Car Price prediction

Submitted by:

Akash Kanjwani

ACKNOWLEDGMENT

During the process of completing this project, I have referred following materials for which I owe them great gratitude.

1.Data collection- Using Web scraping tool i.e. Selenium

2. Car24.com for collecting the data https://www.cars24.com/

3. Data trained video tutorials.

4. Scikit-learn https://scikit-learn.org/stable/

5. Machine Learning for Dummies by John Mueller and Luca Massaron - Easy to understand for a beginner book.

6. Geeksforgeeks. https://www.geeksforgeeks.org/

Besides that all the observation, creations of the models and graphs done by self help.

INTRODUCTION

Problem Statement-

Car price prediction is anyhow interesting and popular problem. Accurate car price prediction involves expert knowledge, because price usually depends on many unique features and factors. The main aim of this project is to predict the price of used cars using the various Machine Learning (ML) models. This can enable the customers to make decisions based on different inputs or factors Generally, most important ones are brand name and model, years, KMs driven and mileage. The fuel type used in the car as well as fuel consumption per mile highly affected price of a car due to often changes in the price of a fuel.

The project Car Price Prediction deals with providing the solution to these problems. Through this project, we will get to know which of the factors are significant and tell us how they affect the car's worth in the market.

Motivation for the Problem Undertaken

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. 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 car price valuation model.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

1) The size of table is 5104*7 i.e. no. of rows are 5100 and no. of columns are 7(including target).

2) Out of 7 columns 2 columns are continuous in nature and rest 5 are object type.

3) Null values are not present in the data set as we can see in ths seaborn heatmap, so there is no need to adopt imputation techniuqe.

4) In case of object data type, we will apply the encoding technique to convert the values in the numeric format.

In order to scrape the data we got some garbage data as well like:

- car name column had many unusefull information.
- year of purches and Kilometers_Driven columns had present in the form fo object datatype.
- In every row of price column and Kilometers_Driven column comma(,) was present.
- we had some empty space in between the columns.
- We seperate the car's company names and created one new column name called company.

All the above garbage data we removed from the dataset with the help of pandas functions.

- In our dataset only 3 columns are showing some outliers that is price, year of purches and kilometers driven. Because the mean is 44586 and standard daviation is 31501 and maximum max values is 400055 which is very higher and reason of outliers.

- High Skewness is present in the kilometers driven column(2.21), shown that data data are not equally distributed.

Data Sources and their formats

Initially we did not data, For getting the data we used web scraping selenium tool.

Data source- https://www.cars24.com/

```
data.head()
```

	car name	year of purches	Kilometers_Driven	Owner	Fuel_type	Transmission	Price
0	2014 Mercedes Benz C Class C 200 AVANTGARDE AU...	January 2014	36,806 km	1st Owner	Petrol	AUTOMATIC	₹ 21,33,299
1	2009 Maruti Wagon R LXI MINOR MANUAL	October 2009	57,473 km	1st Owner	Petrol	MANUAL	₹ 1,56,599
2	2020 KIA SELTOS GTX + AT PETROL AUTOMATIC	September 2020	7,568 km	1st Owner	Petrol	AUTOMATIC	₹ 17,41,799
3	2019 Honda Amaze 1.2 EMT I VTEC MANUAL	June 2019	15,973 km	1st Owner	Petrol	MANUAL	₹ 5,72,399
4	2021 Renault Kiger RXZ 1.0 Easy R Petrol	February 2021	2,747 km	1st Owner	Petrol	NA	₹ 8,56,699

```
data.shape
```

```
(5104, 7)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5104 entries, 0 to 5103
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   car name          5104 non-null   object
 1   year of purches   5104 non-null   object
 2   Kilometers_Driven 5104 non-null   object
 3   Owner             5104 non-null   object
 4   Fuel_type         5104 non-null   object
 5   Transmission      5104 non-null   object
 6   Price             5104 non-null   object
dtypes: object(7)
memory usage: 279.2+ KB
```

Data Pre-processing Done

Outliers, Skewness, Data scaling, Data Cleaning

We had outliers in some columns and also skewed data was present in our data set. To Clean the data we have performed following steps.

- For outliers removing we used z score method.
- For skewness removing we used power transformation.
- For Data scaling we used Min max scaler.

* Encoding the data with the help of label encoder.

* Data cleaning using pandas function.

Data cleaning

```
backup=data.copy()
```

```
data['car name']=data['car name'].str.split(" ").str.slice(1,4).str.join(" ")
```

```
data['company']=data['car name'].str.split(" ").str.slice(0,1).str.join(" ")
```

```
data["year of purches"]=data['year of purches'].str.split(" ").str.slice(1).str.join(" ")
```

```
data=data[data['year of purches']!=""]
```

```
data['year of purches']=data['year of purches'].astype(int)
```

```
data.head()
```

	car name	year of purches	Kilometers_Driven	Owner	Fuel_type	Transmission	Price	company
0	Mercedes Benz C	2014	36,806 km	1st Owner	Petrol	AUTOMATIC	₹ 21,33,299	Mercedes
1	Maruti Wagon R	2009	57,473 km	1st Owner	Petrol	MANUAL	₹ 1,56,599	Maruti
2	KIA SELTOS GTX	2020	7,568 km	1st Owner	Petrol	AUTOMATIC	₹ 17,41,799	KIA
3	Honda Amaze 1.2	2019	15,973 km	1st Owner	Petrol	MANUAL	₹ 5,72,399	Honda
4	Renault Kiger RXZ	2021	2,747 km	1st Owner	Petrol	NA	₹ 8,56,699	Renault

```
data=data[data['Kilometers_Driven']!="---"]
```

```
data=data[data['Kilometers_Driven']!="---"]
```

```
data['Kilometers_Driven']=data['Kilometers_Driven'].str.split(" ").str.slice(0,1).str.join(" ")
```

```
data['Kilometers_Driven']=data['Kilometers_Driven'].str.replace(",","").astype(int)
```

```
data['Price']=data['Price'].str.split(" ").str.slice(1).str.join("").str.replace(",","").astype(int)
```

```
data=data[data['Price']!="---"]
```

```
data=data[data['Transmission']!='NA']
```

```
data
```

	car name	year of purches	Kilometers_Driven	Owner	Fuel_type	Transmission	Price	company
0	Mercedes Benz C	2014	36806	1st Owner	Petrol	AUTOMATIC	2133299	Mercedes
1	Maruti Wagon R	2009	57473	1st Owner	Petrol	MANUAL	156599	Maruti
2	KIA SELTOS GTX	2020	7568	1st Owner	Petrol	AUTOMATIC	1741799	KIA
3	Honda Amaze 1.2	2019	15973	1st Owner	Petrol	MANUAL	572399	Honda
5	Honda WR-V 1.2	2017	17117	1st Owner	Petrol	MANUAL	779899	Honda

Removing Outliers

```
ske=['year of purches','Kilometers_Driven']
```

```
from scipy.stats import zscore
```

```
z=np.abs(zscore(x[ske]))
```

```
z.shape
```

```
(5017, 2)
```

```
threshold=3
print(np.where(z>3))
```

```
(array([   1,    5,    8,   58,  212,  237,  239,  291,  328,  405,  486,
        534,  605,  633,  828,  960, 1058, 1172, 1177, 1181, 1185, 1192,
       1298, 1310, 1435, 1597, 1613, 1767, 1972, 1974, 2053, 2057, 2269,
       2472, 2517, 2540, 2555, 2564, 2722, 2847, 2924, 2929, 2930, 3005,
       3075, 3117, 3152, 3174, 3183, 3241, 3384, 3443, 3488, 3531, 3562,
       3567, 3568, 3643, 3746, 3833, 3838, 3861, 3879, 3939, 4027, 4087,
       4131, 4169, 4175, 4217, 4223, 4231, 4240, 4302, 4313, 4372, 4395,
       4480, 4528, 4538, 4546, 4605, 4732, 4777, 4800, 4815, 4824, 4982],
      dtype=int64), array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1],
      dtype=int64))
```

```
x=x[(z<3).all(axis=1)]
```

Skewness removing

```
col=x.columns
```

```
x['Kilometers_Driven']=np.sqrt(x['Kilometers_Driven'])
```

```
x['year of purches']=np.sqrt(x['year of purches'])
```

Encoding the data

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
enc=['car name','Owner','Fuel_type','Transmission','company']
```

```
for i in enc:
    data[i]=le.fit_transform(data[i])
```

```
data.head()
```

	car name	year of purches	Kilometers_Driven	Owner	Fuel_type	Transmission	Price	company
0	198	2014	36806	0	1	0	2133299	12
1	193	2009	57473	0	1	1	156599	11
2	104	2020	7568	0	1	0	1741799	8
3	31	2019	15973	0	1	1	572399	4
5	49	2017	17117	0	1	1	779899	4

Scaling tha data

```
from sklearn.preprocessing import StandardScaler
```

```
st=StandardScaler()
```

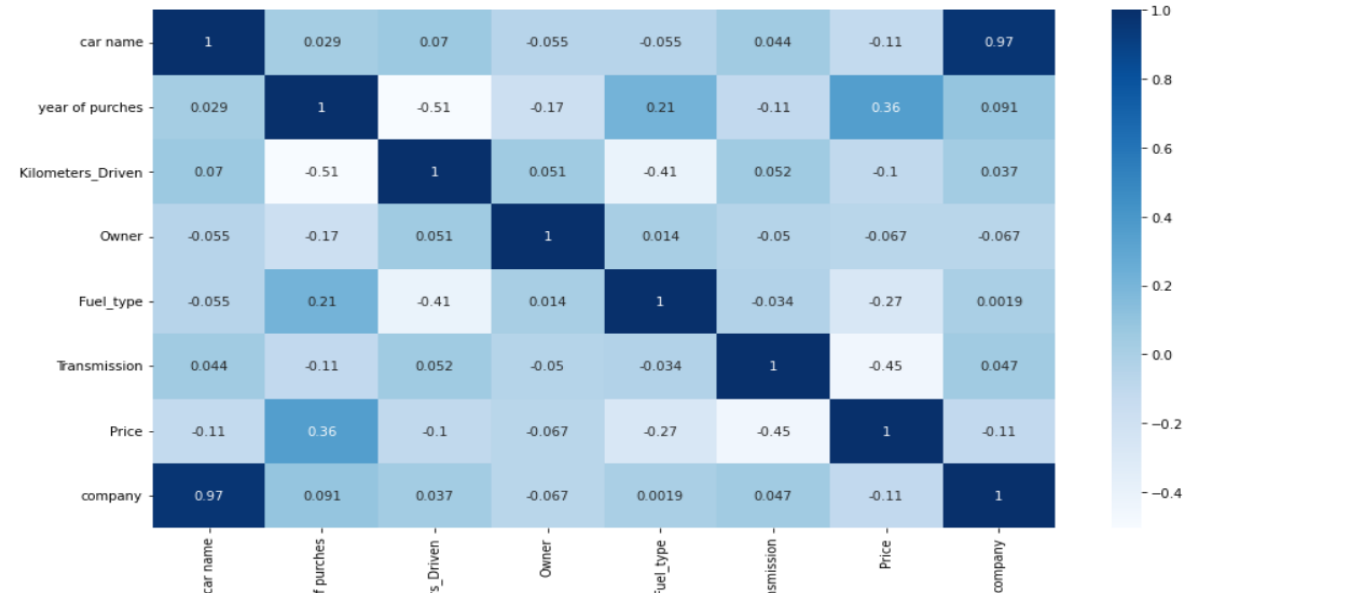
```
x=st.fit_transform(x)
```

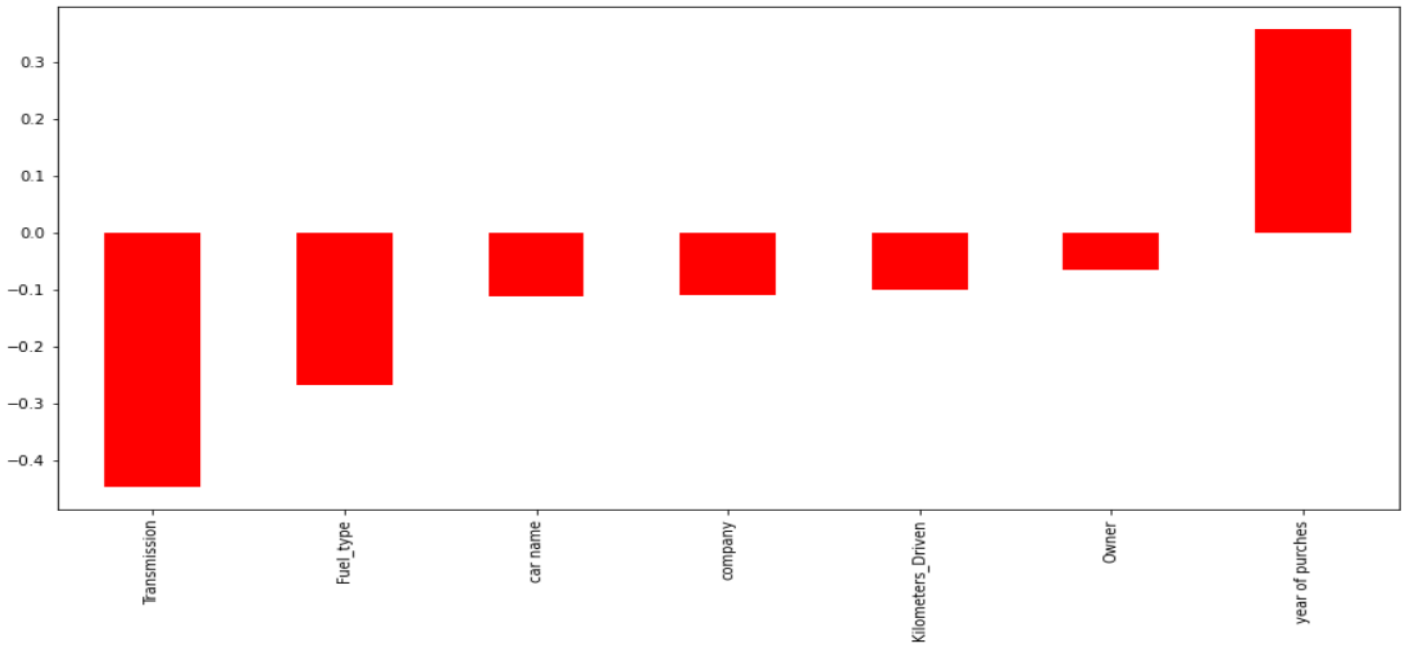
```
x=pd.DataFrame(x,columns=col)
```

```
x
```

	car name	year of purches	Kilometers_Driven	Owner	Fuel_type	Transmission	company
0	0.985187	-1.307065	-0.058532	-0.488409	0.586076	-2.375470	0.782612
1	-0.365786	1.240880	-1.620250	-0.488409	0.586076	-2.375470	-0.222402
2	-1.414947	0.816485	-1.033569	-0.488409	0.586076	0.420969	-1.227416
3	-1.156250	-0.032619	-0.967326	-0.488409	0.586076	0.420969	-1.227416
4	-0.466391	-1.732092	-0.410861	1.662949	-1.568113	-2.375470	-0.724909

Data Inputs- Logic- Output Relationships





- year of purches column making positive correlation with the target column.
- transmission, fule_type, car name, company, owner and kilometers driven column are making negative correlation with the target column.
- car name and company columns making a strong correlation with each other, means one will increase another will also increase.
- Those car prices are high which have been purchesed from the year 2014 to 2021.
- The prices are high when someone is going to buy automatic transmission car.
- The car prices are high when the car owner stats is 1st owner and 2nd owner as compare to others

Hardware and Software Requirements and Tools Used

Anaconda Navigator

Jupyter Notebook

Language-Python

Selenium

Many lib.-------

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
import selenium
warnings.filterwarnings('ignore')
from sklearn.preprocessing import power_transform
from scipy.stats import zscore
from sklearn.preprocessing import MinMaxScaler
import statsmodels.api as si
from scipy import stats from
statsmodels.stats.outliers_influence
import sklearn
from sklearn.linear_model import
LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.model_selection import
train_test_split,GridSearchCV,cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import
RandomForestRegressor,AdaBoostRegressor,GradientBoost
ingRegressor
import xgboost as xg from sklearn.metrics
import
mean_squared_error,mean_absolute_error,r2_score
```

Pandas- For making data frame

Matplotlib and seaborn- For data visualization

Numpy- For numerical python

ZScore- For removing outliers

MInMAxScaler- For data scaling

Power transform-For removing skewness

From metrice - mean_squared_error,mean_absolute_error,r2_score -For checking the model accuracy.

Regression- For regression modeling

Ensamble- For boosting and bagging

Grid search cv- For hyperparameter tuning

Cross_Val_Score- For cross validation

Model/s Development and Evaluation

Approaches Firstly it is import to know about which type of modelling we are going to construct, For this problem we used regression models because our target variable is numerical type and we had to predict the house prices. When we go for regression we have to use some metrics like mean_squared_error, mean_absolute_error, r2_score In order to do this work we have to find out the best random state by which we can achieve good accuracy. Then we split our data set into the train part and test part using train test split. When we done with the modelling we have to use cross validation for real accuracy(without underfitting and overfitting). Hyperpameter is must for increasing the model accuracy in order to build good model for this we use Grid search cv. We followed all the above approaches to build our Machine learning model.

Algorithms

- Linear Regression

- Decision Tree Regressor
- KNeighbors Regressor
- Support Vector Regressor
  - Ridge regressor
  - Lasso regressor

For Bagging and boosting:

- Random Forest Regresssor
- Gradient Bossting Regressor
- AdaBoost Regressor
- XgBoost Regressor

```
import sklearn
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
import xgboost as xg
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

Run and Evaluate selected models and Key Metrics

In order to achieve good accuracy, we find the best random state and then this random state is applied on the train test split.

```
def model_select(model):
    max_score=0
    max_state=0
    for i in range(0,50):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.22)
        md=model()
        md.fit(x_train,y_train)
        predict=md.predict(x_test)
        r2score=r2_score(y_test,predict)
        if r2score > max_score:
            max_score=r2score
            max_state=i
    print('max score is {} at random state {}'.format(max_score,max_state))
```

Linear Regression

Finding best random state by calling model_select function

```
model_select(LinearRegression)
```

```
max score is 0.47614781536471773 at random state 8
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=8)
```

```
LR=LinearRegression()
LR.fit(x_train,y_train)
pred=LR.predict(x_test)

print('Mean_squared error:',mean_squared_error(pred,y_test))
print('Mean absolute error:',mean_absolute_error(pred,y_test))
print('r2_score:',r2_score(pred,y_test))
print('Root mean squared error:',np.sqrt(mean_squared_error(pred,y_test)))
```

```
Mean_squared error: 61097998365.33467
Mean absolute error: 186134.02436226993
r2_score: -0.1594610419611764
Root mean squared error: 247180.09297946037
```

```
for i in range(9,16):
    cvs=cross_val_score(LR,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.38021102793820916
cross validation score when cv is 10 = 0.38964611289842066
cross validation score when cv is 11 = 0.37708300046640164
cross validation score when cv is 12 = 0.3744014522013146
cross validation score when cv is 13 = 0.3613338473993593
cross validation score when cv is 14 = 0.33208062981626046
cross validation score when cv is 15 = 0.3518439621910035
```

Decision tree regressor

Finding best random state by calling model_select function

```
model_select(DecisionTreeRegressor)
```

```
max score is 0.9318122090274361 at random state 40
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=40)
```

```
Dt=DecisionTreeRegressor()
Dt.fit(x_train,y_train)
pred=Dt.predict(x_test)
print("r2 score : ",r2_score(y_test,pred))
print("Mean absoluate error :",mean_absolute_error(y_test,pred))
print("Means squred error:" ,mean_squared_error(y_test,pred))
print("Root mean squred error: ",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score :  0.9240270691094222
Mean absoluate error : 39102.69216589862
Means squred error: 9884841369.949308
Root mean squred error:  99422.5395468719
```

```
for i in range(9,16):
    cvs=cross_val_score(Dt,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.8737534355480465
cross validation score when cv is 10 = 0.8774894943520433
cross validation score when cv is 11 = 0.8790108911836382
cross validation score when cv is 12 = 0.878210852490395
cross validation score when cv is 13 = 0.8880938339355364
cross validation score when cv is 14 = 0.8802303124229143
```

KNeighbor regressor

Finding best random state by calling model_select function

```
model_select(KNeighborsRegressor)
```

```
max score is 0.7746623508395747 at random state 4
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=4)
```

```
knn=KNeighborsRegressor()
knn.fit(x_train,y_train)
pred=knn.predict(x_test)
print("r2 score :",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squred error :",mean_squared_error(y_test,pred))
print("Root mean squred error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score : 0.7746623508395747
Mean absolute error: 108523.79834101383
Mean squred error : 25996739622.171795
Root mean squred error: 161235.04464654016
```

```
for i in range(9,16):
    cvs=cross_val_score(knn,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.6939160931405133
cross validation score when cv is 10 = 0.7053130206353337
cross validation score when cv is 11 = 0.6963700949872088
cross validation score when cv is 12 = 0.687198259015671
cross validation score when cv is 13 = 0.6929133786261492
cross validation score when cv is 14 = 0.6695171571954432
cross validation score when cv is 15 = 0.6911772994148347
```

Lasso Regressor

```
model_select(Lasso)
```

```
max score is 0.4761475245809321 at random state 8
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=8)
```

```
lss=Lasso()
lss.fit(x_train,y_train)
pred=lss.predict(x_test)
print("r2 score :",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squred error :",mean_squared_error(y_test,pred))
print("Root mean squred error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score : 0.4761475245809321
Mean absolute error: 186133.75797305064
Mean squred error : 61098032280.06821
Root mean squred error: 247180.16158273746
```

```
for i in range(9,16):
    cvs=cross_val_score(lss,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.38021111635703975
cross validation score when cv is 10 = 0.3896461500238937
cross validation score when cv is 11 = 0.3770830325569065
cross validation score when cv is 12 = 0.3744015914243806
cross validation score when cv is 13 = 0.3613341051245819
cross validation score when cv is 14 = 0.3320809298942883
cross validation score when cv is 15 = 0.3518442000536705
```

Ridge Regressor

```
model_select(Ridge)
```

```
max score is 0.4761421697439796 at random state 8
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=8)
```

```
ri=Ridge()
ri.fit(x_train,y_train)
pred=ri.predict(x_test)
print("r2 score :",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squred error :",mean_squared_error(y_test,pred))
print("Root mean squred error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score : 0.4761421697439796
Mean absolute error: 186124.41299492292
Mean squred error : 61098656826.13095
Root mean squred error: 247181.4249213135
```

```
for i in range(9,16):
    cvs=cross_val_score(ri,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.38022341152678973
cross validation score when cv is 10 = 0.38965574414198856
cross validation score when cv is 11 = 0.37709463228244466
cross validation score when cv is 12 = 0.3744098760315173
cross validation score when cv is 13 = 0.361348031625079
cross validation score when cv is 14 = 0.3320952648445492
cross validation score when cv is 15 = 0.3518590500538946
```

Random forest classifier

Finding best random state by calling model_select function

```
model_select(RandomForestRegressor)
```

```
max score is 0.9499188456474392 at random state 4
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=4)
```

```
rfr=RandomForestRegressor()
rfr.fit(x_train,y_train)
pred=rfr.predict(x_test)
print("r2 score:",r2_score(y_test,pred))
print("Mean absolute error",mean_absolute_error(y_test,pred))
print("Mean squared error:",mean_squared_error(y_test,pred))
print("Root mean squared error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score: 0.9518885457124674
Mean absolute error 39346.301963133636
Mean squared error: 5550519208.028845
Root mean squared error: 74501.80674338606
```

```
for i in range(9,16):
    cvs=cross_val_score(rfr,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.9128902596960202
cross validation score when cv is 10 = 0.9145716490225946
cross validation score when cv is 11 = 0.9176763355672928
cross validation score when cv is 12 = 0.9185819855644238
cross validation score when cv is 13 = 0.9173015920666102
cross validation score when cv is 14 = 0.9189365036579983
cross validation score when cv is 15 = 0.9208354214215075
```

AdaBoostRegressor

Finding best random state by calling model_select function

```
model_select(AdaBoostRegressor)
```

```
max score is 0.5423343888743941 at random state 40
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=40)
```

```
ada=AdaBoostRegressor()
ada.fit(x_train,y_train)
pred=ada.predict(x_test)
print("r2 score:",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squared error:",mean_squared_error(y_test,pred))
print("Root mean squared error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score: 0.4622011704761785
Mean absolute error: 220360.6355038466
Mean squared error: 70221103821.96092
Root mean squared error: 264992.6486187134
```

```
for i in range(9,16):
    cvs=cross_val_score(ada,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.3014646443843787
cross validation score when cv is 10 = 0.302750477844152
cross validation score when cv is 11 = 0.24307785784428193
cross validation score when cv is 12 = 0.2433367872581751
cross validation score when cv is 13 = 0.22894441640405344
cross validation score when cv is 14 = 0.13614655034812312
cross validation score when cv is 15 = 0.20020940389422137
```

GradientBoostingRegressor

Finding best random state by calling model_select function

```
model_select(GradientBoostingRegressor)
```

```
max score is 0.8620729666427605 at random state 4
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=4)
```

```
gbr=GradientBoostingRegressor()
gbr.fit(x_train,y_train)
pred=gbr.predict(x_test)
print("r2 score:",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squared error:",mean_squared_error(y_test,pred))
print("Root mean squared error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2 score: 0.8649191622997077
Mean absolute error: 88206.18895553515
Mean squared error: 15783180176.931759
Root mean squared error: 125631.12742044369
```

```
for i in range(9,16):
    cvs=cross_val_score(gbr,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

```
cross validation score when cv is 9 = 0.8119233492839667
cross validation score when cv is 10 = 0.8174850150755348
cross validation score when cv is 11 = 0.8134095106786415
cross validation score when cv is 12 = 0.8127054209685873
cross validation score when cv is 13 = 0.81123831477024
cross validation score when cv is 14 = 0.7923510287990877
cross validation score when cv is 15 = 0.803158279707409
```

Xg Boost regressor

```
model_select(xg.XGBRegressor)
```

```
max score is 0.9601971740286308 at random state 40
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=49)
```

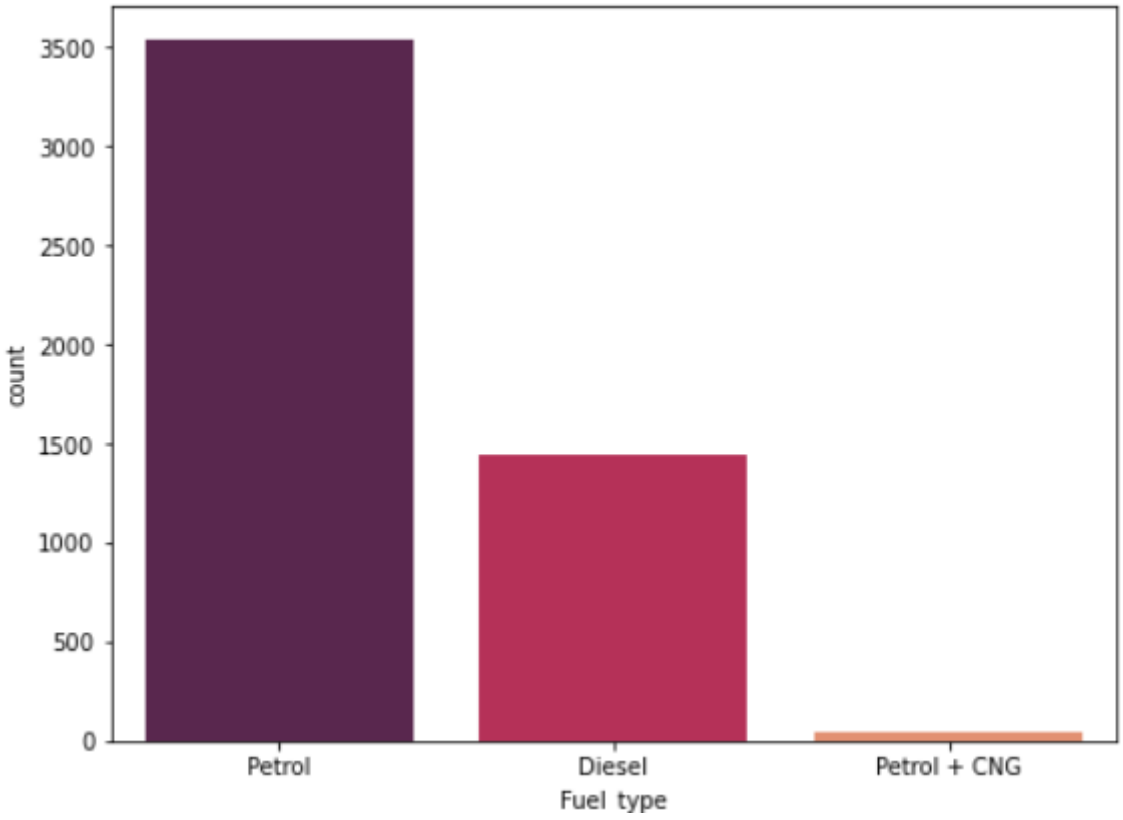
```
XG=xg.XGBRegressor()
XG.fit(x_train,y_train)
pred=XG.predict(x_test)
print("r2 score:",r2_score(y_test,pred))
print("Mean absolute error:",mean_absolute_error(y_test,pred))
print("Mean squared error:",mean_squared_error(y_test,pred))
print("Root mean squared error:",np.sqrt(mean_squared_error(y_test,pred)))
```

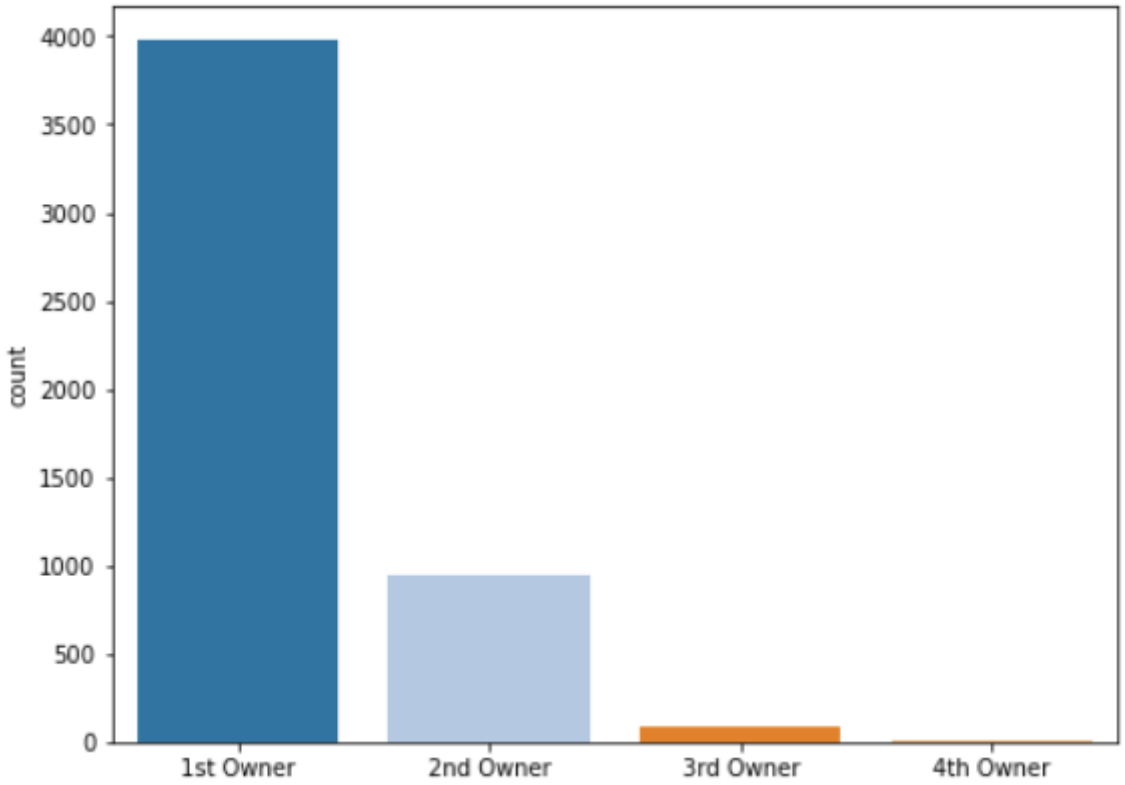
```
r2 score: 0.9560834702248259
Mean absolute error: 42211.923415898615
Mean squared error: 5581440859.501381
Root mean squared error: 74709.0413504375
```

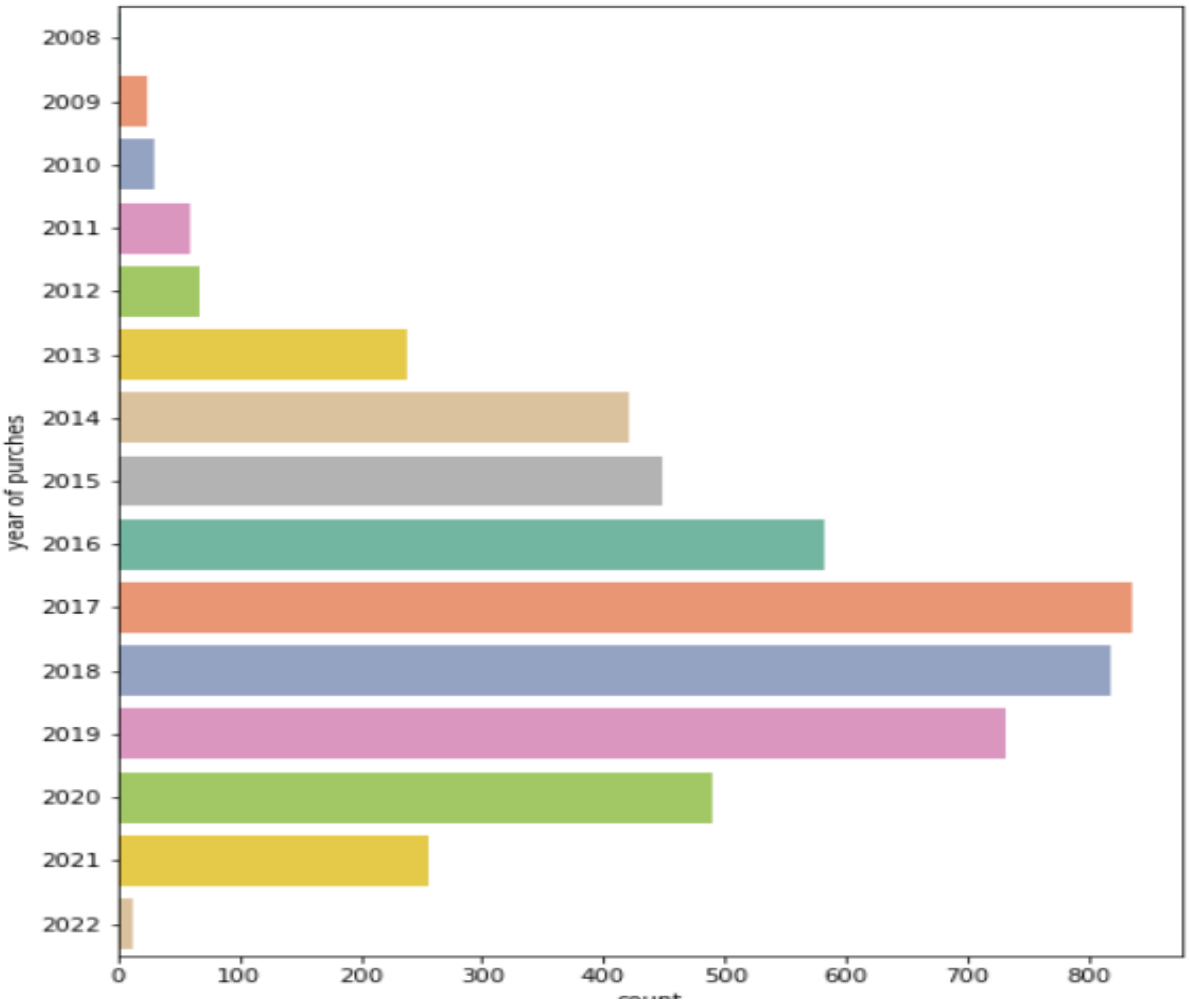
```
for i in range(9,16):
    cvs=cross_val_score(XG,x,y,cv=i)
    print("cross validation score when cv is",i,'=',cvs.mean())
```

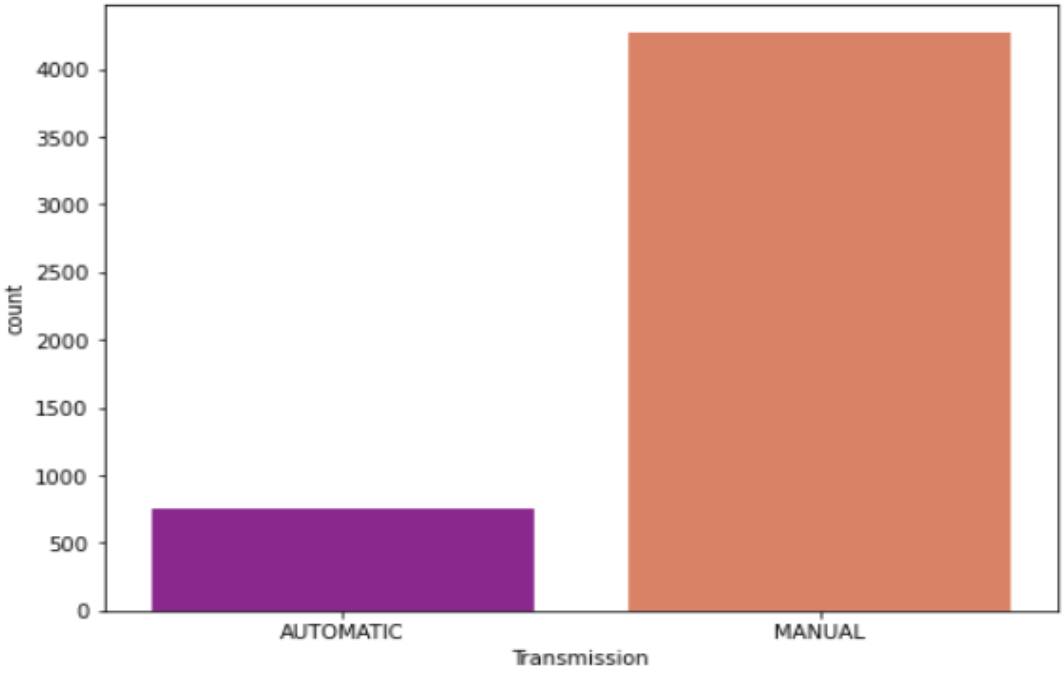
```
cross validation score when cv is 9 = 0.9340325327546434
cross validation score when cv is 10 = 0.9347756270713969
cross validation score when cv is 11 = 0.93495383568659
cross validation score when cv is 12 = 0.9380816737548193
cross validation score when cv is 13 = 0.9380585866314224
cross validation score when cv is 14 = 0.9394404393418794
cross validation score when cv is 15 = 0.9397877482419211
```

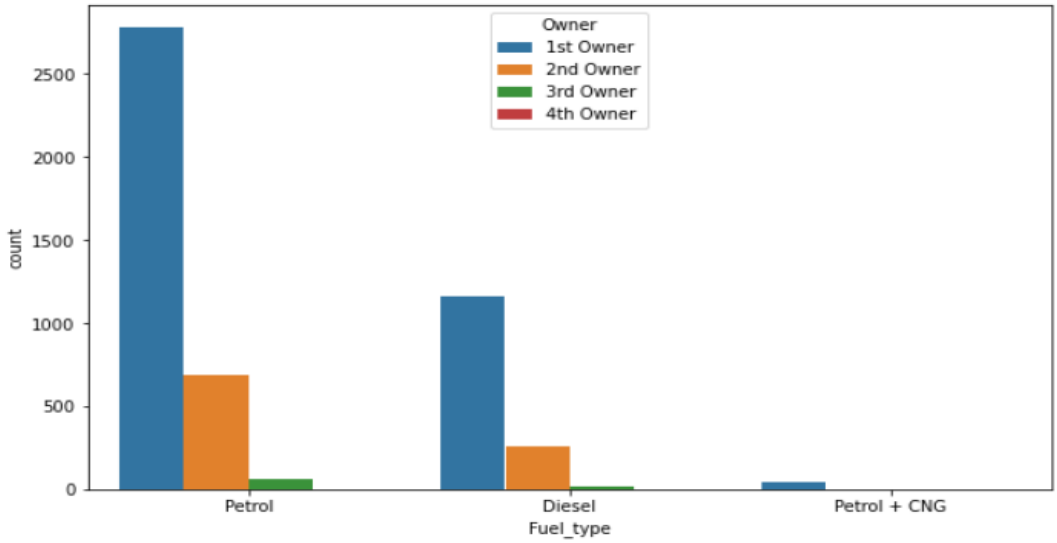
Visualizations

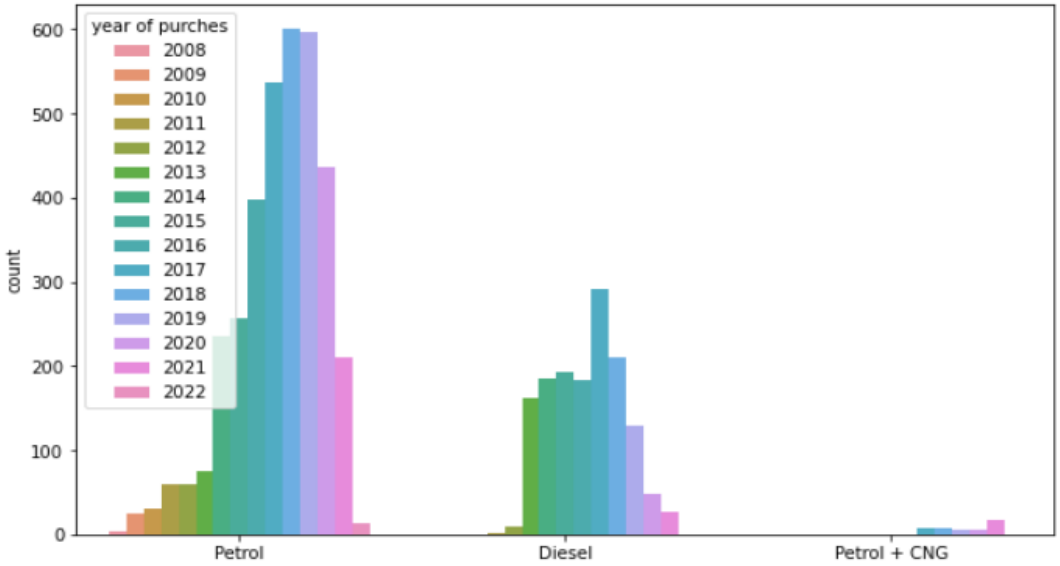


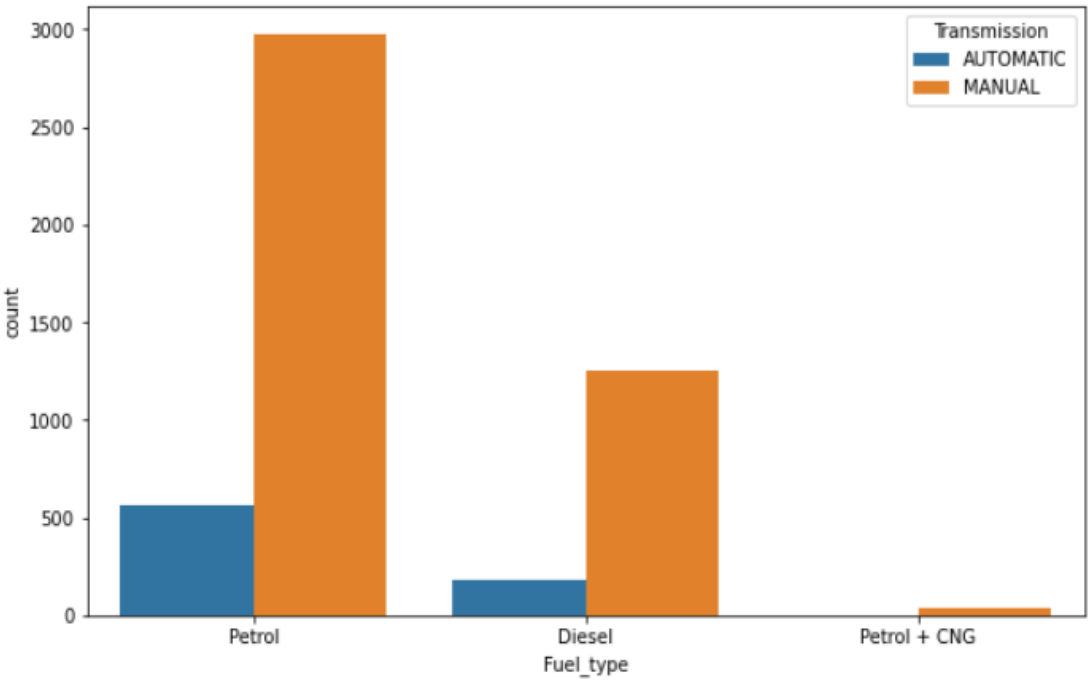



year of purches
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
0
100
200
300
400
500
600
700
800


count
4000
3500
3000
2500
2000
1500
1000
500
0
AUTOMATIC
MANUAL
Transmission


Owner
1st Owner
2nd Owner
3rd Owner
4th Owner
count
2500
2000
1500
1000
500
0
Petrol
Diesel
Petrol + CNG
Fuel_type


year of purches
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
count
600
500
400
300
200
100
0
Petrol
Diesel
Petrol + CNG


Transmission
AUTOMATIC
MANUAL
count
3000
2500
2000
1500
1000
500
0
Petrol
Diesel
Petrol + CNG
Fuel_type

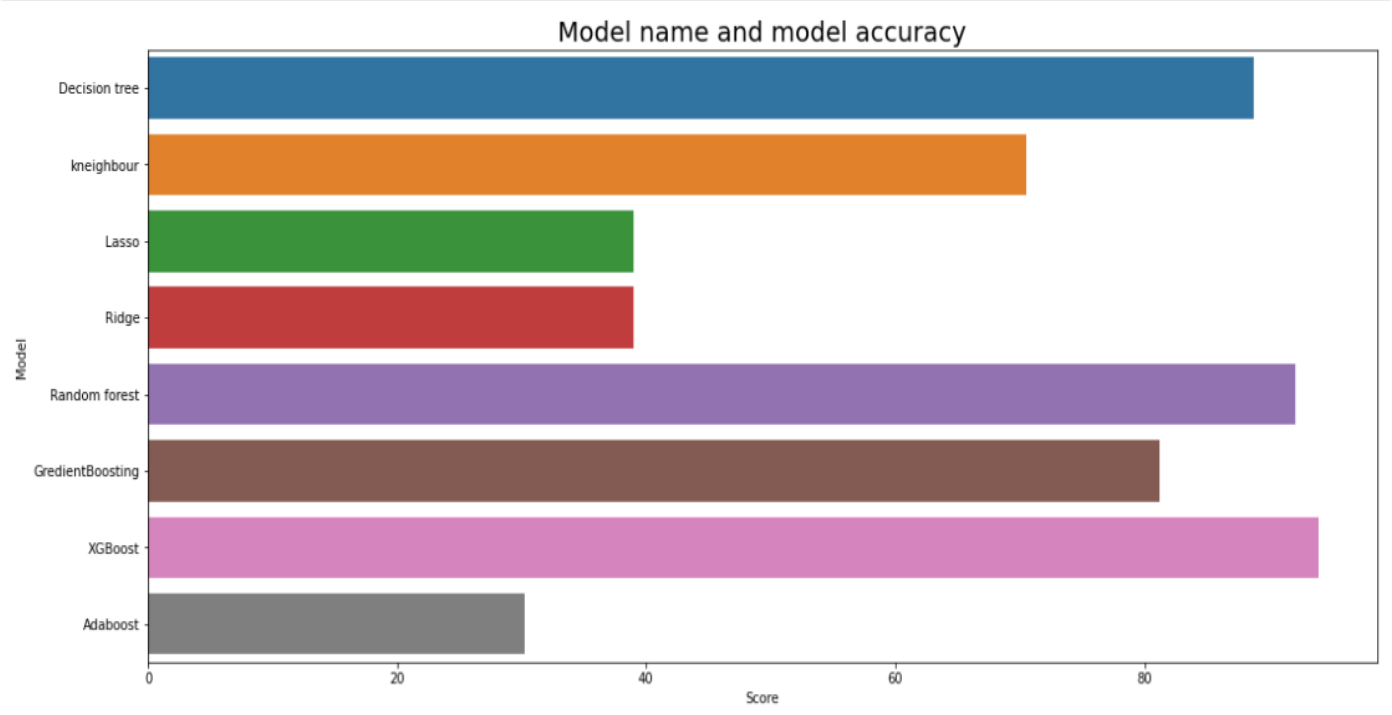
- In our collected dataset, The number of petrol cars are higher(3535), Diesel cars are also being used in a good amount of number(1439), CNG based cars are fewer in number(43).

- Most of the used cars are being sold by the 1st owner, So the chances of price will be increased. Less cars are avilable in the section of 2nd owner and 3rd owner.The count of 4th owner cars are negligible.

- When we see the used car's tranmission, The number of automatic transmission cars is very high and automatic transmission cars are less.

- It is important to know the year of purches, when you go for buying a used car, here we can see that most of the cars are purchesed in the 2013 to 2017.Some cars are avilable in our dataset which had been bought in 2008 to 2012.

CONCLUSION



Best model–

Model name: XG Boost Regressor

r2 score: 95.60

Mean absolute error: 42211.92

Mean squared error: 5581440859.50

Root mean squared error: 74709.04

We got 95.60 % accuracy, which can be considers as a good accuracy for predicting the car price.

About XG boost regressor

XGBoost stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term.

In this paper, we built serveral regression models to predict the price of used car given some of the car features. We eveluated and compared each model to determine the one with

highest performance. We also looked at how some models rank the features according to their importance. In this paper, we followed the data science process starting with getting the data, then cleaning and preprocessing the data, followed by exploring the data and building models, then evaluating the results and communicating them with visualizations.