FLIP ROBO

CAR PRICE PREDICTION

SUBMITTED BY:

MANOJ.I.V

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

INTRODUCTION

- ## Business Problem Framing

  Describe the business problem and how this problem can be related to the real world.

  Answer: The business problem was to scrap the data from the olx website and then use machine learning model for prediction of the car price. In real life this may provide a close cost of the car and car type with model.

- ## Conceptual Background of the Domain Problem

  Describe the domain related concepts that you think will be useful for better understanding of the project.

  Answer: The domain related concepts like Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners etc. The car made and model will make difference as it give the different features in the car. The number of owner also influence the cost based on the maintenance and usage. Driven kilometres influence the cost as the wear and tear of the car. Again manufacturing year and spare parts also makes difference in the price.

- ## Review of Literature

  This is a comprehensive summary of the research done on the topic. The review should enumerate, describe, summarize, evaluate and clarify the research done.

  Answer: We can scrap the details from different websites and then predict the prices.

- ## Motivation for the Problem Undertaken

  Describe your objective behind to make this project, this domain and what is the motivation behind.

Answer: The motivation for the project is to get good condition car for a reasonable price.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

  Answer: The different models used like linear regression, logarithmic regression, K-NN regression, random forest regression, ada-boost regression, EGboost regression, gradient-boost regression and SVC regression.

- Data Sources and their formats

  What are the data sources, their origins, their formats and other details that you find necessary? They can be described here. Provide a proper data description. You can also add a snapshot of the data.

  Answer: The data can be taken by a survey conducted by micro credit company, open source websites like Kaggel etc. The data is in the form of .csv file it may also be in .json or Excel files. In the present form we have taken dataset from olx website.

In [1]:
```
# Importing librabries
import selenium
from selenium import webdriver
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from selenium.webdriver.common.by import By
import time
from bs4 import BeautifulSoup
import requests
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
```

In [22]:
```
# Connecting to the browser
driver=webdriver.Chrome(r'C:\Users\Vishal Manoj\Desktop\Important\pythondata\Sim\chromedriver.exe')
```

In [23]:
```
# Connecting to the webpage
url= 'https://www.olx.in/'
driver.get(url)
```

In [24]:
```
sc=driver.find_element(By.XPATH,'//*[@id="container"]/div[1]/div/div/div[1]/div[2]/div[1]/a')
sc
```

Out[24]:
```
<selenium.webdriver.remote.webelement.WebElement (session="ce2a7f9632cf05bbedef930e3d40dc4d", element="8316a7e8-1ebf-4614-bd40-bbe1ce96e0ca")>
```

In [25]:
```
sc.click()
```

In [19]:
```
dt
```

Out[19]:

	car name	year	price	distance	place	fuel	owner	distance1
0	Volkswagen Tiguan	2017	24,75,000	2017 - 70000.0 km	Amanora	NAN	NAN	70000.0 km
1	Mercedes-Benz New C-Class	2015	27,50,000	2015 - 57,900 km	Tatabad Colony	DIESEL	1st	57,900 km
2	Skoda Yeti	2012	5,99,999	2012 - 103000.0 km	Veterinary Colony	NAN	NAN	103000.0 km
3	Maruti Suzuki Ciaz	2017	7,10,000	2017 - 98,000 km	Sector 29B	DIESEL	1st	98,000 km
4	Hyundai Creta	2018	10,50,000	2018 - 22,900 km	Gahaur	PETROL	1st	22,900 km
...	...	...	...	...	...	...	...	...
217	Hyundai Elantra	2014	6,25,000	2014 - 85,000 km	Anand Vihar	PETROL	NAN	85,000 km
218	Skoda Laura	2009	1,85,000	2009 - 120,000 km	Amrit Varsha Colony	DIESEL	NAN	120,000 km
219	Maruti Suzuki Alto 800	2018	3,40,000	2018 - 1,888 km	Mayur Vihar	PETROL	NAN	1,888 km
220	Mercedes-Benz GLE Class	2019	56,00,000	2019 - 71,000 km	Anchal	--	NAN	71,000 km
221	Maruti Suzuki Vitara Brezza	2018	7,39,999	2018 - 63000.0 km	Preet Vihar	NAN	NAN	63000.0 km

222 rows × 8 columns

In [6]:
```
print('The shape of the dataset is', dt.shape)
```

```
The shape of the dataset is (222, 7)
```

In [7]:
```
# Dataframe columns
dt.columns
```

Out[7]:
```
Index(['car name', 'year', 'price', 'distance', 'place', 'fuel', 'owner'], dtype='object')
```

In [8]:
```
# To find types of data present in the dataset
dt.nunique()
```

Out[8]:
```
car name    105
year         20
price       146
distance    199
place       169
fuel          8
owner         9
dtype: int64
```

- Data Preprocessing Done

  What were the steps followed for the cleaning of the data? What were the assumptions done and what were the next actions steps over that?

  Answer: The place is neglected as it least influencing the car price it is neglected.

In [28]:
```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 222 entries, 0 to 221
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   car name  222 non-null    object
 1   year      222 non-null    int64 
 2   price     222 non-null    object
 3   place     222 non-null    object
 4   fuel      222 non-null    object
 5   owner     222 non-null    object
 6   Distance  222 non-null    object
dtypes: int64(1), object(6)
memory usage: 12.3+ KB
```

In [29]:
```
dt['fuel'].value_counts()
```

Out[29]:
```
DIESEL                          97
PETROL                          87
CNG & HYBRIDS                   15
NAN                             13
CNG                              4
--                               4
LPG                              1
PETROL/COMPRESSED NATURAL GAS    1
Name: fuel, dtype: int64
```

In [30]:
```
dt['fuel'].replace(to_replace='--', value='DIESEL', inplace=True)
```

In [29]:
```
dt['fuel'].value_counts()
```

Out[29]:
```
DIESEL                          97
PETROL                          87
CNG & HYBRIDS                   15
NAN                             13
CNG                              4
--                               4
LPG                              1
PETROL/COMPRESSED NATURAL GAS    1
Name: fuel, dtype: int64
```

In [30]:
```
dt['fuel'].replace(to_replace='--', value='DIESEL', inplace=True)
```

In [31]:
```
dt['fuel'].value_counts()
```

Out[31]:
```
DIESEL                         101
PETROL                          87
CNG & HYBRIDS                   15
NAN                             13
CNG                              4
LPG                              1
PETROL/COMPRESSED NATURAL GAS    1
Name: fuel, dtype: int64
```

```
ownerAs it is catagerical column it was replaced by mode()
```

In [32]:
```
dt['fuel'].replace(to_replace='NAN', value='DIESEL', inplace=True)
```

In [34]:
```
dt['owner'].value_counts()
```

Out[34]:
```
1st       128
2nd        28
NAN        26
First      14
--          9
3rd         7
Second      6
Third       2
Fourth      2
Name: owner, dtype: int64
```

```
ownerAs it is catagerical column it was replaced by mode()
```

In [35]:
```
dt['owner'].replace(to_replace='--', value='1st', inplace=True)
```

In [36]:
```
dt['owner'].replace(to_replace='NAN', value='1st', inplace=True)
```

- Data Inputs- Logic- Output Relationships

  Describe the relationship behind the data input, its format, the logic in between and the output. Describe how the input affects the output.

  Answer: The number of columns are car name, year, price fuel, owner, distance. The car name has characteristics like model, type etc. so this may influence the cost. Manufacturing year also influences the cost of car. Recent manufacturing years costs more and others costs lesser. Owner like 1, 2 or 3 also influences the cost. Distance travelled also influences the price. More the distance travelled lesser the price.

- State the set of assumptions (if any) related to the problem under consideration

  Here, you can describe any presumptions taken by you.

  Answer: Only data from olx of 200 data points are web scraped.

- Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

  Answer: The different hardware and software tools, libraries and packages used are selenium, numpy, pandas, seaborn, matplotlib, sklearn.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

The different models used like linear regression, logarithmic regression, K-NN regression, random forest regression, ada-boost regression, EGboost regression, gradient-boost regression and SVC regression.

- Run and Evaluate selected models

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Answer:

In [161]:
```
# Assignment of X and y
X=dt.drop(['price'], axis=1)
y=dt.price
```

In [162]:
```
# Standardization of the data
scalar= StandardScaler()
X_scaled=scalar.fit_transform(X)
```

In [163]:
```
# Applying machine learning models
# Importing machine learning libraries
# The problem is related regression the regression models are used
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import AdaBoostRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import classification_report, confusion_matrix, mean_squared_error,mean_absolute_error, r2_score
```

In [164]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    li=LinearRegression()
    li.fit(x_train, y_train)
    predtrain=li.predict(x_train)
    predtest=li.predict(x_test)
    print(f"At random state, {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state, 2, the training accuracy is :-11.051225814389335
At random state, 2, the testing accuracy is :0.1012344796786363


At random state, 3, the training accuracy is :-13.092307332795883
At random state, 3, the testing accuracy is :0.14008019268441263


At random state, 4, the training accuracy is :-3.5612601161707085
At random state, 4, the testing accuracy is :-0.03971454707906652


At random state, 5, the training accuracy is :-10.696755730099932
At random state, 5, the testing accuracy is :0.12113805411502332


At random state, 6, the training accuracy is :-10.708339985611913
At random state, 6, the testing accuracy is :0.1060551651717151
```

In [165]:
```
for i in range(1,500):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    re=RandomForestRegressor()
    re.fit(x_train, y_train)
    predtrain=re.predict(x_train)
    predtest=re.predict(x_test)
    print(f"At random state, {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state, 1, the training accuracy is :0.7802983156061528
At random state, 1, the testing accuracy is :0.5655355981867698


At random state, 2, the training accuracy is :0.8578289493301451
At random state, 2, the testing accuracy is :0.4774299616865302


At random state, 3, the training accuracy is :0.8114296436995385
At random state, 3, the testing accuracy is :0.5641128299545135


At random state, 4, the training accuracy is :0.8746341713117103
At random state, 4, the testing accuracy is :0.1812569493287175


At random state, 5, the training accuracy is :0.8024148151593071
At random state, 5, the testing accuracy is :0.6256021718716326
```

```
At random state, 102, the training accuracy is :0.8758329939993625
At random state, 102, the testing accuracy is :0.799398488898343
```

In [166]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    de=DecisionTreeRegressor()
    de.fit(x_train, y_train)
    predtrain=de.predict(x_train)
    predtest=de.predict(x_test)
    print(f"At random state {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state, 4, the testing accuracy is :0.33235665641970447


At random state 5, the training accuracy is :1.0
At random state, 5, the testing accuracy is :0.49400892404211294


At random state 6, the training accuracy is :1.0
At random state, 6, the testing accuracy is :0.5072495726434763


At random state 7, the training accuracy is :1.0
At random state, 7, the testing accuracy is :-1.2475081124622593


At random state 8, the training accuracy is :1.0
At random state, 8, the testing accuracy is :0.08816043602015489
```

```
At random state 788, the training accuracy is :1.0
At random state, 788, the testing accuracy is :0.7518645784175243
```

In [167]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    xg=XGBRegressor()
    xg.fit(x_train, y_train)
    predtrain=xg.predict(x_train)
    predtest=xg.predict(x_test)
    print(f"At random state {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state 1, the training accuracy is :0.9999997450670229
At random state, 1, the testing accuracy is :0.2866737277754353


At random state 2, the training accuracy is :0.9999995611076622
At random state, 2, the testing accuracy is :0.5471279339741839


At random state 3, the training accuracy is :0.999999445999977
At random state, 3, the testing accuracy is :0.7204556183080062


At random state 4, the training accuracy is :0.9999991224577678
At random state, 4, the testing accuracy is :0.3174033615975186


At random state 5, the training accuracy is :0.9999995131887265
At random state, 5, the testing accuracy is :0.6743257827765601
```

```
At random state 584, the training accuracy is :0.9999998055813134
At random state, 584, the testing accuracy is :0.8458231062383801
```

In [170]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    ad=AdaBoostRegressor()
    ad.fit(x_train, y_train)
    predtrain=ad.predict(x_train)
    predtest=ad.predict(x_test)
    print(f"At random state, {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state, 1, the training accuracy is :0.8139086705189053
At random state, 1, the testing accuracy is :0.25657884077097126


At random state, 2, the training accuracy is :0.8125385947541129
At random state, 2, the testing accuracy is :0.40754537276547276


At random state, 3, the training accuracy is :0.7224124671503926
At random state, 3, the testing accuracy is :0.4608426563774738


At random state, 4, the training accuracy is :0.676633429177464
At random state, 4, the testing accuracy is :0.20471251001758284


At random state, 5, the training accuracy is :0.7916415260043229
At random state, 5, the testing accuracy is :0.3134725946841159
```

```
At random state, 590, the training accuracy is :0.793074494122773
At random state, 590, the testing accuracy is :0.7104258136941397
```

In [168]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    kn=KNeighborsRegressor()
    kn.fit(x_train, y_train)
    predtrain=kn.predict(x_train)
    predtest=kn.predict(x_test)
    print(f"At random state {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state 13, the training accuracy is :-0.7919651510733199
At random state, 13, the testing accuracy is :0.12687152650061684


At random state 14, the training accuracy is :-0.9710601669553656
At random state, 14, the testing accuracy is :0.13807683416936323


At random state 15, the training accuracy is :-1.0633809681861943
At random state, 15, the testing accuracy is :-0.15610125792136897


At random state 16, the training accuracy is :-0.8562084494347884
At random state, 16, the testing accuracy is :-1.5232608585714202


At random state 17, the training accuracy is :-0.5642777433176664
At random state, 17, the testing accuracy is :-1.8324488470482603
```

In [171]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    gc=GradientBoostingRegressor()
    gc.fit(x_train, y_train)
    predtrain=gc.predict(x_train)
    predtest=gc.predict(x_test)
    print(f"At random state, {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state, 1, the training accuracy is :0.9851113909448815
At random state, 1, the testing accuracy is :0.13561616322294956


At random state, 2, the training accuracy is :0.9753328668472441
At random state, 2, the testing accuracy is :0.5849667907964766


At random state, 3, the training accuracy is :0.9774208557204899
At random state, 3, the testing accuracy is :0.504911934216784


At random state, 4, the training accuracy is :0.9759226749079913
At random state, 4, the testing accuracy is :0.21679274952997218


At random state, 5, the training accuracy is :0.9733341434277769
At random state, 5, the testing accuracy is :0.5780499125062677
```

```
At random state, 487, the training accuracy is :0.979025623895855
At random state, 487, the testing accuracy is :0.8260123726397017
```

In [172]:
```
for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=i)
    sr=SVR()
    sr.fit(x_train, y_train)
    predtrain=sr.predict(x_train)
    predtest=sr.predict(x_test)
    print(f"At random state {i}, the training accuracy is :{r2_score(predtrain,y_train)}")
    print(f"At random state, {i}, the testing accuracy is :{r2_score(y_test, predtest)}")
    print("\n")
```

```
At random state 1, the training accuracy is :-52732079981.06654
At random state, 1, the testing accuracy is :-0.0752093102363054


At random state 2, the training accuracy is :-44366433006.06865
At random state, 2, the testing accuracy is :-0.07901835975923577


At random state 3, the training accuracy is :-48147328657.71272
At random state, 3, the testing accuracy is :-0.06707252987950985


At random state 4, the training accuracy is :-25272745049.199245
At random state, 4, the testing accuracy is :-0.06175002151025977


At random state 5, the training accuracy is :-54717289671.74118
At random state, 5, the testing accuracy is :-0.05953303358583706
```

In [68]:
```
x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=584)
xg=XGBRegressor()
xg.fit(x_train, y_train)
predtrain=xg.predict(x_train)
predtest=xg.predict(x_test)
print(f"At random state {584}, the training accuracy is :{r2_score(predtrain,y_train)}")
print(f"At random state, {584}, the testing accuracy is :{r2_score(y_test, predtest)}")
```

```
At random state 584, the training accuracy is :0.9999998055813134
At random state, 584, the testing accuracy is :0.8458231062383801
```

In [70]:
```
#Hyperparameter Tuning
from sklearn.model_selection import GridSearchCV
params={'n_estimators': range(2,15),
        'learning_rate': [0.001, 0.01, 0.1, 10, 100],
              'n_jobs': range(2,15)}
```

In [72]:
```
c=GridSearchCV(XGBRegressor(),param_grid=params)
c.fit(x_train, y_train)
print('The beast combination of the parameters are ',c.best_params_)
```

```
The beast combination of the parameters are  {'learning_rate': 0.1, 'n_estimators': 10, 'n_jobs': 2}
```

In [90]:
```
x_train,x_test,y_train, y_test=train_test_split(X_scaled,y,test_size=.20, random_state=584)
xg=XGBRegressor(learning_rate=0.01, n_estimators=10, n_jobs=2)
xg.fit(x_train, y_train)
predtrain=xg.predict(x_train)
predtest=xg.predict(x_test)
print(f"At random state {584}, the training accuracy is :{r2_score(predtrain,y_train)}")
print(f"At random state, {584}, the testing accuracy is :{r2_score(y_test, predtest)}")
```

```
At random state 584, the training accuracy is :0.9992329390126463
At random state, 584, the testing accuracy is :0.811913309936757
```

In [91]:
```
# Calculating the MSE, RMSE and MAE for K-NN model
mse=mean_absolute_error(y_test, predtest)
rmse=np.sqrt(mse)
print('The MAE is', mean_absolute_error(y_test, predtest))
print ('The MSE is', mse, 'and RMSE is', rmse)
```

```
The MAE is 330812.5414930555
The MSE is 330812.5414930555 and RMSE is 575.1630564396983
```

- Key Metrics for success in solving problem under consideration

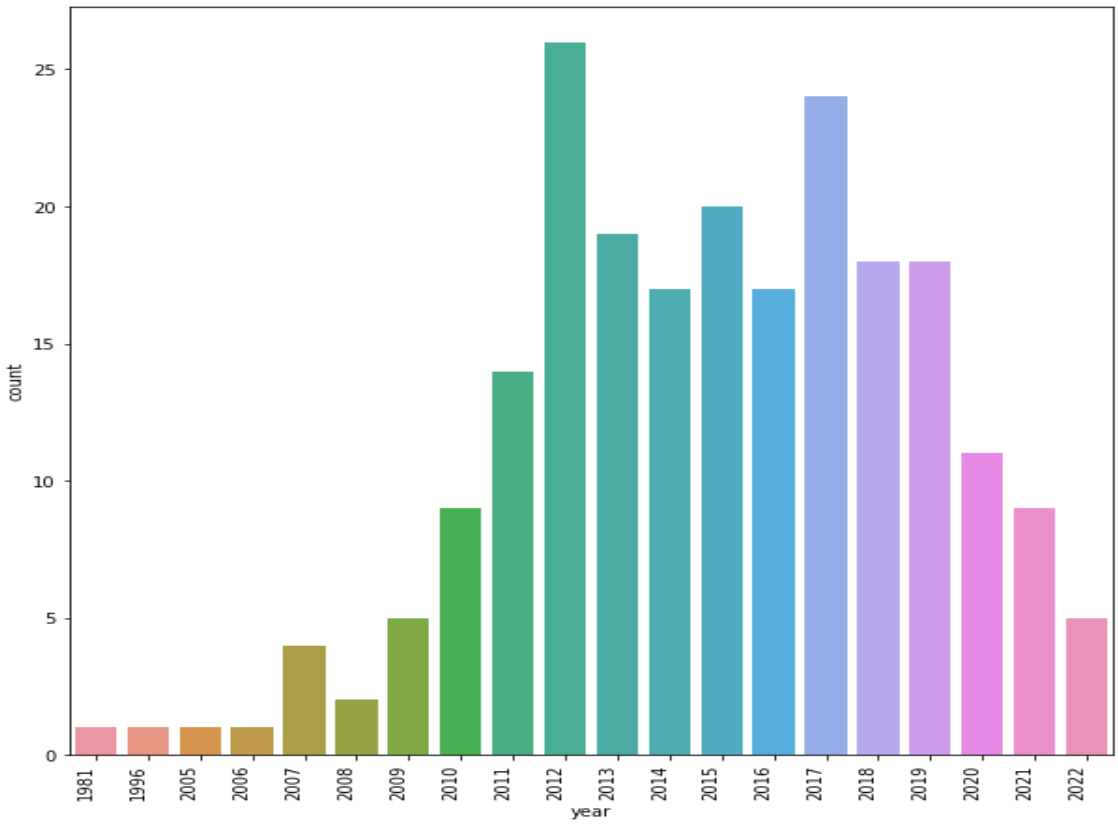
  What were the key metrics used along with justification for using it? You may also include statistical metrics used if any.

- Visualizations

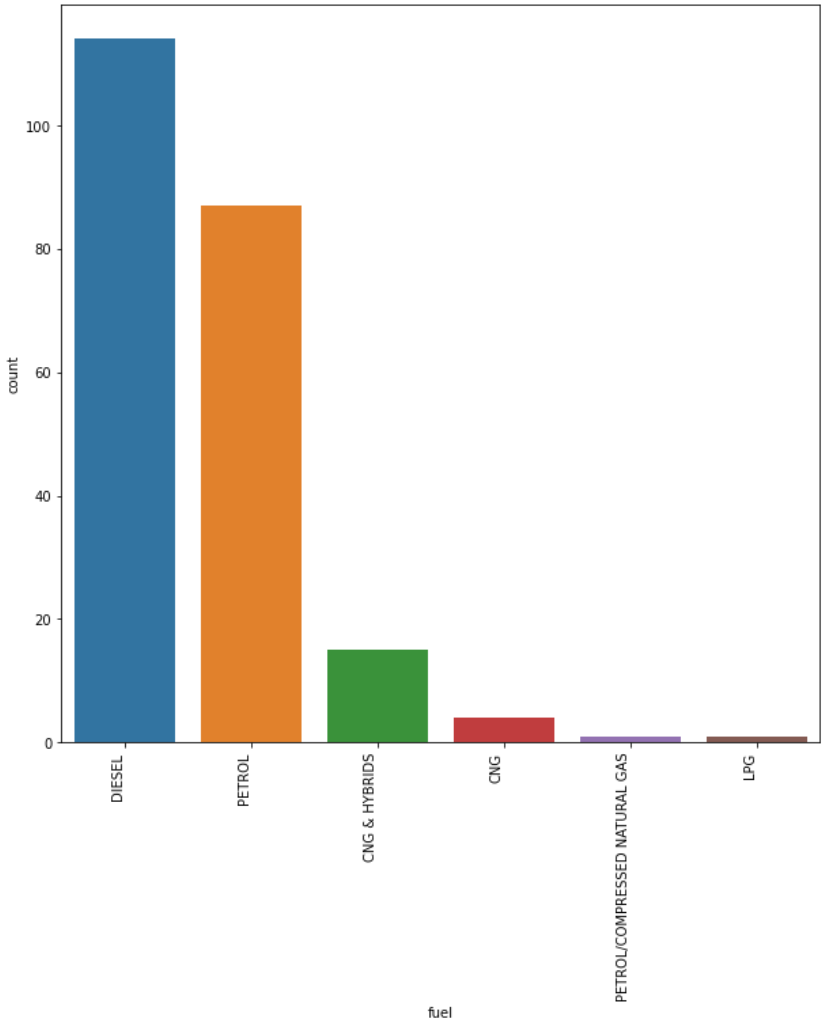
  Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

  If different platforms were used, mention that as well.

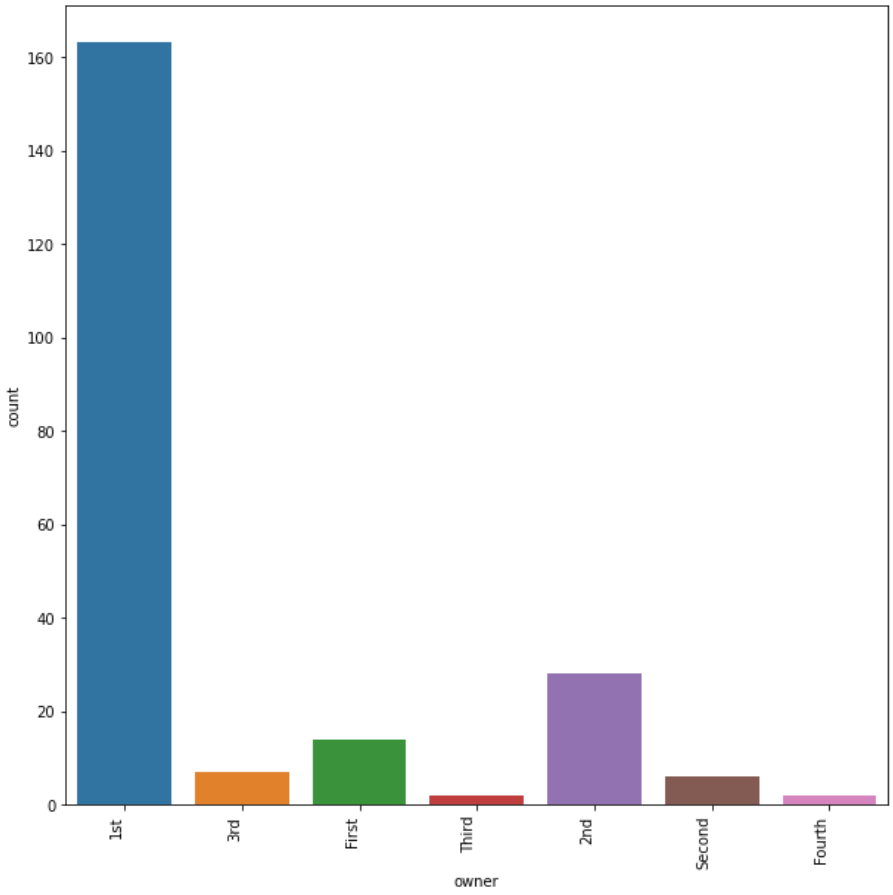
  Answer:



From the graph we can see that 2012 and 2017 manufactured vehicles are for sales in the market.



The diesel fuel is the most selling vehicles in the olx website followed by petrol and CNG and hybrids.



The 1st owner vehicles are the most selling followed by 2nd owner vehicles.

- Interpretation of the Results

  Give a summary of what results were interpreted from the visualizations, pre-processing and modelling.

  Answer: We can see that 2012 and 2017 manufactured vehicles, 1st owner vehicles and diesel fuel are for sales.

CONCLUSION

- Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem.

Answer: We can see that 2012 and 2017 manufactured vehicles, 1st owner vehicles and diesel fuel are for sales.

- Learning Outcomes of the Study in respect of Data Science

  List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

  Answer: It can be seen the model was web scrapped. Then it was made as a dataset which was used for modelling. The data is cleaned and the different graphs were plotted for explorative data analysis. Then the label encoder is used and categorical variables were converted to numerical format. Different model were used and XgBoost model was seen to be most efficient. After hyper-parameter tuning is at random state 584, the training accuracy is 0.9992329390126463. At random state, 584, the testing accuracy is 0.811913309936757

- Limitations of this work and Scope for Future Work

  What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results?

  Answer: More data can be scrapped and applied for prediction. The solution provided only has the accuracy of 81% approximately by using neural networks this accuracy can be increased.