CAR PRICE PREDICTION USING SIMPLE REGRESSION METHOD

Data Analytics –Project Report

GROUP NAME : MECHTIQ
SRN1 : PES1UG20ME160
SRN2 : PES1UG20ME175
SRN3 : PES1UG20ME214

Car Price Prediction Using Auto Regression Model

Objective :

- To Find the Car Price based on the Type of Fuel , Kilometres Driven , Year of the model using Auto Regression Model.

Methodology :

- Quiker Car Sales Data is used .
- Linear , Non Linear and Classification Models are built and their accuracies are noted and the best one is chosen.
- The data is Pre-processed to obtain the columns in a usable format .
- A plot is obtained to visualize how dependent and independent variables behave.
- A Simple Linear Regression model is built to predict the values for the varying independent values .
- Various Plots are plotted between the dependent and independent variables to visualize the models filtering .
- The pipe.predict() function is used to predict the values of the Car Price
  Depending upon the Type_Fuel, Kms_Driven, Year it was brought.
- Plot is Obtained for the same.
- Finding the model with a random state of Train_Test Split where it gives the maximum accuracy.
- All Model are compared to get the better model.

Dataset :

- Quiker Car Sales GitHub

URL :

- **https://github.com/rajtilakls2510/car_price_predictor/blob/master/quikr_car.csv**

Columns :

- Column 1 : Name
- Column 2 : Company
- Column 3 : Year
- Column 4 : Price
- Column 5 : Kms_Driven
- Column 6 : Fuel_Type

PreProcessing of Data :

- Importing Data :
  Code :

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib as mpl
        mpl.style.use('ggplot')

In [2]: car=pd.read_csv('quikr_car.csv')

In [3]: car.head()

Out[3]:
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel

```
In [4]: car.shape
Out[4]: (892, 6)
```

- Since the Year Column's Dtype was object and it had a large number of non-numeric values, we had to alter it.
  So we used .astype(int) to change it from Object to integer value.
- Code :

Year has many non-year values

```
In [7]: car=car[car['year'].str.isnumeric()]
```

Year is in object. Change to integer

```
In [8]: car['year']=car['year'].astype(int)
```

```
In [9]: car.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 842 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        842 non-null    object
 1   company     842 non-null    object
 2   year        842 non-null    int64
 3   Price       842 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: int64(1), object(5)
memory usage: 46.0+ KB
```

- Even the Price Column had so many Garbage values in that like ‘Ask for Price ’ so had remove it .
- Code :

Price has Ask for Price

```
car=car[car['Price']!='Ask For Price']
```

- Price had commas in them which and its Dtype was object so we need to remove the commas as well as convert it to Integer.
- Code :

Price has commas in its prices and is in object ¶

```
car['Price']=car['Price'].str.replace(',','').astype(int)
```

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 819 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        819 non-null    object
 1   company     819 non-null    object
 2   year        819 non-null    int64 
 3   Price       819 non-null    int64 
 4   kms_driven  819 non-null    object
 5   fuel_type   816 non-null    object
dtypes: int64(2), object(4)
memory usage: 44.8+ KB
```

- The kms in the Kms Driven columns' values needed to be eliminated, and their Dtype was likewise an object.
- Therefore, using the.split() and.replace() methods, we must remove the kms from the end of the values before converting them to an integer.
- Code :

KmsDriven has object values with KMS at last.

```
car['kms_driven']=car['kms_driven'].str.split().str.get(0).str.replace(',','')
```

It has Nan values and Two rows have Petrol in them

```
car=car[car['kms_driven'].str.isnumeric()]
```

```
car['kms_driven']=car['kms_driven'].astype(int)
```

- Fuel_Type Columns had Nan values in that and we removed using ~
- Code :

FuelType has Nan values

```
car=car[~car['fuel_type'].isna()]
```

- Car Name was not correctly presented so we need change it using .split() function and .slice() function.
  Code :

Changing Car names & keeping only the first three words

```
car['name']=car['name'].str.split().str.slice(start=0,stop=3).str.join(' ')
```

In the End we removed the Index Column so that it doesn't effect the Model
Code :

```
car=car.reset_index(drop=True)
```

Going Through The Cleaned Data :

- Car Info :
- Code :

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   name         816 non-null    object
 1   company      816 non-null    object
 2   year         816 non-null    int64 
 3   Price        816 non-null    int64 
 4   kms_driven   816 non-null    int64 
 5   fuel_type    816 non-null    object
dtypes: int64(3), object(3)
memory usage: 38.4+ KB
```

- Every Column is assigned with correct Dtype.

- Car Describe :
- Code :

```
car.describe(include='all')
```

	name	company	year	Price	kms_driven	fuel_type
count	816	816	816.000000	8.160000e+02	816.000000	816
unique	254	25	NaN	NaN	NaN	3
top	Maruti Suzuki Swift	Maruti	NaN	NaN	NaN	Petrol
freq	51	221	NaN	NaN	NaN	428
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863	NaN
std	NaN	NaN	4.002992	4.751844e+05	34297.428044	NaN
min	NaN	NaN	1995.000000	3.000000e+04	0.000000	NaN
25%	NaN	NaN	2010.000000	1.750000e+05	27000.000000	NaN
50%	NaN	NaN	2013.000000	2.999990e+05	41000.000000	NaN
75%	NaN	NaN	2015.000000	4.912500e+05	56818.500000	NaN
max	NaN	NaN	2019.000000	8.500003e+06	400000.000000	NaN

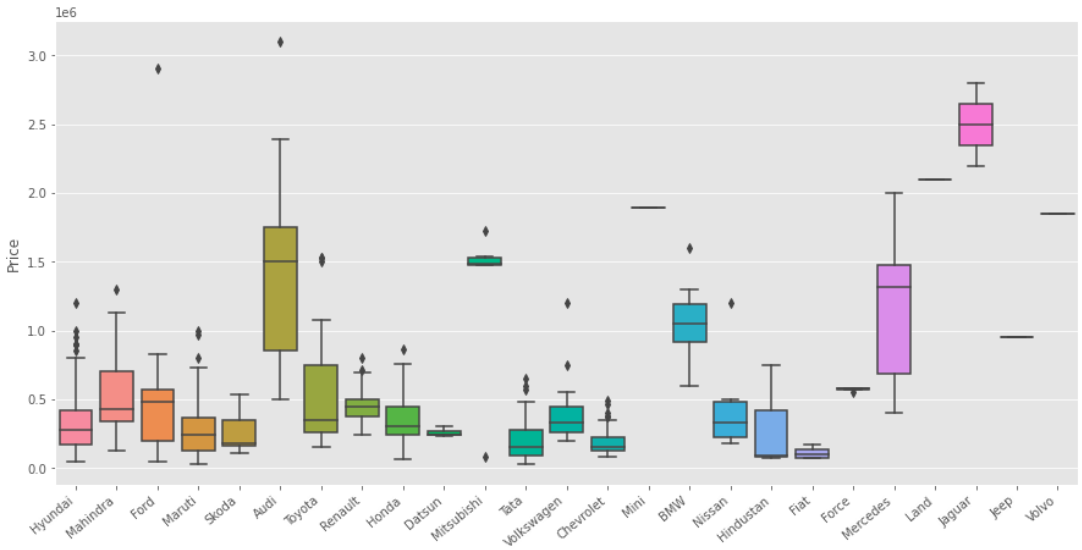
- Finding The Outliers and removing it .
- There was an Outlier so we removed it.
- Code :

```
car = car[car['Price']<6000000].reset_index(drop=True)
```

Visualization of Data using Graphs

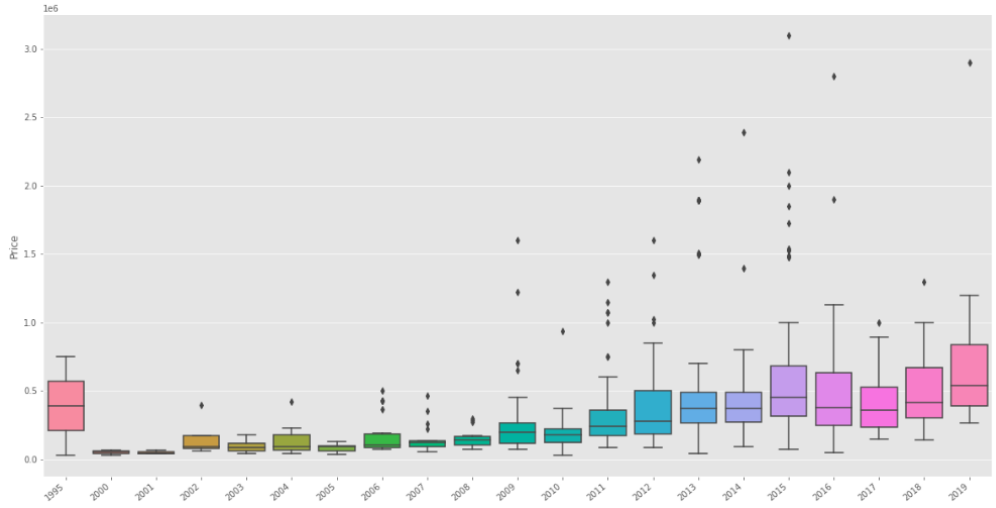
- Checking Relation Between Company with Price
- Graph :

```
plt.subplots(figsize=(15,7))
ax=sns.boxplot(x='company',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```



- Checking Relation Between Year with Price .
- Graph :

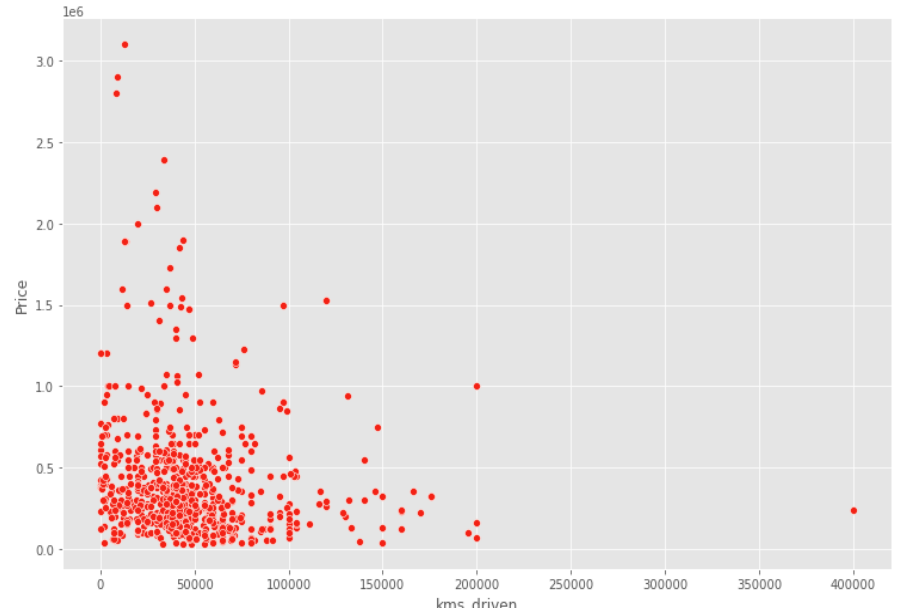
```
plt.subplots(figsize=(20,10))
ax=sns.boxplot(x='year',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```



- Checking Relation Between Kms_Driven with Price
- Graph :

```
sns.relplot(x='kms_driven',y='Price',data=car,height=7,aspect=1.5)
```

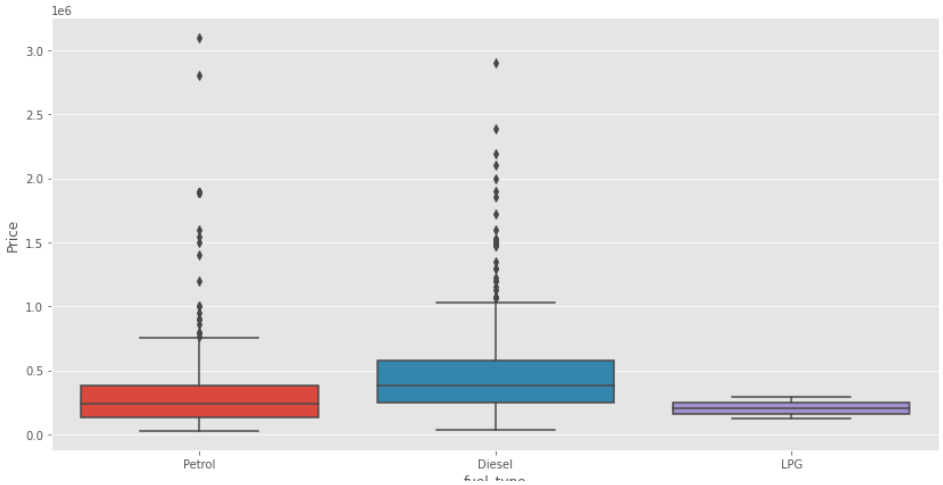
```
<seaborn.axisgrid.FacetGrid at 0x120b188b0>
```



- Checking Relation Between Fuel_Type with Price
- Graph :

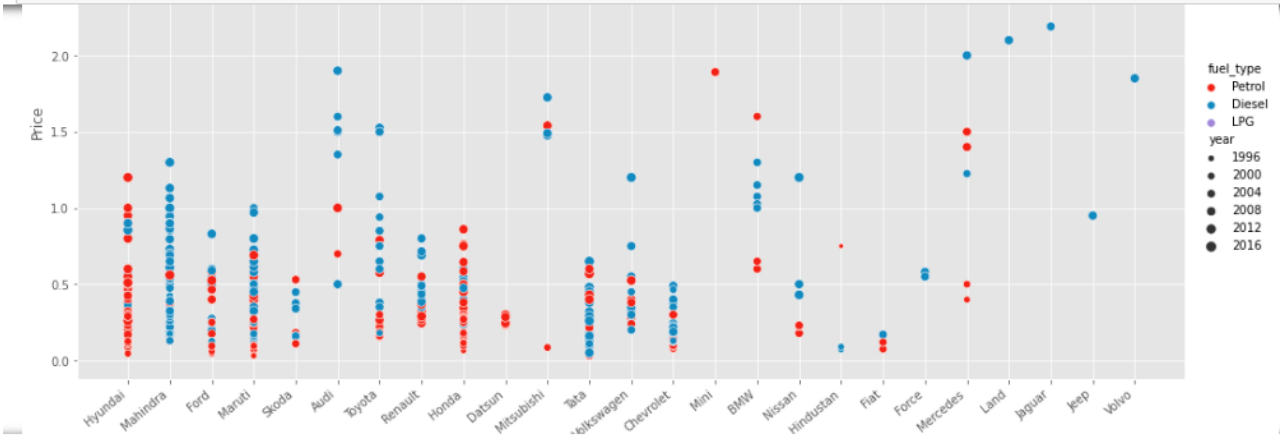
```
plt.subplots(figsize=(14,7))
sns.boxplot(x='fuel_type',y='Price',data=car)
```

```
<AxesSubplot:xlabel='fuel_type', ylabel='Price'>
```



- Checking Relation Between Fuel_Type ,Year and Company
- Graph :

```
ax=sns.relplot(x='company',y='Price',data=car,hue='fuel_type',size='year',height=7,aspect=2)
ax.set_xticklabels(rotation=40,ha='right')
```



Draw Conclusions From Graph :

The conclusion drawn is that this data which is going to be used to build the model is free of any outliers or errors.

So we can say that this dataset is now free to be used to build our required model.

Extracting Training and Testing Data .

Extracting Training Data

```
X=car[['name','company','year','kms_driven','fuel_type']]
y=car['Price']
```

- Applying Train and Test Split.
- Using sklearn.model_selection and importing train_test_split
- Code :

Applying Train Test Split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

- Using OneHotEncoder to handle Categorical Values.
- Code :

Creating an OneHotEncoder object to contain all the possible categories

```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
```

Creating a column transformer to transform categorical columns

```
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
```

Model Creation

Linear Regression :

Code :

Linear Regression Model

```
lr=LinearRegression()
```

```
pipe=make_pipeline(column_trans,lr)
```

Fitting the model

```
pipe.fit(X_train,y_train)
```

Methodology :

- **Why Linear Regression Model is used and why not any other model?**
- We can use regression analysis to gauge how strongly two variables are related. Regression analysis can tell you how much of the total variability in the data is explained by your mode using statistical metrics like R-squared / adjusted R-squared.
- Code :

- Creating Prediction Variable for the measure of Accuracy :

```
: y_pred=pipe.predict(X_test)
```

Accuracy Metric :
Adjusted R^2 :

Why Adjusted R^2 value : Because adjusted R-squared can produce a more accurate picture of the correlation between two variables, it may be preferred over R-squared.

Checking R2 Score

```
: r2_score(y_test,y_pred)
: 0.8264604574769455
```

Methodology :

- As we know that the R^2 accuracy depends on the random state.
- So Finding the Correct Random State is very important .

- So we are using for loop and finding the perfect random state which gives maximum accuracy .
- Code :

Finding the model with a random state of TrainTestSplit where the model was found to give almost 0.92 as r2_score

```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=LinearRegression()
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
655
```

```
scores[np.argmax(scores)]
```

```
0.9200884351998813
```

Prediction From Training Dataset

Methodology :

- After finding the perfect Random State which gives the max value for our accuracy metric

- We need to Predict the Price Value using Train Set which depends on independent value like Fuel Type ,KM Driven, and the Year it was Brought .
- So we use pipe.predict() function which takes independent variables Names as one argument and its Values as another argument .
- Finding the corresponding Price for the above values.

Code :

```
pipe.predict(pd.DataFrame(columns=['name','company','year','kms_driven','fuel_type'],
data=np.array(['Toyota Corolla','Toyota',2006,5100,'Petrol']).reshape(1,5)))
```

Here we have taken Toyota Corolla as an Example and corresponding values are given for Year it was brought , KM Driven , Type of Fuel used.

Predicted Value For the above :

```
array([428873.635597])
```

As a result, the 2006 Toyota Corolla, which has been driven about 5100 miles and has a petrol fuel type, is worth about 428873.63 (four hundred twenty-eight thousand eight hundred seventy-three).

Thank You