PREDCTING PRICE OF PRE- OWNED CARS

A Comparison of Machine Learning Regression Models

Tanushka Raj
Vaishnavi
Internship project
Field of study: Application development in data analysis using python
Semester/year: 5/3
Guide: Ms Ramya
Programme: Computer Science and Engineering

Abstract

The automotive industry has witnessed a surge in the pre-owned car market, making accurate pricing crucial for both sellers and buyers. This study employs data analysis techniques and machine learning algorithms implemented in Python to predict the prices of pre-owned cars. The dataset comprises diverse attributes such as make, model, year of manufacture, mileage, fuel type, and various other features that influence car pricing. The analysis begins with exploratory data analysis (EDA) to gain insights into the dataset's structure and identify patterns. Next, a machine learning model is developed using Python's popular libraries, such as NumPy, Pandas, Seaborn, Matplotlib etc. The dataset is split into training and testing sets, and various regression algorithms, including linear regression and ensemble methods are applied and compared. The Python-based approach ensures transparency, reproducibility, and scalability, making it applicable for large-scale datasets and industry-wide implementations.

Keywords: Exploratory data analysis, Feature engineering, Regression algorithms, Reproducibility, Scalability

1.Introduction

This chapter is dedicated to providing the reader with a comprehensive understanding of the background, the motivation behind the problem at hand, and the overall purpose and significance of the research outlined in this report.

1.1.Background and motivation

The background for conducting data analysis using Python to predict the prices of pre-owned cars lies in the growing significance of the pre-owned car market. As more consumers opt for used vehicles, understanding and accurately predicting their prices become crucial for both sellers and buyers.

The motivation for addressing this problem stems from the challenges associated with determining fair and competitive prices in the dynamic pre-owned car market. Factors such as the make, model, year, mileage, and various other features influence the pricing, creating a complex landscape for sellers and buyers. Traditional methods often fall short in capturing these intricacies, emphasizing the need for a data-driven approach.

Machine learning (ML) is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to make useful inferences from data. Machine learning algorithms are well suited to problems entailing large amounts of data which would not be possible to process without such algorithms. ML works algorithmically rather than mathematically and permit a machine to "learn" and adapt its predictions to best fit the data it has trained on. [1]

1.2.Overall aim

This project aims build a Linear Regression and Random Forest model on two sets of data:

1. Data obtained by omitting rows with missing values.
2. Data obtained by imputing missing values.

To assess these machine learning models in predicting the prices of used cars and draw insights into their behavior. The goal is to enhance understanding of how machine learning can be effectively employed in valuing cars and similar price prediction challenges.

1.3.Problem statement

Storm Motors is an E-Commerce Company who act as mediators between parties interested in selling and buying pre-owned cars. They have recorded data about the seller and car details, registration retails, web advertisement details, make and model information and price. The company wishes to develop an algorithm to predict the price of pre-owned cars based on various attributes associated with the car.

1.4.Research question

The research questions that this study will answer is:
Which ML model and parameters gives the best overall accuracy in making price predictions for used cars?

1.5.Scope

This study will focus on answering specific research questions that involve comparing different machine learning algorithms for predicting car prices. To achieve this, we will gather and prepare a dataset that allows a fair comparison among these algorithms. It's important that the selected algorithms are similar enough to use the same dataset for training and comparison. Our goal is to compare these algorithms on an equal playing field rather than maximizing the performance of any single algorithm without improving comparability.

2.Theory

In this chapter, the reader can explore important theories and research connected to the study. This includes understanding how regression learning works, the metrics used to measure how well the models perform etc.

2.1. Linear Regression

Linear Regression is a technique to estimate the linear relationship between each of a number of independent variables and a dependent variable. Linear Regression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. [2]

2.2. Random Forest Regression

Random Forest is an ensemble learning technique for classification and regression tasks. The algorithm makes use of Decision Trees. They consist of a set of independent binary trees, each stochastically trained on random subsets of data. Although these trees individually may be overstrained, the randomness in the process of training results in the trees producing independent estimates, which are then combined to produce a result. Random Forests have been shown to be effective in a wide range of classification and regression problems. The generalization error for forests converges asymptotically to a limit as the number of trees in the forest becomes large. The generalization error of a forest of Decision Tree Regressors depends on the strength of the individual trees in the forest and the correlation between them [3]. Random Forest Regression is a stochastic process in that each tree is trained on a random subset of data, meaning that the algorithm will behave differently each time it is trained. The algorithm therefore combines the results of many Decision Trees utilizing regression.

3.Methodology

To address the task at hand, we implemented a Python-based solution for data analysis and prediction of car prices. This involved a dataset containing information on 50,000 cars, including details like brand, model, and other relevant features. The dataset underwent preprocessing to ensure its cleanliness and removal of unnecessary information.

Subsequently, we conducted a comprehensive analysis of the data using graphical representations such as graphs and charts.. An emphasis was placed on identifying and handling outliers that could influence the accuracy of our predictions.

To facilitate prediction, we implemented two distinct methods: Linear Regression and Random Forest. These machine learning techniques were applied to forecast car prices based on their characteristics, and a comparative analysis was conducted to evaluate their respective effectiveness.

The ultimate objective of this effort is to provide valuable insights for potential car buyers. By examining and contrasting the predictions generated by our models, buyers can make informed decisions about the relative value of different cars.

4.Data set description

The car data set used in this research were collected from NPTEL website. This dataset consists of 50,001 car observations and the 19 attributes of pre-owned car are from an e-commerce site (Storm Motors). These datasets may contain a significant number of pre-owned cars information with several presumably requiring some tweaking and engineering. For example, duplicated observations can affect model performance and must be removed in advance [5].

A descriptive statistic of categorical variables is shown. Technically, attributes such as dateCrawled, lastSeen, postal-code, and dateCreated have no effect on price prediction, hence are removed to improve model performance. [6] Since their values are highly unbalanced, attributes such as seller, offerType and abtest were also removed with the data preparation process by inspecting more detail on dataset. Finally, the name was removed as well, because it contains too many unique values. [4]

5.EDA

5.1. Importing necessary libraries and reading the csv file

In [1]:
```
#importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
#setting dimension for plots
sns.set(rc={'figure.figsize':(9,6)})
```

In [4]:
```
#read cvs file
cars_data=pd.read_csv('cars_sampled.csv')
```

In [5]:
```
#creating copy
cars=cars_data.copy()
```

Importing these libraries enables to gain access to a comprehensive set of tools for data manipulation (pandas and numpy) and data visualization (seaborn and matplotlib). This combination is often used in data science and analysis workflows to efficiently process and explore data and create visualizations for better insights. The dataset cars_sampled is read into the notebook.

5.2. Cleaning and processing data

```
In [11]: #data cleaning
         #no.of missing values in each column
         cars.isnull().sum()

Out[11]: seller                   0
         offerType                0
         price                    0
         abtest                   0
         vehicleType           5152
         yearOfRegistration       0
         gearbox               2765
         powerPS                  0
         model                 2730
         kilometer                0
         monthOfRegistration      0
         fuelType              4467
         brand                    0
         notRepairedDamage     9640
         dtype: int64
```

This method provides a summary of the number of missing values in each column of the cars DataFrame. The output will be a series where each entry corresponds to a column in the DataFrame, and the value represents the count of missing values in that column. This information is valuable for understanding the extent of missing data in the dataset and deciding how to handle it during the data cleaning process. A majority of 9640 records were missing from notRepairedDamage followed by 5152 missing records in vehicleType etc.

5.2.1. Year of registration

```
In [12]: #variable yearOfRegistration
         yearwise_count=cars['yearOfRegistration'].value_counts().sort_index()

In [13]: sum(cars['yearOfRegistration']>2023)

Out[13]: 24

In [14]: sum(cars['yearOfRegistration']<1950)

Out[14]: 38
```

The dataset consists of records of cars in the future years which are not relevant for analysis. Hence the year of registration of cars after 2023 and before 1950 which are 24 and 38 respectively are filter out so they don’t smear the effect of the model.

5.2.2. Price

In [23]:
```
#variable price
price_count=cars['price'].value_counts().sort_index()
```

In [24]:
```
cars['price'].describe()
```

Out[24]:
```
count      49531.000
mean        6567.220
std        86222.378
min            0.000
25%         1150.000
50%         2950.000
75%         7100.000
max     12345678.000
Name: price, dtype: float64
```

In [25]:
```
sum(cars['price']>150000)
```

Out[25]:
```
34
```

In [26]:
```
sum(cars['price']<100)
```

Out[26]:
```
1748
```

In order to generalizing the model for a workable range of data the lower range of price category is cleared such that they do not sway the effect of the model. The mean here is 6567.220 and median is 2950 which is largely skewed. Thus, cars below $100 and above $150000 are filtered.

5.2.3. PowerPS

In [27]:
```
#variable powerPS
power_count=cars['powerPS'].value_counts().sort_index()
```

In [28]:
```
cars['powerPS'].describe()
```

Out[28]:
```
count   49531.000
mean      116.501
std       231.536
min         0.000
25%        69.000
50%       105.000
75%       150.000
max     19312.000
Name: powerPS, dtype: float64
```

In [29]:
```
sum(cars['powerPS']>500)
```

Out[29]:
```
115
```

In [30]:
```
sum(cars['powerPS']<10)
```

Out[30]:
```
5565
```

It is seen that in powerPS mean is 116 and median is 105 which is not a huge difference, yet the standard deviation is around 200 which makes the data to be skewed and hence higher power values need to be cut down. Hence the modified range is set from 10 to 500.

5.3. Working range of data after cleaning

```
In [31]: cars=cars[
             (cars.yearOfRegistration <= 2023)
             &(cars.yearOfRegistration >= 1950)
             &(cars.price >= 100)
             &(cars.price <= 150000)
             &(cars.powerPS >= 10)
             &(cars.powerPS <= 500)
         ]
```

```
In [32]: cars.describe()
```

Out[32]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration
count	42772.000	42772.000	42772.000	42772.000	42772.000
mean	6135.107	2003.627	126.050	125815.253	5.996
std	7946.682	7.093	60.530	39078.120	3.579
min	100.000	1951.000	10.000	5000.000	0.000
25%	1450.000	1999.000	80.000	100000.000	3.000
50%	3499.000	2004.000	116.000	150000.000	6.000
75%	7900.000	2008.000	150.000	150000.000	9.000
max	149000.000	2018.000	500.000	150000.000	12.000

It is observed that after cleaning the data, 7229 records were dropped. The new workable set of data consists of 42772 records.

5.4. Variable reduction

```
In [104]: #combining yearOfRegistration and monthOfRegistration
          cars['monthOfRegistration']/=12
```

```
In [107]: #create age = yearOfRegistration + monthOfRegstration
          cars['Age']=(2023-cars['yearOfRegistration'])+cars['monthOfRegistration']
          cars['Age']=round(cars['Age'],2)
          cars['Age'].describe()

Out[107]: count    42772.000
          mean        19.414
          std          7.093
          min          5.000
          25%         15.030
          50%         19.070
          75%         24.010
          max         72.060
          Name: Age, dtype: float64
```

On purpose of dealing year of registration and month of registration in a better way, they are reduced to a new variable Age (obtained by adding yearOfRegistration and monthOfRegistration) whose mean is 19.41 and median is 19.07 thus making the data stable.

5.5. Visualizing parameters after narrowing working range

5.5.1. Age VS price

```
In [42]: #visualizing parameter after narrowing working range
         #Age vs price
         sns.regplot(x='Age',y='price',scatter=True,fit_reg=False,data=cars)

Out[42]: <Axes: xlabel='Age', ylabel='price'>
```

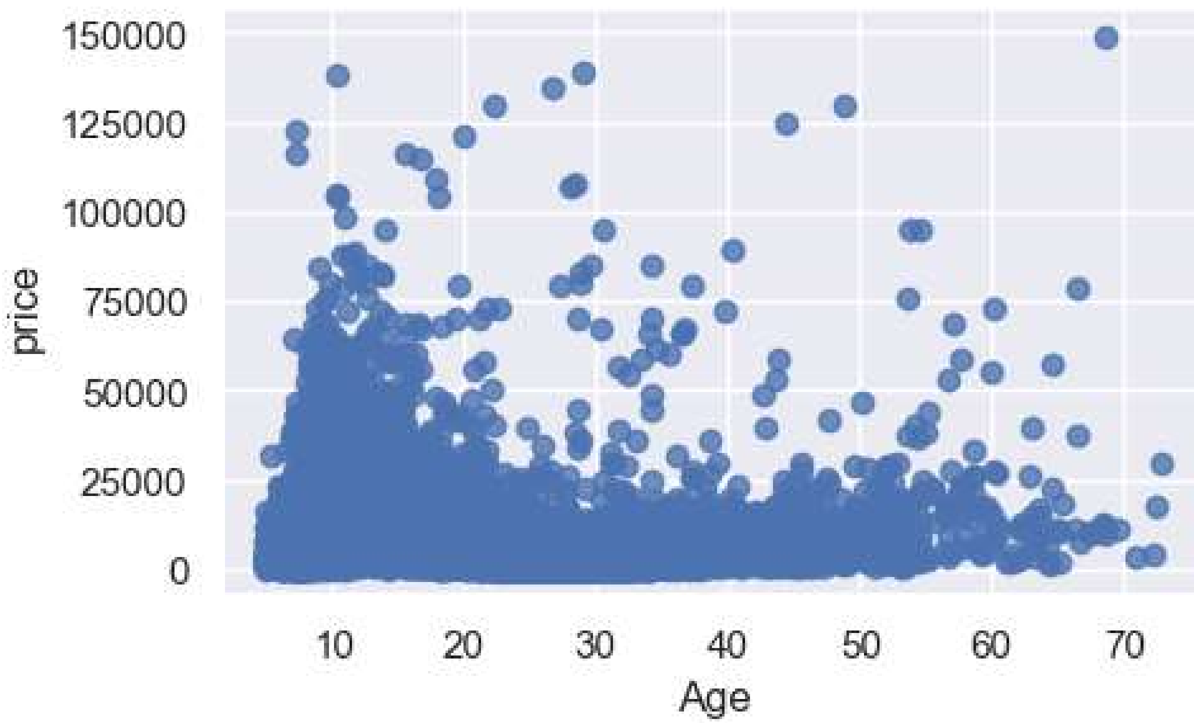

Fig1: Scatter plot representing Age VS price

From fig1, cars which are priced higher are fairly newer. There are few cars which are priced higher yet older which can be considered as vintage. On a general note, with increase in age price also drop.

5.5.2. powerPS VS price

```
In [43]: #powerPS vs price
         sns.regplot(x='powerPS',y='price',scatter=True,fit_reg=False,data=cars)

Out[43]: <Axes: xlabel='powerPS', ylabel='price'>
```

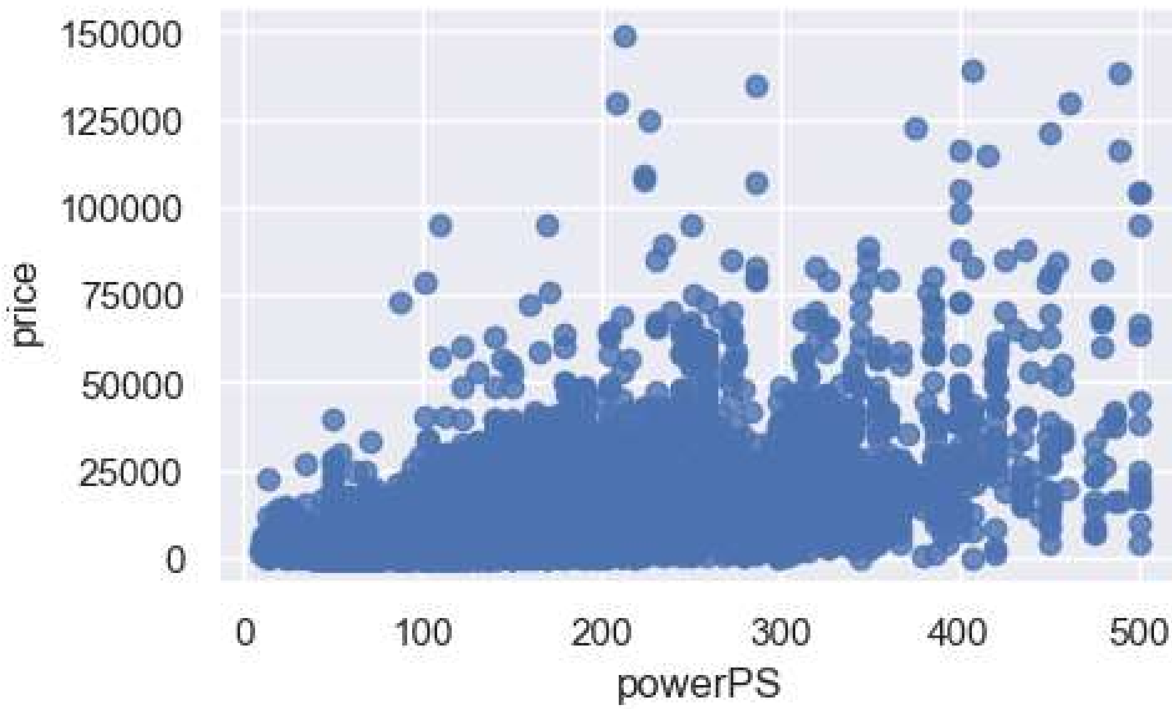

Fig2: Scatter plot representing powerPS VS price

It is clear from fig2 that as power of the car increases the price associated with it also increses. Here with increase in power the price increases.

5.6. Analyzing categorical variables

5.6.1. Seller

In [44]:
```
#variable seller
cars['seller'].value_counts()
```

Out[44]:
```
private       42771
commercial        1
Name: seller, dtype: int64
```

In [45]:
```
pd.crosstab(cars['seller'],columns='count',normalize=True)
```

Out[45]:

col_0	count
seller	
commercial	0.000
private	1.000

In [46]:
```
sns.countplot(x='seller',data=cars)
```

Out[46]:
```
<Axes: xlabel='seller', ylabel='count'>
```

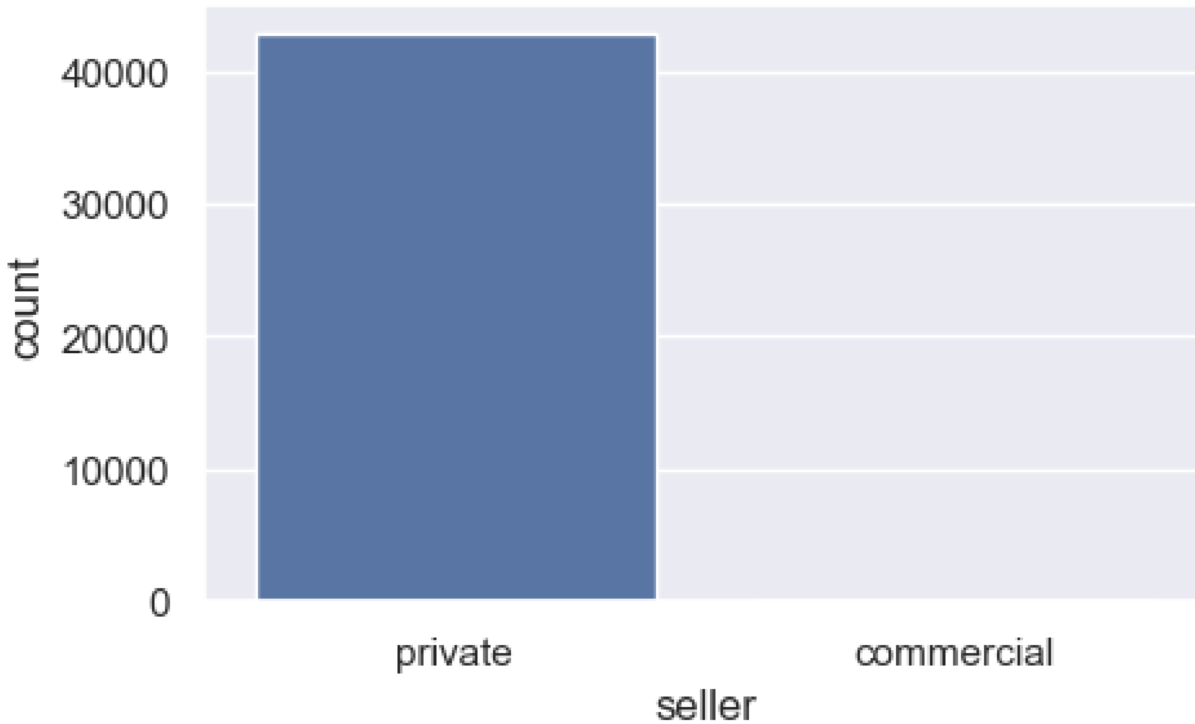

Fig 3: Representing proportion of seller

The analysis shows that majority of 42771 were private seller and only 1 was commercial seller. Private sellers almost accounts to 100% of the proportion. Fewer cars have commercial sellers which makes it insignificant.

5.6.2 OfferType

```
In [47]: #variable offerType
         cars['offerType'].value_counts()

Out[47]: offer    42772
         Name: offerType, dtype: int64

In [48]: sns.countplot(x='offerType',data=cars)

Out[48]: <Axes: xlabel='offerType', ylabel='count'>
```

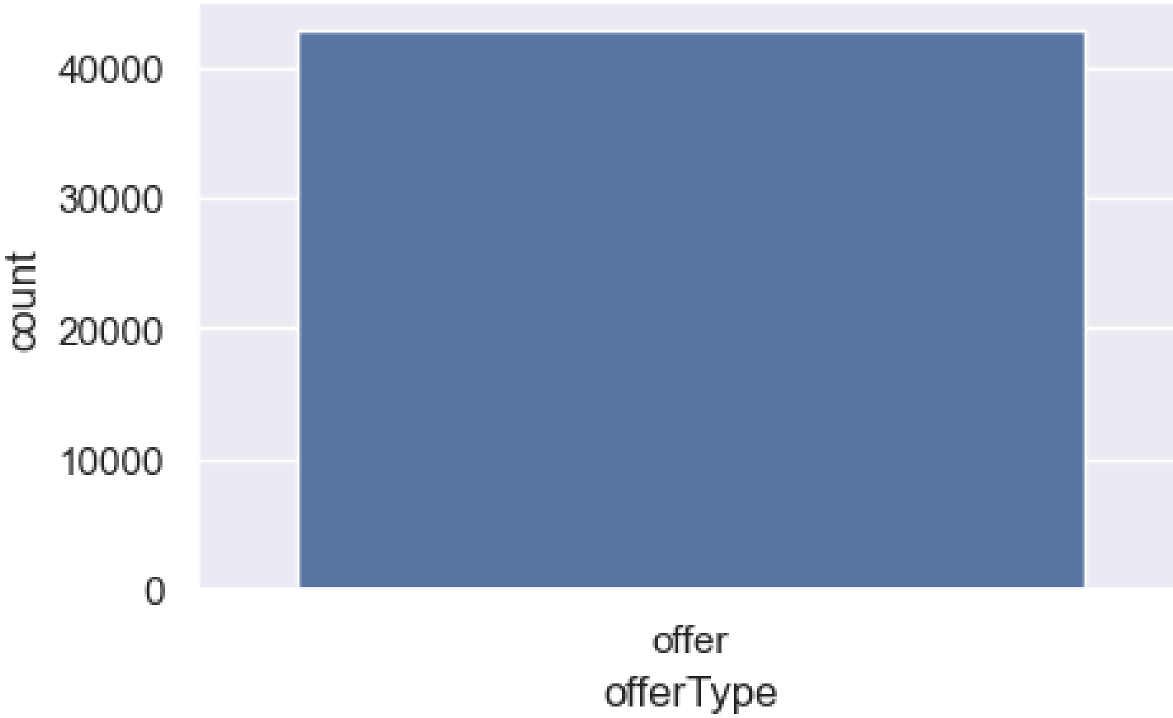

Fig 4: Representing proportion of offerType

Here all the 42772 cars are under the variable type offer and none of them are under request, which further makes then insignificant.

5.6.3. abtest

```
In [49]: #variavle abtest
         cars['abtest'].value_counts()

Out[49]: test       22128
         control    20644
         Name: abtest, dtype: int64

In [50]: pd.crosstab(cars['abtest'],columns='count',normalize=True)

Out[50]:
```

col_0	count
abtest	
control	0.483
test	0.517

```
In [51]: sns.countplot(x='abtest',data=cars)

Out[51]: <Axes: xlabel='abtest', ylabel='count'>
```

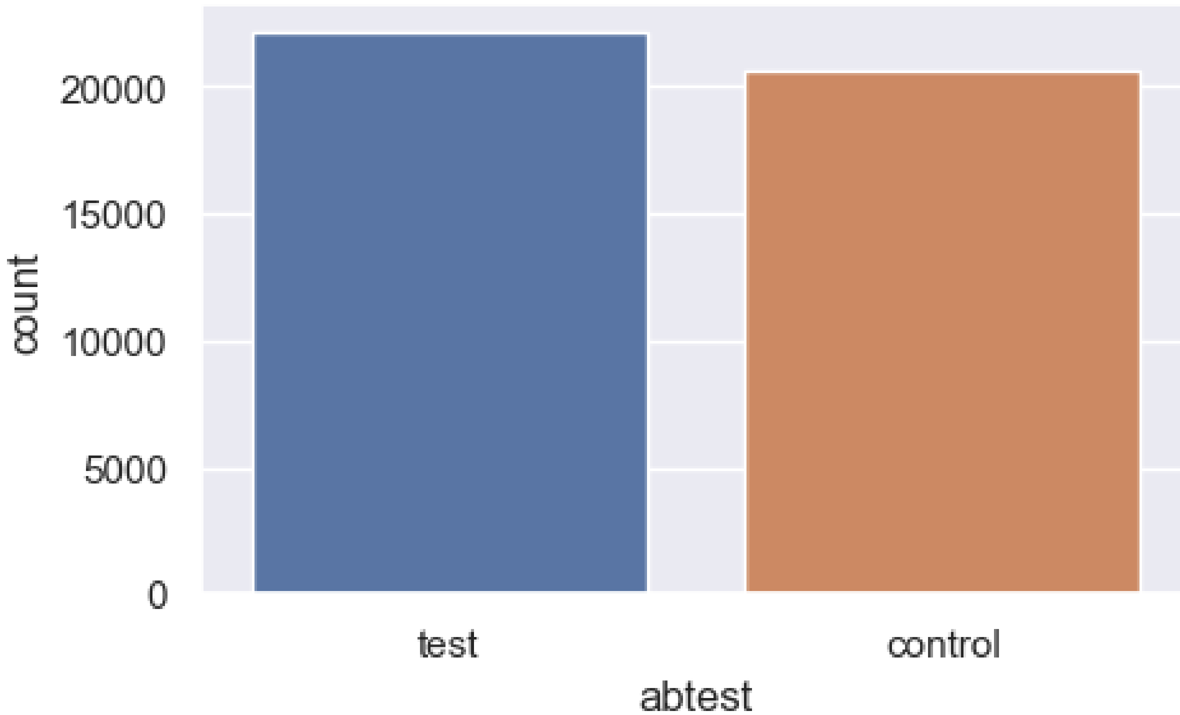

Fig 5: Representing proportion of abtest

Here 22128 cars are under test group and 20644 cars are under control group. This comprises of 49% and 51% of the total cars respectively. Fig 5 makes it more conclusive with a close cut of both categories hence making the affect on price insignificant.

5.6.4. vehicleType

```
In [52]: #variable vehicalType
         cars['vehicleType'].value_counts()

Out[52]: limousine        11746
         small car         9285
         station wagon     8076
         bus               3597
         cabrio            2792
         coupe             2261
         suv               1813
         others             326
         Name: vehicleType, dtype: int64

In [57]: sns.boxplot(x='vehicleType',y='price',data=cars)
         plt.xticks(rotation='vertical')
         plt.show()
```

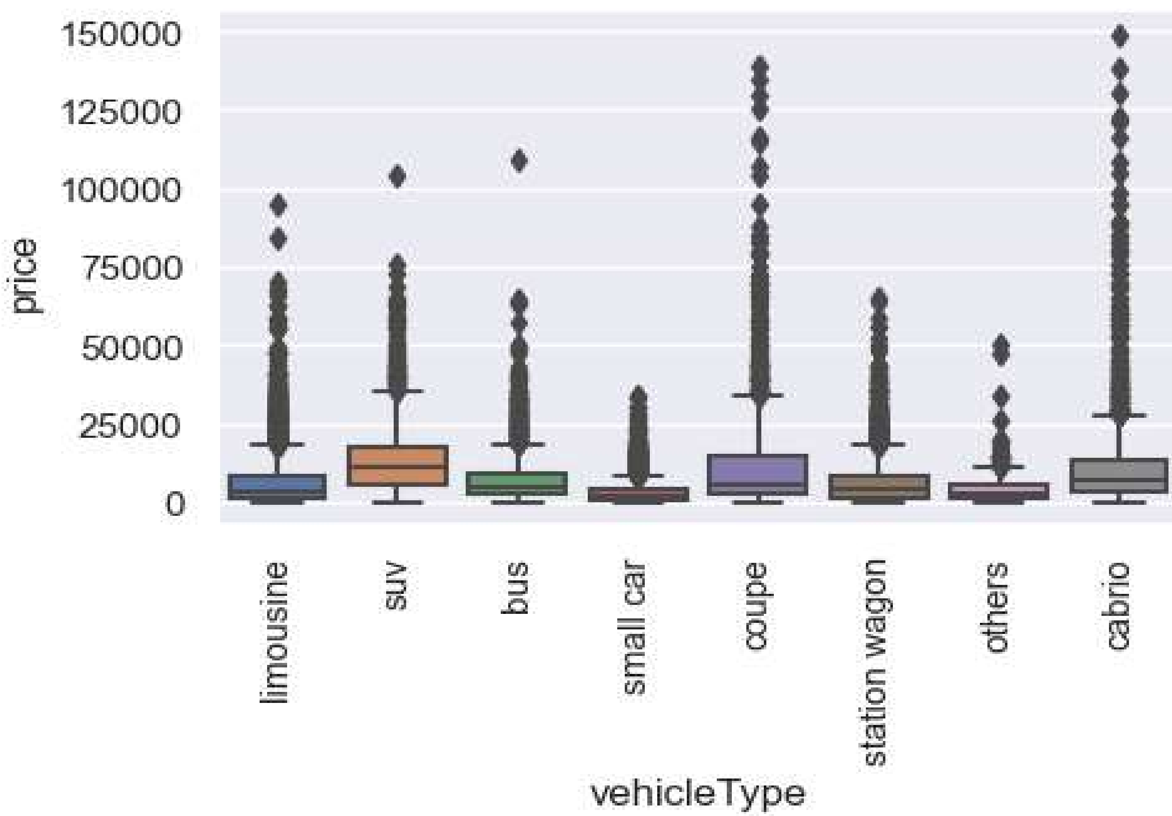

Fig 6: Box plot of vehicle type

It is observed that limousine occupies majority of the proportion. From fig 6 it can be stated that different vehicle types affect the price differently. Hence vehicleType affects price.

5.6.5. GearBox

```
cars['gearbox'].value_counts()

Out[59]: manual       32582
         automatic     9396
         Name: gearbox, dtype: int64

In [60]: pd.crosstab(cars['gearbox'],columns='count',normalize=True)
Out[60]:
```

col_0	count
gearbox	
automatic	0.224
manual	0.776

```
In [62]: sns.boxplot(x='gearbox',y='price',data=cars)
Out[62]: <Axes: xlabel='gearbox', ylabel='price'>
```

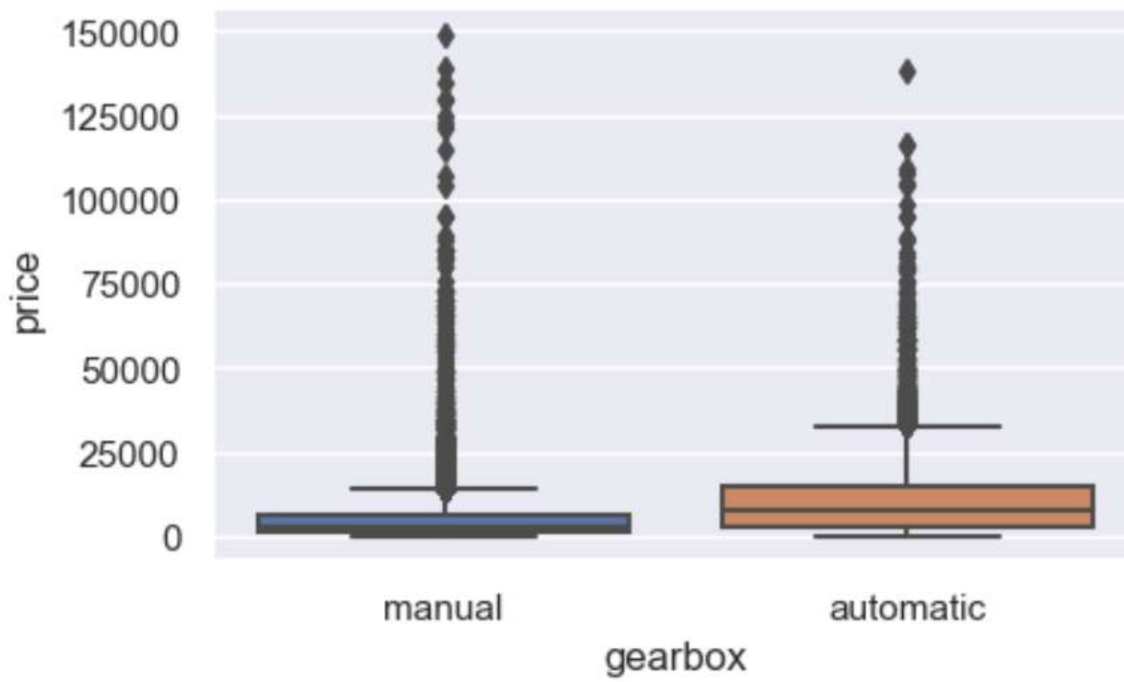

Fig 7: Box plot of gearbox

There were 32582 records of manual gear and 9396 records of automatic gear which counts for 78% and 22% respectively. From fig 7 it can be concluded that manual is priced lower than automatic and hece gearbox affects pice.

5.6.6. Kilometer

```
In [142]: #variable kilometer
          cars['kilometer'].value_counts()

Out[142]: 150000    27430
          125000     4597
          100000     1824
          90000      1484
          80000      1378
          70000      1182
          60000      1101
          50000       932
          40000       795
          30000       712
          20000       651
          5000        479
          10000       207
          Name: kilometer, dtype: int64

In [147]: sns.boxplot(x='kilometer',y='price',data=cars)
Out[147]: <Axes: xlabel='kilometer', ylabel='price'>
```

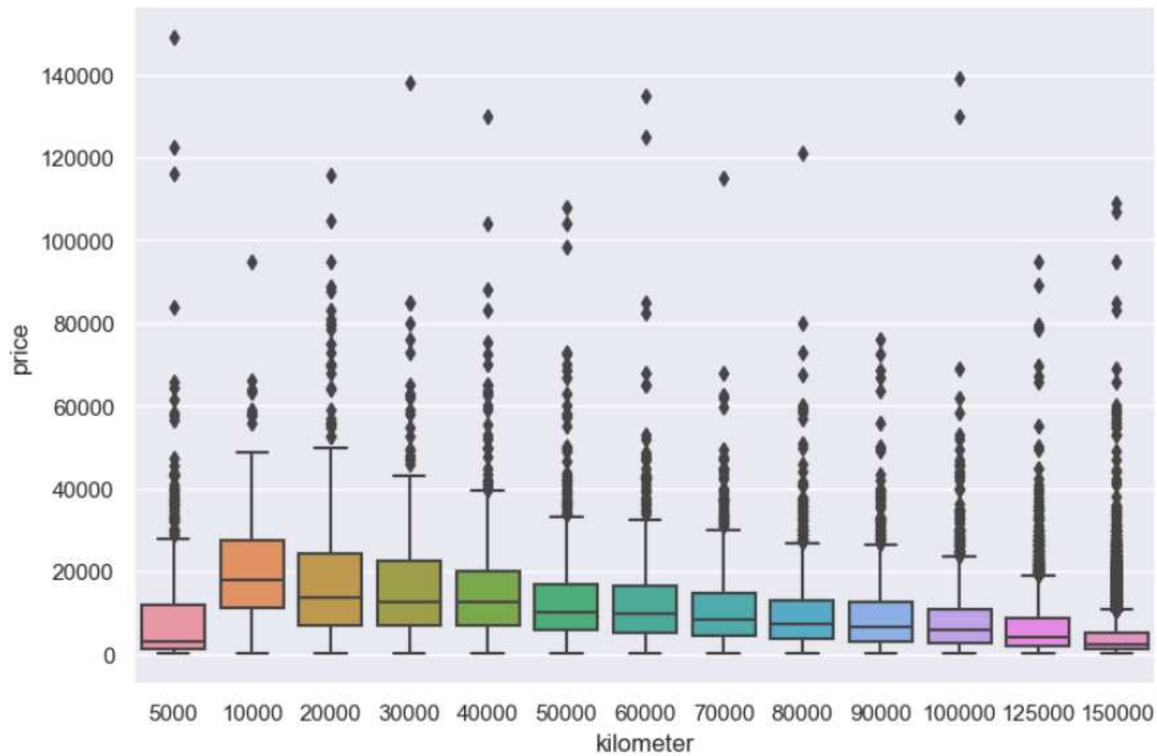

Fig 8: Box plot of kilometer

On the basis of kilometer, cars that have travelled less distance costs higher. Hence kilometer affects price.

5.6.7. fuelType

```
In [64]: #variable fuelType
         cars['fuelType'].value_counts()

Out[64]: petrol     26509
         diesel     12854
         lpg          690
         cng           70
         hybrid        36
         electro       10
         other          6
         Name: fuelType, dtype: int64

In [67]: sns.boxplot(x='fuelType',y='price',data=cars)

Out[67]: <Axes: xlabel='fuelType', ylabel='price'>
```

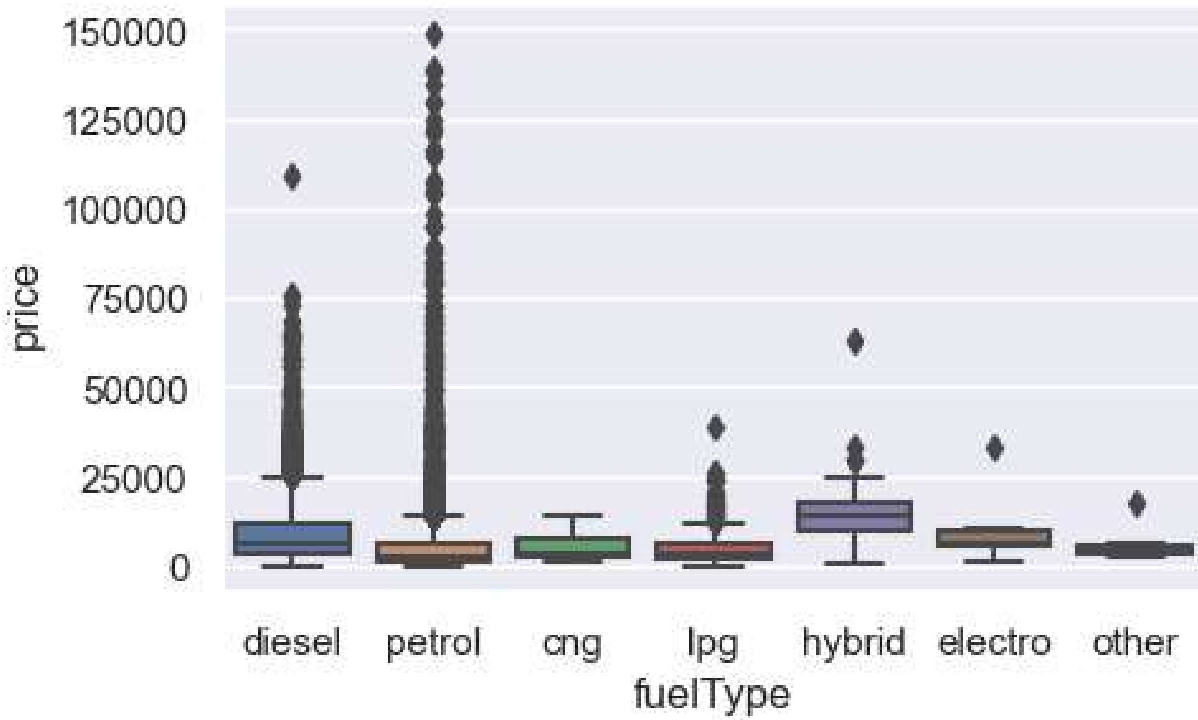

Fig 9: Box plot of fuelType

Petrol and diesel occupy majority of the proportion in fuel type. Various fuel types have different range of price hence fuelType affects price.

5.6.8. notRepairedDamage

```
In [16]: #variable notRepairedDamage
         #yes- damaged and not rectified
         #no- damaged but rectified
         cars['notRepairedDamage'].value_counts()

Out[16]: no     35337
         yes     4948
         Name: notRepairedDamage, dtype: int64
```

```
In [17]: pd.crosstab(cars['notRepairedDamage'],columns='count',normalize=True)
```

Out[17]:

col_0	count
notRepairedDamage	
no	0.877175
yes	0.122825

```
In [18]: sns.countplot(x='notRepairedDamage',data=cars)

Out[18]: <Axes: xlabel='notRepairedDamage', ylabel='count'>
```

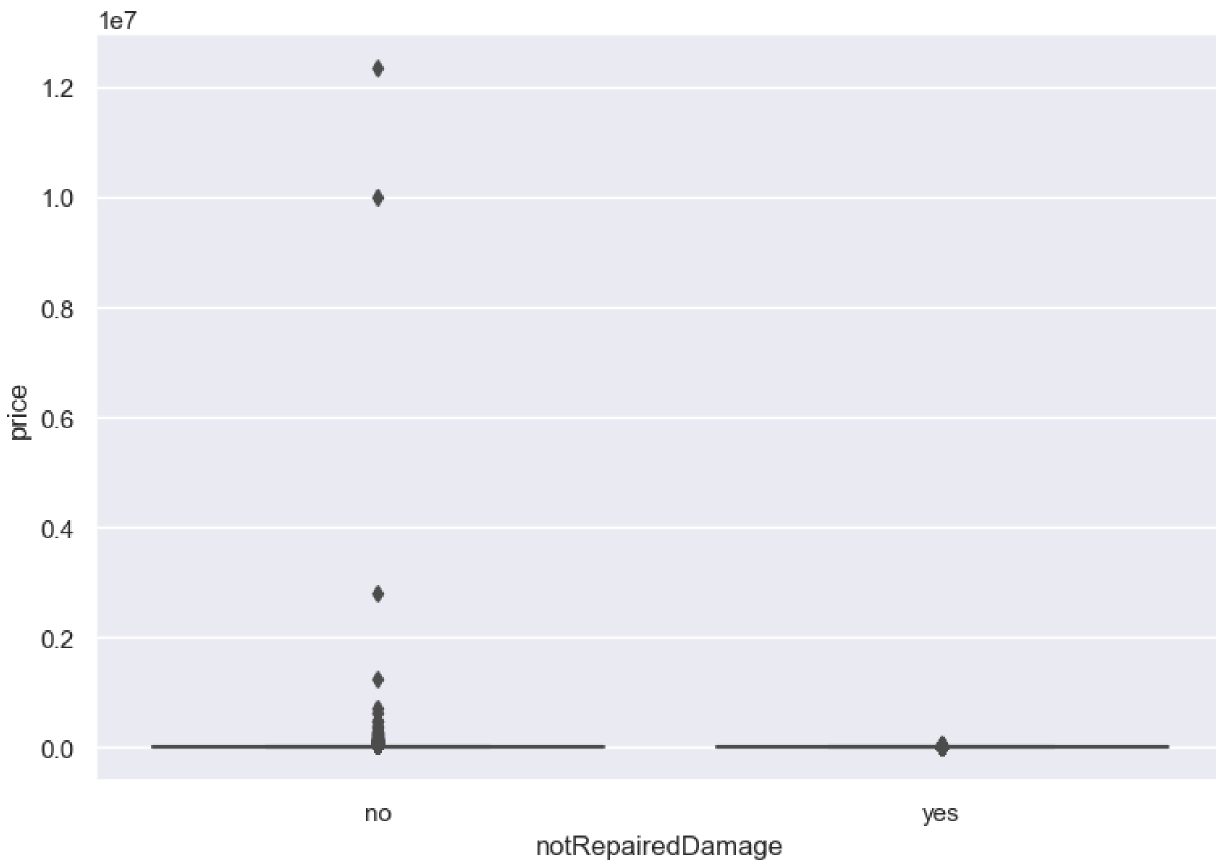

Fig 10: Box plot of notRepairedDamage

It is observed that 35337 cars were damaged but repaired and 4948 cars were damaged but not repaired. The box plot in fig10 confirms that cars which were repaired have higher price. Hence notRepairedDamage affects price.

5.7. A Linear Regression and Random Forest model on data obtained by omitting rows with missing values

Import necessary libraries and omit missing values

```
In [63]: #import libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

```
In [54]: #linear regression and random forest
#data obtained by omitting missing rows
cars_omit=cars.dropna(axis=0)
cars_omit=pd.get_dummies(cars_omit,drop_first=True)
```

```
In [86]: #plotting variable price
prices=pd.DataFrame({"1. Before":y1,"2. After":np.log(y1)})
prices.hist()

Out[86]: array([[<Axes: title={'center': '1. Before'}>,
        <Axes: title={'center': '2. After'}>]], dtype=object)
```

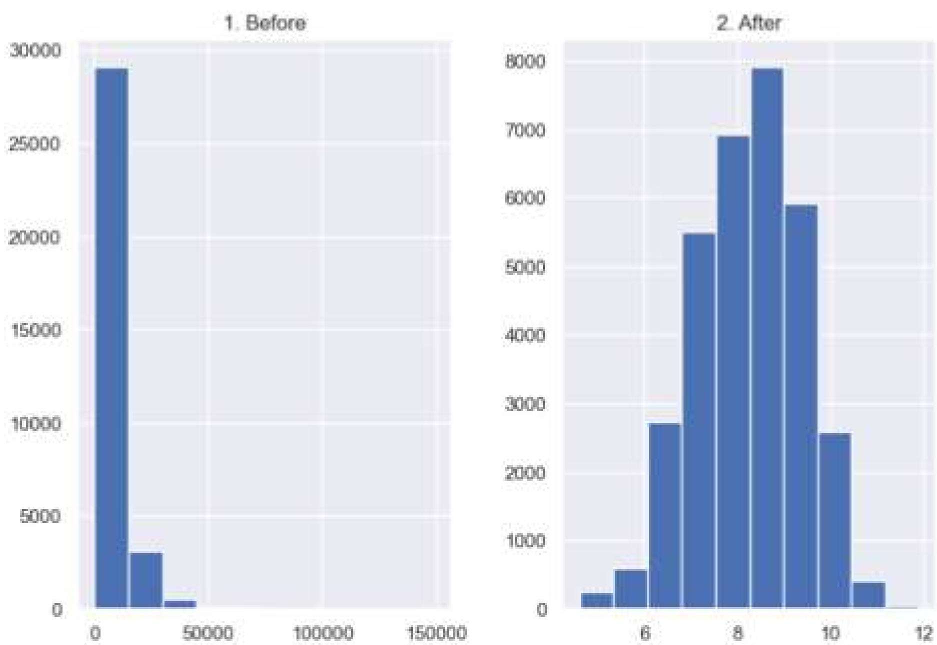

Fig 11(1): A right skewed distribution of price before log transformation
Fig 11(2): A bell curved distribution of price after log transformation

5.8. A Linear Regression and Random Forest model on data obtained by imputing rows with missing values

Imputing missing values

```
In [140]:
cars_imputed=cars.apply(lambda x:x.fillna(x.median())\
if x.dtype=='float' else\
x.fillna(x.value_counts().index[0]))
cars_imputed.isnull().sum()
```

```
Out[140]: price                   0
          vehicleType             0
          yearOfRegistration      0
          gearbox                 0
          powerPS                 0
          model                   0
          kilometer               0
          monthOfRegistration     0
          fuelType                0
          brand                   0
          notRepairedDamage       0
          Age                     0
          dtype: int64
```

```
In [141]:
#converting categorical variables to dummy variables
cars_imputed=pd.get_dummies(cars_imputed,drop_first=True)
```

```
In [143]:
#separating input and output feature
x2=cars_imputed.drop(['price'],axis='columns',inplace=False)
y2=cars_imputed['price']
```

```
In [144]:
#plotting the variable price
prices=pd.DataFrame({"1.Before":y2,"2.After":np.log(y2)})
prices.hist()
```

```
Out[144]: array([[<Axes: title={'center': '1.Before'}>,
                  <Axes: title={'center': '2.After'}>]], dtype=object)
```

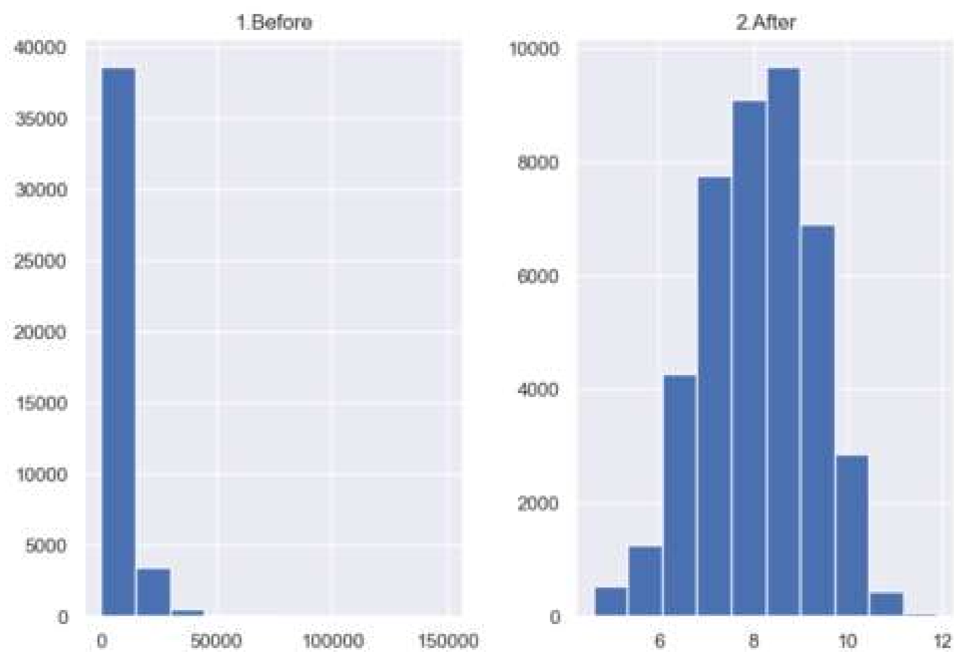

Fig 12(1): A right skewed distribution of price before log transformation
Fig 12(2): A bell curved distribution of price after log transformation

5.9. Result of analysis with omitted and imputed data

In [172]:

```
print('Metrics for models built from data where missing values were omitted')
print('R squared value for train from linear Regrission= %s'%r2_lin_train1)
print('R squared value for test from linear Regrission= %s'%r2_lin_test1)
print('R squared value for train from Random Forest = %s'%r2_lin_train1)
print('R squared value for test from Random Forest= %s'%r2_lin_test1)
print('Base RMSE of model built from data where missing values are omitted=%s'%base_root_mean_square_erroe_impute
print('RMSE value for test from Linear Regression=%s'%lin_rmse1)
print('RMSE value for test from Random Forest=%s'%rf_rmse1)
print('\n\n')
print('Metrics for mpdels built from data where missing values were imputed')
print('R squared value for train from linear Regression= %s'%r2_lin_train2)
print('R squared value for test from linear Regression= %s'%r2_lin_test2)
print('R squared value for train from Random Forests= %s'%r2_rf_train2)
print('R squared value for test from Random Forests= %s'%r2_rf_test2)
print('Base RMSE of model built from data where missing values are imputed=%s'%base_root_mean_square_erroe_impute
print('RMSE value for test from Linear Regression=%s'%lin_rmse2)
print('RMSE value for test from Random Forest=%s'%rf_rmse2)
```

```
Metrics for models built from data where missing values were omitted
R squared value for train from linear Regrission= 0.7809059791496729
R squared value for test from linear Regrission= 0.7670765769569896
R squared value for train from Random Forest = 0.9787434435675523
R squared value for test from Random Forest= 0.85885713162442
Base RMSE of model built from data where missing values are omitted=0.1526
810336802738
RMSE value for test from Linear Regression=0.5441307172375649
RMSE value for test from Random Forest=0.42357091488689363
```

```
Metrics for mpdels built from data where missing values were imputed
R squared value for train from linear Regression= 0.6678532994352833
R squared value for test from linear Regression= 0.6730820501214233
R squared value for train from Random Forests= 0.9684370289685181
R squared value for test from Random Forests= 0.8046473530837177
Base RMSE of model built from data where missing values are imputed=0.1526
810336802738
RMSE value for test from Linear Regression=0.08799339271270556
RMSE value for test from Random Forest=0.06748305599783345
```

The study shows that, Random Forest is better than Linear Regression model in both cases where missing values were omitted and where missing values were imputed.

6.Conclusion

This project holds significant value for e-commerce platforms engaged in mediating pre-owned car transactions. It simplifies the decision-making process for customers, empowering them to compare and assess diverse car models and features. Sellers benefit from the ability to effectively communicate the merits of various models, ensuring satisfaction for both buyers and sellers.

The project includes a detailed comparative study evaluating the performance of regression-based supervised machine learning models. Each model undergoes training using data collected from an e-commerce website specializing in the used car market. Notably, the findings reveal that Linear Regression stands out with exceptional performance, achieving the lowest Root Mean Square Error (RMSE) at 8902.410. This underscores the model's accuracy in predicting pre-owned car prices, making it a valuable tool for enhancing the efficiency of the e-commerce platform.

References

[1] Annina S, Mahima SD, Ramesh B, “An Overview of Machine Learning and its Applications”. International Journal of Electrical Sciences & Engineering (IJESE), 2015. pp. 22-24.

[2] Skikit-learn, supervised-learning [Accessed November 22, 2023]

[3] Leo Breiman, “Random Forests”. University of California Berkeley, 2001.

[4] Nitis Monburinon, Prajak Chertchom,Thongchai Kaewkiriya, Suwat Rungpheung, Sabir Buya, Pitchayakit Boonpou Title” Prediction Of Prices For Used Car By Using Regression Models”

[5] G.Rossum, “Python Reference Manual,” Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 1995.

[6] G.Chandrashekar And F. Sahin, “A Survey On Featureselection Methods,” Computers & Electrical Engineering, Vol. 40, No. 1, Pp. 16–28, 2014. [Accessed November 22, 2023]