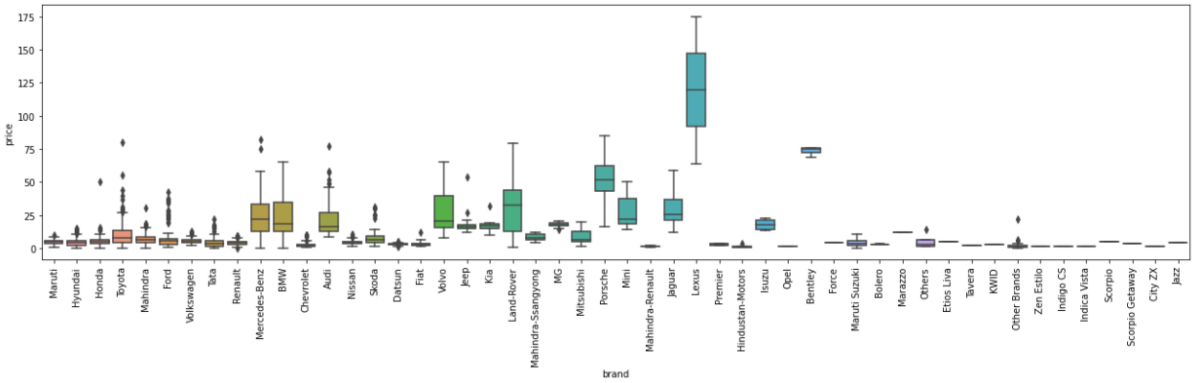
- Key Metrics for success in solving problem under consideration
  - R2 Score is used as the primary key metric for evaluation.
  - MSE and RMSE are used as secondary metrics.
- Visualizations
  1. Count of brands



Observations:

- Most of the data are from Hyundai, honda, Toyota, Mahindra cars.

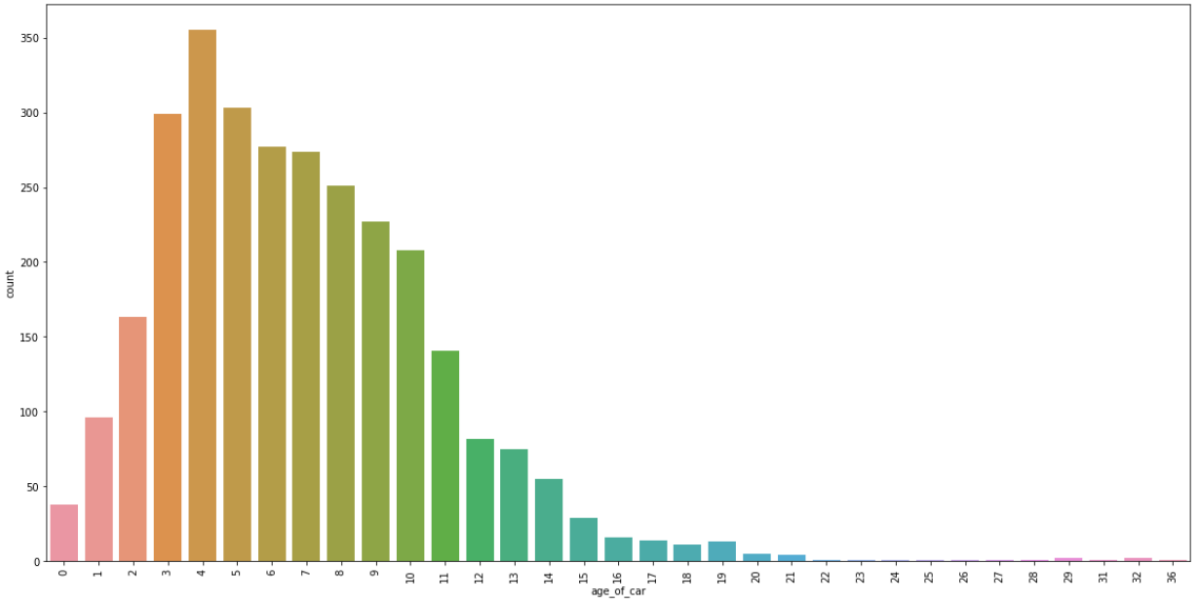
2. Car prices based on brands:



Observation:

- Lexus brand is the most expensive.

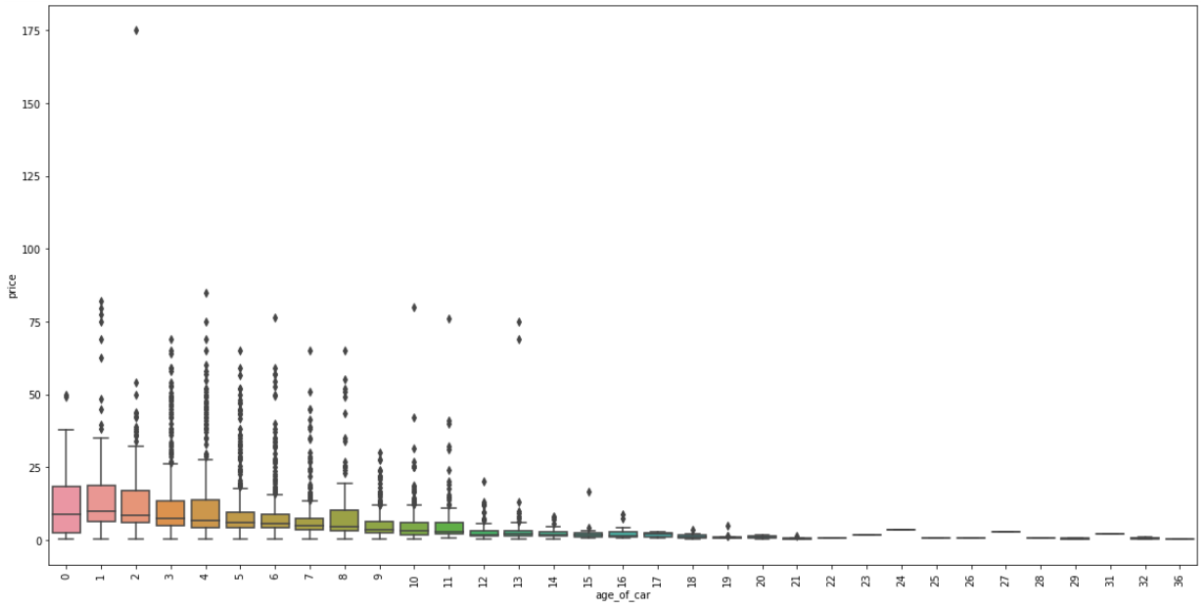
3. Car count based on age of car:



Observations:

- Most of the cars are 4 years old.
- Many cars fall in the range of 3 to 10 years of age.

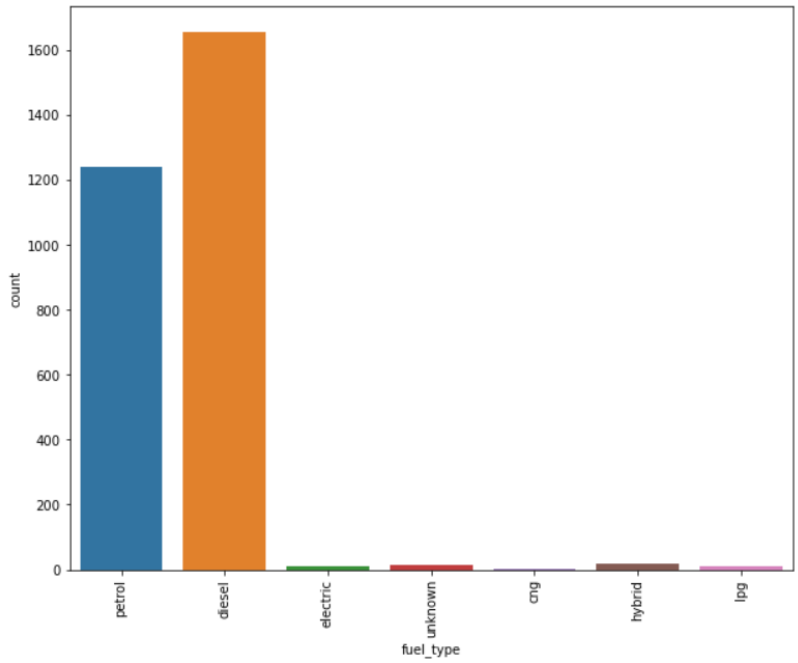
4. Price based on age of the cars:



Observations:

- There are expensive cars in the 1 to 13 years range.

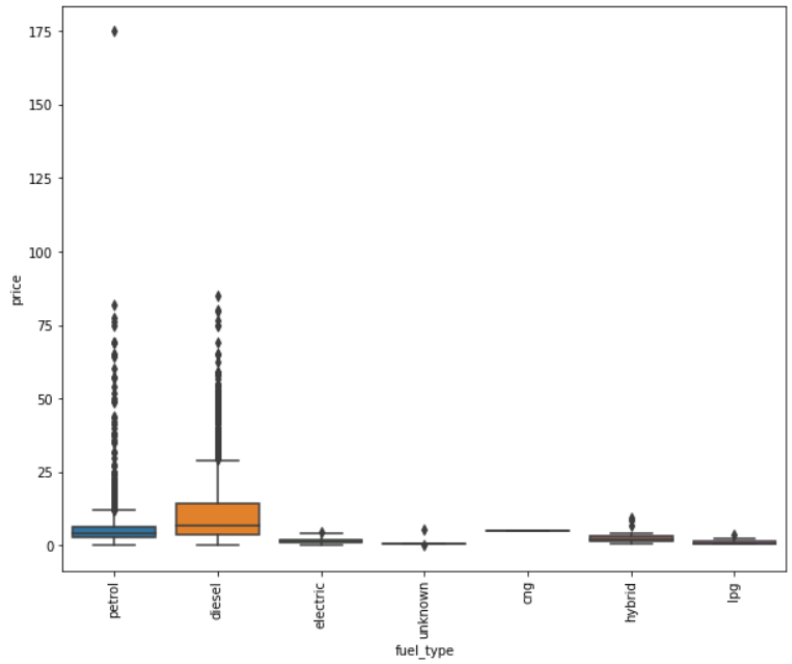
5. Number of cars based on fuel type:



Observations:

- A large number of cars are petrol or diesel cars in the dataset.

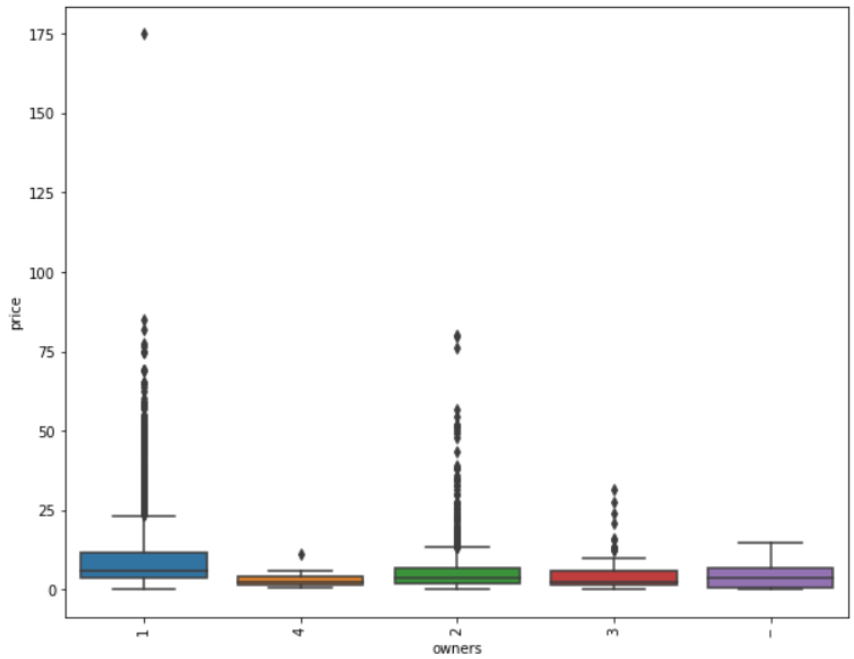
6. Price based on fuel type:



Observations:

- The most expensive cars are Diesel type in general although petrol type has one natural outlier datapoint.

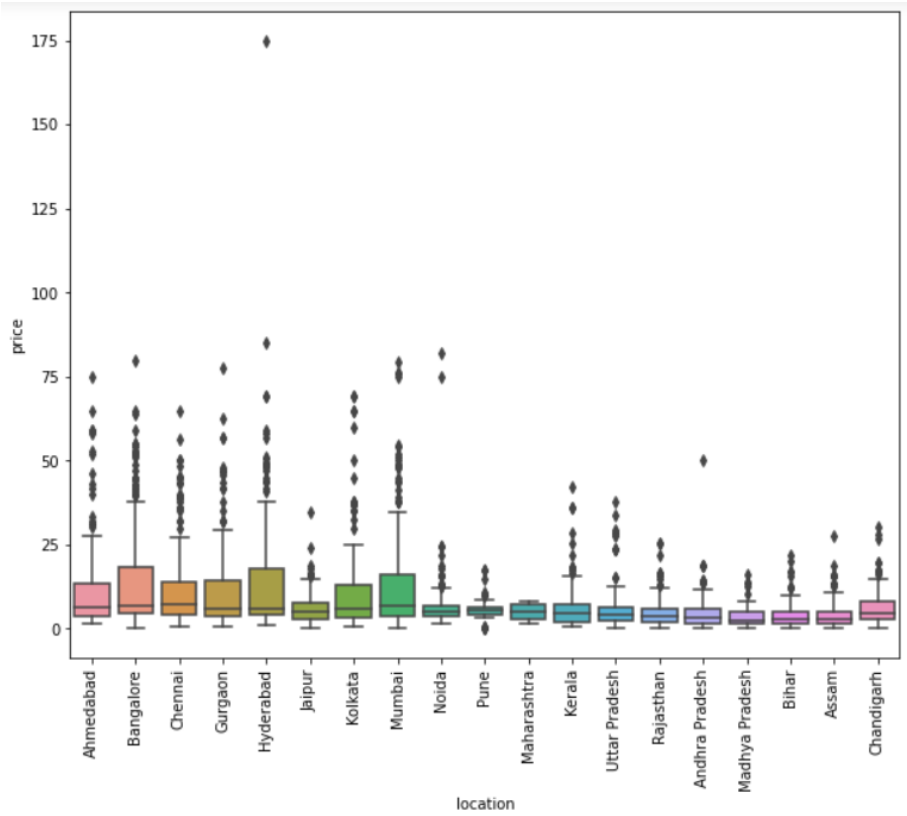
7. Price based on number of owners:



Observations:

- The price is more if the number of previous owners is less.

8. Price based on locations:



Observations:

- Bangalore and Hyderabad see to have more expensive cars.

- Interpretation of the Results
  - We can see from the visuals that the features are impacting the price.
  - There are categorical data that needs to be encoded.

CONCLUSION

- Key Findings and Conclusions of the Study
  - The Brand of the car, Age of the car and the number of previous owners of the car have more influence on the price of the car.
- Limitations of this work and Scope for Future Work

  Retraining of the model is important at frequent intervals so that the predicted prices stay relevant to the economic situation.