FLIP ROBO

CAR PRICE PREDICTION PROJECT

Submitted by:

ASHOK KUMAR SHARMA

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Ms. Rashi Mathur** as well as **Flip Robo Technologies** who gave me the golden opportunity to do project on **Car Price Prediction**, which also helped me in doing lots of research and I came to know about so many new things.

I am very much thankful to **Dr. Deepika Sharma, Trainer (DataTrained)**, for their valuable guidance, keen interest and encouragement at various stages of my training period which eventually helped me a lot in doing this project.

I also acknowledge with thanks for suggestion and timely guidance, which I have received from my SME Ms. Rashi Mathur during this project, which immensely helped me in the evaluation of my ideas on the project.

ASHOK KUMAR SHARMA

INTRODUCTION

- Business Problem Framing

  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. With the change in market due to Covid 19 impact, small traders are facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data.

- ## Conceptual Background of the Domain Problem

  Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including, brand, model, variant, driven kilometres, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on collected data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

- ## Motivation for the Problem Undertaken

  Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including, brand, model, variant, driven kilometres, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

For checking datatypes and null values, pandas.DataFrame.info() and pandas.Series.isnull().sum() method has been used. To drop the null values pandas.DataFrame.dropna() method has been used. To replace and remove the certain terms and punctuations, pandas.Series.str.replace() method with regular expression has been used.

- ## Data Sources and their formats

The dataset is in the form of .CSV (Comma Separated Value) format and consists of 10 columns (9 features and 1 label) with 10083 number of records as explained below:

1. Brand: Brand name of the car.
2. Model: Model name of the car.
3. Varient: Model Varient of the car.
4. Manufacture Year: Year in which car was manufactured.
5. Driven Kilometers: How many kilometers car has been driven till the date.
6. Fuel: Type of fuel can be used to operate the car.
7. Number of Owner: How many times car has been sold.
8. Body Type: Type of car body ie. Hatchback, Sedan, SUV, MUV, Coupe, Luxury, Super Luxury, Minivan, Luxury Sedan, Luxury SUV.
9. Location: Location in which car is available for selling.
10. Price: Price of the car.

- ## Data Pre-processing Done

The following pre-processing pipeline is required to perform model prediction:

1. **Load Dataset**
2. **Perform Data Cleansing as follows:**
   - Extract numbers from `Driven Kilometers` and store it as float64 type.
   - Rename columns as *{"Manufacture Year":"ManufactureYear","Driven Kilometers":"DrivenKilometers","Number of Owner":"NumberOfOwner","Body Type":"BodyType"}*
   - Convert all the object type features to lower case.
   - Change the data type of `Manufacture Year` from *int64 to float46*.

3. **Encode descrete features using pandas *get_dummies()* function**.

4. **Remove outliers using scipy.stats *zscore()* function keeping threshold -3 to +3 and % of data loss <=5%.**
5. **Seperate input and output variables.**
6. **Treat skewness in continuous features using sklearn.preprocessing *power_transform()* function.**
7. **Scale continuous features using sklearn.preprocessing *StandardScaler()* function.**
8. **Apply decomposition on input variables using sklearn.decomposition *PCA*.**
9. **Load saved or serialized model using *joblib.load()* and predict values.**

- Data Inputs- Logic- Output Relationships

Input	Logic (algorithm)	Output
Brand Model Varient ManufactureYear DrivenKilometers Fuel NumberOfOwner BodyType Location	LinearRegression SGDRegressor Ridge Lasso **KNeighborsRegressor**	Price

There is 9 input variable needs to be provided to the logic to get the output i.e. Price. Logic highlighted in green i.e. KNeighborsRegressor is the best performing algorithm among all other algorithms on this dataset.

- Hardware and Software Requirements and Tools Used

During this project, following set of hardware is being used:

RAM: 8 GB

PAGE_FILE: 90GB on SSD

CPU: AMD A8 Quad Core 2.2 Ghz

GPU: AMD Redon R5 Graphics

and the following software and tools is being used:

a. Python
b. Jupyter Notebook
c. Anaconda

With following libraries and packages:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- scipy
- sys
- tqdm.notebook
- timeit
- sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  To solve this problem following steps are used:

1. **Load Dataset**
2. **Perform Data Cleansing as follows:**
   - Extract numbers from `Driven Kilometers` and store it as float64 type.
   - Rename columns as *{"Manufacture Year":"ManufactureYear","Driven Kilometers":"DrivenKilometers","Number of Owner":"NumberOfOwner","Body Type":"BodyType"}*
   - Convert all the object type features to lower case.
   - Change the data type of `Manufacture Year` from *int64 to float46*.
3. **Encode descrete features using pandas *get_dummies()* function**.
4. **Remove outliers using scipy.stats *zscore()* function keeping threshold -3 to +3 and % of data loss <=5%.**
5. **Seperate input and output variables**.
6. **Treat skewness in continuous features using sklearn.preprocessing *power_transform()* function**.
7. **Scale continuous features using sklearn.preprocessing *StandardScaler()* function**.
8. **Apply decomposition on input variables using sklearn.decomposition *PCA***.
9. **Train & Test Model**

- Testing of Identified Approaches (Algorithms)

Following are the list of algorithms used for training and testing:

1. LinearRegression
2. SGDRegressor
3. Ridge
4. Lasso
5. KNeighborsRegressor

- Run and Evaluate selected models

  A total of 5 algorithm has been used on this dataset for training testing purpose, these are LinearRegression, SGDRegressor, Ridge, Lasso and KNeighborsRegressor. To perform training and testing operation(s) following functions has been defined for which codes are as follows:

```
#importing required libraries

from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV

from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error

from tqdm.notebook import tqdm

import sys, timeit


#defining function for best random state

def get_best_random_state(model,x,y,r_state=[38,40,42,44,46],t_size=0.33):

    best_random_state = None

    best_r2_score = None

    msg = ""


    for r in tqdm(r_state,desc="Finding Best Random State"):

        x_train,x_test,y_train,y_test =
train_test_split(x,y,random_state=r,test_size=t_size)


        model.fit(x_train,y_train)

        y_predict = model.predict(x_test)


        temp_r2_score = r2_score(y_test,y_predict)
```

```
        if best_r2_score == None:
            best_r2_score = temp_r2_score
            best_random_state = r

        if temp_r2_score > best_r2_score:
            best_r2_score = temp_r2_score
            best_random_state = r

        msg += f"[{r}: {temp_r2_score}] "
        sys.stdout.write(f"\r{msg}    ")

    return best_random_state, best_r2_score



#defining function to get best CV score
def get_best_cv(model,x_train,y_train,cv_range=[3,5,7,9,11]):
    best_cv_score = None
    best_cv = None
    msg = ""

    for cv in tqdm(cv_range,desc="Finding Best CV"):
        temp_cv_score = cross_val_score(model,x_train,y_train,cv=cv).mean()

        if best_cv_score == None:
            best_cv_score = temp_cv_score
            best_cv = cv

        if temp_cv_score > best_cv_score:
            best_cv_score = temp_cv_score
            best_cv = cv

        msg += f"[{cv}: {temp_cv_score}] "
        sys.stdout.write(f"\r{msg}      ")
```

```
    return best_cv,best_cv_score



#defining function for model training & testing

def
build_models(models,X,Y,r_state=[38,40,42,44,46],t_size=0.33,cv_range=[3,5,
7,9,11]):

    for m in tqdm(models,desc="Training & Testing Models"):

        print(f"====================================================")

        print(f"Processing: {m}")

        print(f"====================================================")


        #start timer

        start_time = timeit.default_timer()


        #initializing model

        model = models[m]


        #getting best random state

        best_random_state, initial_r2_score =
get_best_random_state(model['name'],X,Y,r_state=r_state,t_size=t_size)


        #split train & test data

        x_train,x_test,y_train,y_test =
train_test_split(X,Y,random_state=best_random_state,test_size=t_size)


        #getting best cv

        best_cv, best_cv_score =
get_best_cv(model['name'],x_train,y_train,cv_range=cv_range)


        #training model using GridSearchCV

        gscv = GridSearchCV(model['name'],model['parameters'],cv=best_cv)

        gscv.fit(x_train,y_train)


        #testing model
```

```
        y_predict = gscv.best_estimator_.predict(x_test)
        final_r2_score = r2_score(y_test,y_predict)
        mse = mean_squared_error(y_test,y_predict)
        mae = mean_absolute_error(y_test,y_predict)

        #storing values
        models[m]["random_state"] = best_random_state
        models[m]["initial_r2_score"] = initial_r2_score
        models[m]["cv"] = best_cv
        models[m]["cross_val_score"] = best_cv_score
        models[m]["gscv"] = gscv
        models[m]["final_r2_score"] = final_r2_score
        models[m]["mse"] = mse
        models[m]["rmse"] = np.sqrt(mse)
        models[m]["mae"] = mae
        models[m]["x_train"] = x_train
        models[m]["x_test"] = x_test
        models[m]["y_train"] = y_train
        models[m]["y_test"] = y_test
        models[m]["y_predict"] = y_predict

        print("\n\n")

    return models



#defining function to display model performance
def display_performance(models):
    model_names = []
    model_initial_scores = []
    model_cross_val_scores = []
    model_final_scores = []
    model_mse = []
    model_rmse = []
```

```
    model_mae = []

    for m in models:
        model_names.append(m)
        model_initial_scores.append(models[m]["initial_r2_score"])
        model_cross_val_scores.append(models[m]["cross_val_score"])
        model_final_scores.append(models[m]["final_r2_score"])
        model_mse.append(models[m]["mse"])
        model_rmse.append(models[m]["rmse"])
        model_mae.append(models[m]["mae"])

    model_performances = pd.DataFrame({
        "Model Name": model_names,
        "Initial R2 Score": model_initial_scores,
        "Cross Val Score": model_cross_val_scores,
        "Final R2 Score": model_final_scores,
        "MSE": model_mse,
        "RMSE": model_rmse,
        "MAE": model_mae
    })

    model_performances["Final R2 Score - Cross Val Score"] =
model_performances["Final R2 Score"] - model_performances["Cross Val
Score"]
    return model_performances



#importing required model algorithms
from sklearn.linear_model import LinearRegression, SGDRegressor, Ridge,
Lasso
from sklearn.neighbors import KNeighborsRegressor

#preparing list of models
models = {
    "LinearRegression": {
        "name": LinearRegression(),
```

```
    "parameters": {
        "fit_intercept": [True],
        "normalize": [True, False],
        "n_jobs": [-1]
    }
},
"SGDRegressor": {
    "name": SGDRegressor(),
    "parameters": {
        "loss": ['huber','squared_loss'],
        "penalty": ['l2'],
        "max_iter": [3000],

    }
},
"Ridge": {
    "name": Ridge(),
    "parameters": {
        "max_iter": [3000],
        "solver": ['saga','sparse_cg','lsqr'],
    }
},
"Lasso": {
    "name": Lasso(),
    "parameters": {
        "max_iter": [3000],
        "selection": ['random','cyclic'],
    }
},
"KNeighborsRegressor": {
    "name": KNeighborsRegressor(),
    "parameters": {
        "weights": ['uniform','distance'],
        "algorithm": ['ball_tree','kd_tree','brute'],
```

```
            "leaf_size": [40],
            "n_jobs": [-1]
        }
    }
}


#training & testing models
trained_models = build_models(models,X,Y)
```

Training & Testing Models: 100%
5/5 [24:30<00:00, 235.90s/it]

```
=====================================================
Processing: LinearRegression
=====================================================
```

Finding Best Random State: 100%
5/5 [00:03<00:00, 1.37it/s]

```
[38: 0.8402767713985033] [40: 0.8622084616467132] [42: 0.8127623722340922]
[44: 0.8607290464810919] [46: 0.811645810496689]
```

Finding Best CV: 100%
5/5 [00:23<00:00, 5.52s/it]

```
[3: 0.7714657769739143] [5: 0.8110118974625147] [7: 0.8086854874224455] [9:
0.8135738394744733] [11: 0.806098487216493]


=====================================================
Processing: SGDRegressor
=====================================================
```

Finding Best Random State: 100%
5/5 [01:12<00:00, 13.79s/it]

```
[38: 0.8377256181965081] [40: 0.8559232842129666] [42: 0.8048768466150479]
[44: 0.8546225616490616] [46: 0.8141312720996305]
```

Finding Best CV: 100%
5/5 [07:15<00:00, 105.07s/it]

```
[3: 0.8044680144511718] [5: 0.8284509090701813] [7: 0.8247831504777209] [9:
0.8312200028672447] [11: 0.8240238302264568]


=====================================================
Processing: Ridge
=====================================================
```

Finding Best Random State: 100%
5/5 [00:01<00:00, 2.80it/s]

```
[38: 0.8451404960083606] [40: 0.864318488755457] [42: 0.815237812639208] [4
4: 0.861756638259592] [46: 0.8288639711687115]
```

Finding Best CV: 100%
5/5 [00:10<00:00, 2.46s/it]

```
[3: 0.8132499997033843] [5: 0.835751537584971] [7: 0.8313844201911927] [9: 0.8360438801469606] [11: 0.8297068704500506]
```

```
=====================================================
Processing: Lasso
=====================================================
```

Finding Best Random State: 100%
5/5 [00:02<00:00, 1.84it/s]

```
[38: 0.840403813970471] [40: 0.8622868525232632] [42: 0.8128651145904029] [44: 0.8607496170508594] [46: 0.8120546701354745]
```

Finding Best CV: 100%
5/5 [00:17<00:00, 3.85s/it]

```
[3: 0.7727113954303088] [5: 0.8119365417960409] [7: 0.8093092263923621] [9: 0.8141666752046659] [11: 0.806722513901845]
```

```
=====================================================
Processing: KNeighborsRegressor
=====================================================
```

Finding Best Random State: 100%
5/5 [00:12<00:00, 2.58s/it]

```
[38: 0.8346546483826458] [40: 0.8823920844635241] [42: 0.8173145007940013] [44: 0.8807213601362528] [46: 0.8195735842911841]
```

Finding Best CV: 100%
5/5 [00:23<00:00, 5.07s/it]

```
[3: 0.7967718823336841] [5: 0.8274289644982492] [7: 0.8358203214256463] [9: 0.8431421050597832] [11: 0.842011091661163]
```

	Model Name	Initial R2 Score	Cross Val Score	Final R2 Score	MSE	RMSE	MAE	Final R2 Score - Cross Val Score
0	LinearRegression	0.862208	0.813574	0.862208	1.515523e+11	389297.197298	200975.791866	0.048635
1	SGDRegressor	0.855923	0.831220	0.856501	1.578302e+11	397278.442102	195495.065156	0.025281
2	Ridge	0.864318	0.836044	0.864020	1.495598e+11	386729.581656	195816.892216	0.027976
3	Lasso	0.862287	0.814167	0.862287	1.514661e+11	389186.468918	200886.755499	0.048120
4	KNeighborsRegressor	0.882392	0.843142	0.907573	1.016570e+11	318837.021160	104708.315430	0.064431

From the above model comparison it is clear that **KNeighborsRegressor** performs better with **R2 Score: 90.76%** and **Cross Val Score: 84.43%** than other models.

- ## Key Metrics for success in solving problem under consideration

  To find out best performing model following metrics are used:

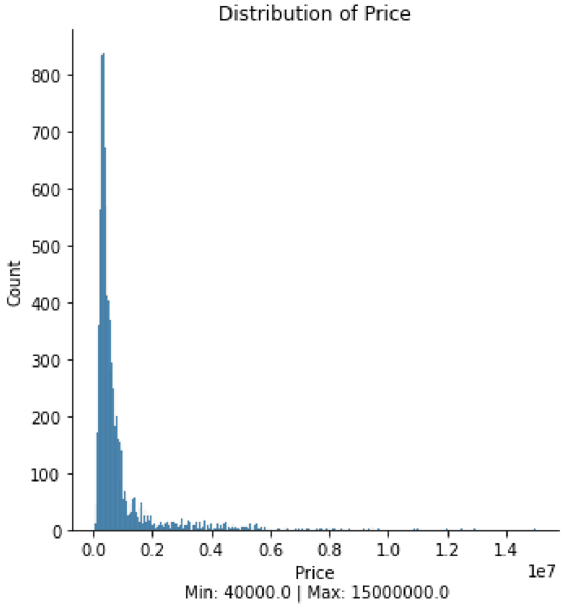
  1. R2 Score: It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance.
  2. Mean Squared Error: A risk metric corresponding to the expected value of the squared (quadratic) error or loss.
  3. Mean Absolute Error: A risk metric corresponding to the expected value of the absolute error loss or -norm loss.
  4. Root Mean Squared Error: Root of mean squared error.

- ## Visualizations

  To better understand the data, following types of visualizations have been used: 1. Univariate.

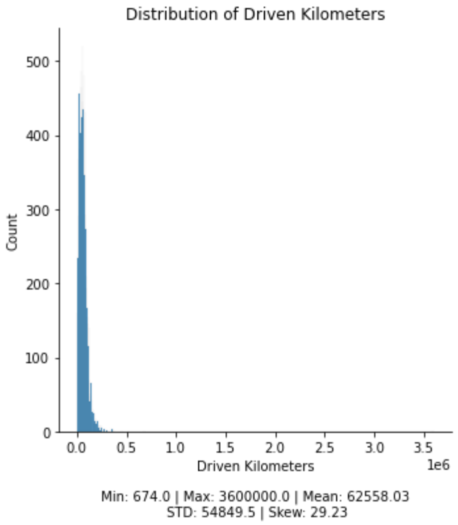
  1. Univariate Analysis: Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. In this project, distribution plot, count plot and box plot has been used.

Displot:



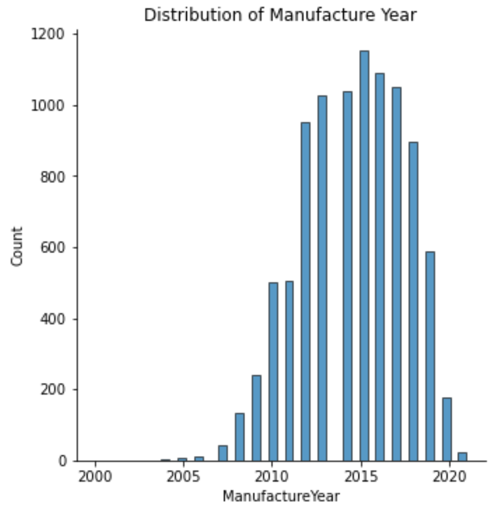
Remarks:

- Most of car **price lies between 40000 to 2000000**.
- **Mininum** price of car is **40000** and **Maximum** price is **15000000**.



Remarks:

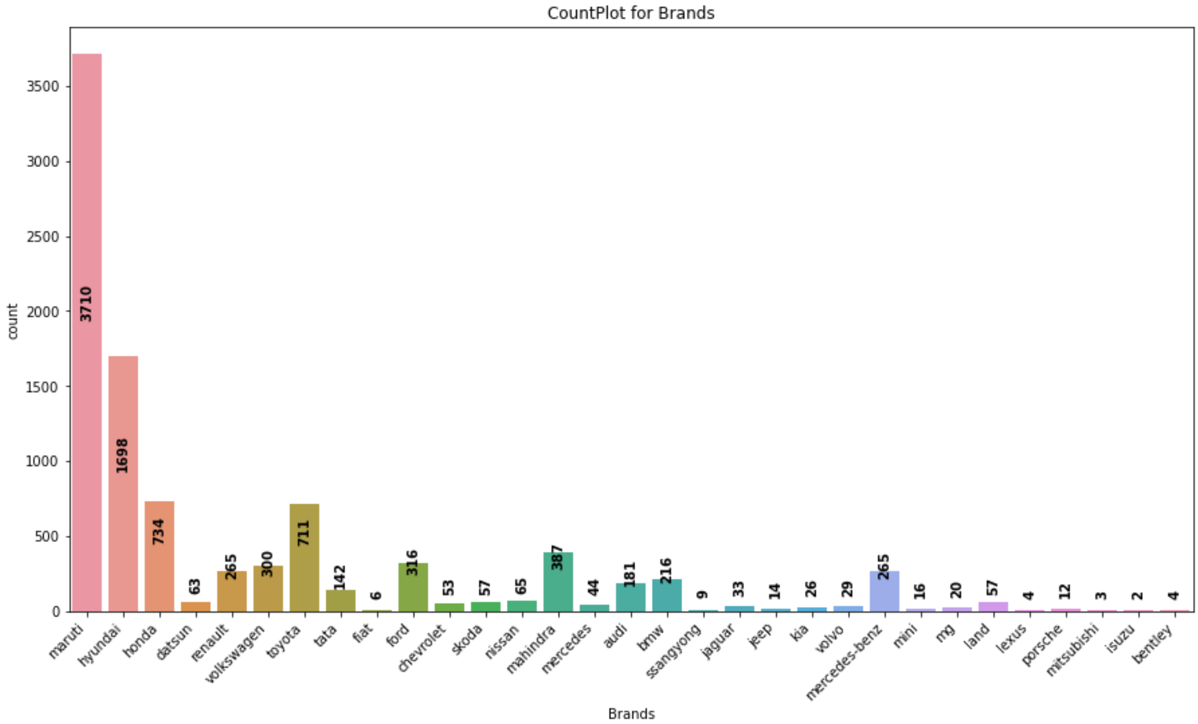
- Data is **not distributed normally** or not in bell-curve.
- Data is **highly spread**.
- Data is **positively skewed** and needs to be treated accordingly before providing to for model training.
- Mininum driven kilometer is 674 while maximum is 36 lakh.



Remarks:

- Most of the cars are from year 2012 to 2019.
- Maximum number of cars are of manufacture year 2015.
- Minimum number of cars are of manufacture year 2000.

CountPlot:

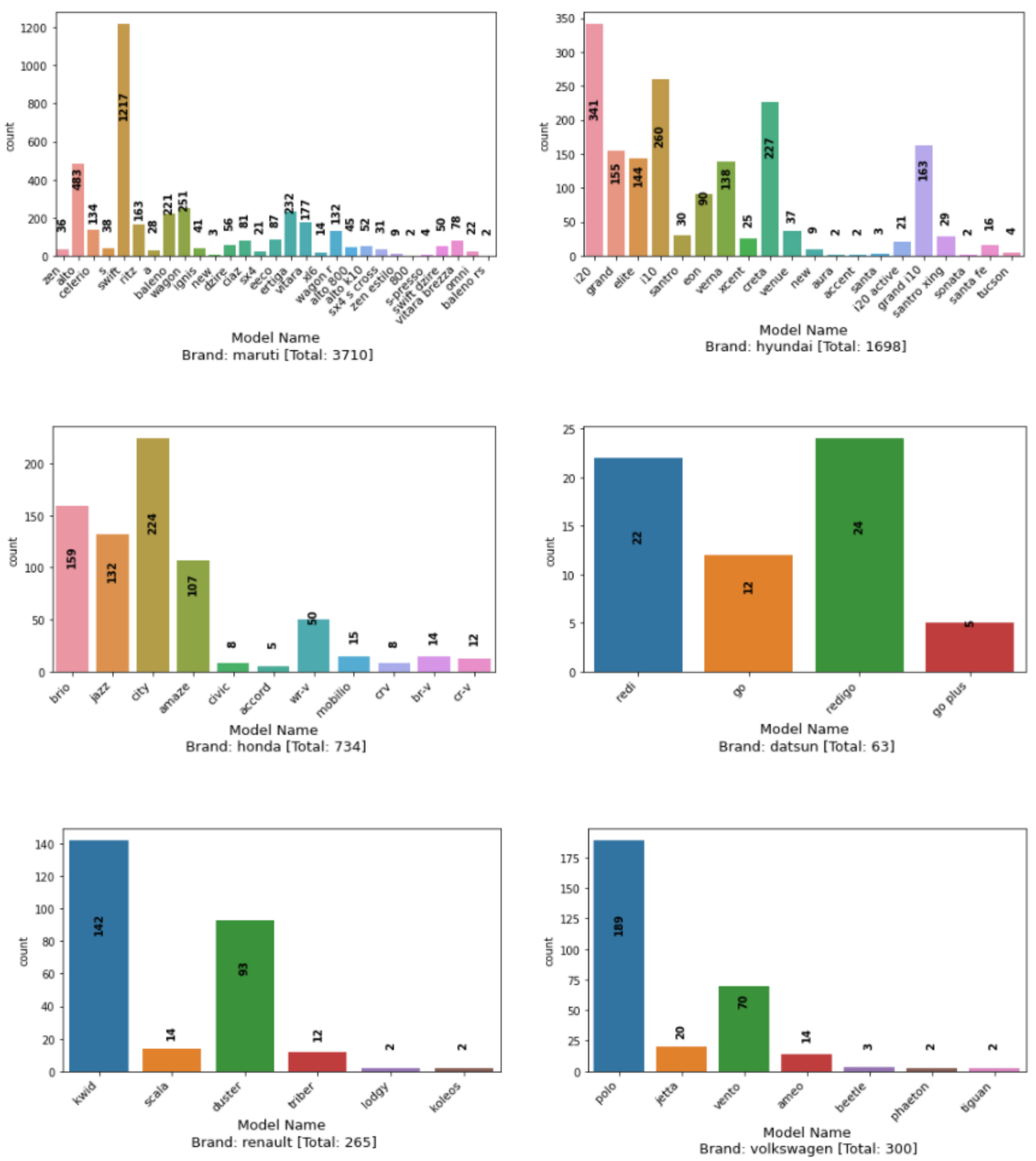


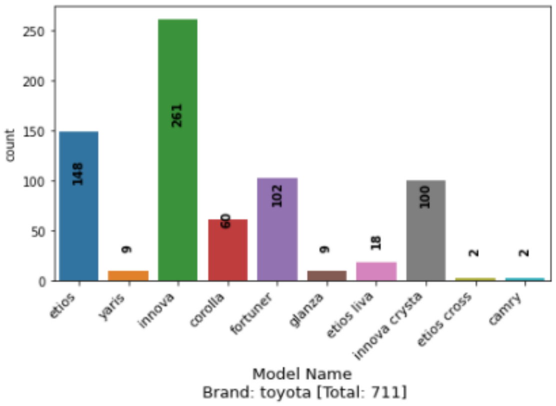
Remarks:

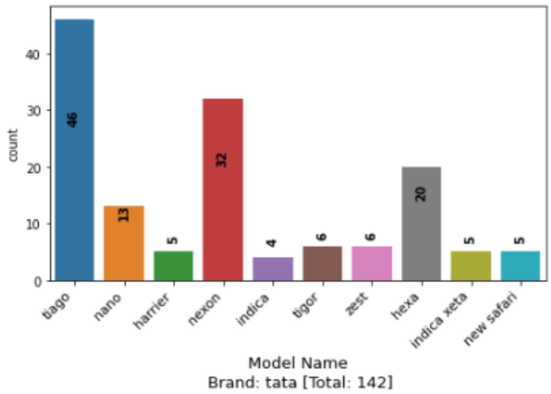
- Most of the records are of brand **maruti, hyundai, honda, toyota, ford, mahindra & volkswagen**.

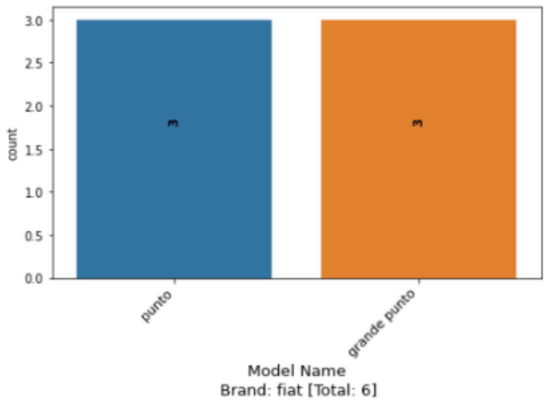
- Maximum number of records are of brand **maruti**.
- Minimum number of records are of brand **isuzu**.

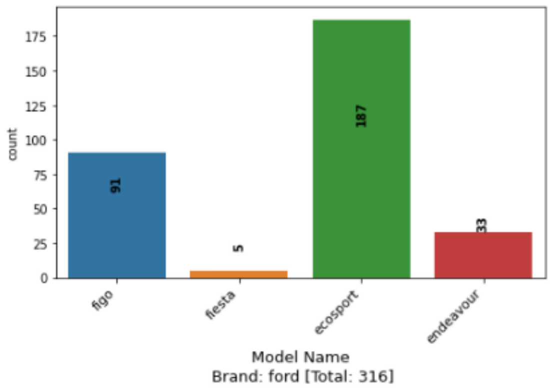
Count Plot for each Brand with their respective Model

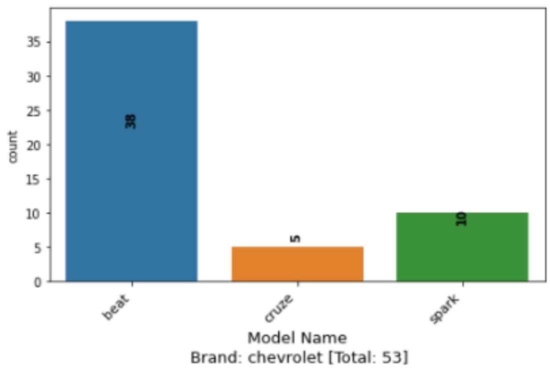


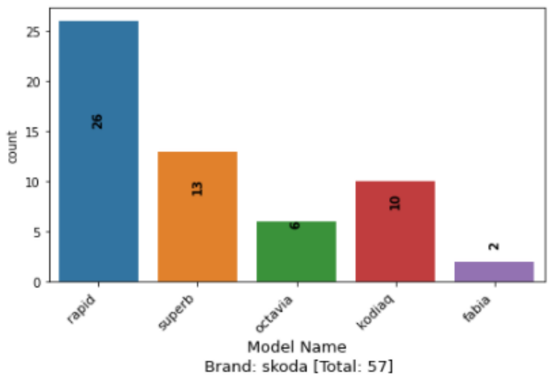

count
148
9
261
60
102
9
18
100
2
2
etios
yaris
innova
corolla
fortuner
glanza
etios liva
innova crysta
etios cross
camry
Model Name
Brand: toyota [Total: 711]

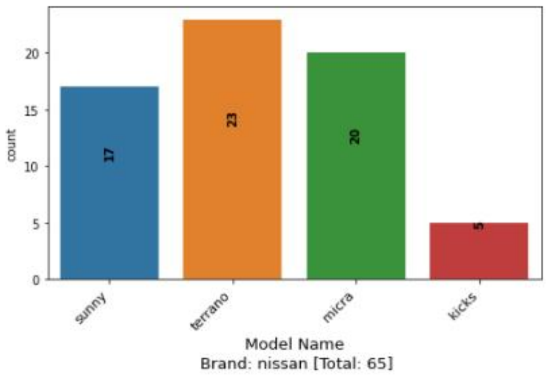

count
46
13
5
32
4
6
6
20
5
5
tiago
nano
harrier
nexon
indica
tigor
zest
hexa
indica xeta
new safari
Model Name
Brand: tata [Total: 142]

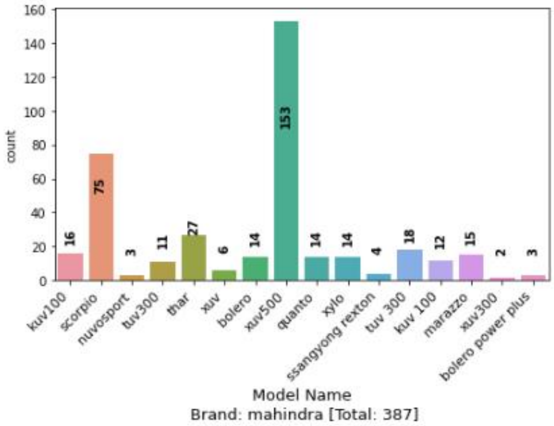

count
3
3
punto
grande punto
Model Name
Brand: fiat [Total: 6]

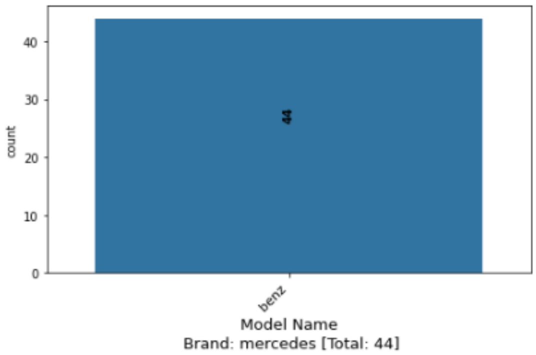

count
91
5
187
33
figo
fiesta
ecosport
endeavour
Model Name
Brand: ford [Total: 316]

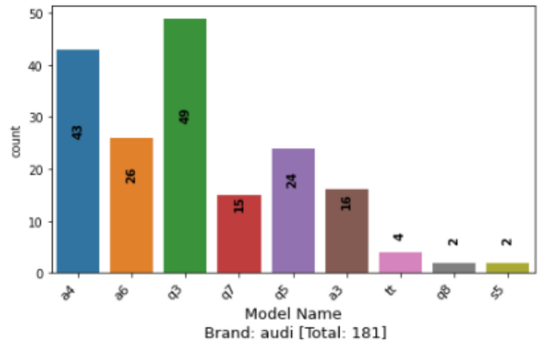

count
38
5
10
beat
cruze
spark
Model Name
Brand: chevrolet [Total: 53]

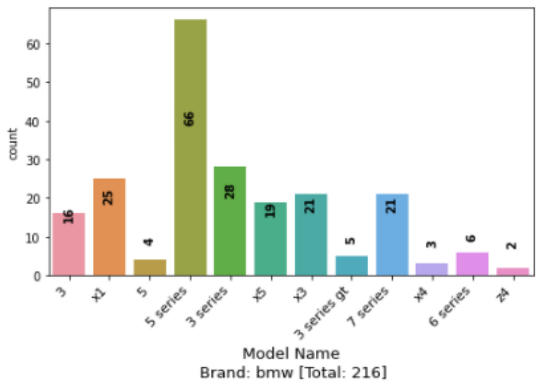

count
26
13
6
10
2
rapid
superb
octavia
kodiaq
fabia
Model Name
Brand: skoda [Total: 57]

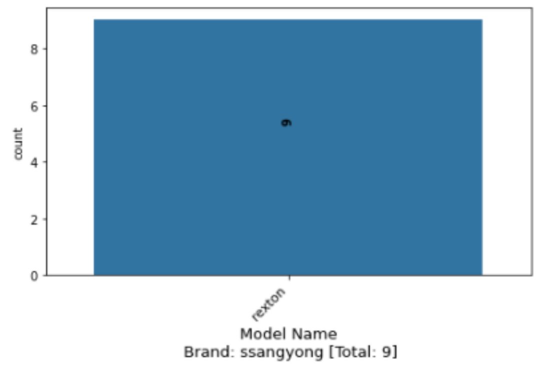

count
17
23
20
5
sunny
terrano
micra
kicks
Model Name
Brand: nissan [Total: 65]

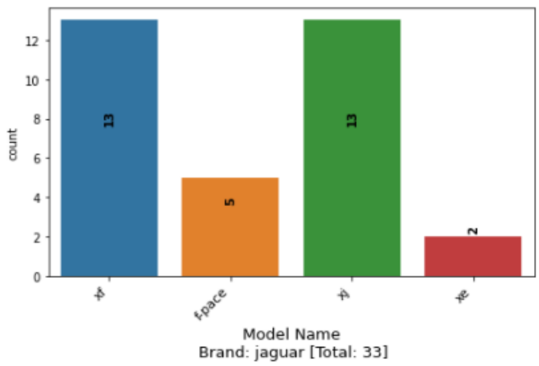

count
16
75
3
11
27
6
14
153
14
14
4
18
12
15
2
3
kuv100
scorpio
nuvosport
tuv300
thar
xuv
bolero
xuv500
quanto
xylo
ssangyong rexton
tuv 300
kuv 100
marazzo
xuv300
bolero power plus
Model Name
Brand: mahindra [Total: 387]

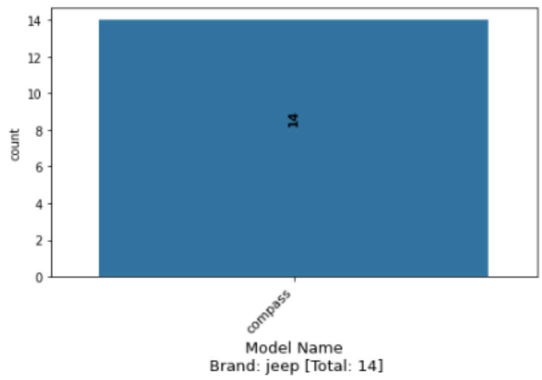

count
44
benz
Model Name
Brand: mercedes [Total: 44]

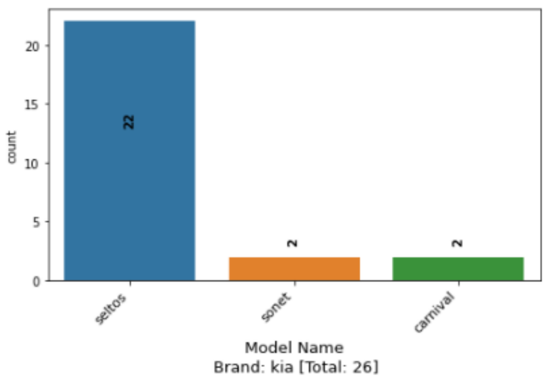

count
43
26
49
15
24
16
4
2
2
a4
a6
q3
q7
q5
a3
tt
q8
s5
Model Name
Brand: audi [Total: 181]

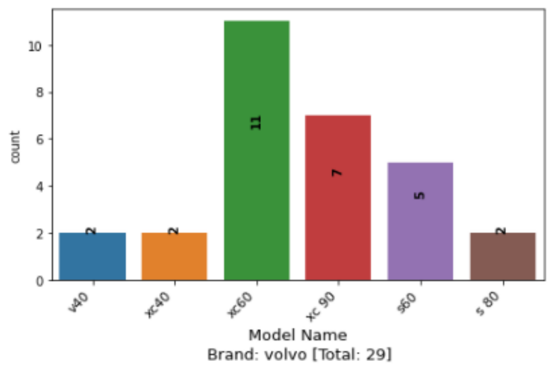

count
16
25
4
66
28
19
21
5
21
3
6
2
3
x1
5
5 series
3 series
x5
x3
3 series gt
7 series
x4
6 series
z4
Model Name
Brand: bmw [Total: 216]

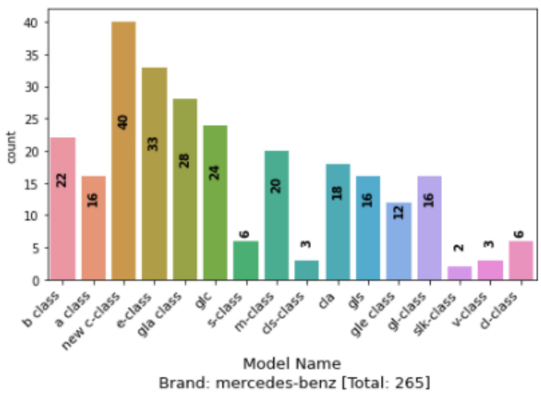

count
9
rexton
Model Name
Brand: ssangyong [Total: 9]

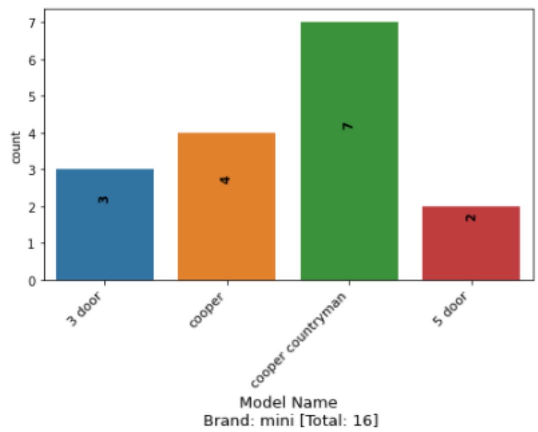

count
13
5
13
2
xf
f-pace
xj
xe
Model Name
Brand: jaguar [Total: 33]

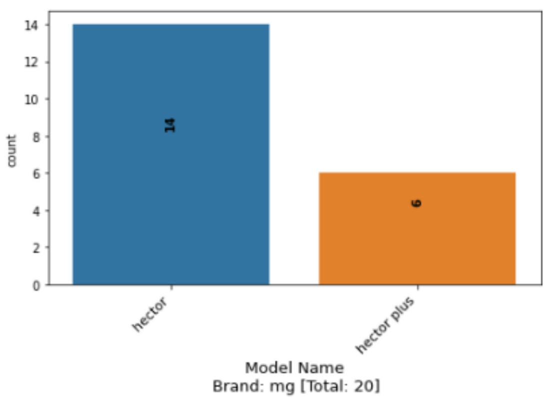

count
14
compass
Model Name
Brand: jeep [Total: 14]

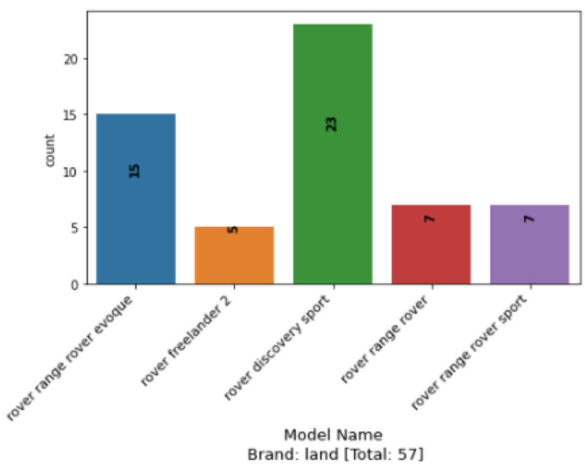

count
22
2
2
seltos
sonet
carnival
Model Name
Brand: kia [Total: 26]

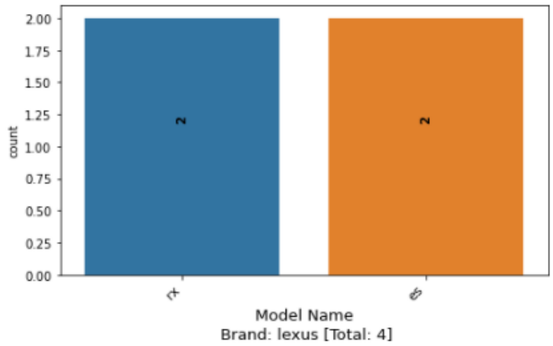

count
2
2
11
7
5
2
v40
xc40
xc60
xc 90
s60
s 80
Model Name
Brand: volvo [Total: 29]

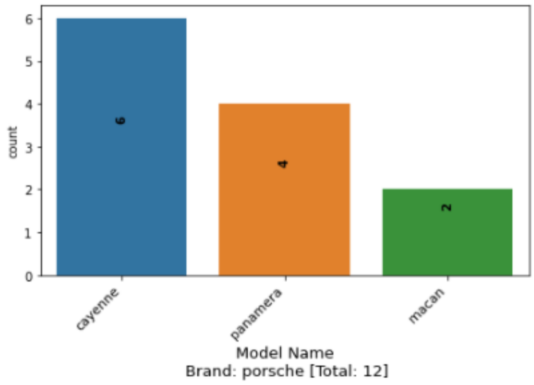

count
22
16
40
33
28
24
6
20
3
18
16
12
16
2
3
6
b class
a class
new c-class
e-class
gla class
glc
s-class
m-class
cls-class
cla
gls
gle class
gl-class
slk-class
v-class
cl-class
Model Name
Brand: mercedes-benz [Total: 265]

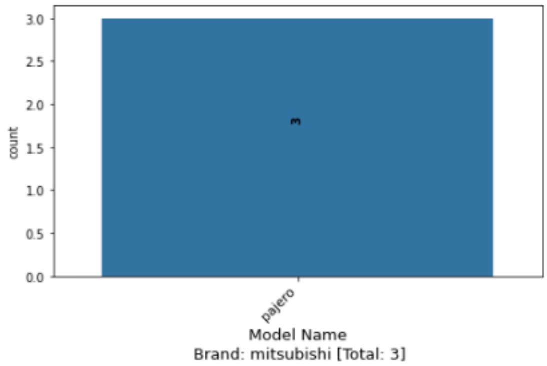

count
3
4
7
2
3 door
cooper
cooper countryman
5 door
Model Name
Brand: mini [Total: 16]

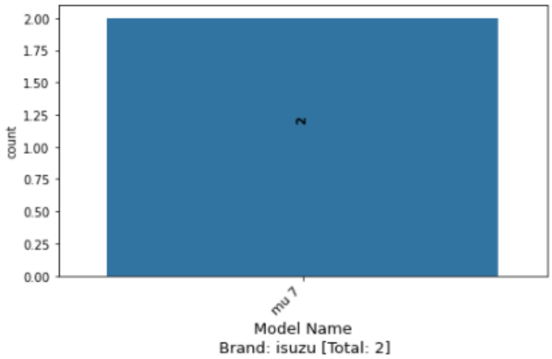

count
14
6
hector
hector plus
Model Name
Brand: mg [Total: 20]

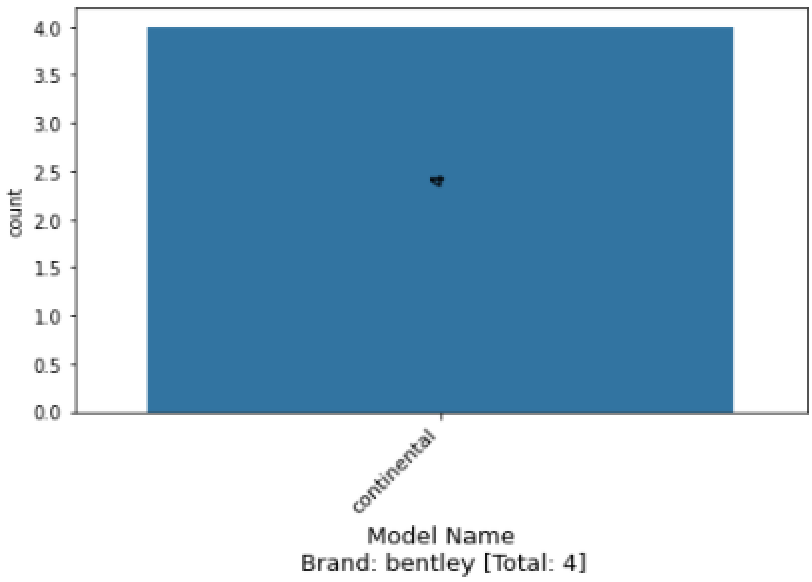

count
15
5
23
7
7
rover range rover evoque
rover freelander 2
rover discovery sport
rover range rover
rover range rover sport
Model Name
Brand: land [Total: 57]











Remarks:

1. `for brand maruti:`
    - Most of the records are for model **swift, alto, wagon, ertiga, baleno, vitara, ritz, celerio & wagon r**.
    - Maximum number of records are for model **swift**.
    - Minimum number of records are for model **800 & baleno rs**.

2. `for brand hyundai:`
    - Most of the records are for model **i20, i10, creta, grand i10, grand, elite & verna**.
    - Maximum number of records are for model **i20**.
    - Minimum number of records are for model **aura, accent & sonata**.

3. `for brand honda:`
    - Most of the records are for model **city, brio, jazz & amaze**.
    - Maximum number of records are for model **city**.
    - Minimum number of records are for model **accord**.

4. `for brand datsun:`
    - Most of the records are for model **redigo & redi**.
    - Maximum number of records are for model redigo
    - Minimum number of records are for model **go plus**.

5. `for brand renault:`
    - Most of the records are for model ** kwid & duster **.
    - Maximum number of records are for model ** kwid **.
    - Minimum number of records are for model ** lodgy & koleos **.

5. `for brand volkswagen:`
    - Most of the records are for model ** polo & vento **.
    - Maximum number of records are for model ** polo **.
    - Minimum number of records are for model ** phaeton & tiguan **.

6. `for brand toyota:`
    - Most of the records are for model ** innova, etios, fortuner & innova crysta **.
    - Maximum number of records are for model ** innova **.
    - Minimum number of records are for model ** etios cross & camry **.

7. `for brand tata:`
    - Most of the records are for model ** tiago, nexon & hexa **.
    - Maximum number of records are for model ** tiago**.
    - Minimum number of records are for model ** indica **.

8. `for brand fiat:`
    - Equal number of records are there for all model.

9. `for brand ford:`
    - Most of the records are for model ** ecosport & figo **.
    - Maximum number of records are for model ** ecosport **.
    - Minimum number of records are for model ** endeavour **.

10. `for brand chevrolet:`
    - Most of the records are for model ** beat & spark **.
    - Maximum number of records are for model ** beat **.
    - Minimum number of records are for model ** cruze **.

11. `for brand skoda:`
    - Most of the records are for model ** rapid, superb & kodiaq **.
    - Maximum number of records are for model ** rapid **.
    - Minimum number of records are for model ** fabia **.

12. **`for brand nissan:`**
    - Most of the records are for model ** terrano, micra & sunny **.
    - Maximum number of records are for model ** terrano **.
    - Minimum number of records are for model ** kicks **.

13. **`for brand mahindra:`**
    - Most of the records are for model ** xuv500, scorpio & thar **.
    - Maximum number of records are for model ** xuv500 **.
    - Minimum number of records are for model ** xuv300 **.

14. **`for brand mercedes:`**
    - Has only one model named benz in the dataset.

15. **`for brand audi:`**
    - Most of the records are for model ** q3, a4, a6 & q5 **.
    - Maximum number of records are for model ** q3 **.
    - Minimum number of records are for model ** q8 & s5 **.

16. **`for brand bmw:`**
    - Most of the records are for model ** 5 series, 3 series, x1, x3 & 7 series **.
    - Maximum number of records are for model ** 5 series **.
    - Minimum number of records are for model ** z4 **.

17. **`for brand ssangyong:`**
    - Has only one model named rexton in the dataset.

18. **`for brand jaguar:`**
    - Most of the records are for model ** xf & xj **.
    - Maximum number of records are for model ** xf & xj**.
    - Minimum number of records are for model ** xe **.

19. **`for brand jeep:`**
    - Has only one model named compass in the dataset.

20. **`for brand kia:`**
    - Maximum number of records are for model ** seltos **.
    - Minimum number of records are for model ** sonet & carnival **.

21. **`for brand volvo:`**
    - Most of the records are for model ** xc60, xc 90 & s60 **.
    - Maximum number of records are for model ** xc60 **.
    - Minimum number of records are for model ** v40, xc40 & s 80 **.

22. **`for brand mercedes-benz:`**
    - Most of the records are for model ** new c-class, e-class, gla class, glc, b class, m-class & cla **.
    - Maximum number of records are for model ** new c-class **.
    - Minimum number of records are for model ** slk-class **.

23. **`for brand mini:`**
    - Maximum number of records are for model ** cooper countryman **.
    - Minimum number of records are for model ** 3 door **.

24. **`for brand mg:`**

- Maximum number of records are for model ** hector **.
- Minimum number of records are for model ** hector plus **.

25. `for brand land:`
    - Most of the records are for model ** rover discovery sport & rover range rover evoque **.
    - Maximum number of records are for model ** rover discovery sport**.
    - Minimum number of records are for model ** rover freelander 2 **.

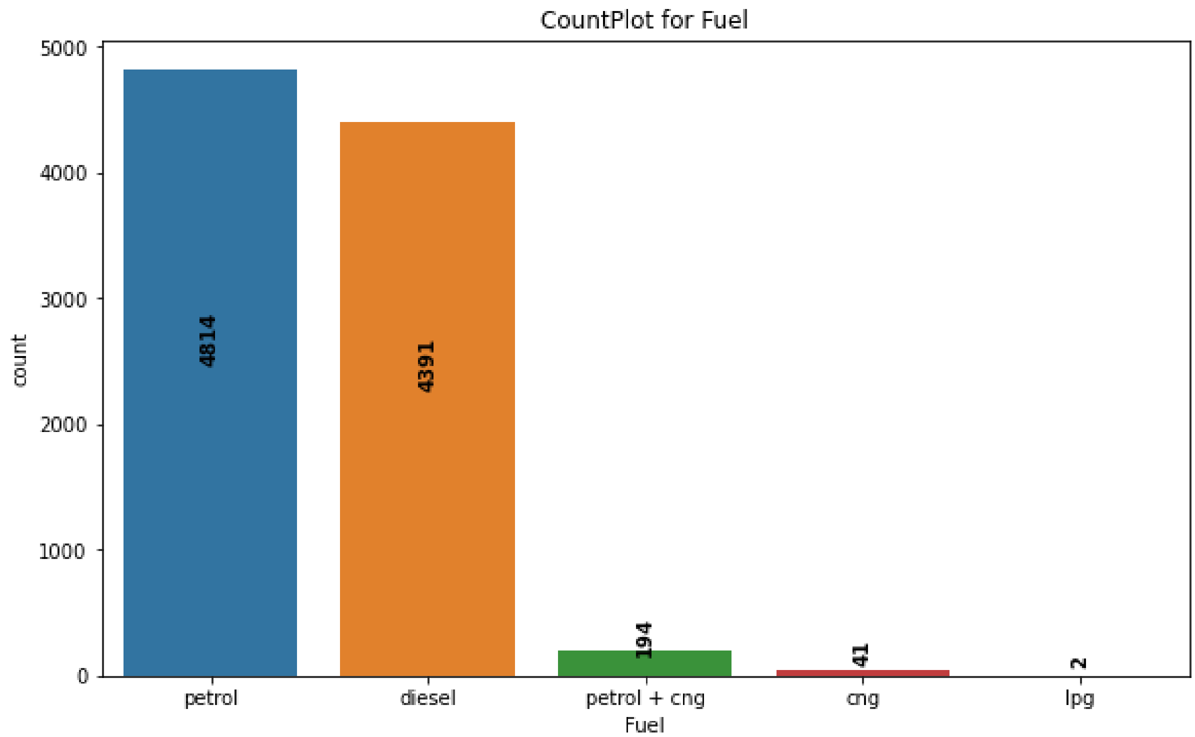
26. `for brand lexus:`
    - All models have equal number of records.

27. `for brand porsche:`
    - Maximum number of records are for model ** cayenne **.
    - Minimum number of records are for model ** macan **.

28. `for brand mitsubishi:`
    - Has only one model named pajero.

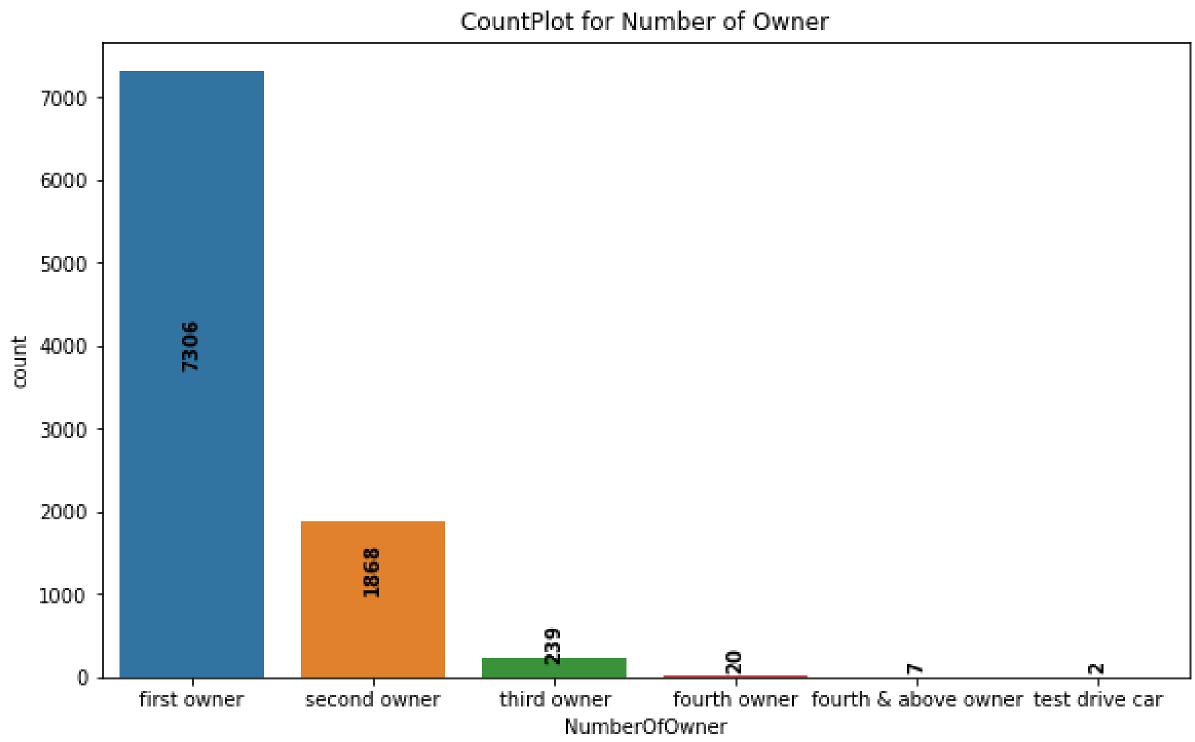
29. `for brand isuzu:`
    - Has only one model named mu 7.

30. `for brand bentley:`
    - Has only one model named continental.



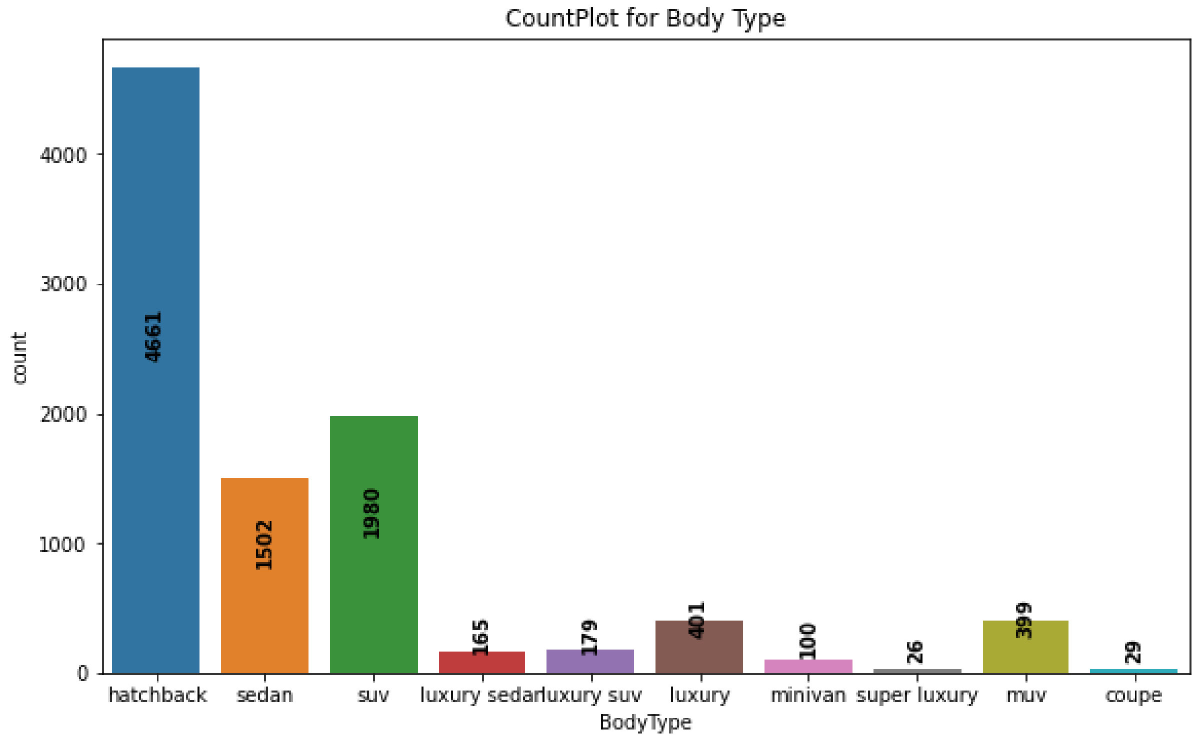
Remarks:

- Most of the records are for **petrol and diesel** engine type cars.
- Maximum number of cars are of **petrol** engine.
- Minimum number of cars are of **lpg** engine.



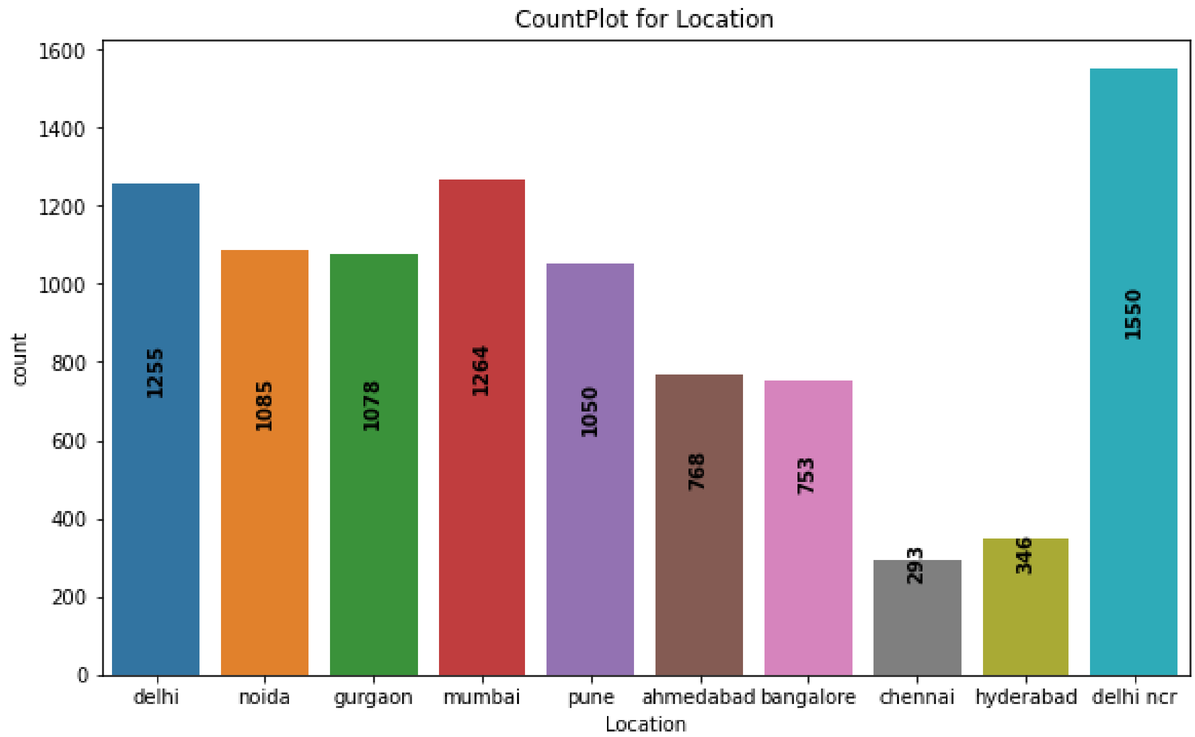
Remarks:

- Most of the records are for **first and second** owner.
- Maximum number of cars are of **first** owner.
- Minimum number of cars are of **test driver**.



Remarks:

- Most of the records are for **hatchback, suv, sedan, luxury & muv**.
- Maximum number of cars are of **hatchback** owner.
- Minimum number of cars are of **super luxury**.

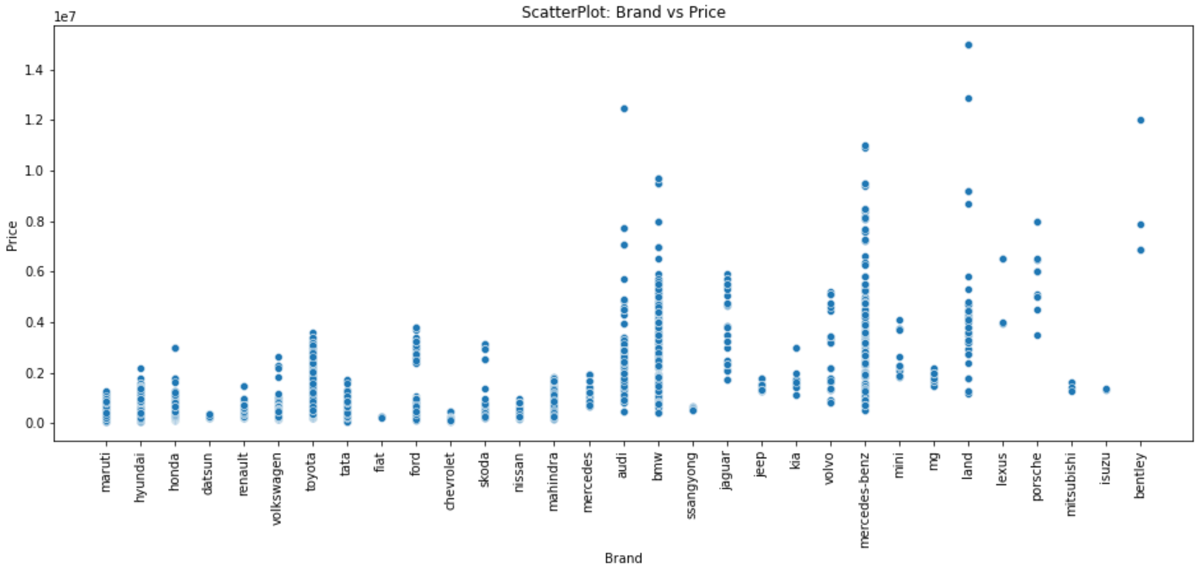


Remarks:

- Most of the records are for **delhi ncr, delhi, mumbai, noida, gurgaon & pune**.
- Maximum number of cars are of **delhi ncr**.
- Minimum number of cars are of **chennai**.

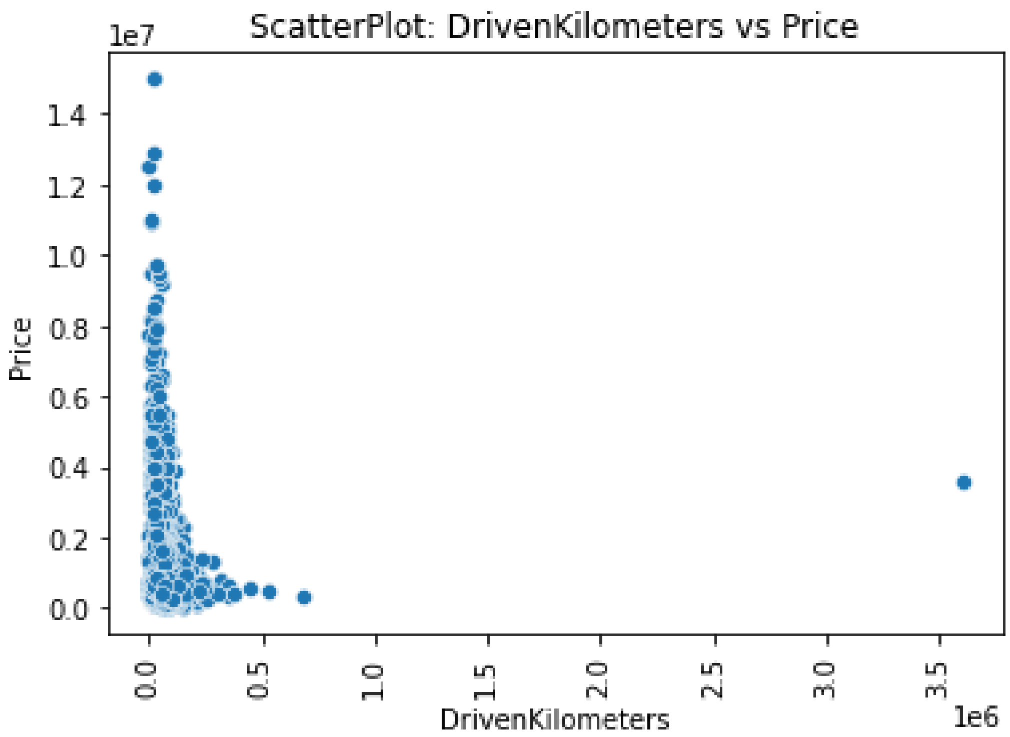
2. Bivariate Analysis: Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them.

   Scatterplot:



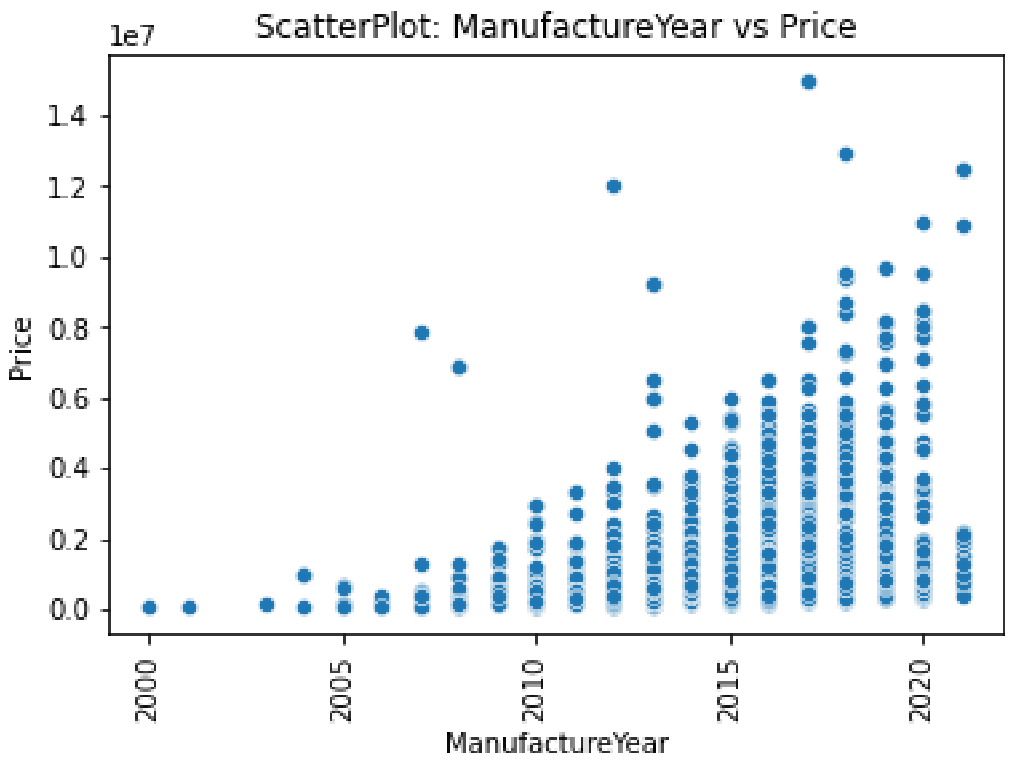
Remarks:

- Most of the car price ranges from 40000 to 2000000.
- **Maruti** brand car **price ranges from 40000 to 1500000** (aproximately).
- **Hyundai** brand car **price ranges from 40000 to 2200000** (aproximately).
- **Honda** brand car **price ranges from 40000 to 3000000** (aproximately).
- **Datsun** brand car **price ranges from 100000 to 500000** (aproximately).
- **Renault** brand car **price ranges from 80000 to 1800000** (aproximately).
- **Volkswagen** brand car **price ranges from 100000 to 2500000** (aproximately).
- **Toyota** brand car **price ranges from 100000 to 3800000** (aproximately).
- **Tata** brand car **price ranges from 40000 to 1800000** (aproximately).
- **Fiat** brand car **price ranges from 400000 to 500000** (aproximately).
- **Ford** brand car **price ranges from 300000 to 4000000** (aproximately).
- **Chevrolet** brand car **price ranges from 100000 to 500000** (aproximately).
- **Skoda** brand car **price ranges from 400000 to 3500000** (aproximately).
- **Nissan** brand car **price ranges from 400000 to 1200000** (aproximately).
- **Mahindra** brand car **price ranges from 400000 to 1800000** (aproximately).
- **Mercedes** brand car **price ranges from 1000000 to 2000000** (aproximately).
- **Audi** brand car **price ranges from 400000 to 13000000** (aproximately).
- **BMW** brand car **price ranges from 400000 to 10000000** (aproximately).
- **ssangyong** brand car **price ranges from 400000 to 500000** (aproximately).
- **Jaguar** brand car **price ranges from 1200000 to 5800000** (aproximately).
- **Jeep** brand car **price ranges from 1000000 to 1500000** (aproximately).
- **Kia** brand car **price ranges from 1000000 to 2400000** (aproximately).
- **Volvo** brand car **price ranges from 8000000 to 4500000** (aproximately).
- **Mercedes-Benz** brand car **price ranges from 600000 to 12000000** (aproximately).
- **Mini** brand car **price ranges from 1800000 to 3800000** (aproximately).
- **MG** brand car **price ranges from 1200000 to 2000000** (aproximately).
- **Land** brand car **price ranges from 1000000 to 16000000** (aproximately).
- **Lexus** brand car **price ranges from 4000000 to 6200000** (aproximately).
- **Porsche** brand car **price ranges from 3000000 to 7200000** (aproximately).
- **Mitsubishi** brand car **price ranges from 1200000 to 1500000** (aproximately).
- **Isuzu** brand car **price ranges from 1200000 to 1400000** (aproximately).
- **Bentley** brand car **price ranges from 6000000 to 12000000** (aproximately).



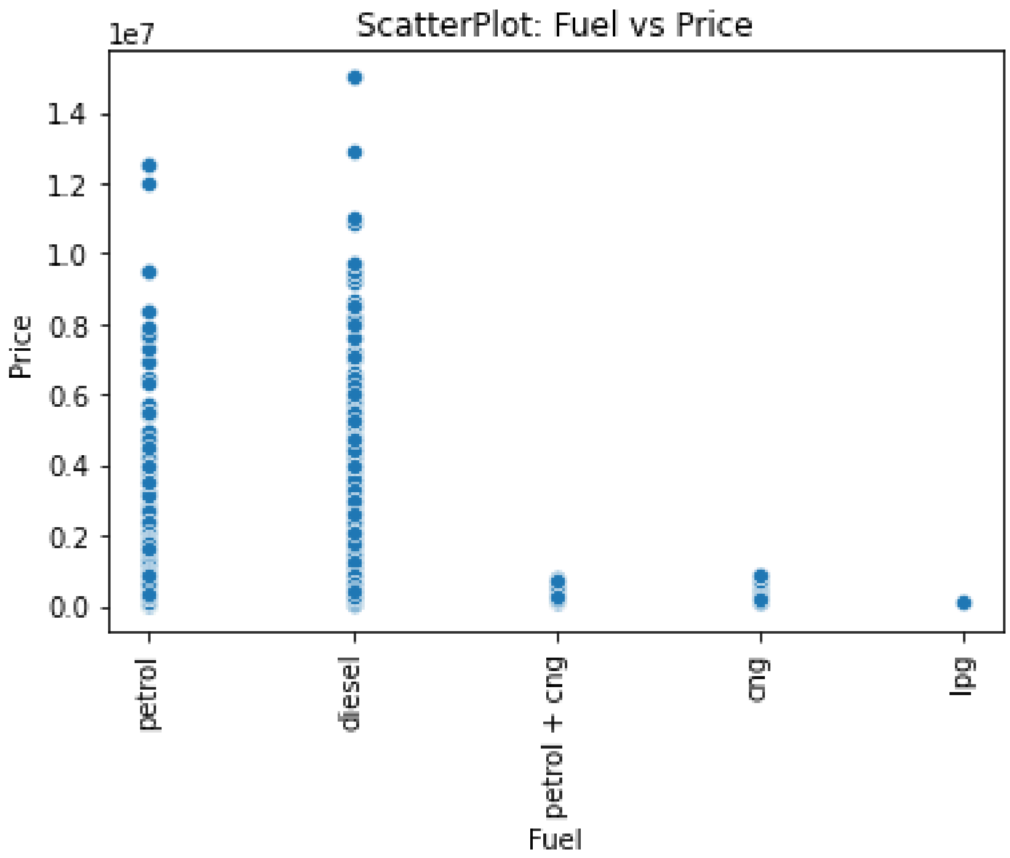
Remarks:

- **Price decreases as the Driven Kilometers increases.**



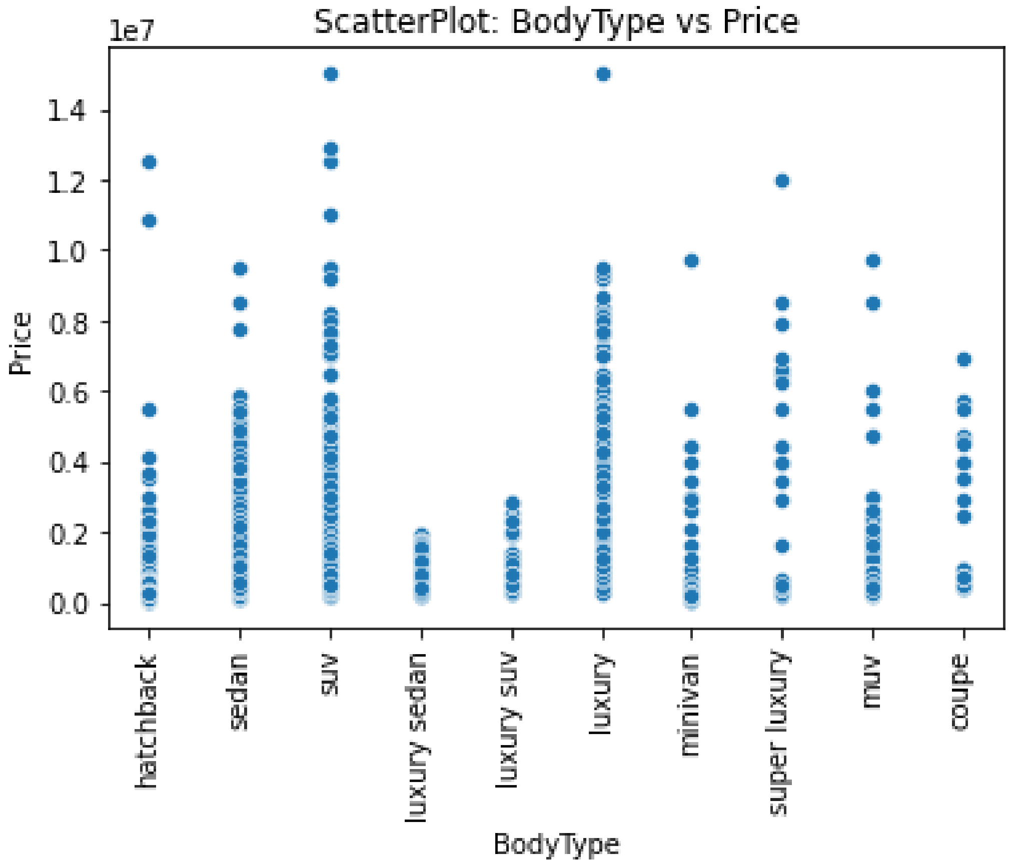
Remarks:

- **Price increases as the number of year increases, i.e., newer the car higher the price.**



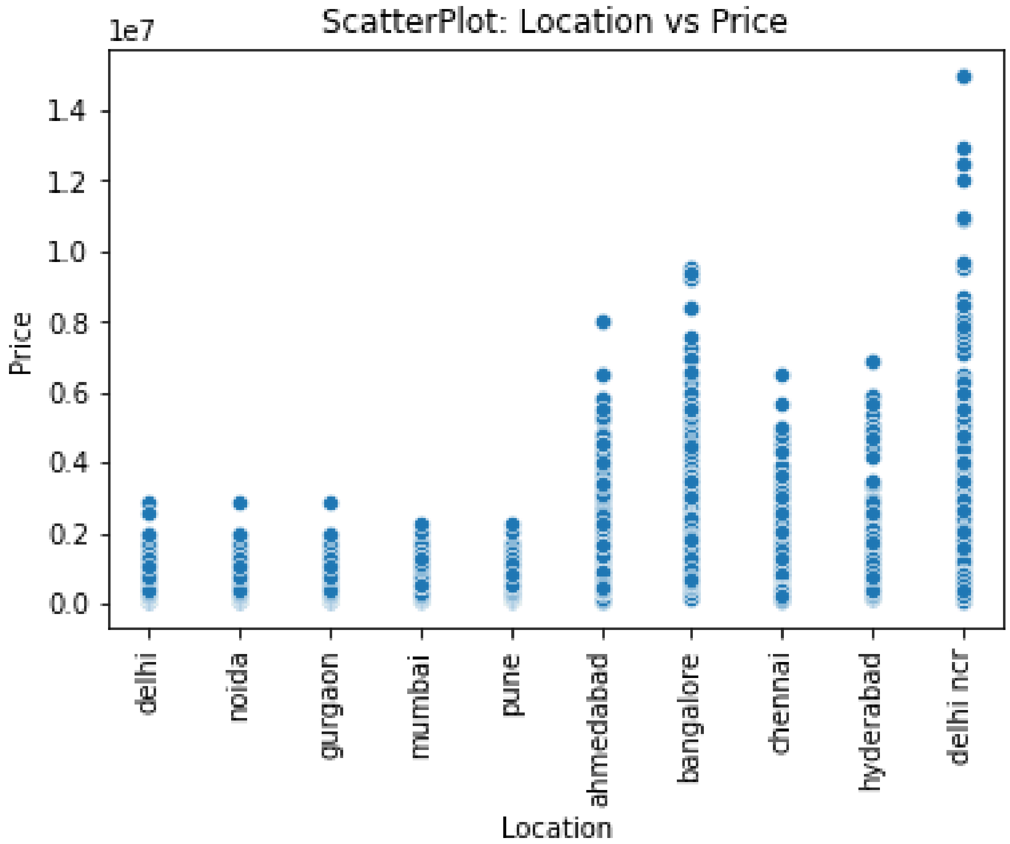
Remarks:

- Price of **petrol and diesel cars are higher** as compared to the price of petrol+cng, cng and lpg.



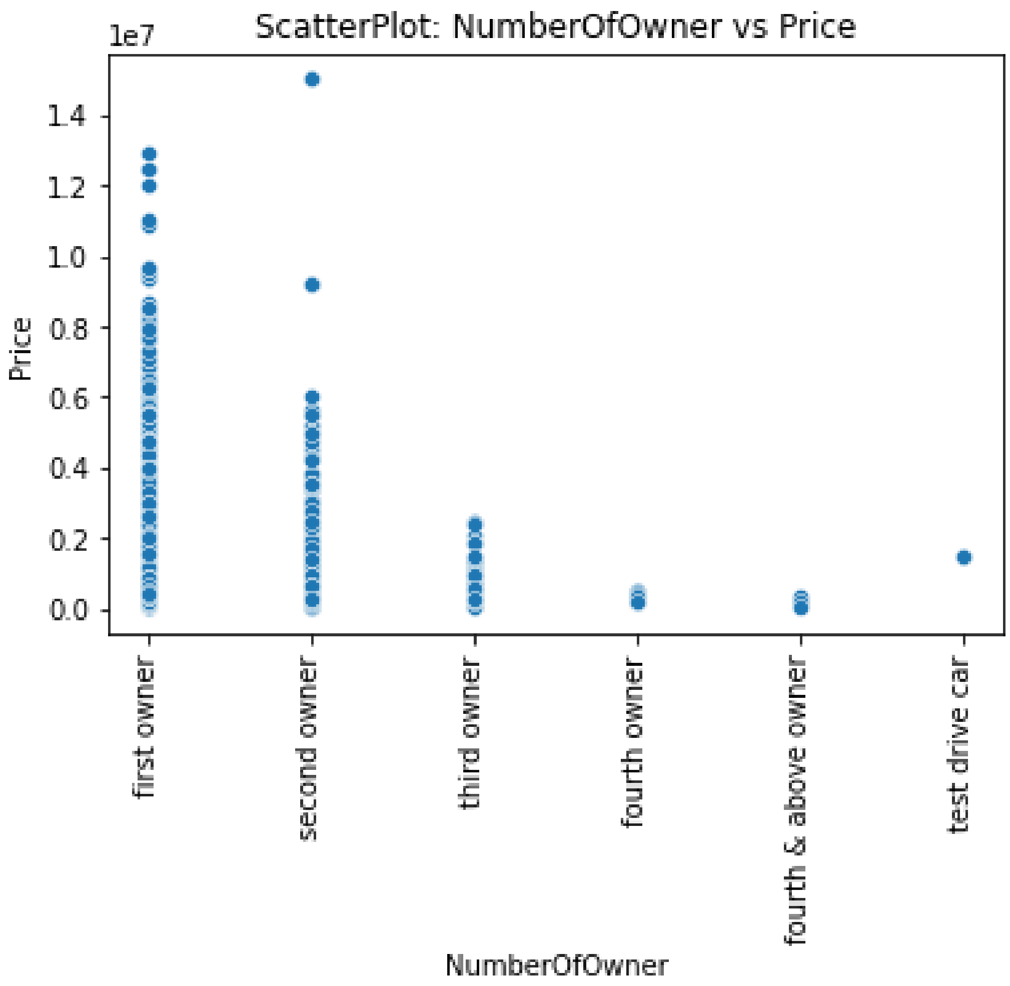
Remarks:

- Price of **SUV and Luxury are on higher side** as compared to others while Price of luxury sedan and luxury suv are on lower side.



Remarks:

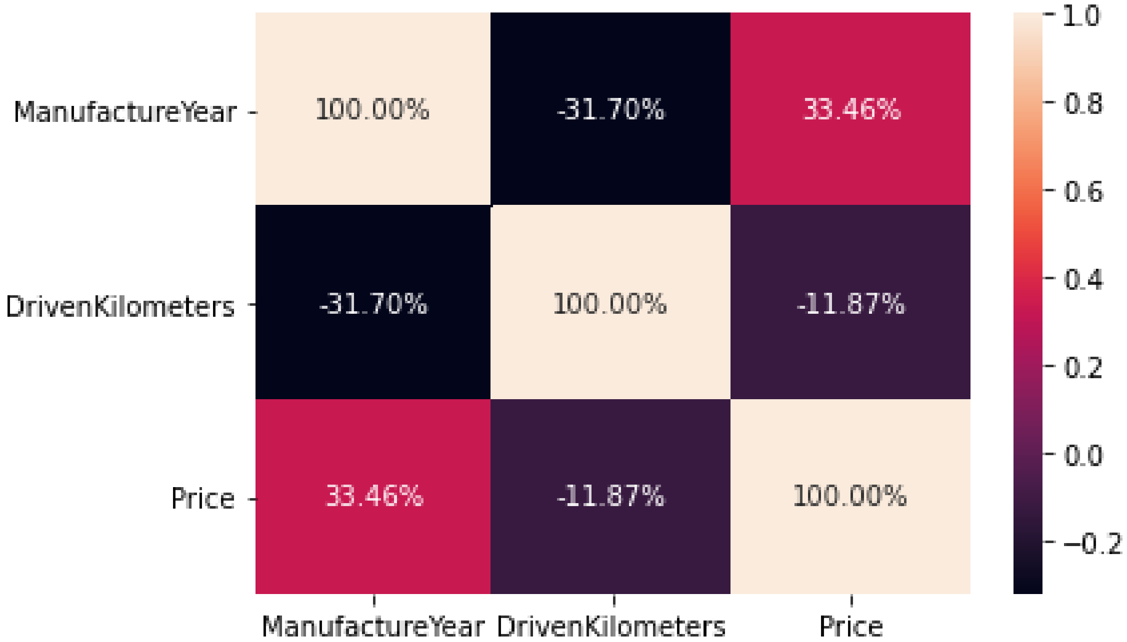
- Price of car in delhi ncr ranges from as low as 40000 to as high as 13000000. Similerly in bangalore it ranges from 80000 to 9000000, in ahmedabad it ranges from 40000 to 8000000, in hyderabad it ranges from 80000 to 7000000 and in chennai it ranges from 80000 to 6000000 while in all other location it ranges from 40000 to 3000000.



Remarks:

- **As the number of owner increases, price of car decreases.**

3. Multivariate Analysis: Multivariate statistics is a subdivision of statistics encompassing the simultaneous observation and analysis of more than one outcome variable.



Remarks:

- **Manufacture Year is positively good correlated to Price**.
- **Driven Kilometers is negatively correlated to Price**.

- Interpretation of the Results

Starting with univariate analysis, with the help of distplot, it was found that the minimum price of a car in the dataset is 40,000 while maximum is 1.5 Crore. It was also found that the data is positively skewed in Driven Kilometers and in Manufacture Year, the oldest car is of year 2000 and the newest car is of year 2021. Moving further with countplot, it was found that the most of the records in the dataset are of Maruti, Hyundai, Honda, Toyota, Ford, Mahindra & Volkswagen. Also, the maximum number of records are of Maruti and minimum numbers are for Isuzu. Also, with bivariate analysis, with the help of scatterplot, it was found that Maruti brand car price ranges from 40000 to 1500000 (aproximately), Hyundai brand car price ranges from 40000 to 2200000 (aproximately), Honda brand car price ranges from 40000 to 3000000 (aproximately), Datsun brand car price ranges from 100000 to 500000 (aproximately), Renault brand car price ranges from 80000 to 1800000 (aproximately), Volkswagen brand car price ranges from

100000 to 2500000 (aproximately), Toyota brand car price ranges from 100000 to 3800000 (aproximately), Tata brand car price ranges from 40000 to 1800000 (aproximately), Fiat brand car price ranges from 400000 to 500000 (aproximately), Ford brand car price ranges from 300000 to 4000000 (aproximately), Chevrolet brand car price ranges from 100000 to 500000 (aproximately), Skoda brand car price ranges from 400000 to 3500000 (aproximately), Nissan brand car price ranges from 400000 to 1200000 (aproximately), Mahindra brand car price ranges from 400000 to 1800000 (aproximately), Mercedes brand car price ranges from 1000000 to 2000000 (aproximately), Audi brand car price ranges from 400000 to 13000000 (aproximately), BMW brand car price ranges from 400000 to 10000000 (aproximately), Ssangyong brand car price ranges from 400000 to 500000 (aproximately), Jaguar brand car price ranges from 1200000 to 5800000 (aproximately), Jeep brand car price ranges from 1000000 to 1500000 (aproximately), Kia brand car price ranges from 1000000 to 2400000 (aproximately), Volvo brand car price ranges from 8000000 to 4500000 (aproximately), Mercedes-Benz brand car price ranges from 600000 to 12000000 (aproximately), Mini brand car price ranges from 1800000 to 3800000 (aproximately), MG brand car price ranges from 1200000 to 2000000 (aproximately), Land brand car price ranges from 1000000 to 16000000 (aproximately), Lexus brand car price ranges from 4000000 to 6200000 (aproximately), Porsche brand car price ranges from 3000000 to 7200000 (aproximately), Mitsubishi brand car price ranges from 1200000 to 1500000 (aproximately), Isuzu brand car price ranges from 1200000 to 1400000 (aproximately) and Bentley brand car price ranges from 6000000 to 12000000 (aproximately). It was also found that as the driven kilometers increases, price of car decreases and as the number of years increases, price also increases. Also, price of petrol and diesel cars are higher as compared to the price of petrol+cng, cng and lpg. Price of SUV and Luxury are on higher side. Price of car decreases as the number of owner increases. With the help of multivariate analysis using heatmap, it was found that Manufacture Year is positively good correlated to price while driven kilometers are negatively correlated to price.

CONCLUSION

- Key Findings and Conclusions of the Study

Final model **KNeighborsRegressor** performs better with **R2 Score: 90.76% and Cross Val Score: 84.31%** and can further be improved by training with more specific data.

- Learning Outcomes of the Study in respect of Data Science

During the data analysis, I have considered Brand, Model and Varient but one can also proceed with only Varient by dropping Brand and Model which will reduced the number of features at the time of feature encoding, resulting in less training time, might impact the model performance either in positive or negative way. As of now, I am finishing this project with my current approach which gives the **final R2 Score of 90.76% and Cross Val Score: 84.31%** and this can be further improved by training with more specific data.

- Limitations of this work and Scope for Future Work

Current model is limited to used car data but this can further be improved for other sectors of automobiles by training the model accordingly. The overall score can also be improved further by training the model with more specific data.