CAR PRICE PREDICTION

Submitted by:
JAMEEL KHAN

ACKNOWLEDGMENT

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

I wish to express my sincere gratitude to Mr. Shubham Yadav, for providing me an opportunity to do my internship and project work in “FLIPROBO”.

It gives me immense pleasure in presenting this project report on “Car Price Prediction Model”. It has been my privilege to have a team of project guide who have assisted me from the commencement of this project. The success of this project is a result of sheer hard work, and determination put in by me with the help of You Tube videos, references taken from Kaggle.com, skikit-learn.org.. To know more about micro finance, I read

https://www.geeksforgeeks.org/

https://github.com/

https://www.mckinsey.com/

https://www.counterpointresearch.com/

I hereby take this opportunity to add a special note of thanks for to Mr. Shubham Yadav, who undertook to act as my mentor despite his many other professional commitments. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. Without his insight,support this project wouldn’t have reached fruitfulness.

The project is dedicated to all those people of Fliprobo, Datatrained who helped me while doing this project.

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, the previous ML models are not performing well.

- Conceptual Background of the Domain Problem

  A good knowledge of after sales market of cars is necessary. What makes a car valuable will be key.

- Review of Literature

  Not a lot of research is available on car prices after covid-19 impact.

  https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/how-consumers-behavior-in-car-buying-and-mobility-changes- amid-covid-19

  https://www.counterpointresearch.com/weekly-updates-covid-19-impact- global-automotive-industry/

- Motivation for the Problem Undertaken
  Due to covid-19 the car market has changed a lot, some cars have shot up in popularity and some gone down in price.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

Inbuilt function such as standardizing and log will be used in tackling this problem.

R-square is a comparison of residual sum of squares *(SS_{res})* with total sum of squares*(SS_{tot})*. Total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where SS_{res} is the residual sum of squares and SS_{tot} is the total sum of squares.

R-square is the main metric which I will use in this regression analysis.

Concordance index was also used. The concordance index or c-index is a metric to evaluate the predictions made by an algorithm. It is defined as the proportion of concordant pairs divided by the total number of possible evaluation pairs.

- Data Sources and their formats
  The data was scraped from cars24 website; data was scraped for more than 20 cities where prices differ.

	Unnamed: 0	Brand	Model	Year	Fuel	KM driven	Price
0	0	KIA	SELTOS	2020	Petrol	5,822 km	₹11,33,599
1	1	Maruti	Swift	2012	Diesel	1,18,117 km	₹3,43,199
2	2	Hyundai	Verna	2013	Diesel	53,517 km	₹4,82,299
3	3	Maruti	Swift	2013	Diesel	49,944 km	₹3,54,299
4	4	Maruti	Swift	2016	Diesel	1,06,080 km	₹4,49,299
...	...	...	...	...	...	...	...
5151	5151	Maruti	Alto	2020	Petrol	4,179 km	₹3,30,799
5152	5152	Maruti	Alto	2020	Petrol	38,222 km	₹3,61,899
5153	5153	Maruti	Alto	2018	Petrol	24,853 km	₹2,96,799
5154	5154	Maruti	Alto	2018	Petrol	50,463 km	₹3,32,599
5155	5155	Maruti	Celerio	2018	Petrol	41,352 km	₹4,40,199

5156 rows × 7 columns

It consists of total 7 different columns which are as follows out of which our target variable column named as Price in it

Data Pre-processing Done

After loading of dataset in Jupyter, I first checked the shape of the dataset then I moved further in next step for checking any null values present in it.
Now moving further, unnamed 0 column is of no use in the dataset as it gives no insights with our dataset so we will drop this column.
The years were extracted from the name of the car which contained lot of information.

Numerical variables were converted to integer type (from string) so I could perform deeper analysis on them.

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

The main assumption is that there is no selection bias in the data which we have.

This is because we have cars from varying years and varying city; each city doesn't have equal amount of data.

Here we can see the count of data per city.

Data Inputs- Logic- Output Relationships

The input data for the processing and getting the output is converted in to the numerical forms or data words, in to the numerical form. It feed to the model in

the form of series (one by one) which analyses by the model providing the certain score through the medium of performance metrics.

Hardware and Software Requirements and Tools Used

Data Science task should be done with sophisticated machine with high end machine configuration. The machine which I'm currently using is powered by intel core i5 processor with 4GB of RAM. With this above-mentioned configuration, I managed to work with the data set in Jupyter Notebook which help us to write Python codes. As I'm using low configuration machine so it
took more time than usual to execute codes. The library used for the assignment are Numpy, Pandas, Matplotlib, Seaborn, Scikit learn

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sn

import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.model_selection import train_test_split,cross_val_score

#importing models
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
from xgboost import XGBRegressor

#importing metrics
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
```

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor

Identification of possible problem-solving approaches (methods)

These are the following approaches I have used here: Importing and displaying insights of the data using pandas Dataframe. Analyzing the data and use proper pre-processing techniques to clean it. Applied log transformation to remove the skewness from the data. Lastly, scaled the data, after all pre-processing were completed.

As I went through the dataset, I found some outliers and skewness are present in the dataset. The skewness was also reduced using power-transformation wherever applicable. There were certain columns which had least importance with our target variable, hence those were dropped. After data cleaning and data transformation, data visualization was done to represent data graphically. At last, the most important part was to build model for the data set.

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

```
z=np.abs(zscore(df))
z.shape
```

```
(5156, 6)
```

```
df1=df[(z<3).all(axis=1)]
print(df.shape)
print(df1.shape)
```

```
(5156, 6)
(4930, 6)
```

```
loss_percent = (5156-4930)/5156*100
print(loss_percent)
```

```
4.383242823894492
```

```
from sklearn.preprocessing import power_transform
```

```
df1[['Fuel']]=power_transform(df1[['Fuel']])
```

```
df1.skew()
```

```
Brand        0.261146
Model        0.369618
Year        -0.658344
Fuel         0.013844
KM driven    1.003594
Price        0.968566
dtype: float64
```

After removing of outliers using zscore I checked my dataset shape as if how much of data It is been removed. We check this thing because the dataset is highly precious and we cannot lose a high percent of data.
After removing of outliers, I got to that the data is properly scaled so for scaling the values of columns we will be using MinMaxScaler from Sklearn. As it will scale down values of columns in a similar way

```
from sklearn.preprocessing import MinMaxScaler

sc=MinMaxScaler()
x=sc.fit_transform(x)
```

Testing of Identified Approaches (Algorithms)

Following are the algorithms used for the training and testing: -

- **Linear Regression**
- **RandomForestRegressor**
- **DecisionTreeRegressor**
- **GradientBoostingRegressor**
- **KNeighborsRegressor**
- **SVR**
- **Lasso**
- **Ridge**
- **ElasticNet**
- **AdaBoostRegressor**
- **XGBRegressor**

```
#Creating list of models and another list mapped to their names
models=[KNeighborsRegressor(),SVR(),LinearRegression(),Lasso(),Ridge(),ElasticNet(),DecisionTreeRegressor(),
       RandomForestRegressor(),AdaBoostRegressor(),GradientBoostingRegressor(),XGBRegressor()]

names=['KNeighborsRegressor','SVR','LinearRegression','Lasso','Ridge','ElasticNet','DecisionTreeRegressor',
       'RandomForestRegressor','AdaBoostRegressor','GradientBoostingRegressor','XGBRegressor']
```

Model Function

```
def createmodels(model_list,independent,dependent,n):
    xtrain,xtest,ytrain,ytest=train_test_split(independent,dependent,test_size=0.25,random_state=12)
    name=[]
    meanabs=[]
    meansqd=[]
    rootmeansqd=[]
    r2=[]
    mcv=[]

    #Creating models
    for i,model in enumerate(model_list):
        model.fit(xtrain,ytrain)
        p=model.predict(xtest)
        score=cross_val_score(model,independent,dependent,cv=10)

        #Calculating scores of the model and appending them to a list
        name.append(n[i])
        meanabs.append(np.round(mean_absolute_error(p,ytest),4))
        meansqd.append(np.round(mean_squared_error(p,ytest),4))
        rootmeansqd.append(np.round(np.sqrt(mean_squared_error(p,ytest)),4))
        r2.append(np.round(r2_score(p,ytest),2))
        mcv.append(np.round(np.mean(score),4))

    #Creating Dataframe
    data=pd.DataFrame()
    data['Model']=name
    data['Mean Absolute Error']=meanabs
    data['Mean Squared Error']=meansqd
    data['Root Mean Squared Error']=rootmeansqd
    data['R2 Score']=r2
    data['Mean of Cross validaton Score']=mcv
    data.set_index('Model',inplace = True)
    return data
```

```
createmodels(models,x,y,names)
```

Model	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	R2 Score	Mean of Cross validaton Score
KNeighborsRegressor	47054.7038	7.419576e+09	86136.9635	0.83	0.8156
SVR	176225.4275	5.517308e+10	234889.5114	-4161205.58	-0.0537
LinearRegression	136993.5675	3.507853e+10	187292.6362	-0.88	0.3226
Lasso	136993.1846	3.507847e+10	187292.4794	-0.88	0.3226
Ridge	137002.8494	3.508339e+10	187305.6057	-0.91	0.3228
ElasticNet	175703.5602	5.128227e+10	226455.8802	-456.96	0.0239
DecisionTreeRegressor	14845.3277	2.885387e+09	53715.7964	0.95	0.8743
RandomForestRegressor	18364.5754	2.493327e+09	49933.2272	0.95	0.9108
AdaBoostRegressor	151802.4696	3.261868e+10	180606.4183	-1.04	0.3700
GradientBoostingRegressor	67729.0097	9.915740e+09	99577.8074	0.72	0.7962
XGBRegressor	24401.2314	2.419185e+09	49185.2102	0.95	0.9109

Hyperparameter Tuning

We have applied Hyperparameter tuning on Random Forest and Xtreme Gradient Boost as they are giving the best performance for our dataset

Random Forest Regressor

```
from sklearn.model_selection import RandomizedSearchCV
```

```
params={'n_estimators':[100, 300, 500, 700],
        'min_samples_split':[1,2,3,4],
        'min_samples_leaf':[1,2,3,4],
        'max_depth':[None,1,2,3,4,5,6,7,8]}

g=RandomizedSearchCV(RandomForestRegressor(),params,cv=10)

g.fit(xtrain,ytrain)
```

```
RandomizedSearchCV(cv=10, estimator=RandomForestRegressor(),
                   param_distributions={'max_depth': [None, 1, 2, 3, 4, 5, 6, 7,
                                                      8],
                                        'min_samples_leaf': [1, 2, 3, 4],
                                        'min_samples_split': [1, 2, 3, 4],
                                        'n_estimators': [100, 300, 500, 700]})
```

```
print(g.best_estimator_)
print(g.best_params_)
print(g.best_score_)
```

```
RandomForestRegressor(max_depth=7, n_estimators=500)
{'n_estimators': 500, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': 7}
0.8021445929768415
```

```
m=RandomForestRegressor(max_depth=7, min_samples_leaf=1, min_samples_split=2,n_estimators=500)
m.fit(xtrain,ytrain)
p=m.predict(xtest)
score=cross_val_score(m,x,y,cv=10)
```

```
print('Mean Absolute Error is',np.round(mean_absolute_error(p,ytest),4))
print('Mean Squared Error is',np.round(mean_squared_error(p,ytest),4))
print('Root Mean Squared Error is',np.round(np.sqrt(mean_squared_error(p,ytest)),4))
print('R2 Score is',np.round(r2_score(p,ytest),4)*100)
print('Mean of cross validaton Score is',np.round(np.mean(score)*100,4))
```

```
Mean Absolute Error is 68703.8115
Mean Squared Error is 9803856992.1754
Root Mean Squared Error is 99014.4282
R2 Score is 72.76
Mean of cross validaton Score is 77.1946
```

Xtreme Gradient Boost

```
params={
 "learning_rate"    : [0.001,0.05, 0.10, ] ,
 "max_depth"        : [ 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]
    }

g=RandomizedSearchCV(XGBRegressor(verbosity=0),params,cv=10)

g.fit(xtrain,ytrain)
```

```
print(g.best_estimator_)
print(g.best_params_)
print(g.best_score_)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0.0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.05, max_delta_step=0, max_depth=12,
             min_child_weight=7, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=0)
{'min_child_weight': 7, 'max_depth': 12, 'learning_rate': 0.05, 'gamma': 0.0, 'colsample_bytree': 0.7}
0.9274233031672431
```

```
m=XGBRegressor(colsample_bytree=0.7,gamma=0.0,learning_rate=0.05,max_depth=12,min_child_weight=7)
m.fit(xtrain,ytrain)
p=m.predict(xtest)
score=cross_val_score(m,x,y,cv=10)
```

```
print('Mean Absolute Error is',np.round(mean_absolute_error(p,ytest),4))
print('Mean Squared Error is',np.round(mean_squared_error(p,ytest),4))
print('Root Mean Squared Error is',np.round(np.sqrt(mean_squared_error(p,ytest)),4))
print('R2 Score is',np.round(r2_score(p,ytest),4)*100)
print('Mean of cross validaton Score is',np.round(np.mean(score)*100,4))
```

```
Mean Absolute Error is 33457.5115
Mean Squared Error is 3510077058.4174
Root Mean Squared Error is 59245.9033
R2 Score is 91.97999999999999
Mean of cross validaton Score is 91.1656
```

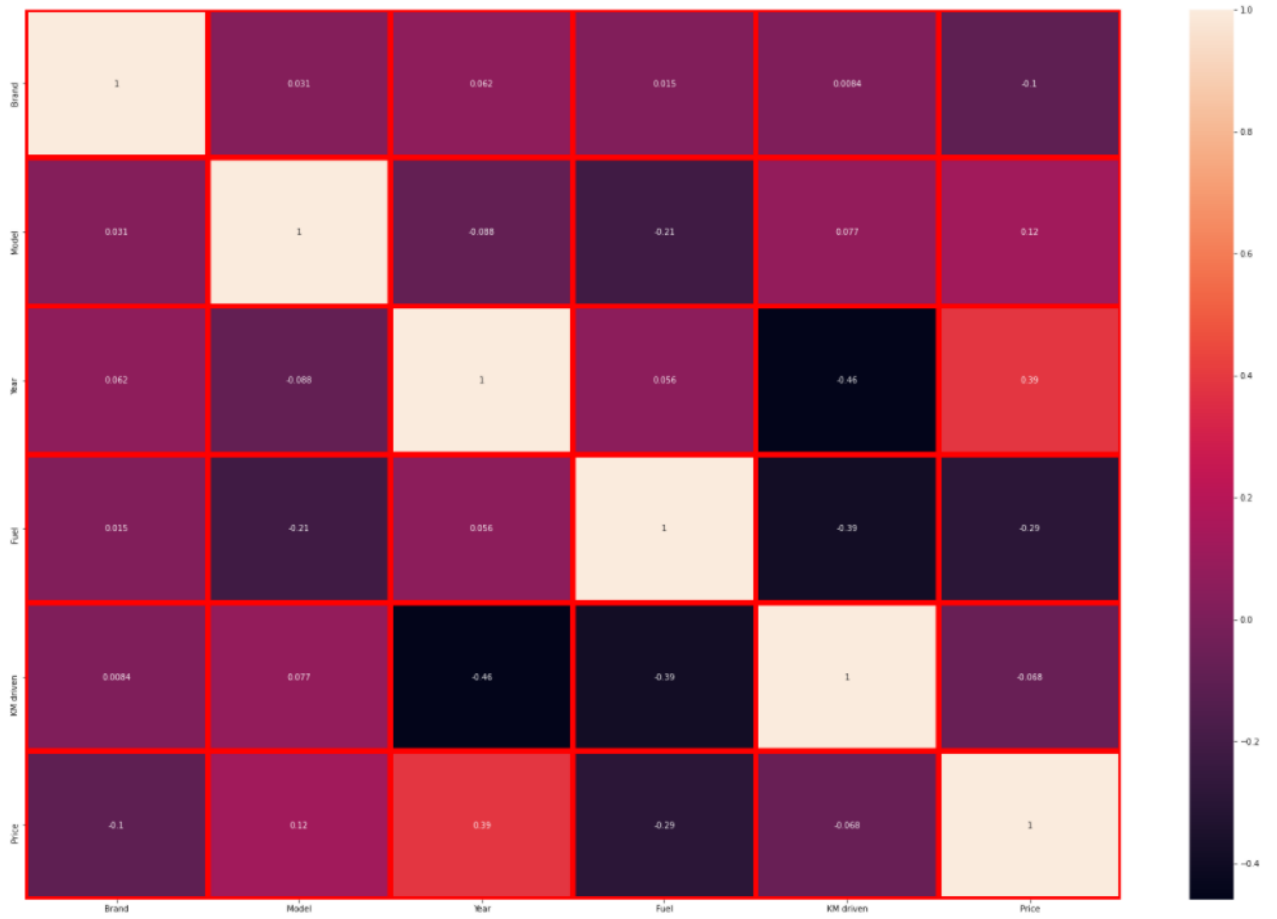
Visualizations

The plots used to visualize the data are: -

- Heatmap
- Count plot
- Pie plot
- Dist plot
- Hist plot

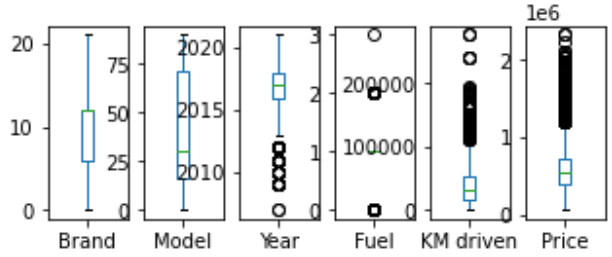
```
plt.figure(figsize=[30,20])
sn.heatmap(cor,annot=True,linewidths=6,linecolor='r')
```

```
<AxesSubplot:>
```



```
df.plot(kind="box",subplots=True,layout=(2,7))
```

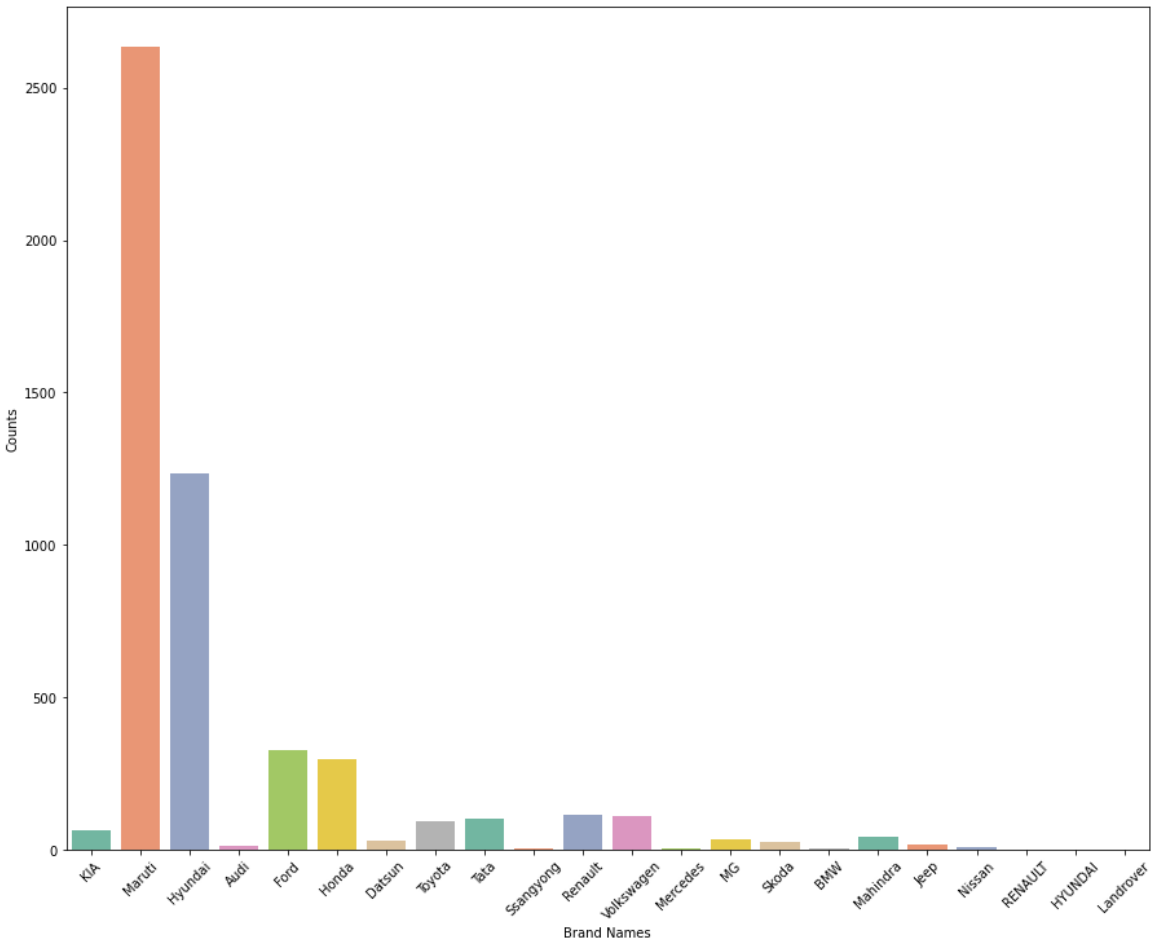
```
Brand           AxesSubplot(0.125,0.536818;0.0945122x0.343182)
Model        AxesSubplot(0.238415,0.536818;0.0945122x0.343182)
Year         AxesSubplot(0.351829,0.536818;0.0945122x0.343182)
Fuel         AxesSubplot(0.465244,0.536818;0.0945122x0.343182)
KM driven    AxesSubplot(0.578659,0.536818;0.0945122x0.343182)
Price        AxesSubplot(0.692073,0.536818;0.0945122x0.343182)
dtype: object
```



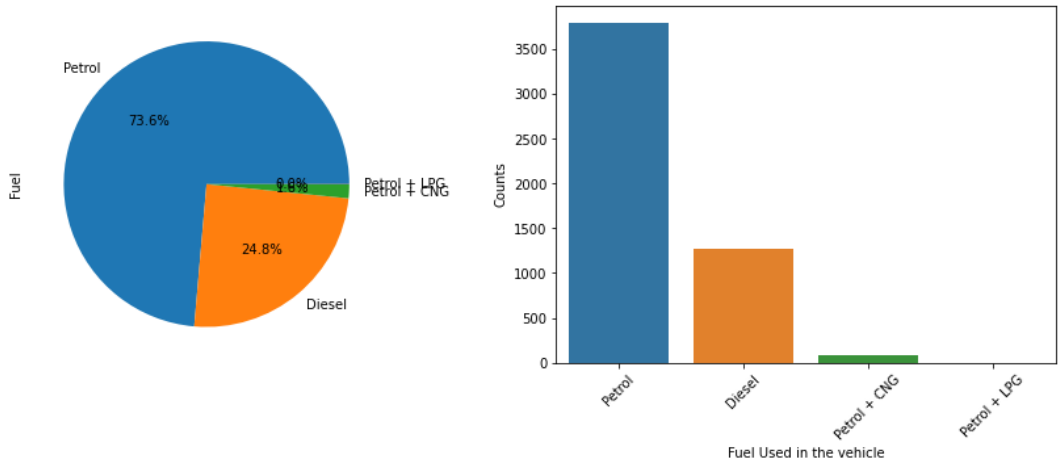
EDA

```
plt.figure(figsize=(15,12))
sn.countplot(x='Brand',data=df,palette='Set2')
plt.xlabel("Brand Names")
plt.xticks(rotation='45')
plt.ylabel("Counts")
plt.show()

print(df['Brand'].value_counts())
```


Counts
2500
2000
1500
1000
500
0
KIA
Maruti
Hyundai
Audi
Ford
Honda
Datsun
Toyota
Tata
Ssangyong
Renault
Volkswagen
Mercedes
MG
Skoda
BMW
Mahindra
Jeep
Nissan
RENAULT
HYUNDAI
Landrover
Brand Names

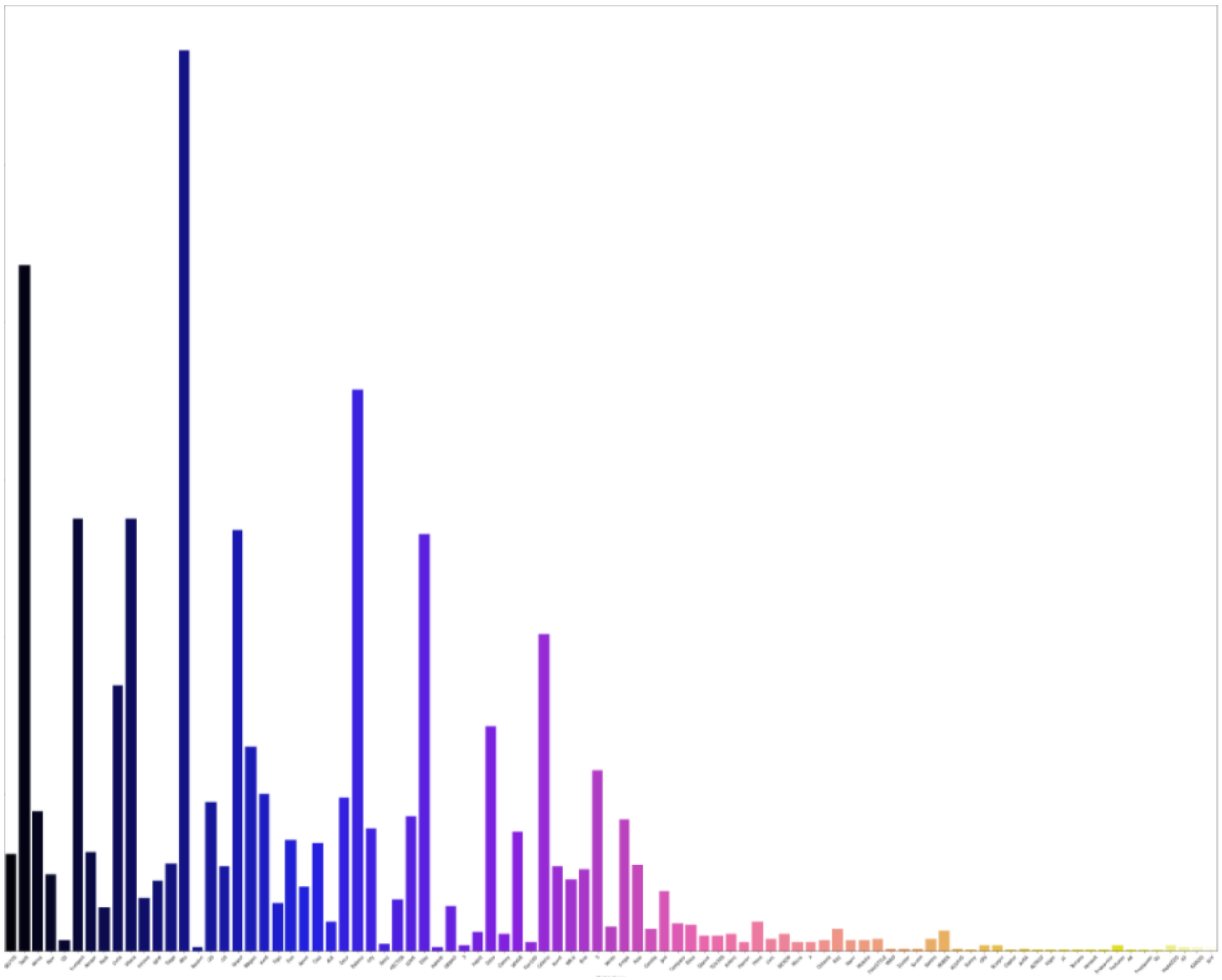
```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
df['Fuel'].value_counts().plot.pie(autopct='%1.1f%%')
plt.subplot(1,2,2)
sn.countplot(df['Fuel'])
plt.xlabel("Fuel Used in the vehicle")
plt.xticks(rotation='45')
plt.ylabel("Counts")
plt.show()
df['Fuel'].value_counts()
```




```
Petrol          3796
Diesel          1278
Petrol + CNG      81
Petrol + LPG       1
Name: Fuel, dtype: int64
```

```
plt.figure(figsize=(50,40))
sn.countplot(x='Model',data=df,palette='gnuplot2')
plt.xlabel("Model of cars")
plt.xticks(rotation='45')
plt.ylabel("Counts")
plt.show()

print(df['Model'].value_counts())
```

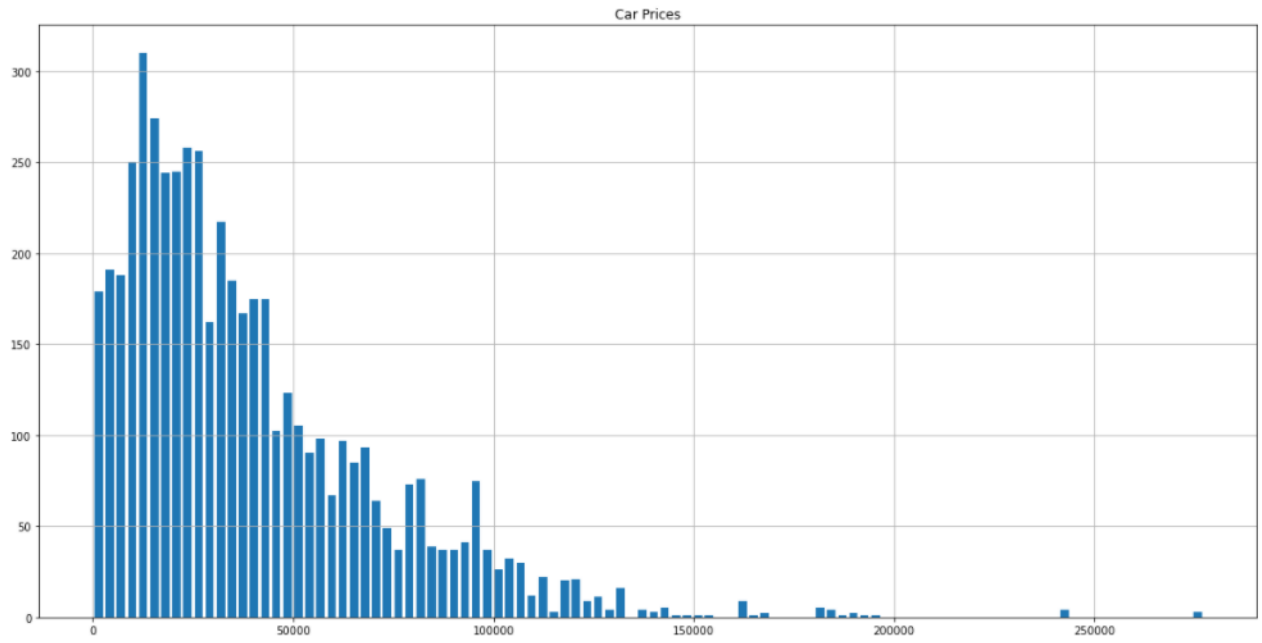


```
Alto          573
Swift         436
Baleno        357
Vitara        275
Ecosport      275
              ...
Terrano         1
Kiger           1
Renault         1
Go              1
ALTROZ          1
Name: Model, Length: 91, dtype: int64
```

Alto is the model which is most sold

S

```
df['KM driven'].hist(bins=100,figsize=(20,10),rwidth = 0.75)
plt.title('Car Prices')
plt.show()
```



CONCLUSION

Key Findings and Conclusions of the Study

The main component on which the price of a car brand, model, the year which car was bought; the mileage on the car etc.

The price also depends on which city the car was registered, as some cities have different tax rates and restrictions. Eg Delhi NCR has 10-year limit on diesel cars and 15 years on petrol cars, but no other city has such restrictions

Most of the used cars on the market for resale are from Maruti and its models. The top 3 models for resale are Swift, Alto and WagonR. Most of the vehicles runs on petrol than any other fuel. The vehicles are mostly driven below 400000 km

The main component on which the price of a car brand, model, the year which car was bought; the mileage on the car etc.

Learning Outcomes of the Study in respect of Data Science

Decision Tree regression works best for this particular data set, hyper parameter tuning was performed and optimal parameters were found.

EDA is very powerful in understanding the data and pre-processing it before feeding it to the algorithm. Statistical methods work the best.

Limitations of Decision Tree regression works best for this particular data set, hyper parameter tuning was performed and optimal parameters were found.

Machine learning algorithms are very helpful in solving real world problems. If properly used they can predict lot of things which can be useful for our daily life. The visualization of the data are used to understand the insight of the dataset. We can create assumptions and conclusion with different types of visualizations. Hence, we can conclude that, machine learning algorithms are a useful tool to create models which help us to overcome real world challenges and give an idea to make future planning according to the situations.

Limitations of this work and Scope for Future Work

Post covid-19 car market is still evolving, and it will keep evolving for the foreseeable future. The algorithms will need to keep changing to keep up with the evolution.

Thank you