Import necessary Libraries

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
```

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)
```

In [5]:

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: object(6)
memory usage: 41.9+ KB
```

In [6]:

```
car["year"].unique() #The year column is with mixed dtypes
```

Out[6]:

```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '...', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o...', 'ture', 'emi', 'car', 'able', 'no.',
       'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'e...', 'D...',
       ', Ac', 'go .', 'k...', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r...', 'zest'], dtype=object)
```

In [7]:

```
car["Price"].unique() #3rd entry is in Object data type which need to be removed and all entries should be in int.
```

Out[7]:

```
array(['80,000', '4,25,000', 'Ask For Price', '3,25,000', '5,75,000',
       '1,75,000', '1,90,000', '8,30,000', '2,50,000', '1,82,000',
       '3,15,000', '4,15,000', '3,20,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,90,000', '95,000', '1,80,000', '3,85,000', '1,05,000',
       '6,50,000', '6,89,999', '4,48,000', '5,49,000', '5,01,000',
       '4,89,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
       '4,99,999', '2,35,000', '2,49,999', '14,75,000', '3,95,000',
       '2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
       '1,10,000', '4,48,999', '18,91,111', '1,59,500', '3,44,999',
       '4,49,999', '8,65,000', '6,99,000', '3,75,000', '2,24,999',
       '12,00,000', '1,95,000', '3,51,000', '2,40,000', '90,000',
       '1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,90,000',
       '1,35,000', '16,00,000', '7,01,000', '2,65,000', '5,25,000',
       '3,72,000', '6,35,000', '5,50,000', '4,85,000', '3,29,500',
       '2,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
       '4,50,000', '2,70,000', '1,58,400', '1,79,000', '1,25,000',
       '2,99,000', '1,50,000', '2,75,000', '2,85,000', '3,40,000',
       '70,000', '2,89,999', '8,49,999', '7,49,999', '2,74,999',
       '9,84,999', '5,99,999', '2,44,999', '4,74,999', '2,45,000',
       '1,69,500', '3,70,000', '1,68,000', '1,45,000', '98,500',
       '2,09,000', '1,85,000', '9,00,000', '6,99,999', '1,99,999',
       '5,44,999', '1,99,000', '5,40,000', '49,000', '7,00,000', '55,000',
       '8,95,000', '3,55,000', '5,65,000', '3,65,000', '40,000',
       '4,00,000', '3,30,000', '5,80,000', '3,79,000', '2,19,000',
       '5,19,000', '7,30,000', '20,00,000', '21,00,000', '14,00,000',
       '3,11,000', '8,55,000', '5,35,000', '1,78,000', '3,00,000',
       '2,55,000', '5,49,999', '3,80,000', '57,000', '4,10,000',
       '2,25,000', '1,20,000', '59,000', '5,99,000', '6,75,000', '72,500',
       '6,10,000', '2,30,000', '5,20,000', '5,24,999', '4,24,999',
       '6,44,999', '5,84,999', '7,99,999', '4,44,999', '6,49,999',
       '9,44,999', '5,74,999', '3,74,999', '1,30,000', '4,01,000',
       '13,50,000', '1,74,999', '2,39,999', '99,999', '3,24,999',
       '10,74,999', '11,30,000', '1,49,000', '7,70,000', '30,000',
       '3,35,000', '3,99,000', '65,000', '1,69,999', '1,65,000',
       '5,60,000', '9,50,000', '7,15,000', '45,000', '9,40,000',
       '1,55,555', '15,00,000', '4,95,000', '8,00,000', '12,99,000',
       '5,30,000', '14,99,000', '32,000', '4,05,000', '7,60,000',
       '7,50,000', '4,19,000', '1,40,000', '15,40,000', '1,23,000',
       '4,98,000', '4,80,000', '4,88,000', '15,25,000', '5,48,900',
       '7,25,000', '99,000', '52,000', '28,00,000', '4,99,000',
       '3,81,000', '2,78,000', '6,90,000', '2,60,000', '90,001',
       '1,15,000', '15,99,000', '1,59,000', '51,999', '2,15,000',
       '35,000', '11,50,000', '2,69,000', '60,000', '4,30,000',
       '85,00,003', '4,01,919', '4,90,000', '4,24,000', '2,05,000',
       '5,49,900', '3,71,500', '4,35,000', '1,89,700', '3,89,700',
       '3,60,000', '2,95,000', '1,14,990', '10,65,000', '4,70,000',
       '48,000', '1,88,000', '4,65,000', '1,79,999', '21,90,000',
       '23,90,000', '10,75,000', '4,75,000', '10,25,000', '6,15,000',
       '19,00,000', '14,90,000', '15,10,000', '18,50,000', '7,90,000',
       '17,25,000', '12,25,000', '68,000', '9,70,000', '31,00,000',
       '8,99,000', '88,000', '53,000', '5,68,500', '71,000', '5,90,000',
       '7,95,000', '42,000', '1,89,000', '1,62,000', '35,999',
       '29,00,000', '39,999', '50,500', '5,10,000', '8,60,000',
       '5,00,001'], dtype=object)
```

In [8]:

```
car["kms_driven"].unique()
```

Out[8]:

```
array(['45,000 kms', '40 kms', '22,000 kms', '28,000 kms', '36,000 kms',
       '59,000 kms', '41,000 kms', '25,000 kms', '24,530 kms',
       '60,000 kms', '30,000 kms', '32,000 kms', '48,660 kms',
       '4,000 kms', '16,934 kms', '43,000 kms', '35,550 kms',
       '39,522 kms', '39,000 kms', '55,000 kms', '72,000 kms',
       '15,975 kms', '70,000 kms', '23,452 kms', '35,522 kms',
       '48,508 kms', '15,487 kms', '82,000 kms', '20,000 kms',
       '68,000 kms', '38,000 kms', '27,000 kms', '33,000 kms',
       '46,000 kms', '16,000 kms', '47,000 kms', '35,000 kms',
       '30,874 kms', '15,000 kms', '29,685 kms', '1,30,000 kms',
       '19,000 kms', nan, '54,000 kms', '13,000 kms', '38,200 kms',
       '50,000 kms', '13,500 kms', '3,600 kms', '45,863 kms',
       '60,500 kms', '12,500 kms', '18,000 kms', '13,349 kms',
       '29,000 kms', '44,000 kms', '42,000 kms', '14,000 kms',
       '49,000 kms', '36,200 kms', '51,000 kms', '1,04,000 kms',
       '33,333 kms', '33,600 kms', '5,600 kms', '7,500 kms', '26,000 kms',
       '24,330 kms', '65,480 kms', '28,028 kms', '2,00,000 kms',
       '99,000 kms', '2,800 kms', '21,000 kms', '11,000 kms',
       '66,000 kms', '3,000 kms', '7,000 kms', '38,500 kms', '37,200 kms',
       '43,200 kms', '24,800 kms', '45,872 kms', '40,000 kms',
       '11,400 kms', '97,200 kms', '52,000 kms', '31,000 kms',
       '1,75,430 kms', '37,000 kms', '65,000 kms', '3,350 kms',
       '75,000 kms', '62,000 kms', '73,000 kms', '2,200 kms',
       '54,870 kms', '34,580 kms', '97,000 kms', '60 kms', '80,200 kms',
       '3,200 kms', '0,000 kms', '5,000 kms', '588 kms', '71,200 kms',
       '1,75,400 kms', '9,300 kms', '56,758 kms', '10,000 kms',
       '56,450 kms', '56,000 kms', '32,700 kms', '9,000 kms', '73 kms',
       '1,60,000 kms', '84,000 kms', '58,559 kms', '57,000 kms',
       '1,70,000 kms', '80,000 kms', '6,821 kms', '23,000 kms',
       '34,000 kms', '1,800 kms', '4,00,000 kms', '48,000 kms',
       '90,000 kms', '12,000 kms', '69,900 kms', '1,66,000 kms',
       '122 kms', '0 kms', '24,000 kms', '36,469 kms', '7,800 kms',
       '24,695 kms', '15,141 kms', '59,910 kms', '1,00,000 kms',
       '4,500 kms', '1,29,000 kms', '300 kms', '1,31,000 kms',
       '1,11,111 kms', '59,466 kms', '25,500 kms', '44,005 kms',
       '2,110 kms', '43,222 kms', '1,00,200 kms', '65 kms',
       '1,40,000 kms', '1,03,553 kms', '58,000 kms', '1,20,000 kms',
       '49,800 kms', '100 kms', '81,876 kms', '6,020 kms', '55,700 kms',
       '18,500 kms', '1,80,000 kms', '53,000 kms', '35,500 kms',
       '22,134 kms', '1,000 kms', '8,500 kms', '87,000 kms', '6,000 kms',
       '15,574 kms', '8,000 kms', '55,800 kms', '56,400 kms',
       '72,160 kms', '11,500 kms', '1,33,000 kms', '2,000 kms',
       '88,000 kms', '65,422 kms', '1,17,000 kms', '1,50,000 kms',
       '10,750 kms', '6,800 kms', '5 kms', '9,800 kms', '57,923 kms',
       '30,201 kms', '6,200 kms', '37,518 kms', '24,652 kms', '383 kms',
       '95,000 kms', '3,528 kms', '52,500 kms', '47,900 kms',
       '52,800 kms', '1,95,000 kms', '48,008 kms', '48,247 kms',
       '9,400 kms', '64,000 kms', '2,137 kms', '10,544 kms', '49,500 kms',
       '1,47,000 kms', '90,001 kms', '48,006 kms', '74,000 kms',
       '85,000 kms', '29,500 kms', '39,700 kms', '67,000 kms',
       '19,336 kms', '60,105 kms', '45,933 kms', '1,02,563 kms',
       '28,600 kms', '41,800 kms', '1,16,000 kms', '42,590 kms',
       '7,400 kms', '54,500 kms', '76,000 kms', '00 kms', '11,523 kms',
       '38,600 kms', '95,500 kms', '37,458 kms', '85,960 kms',
       '12,516 kms', '30,600 kms', '2,550 kms', '62,500 kms',
       '69,000 kms', '28,400 kms', '68,485 kms', '3,500 kms',
       '85,455 kms', '63,000 kms', '1,600 kms', '77,000 kms',
       '26,500 kms', '2,875 kms', '13,900 kms', '1,500 kms', '2,450 kms',
       '1,625 kms', '33,400 kms', '60,123 kms', '38,900 kms',
       '1,37,495 kms', '91,200 kms', '1,46,000 kms', '1,00,800 kms',
       '2,100 kms', '2,500 kms', '1,32,000 kms', 'Petrol'], dtype=object)
```

In [9]:

```
car["kms_driven"].isna().sum()
```

Out[9]:

```
52
```

In [10]:

```
car["fuel_type"].unique()
```

Out[10]:

```
array(['Petrol', 'Diesel', nan, 'LPG'], dtype=object)
```

Quality

1. Year has many Non-year values and year is in object dtype not in integer.
2. In Price there is one object value so we need to remove that and all entries are in string which should be in integer and there are commas also in price which should also be removed.
3. In Kms_driven there is "kms" in string and commas should be removed. plus need to change object dtype to integer data type.
4. There are 52 null values in kms_driven which should be handaled.
5. Fuel type has nan values.
6. Name is complicated so I will keep first 3 words of name.

Data Cleaning

In [11]:

```
#For Data security I am creating a data copy.
Backup = car.copy()
```

In [12]:

```
#1. first problem in data is :- Year has many Non-year values
car = car[car["year"].str.isnumeric()] #on String .str data we took the numeric values only by .isnumeric()
```

In [13]:

```
#2. year is in object dtype not in integer.
car["year"] = car["year"].astype(int) #data is converted to int with help of .astype
```

In [14]:

```
#3. Price has Ask for Price
car = car[car["Price"]!="Ask For Price"] # Now we have only those entries in which "Ask for price" is not there.
```

In [15]:

```
#4. There are ' , ' in the Price.
car["Price"] = car["Price"].str.replace(',', '').astype(int)
```

In [16]:

```
#5. kms_driven has kms and there are "," with kms.
car["kms_driven"] = car["kms_driven"].str.split( " " ).str.get(0).str.replace(',', '')
#first we split the feature data from space. then i kept 0 index data.
#then i replaced ',' with ' '
```

In [17]:

```
#6. kms_driven has some object data
car = car[car["kms_driven"]!= "Petrol"]
```

In [18]:

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

In [19]:

```
#7. As there was one NaN value in fuel type by ~ that row is not showing
car = car[~car["fuel_type"].isna()]
```

In [20]:

```
#8. Name is very confusing and mixed dtype so i am keeping only first 3 values of the name
car["name"] = car["name"].str.split(" ").str.slice(0,3).str.join( " ")
#first split the name with space by .str.split()
# slice the values by .str.slice()
#then keep the first 3 values
```

In [21]:

```
#The index got currepted after cleaning so Reset_index and removed the old index values by drop=True
car = car.reset_index(drop=True)
```

In [22]:

```
#the car price must have outliers as the max price and upper quartile has too much difference.
car.describe()
```

Out[22]:

	year	Price	kms_driven
count	816.000000	8.160000e+02	816.000000
mean	2012.444853	4.117176e+05	46275.531863
std	4.002992	4.751844e+05	34297.428044
min	1995.000000	3.000000e+04	0.000000
25%	2010.000000	1.750000e+05	27000.000000
50%	2013.000000	2.999990e+05	41000.000000
75%	2015.000000	4.912500e+05	56818.500000
max	2019.000000	8.500003e+06	400000.000000

In [23]:

```
car[car["Price"]>6e5] #There is 1 car which is more than 60million
```

Out[23]:

	name	company	year	Price	kms_driven	fuel_type
6	Ford EcoSport Ambiente	Ford	2016	830000	24530	Diesel
14	Audi A8	Audi	2017	1000000	4000	Petrol
33	Toyota Innova 2.0	Toyota	2012	650000	82000	Diesel
34	Renault Lodgy 85	Renault	2018	689999	20000	Diesel
47	Mitsubishi Pajero Sport	Mitsubishi	2015	1475000	47000	Diesel
...	...	...	...	...	...	...
763	Mahindra Scorpio VLX	Mahindra	2014	650000	77000	Diesel
764	Toyota Innova 2.5	Toyota	2012	750000	75000	Diesel
771	Ford Endeavor 4x4	Ford	2019	2900000	9000	Diesel
777	Toyota Innova 2.5	Toyota	2011	750000	75000	Diesel

In [24]:

```
print(car['Price'].max())
print(car["Price"].min())
print(car["Price"].median())
print(car["Price"].mean())
print(car["Price"].mode())
```

```
8500003
30000
299999.0
411717.61519607843
0    250000
dtype: int32
```

In [25]:

```
car = car[car["Price"]<6e6].reset_index(drop= True)
```

In [26]:

```
#Save the cleaned data in CSV file
car.to_csv("Cleaned Quikr Car data.csv")
```

Seperate Input and Output

In [27]:

```
X = car.drop(columns= "Price")
y = car["Price"]
```

In [28]:

```
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)
```

In [29]:

```
one_hot_encoder = OneHotEncoder()
```

In [30]:

```
one_hot_encoder.fit(X[["name", "company","fuel_type"]])
```

Out[30]:

```
▾ OneHotEncoder
OneHotEncoder()
```

In [31]:

```
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

In [32]:

```
columns_trans = make_column_transformer((OneHotEncoder(categories=one_hot_encoder.categories_),["name", "company","fuel_type"]),remainder= "p
```

In [33]:

```
linear_model = LinearRegression()
```

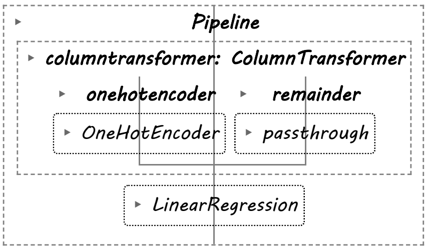
In [34]:

```
pipe = make_pipeline(columns_trans, linear_model)
```

In [35]:

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

Out[35]:



In [36]:

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

In [37]:

```
r2_score(y_test,y_pred)
```

Out[37]:

```
0.5015570684511671
```

In [38]:

```
score = []
for i in range(1000):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.2,random_state= i)
    linear_model= LinearRegression()
    pipe = make_pipeline(columns_trans, linear_model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    print(r2_score(y_test, y_pred), i)
    score.append(r2_score(y_test, y_pred))
```

```
0.7722837172132719 98
0.6764636597583917 99
0.7138119016320059 100
0.7605419753243958 101
0.7224426224200098 102
0.6787495372073729 103
0.720681545450921 104
0.6126558737637817 105
0.6658867671006738 106
0.5844940693868773 107
0.7560397050257291 108
0.807091812218459 109
0.7658862928800008 110
0.6627766734766672 111
0.6256171741614938 112
0.7470583255911711 113
0.742823324955956 114
0.6568506581285807 115
0.6951254734114951 116
0.6661815248788046 117
```

In [39]:

```
np.argmax(score)
```

Out[39]:

```
661
```

In [40]:

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

Out[40]:

```
0.8900345063168427
```

In [41]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.2,random_state= np.argmax(score))
linear_model= LinearRegression()
pipe = make_pipeline(columns_trans, linear_model)
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test, y_pred), i)
```

```
0.8900345063168427 999
```

Model Evaluation

In [42]:

```
import pickle
```

In [43]:

```
pickle.dump(pipe, open("LinearRegressionModel.pkl", "wb"))
```

Let's predict one row

In [44]:

```
pipe.predict(pd.DataFrame([["Maruti Suzuki Swift","Maruti", 2019, 100, "Petrol"]], columns= ["name", "company", "year","kms_driven", "fuel_typ
```

Out[44]:

```
array([401379.05200387])
```