In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
%matplotlib inline
# 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)
```

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]:

```
backup = car.copy()
```

Data Quality

- names are pretty inconsistent
- names have company names attached to it
- some names are spam like 'Maruti Ertiga showroom condition with' and 'Well mentained Tata Sumo'
- company: many of the names are not of any company like 'Used', 'URJENT', and so on.
- year has many non-year values
- year is in object. Change to integer
- Price has Ask for Price
- Price has commas in its prices and is in object
- kms_driven has object values with kms at last.
- It has nan values and two rows have 'Petrol' in them
- fuel_type has nan values

Cleaning Data

Year has many non-year valus

In [7]:

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

Year is in object. change to intger

In [8]:

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

Price has "Ask for Price"

In [9]:

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

In [10]:

```
# car['Price'].unique()
```

Price has commas in its prices and is in object

In [11]:

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

kms_driven has object values with kms at last

In [12]:

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

It has nan value values and two rows have 'Petrol' in them

In [13]:

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

In [14]:

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

fuel_type has nan values

In [15]:

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

In [16]:

```
car.shape
```

Out[16]:

```
(816, 6)
```

name and company had spammed data...but with previous cleaning, thoserows got removed.

company does not need any cleaning now. WChwanging car names. keeping only the first 3 words

In [17]:

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

Resetting the index of the final cleaned data

In [18]:

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

Cleaned Data

In [19]:

```
car.to_csv('Cleaned_Car_data.csv')
```

In [20]:

```
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    int32 
 3   Price       816 non-null    int32 
 4   kms_driven  816 non-null    int32 
 5   fuel_type   816 non-null    object
dtypes: int32(3), object(3)
memory usage: 28.8+ KB
```

In [21]:

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

Out[21]:

	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

In [22]:

```
car=car[car['Price']<6000000]
```

Extracting Data

In [23]:

```
x=car.drop(columns='Price')
y=car['Price']
```

In [24]:

```
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 [25]:

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

In [26]:

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

Out[26]:

```
OneHotEncoder()
```

In [27]:

```
column_trans = make_column_transformer((OneHotEncoder(categories=ohe.categories_),['nar
                                      remainder='passthrough')
```

In [28]:

```
lr = LinearRegression()
```

In [29]:

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

In [30]:

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

Out[30]:

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(categories=
[array(['Audi A3 Cabriolet', 'Audi A4 1.8', 'Audi A4 2.0', 'Audi A6 2.0',
       'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3 Series',
       'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW X1 sDrive20d',
       'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat...

array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford',
       'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
       'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
       'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
      dtype=object),

array(['Diesel', 'LPG', 'Petrol'], dtype=object)]),
                                                  ['name', 'company',
                                                   'fuel_type'])])),
                ('linearregression', LinearRegression())])
```

In [31]:

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

In [32]:

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

Out[32]:

```
0.6954629091832226
```

Finding the model with a random state of TrainTestSplit where the model was found to give almost 0.88

In [33]:

```
scores=[]
for i in range(1000):
    x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,random_state
    lr=LinearRegression()
    pipe=make_pipeline(column_trans, lr)
    pipe.fit(x_train, y_train)
    y_pred=pipe.predict(x_test)
#     print(r2_score(y_test, y_pred), i)
    scores.append(r2_score(y_test, y_pred))
```

In [34]:

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

Out[34]:

```
661
```

In [35]:

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

Out[35]:

```
0.889770931767991
```

In [36]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,random_state=np.
lr=LinearRegression()
pipe=make_pipeline(column_trans, lr)
pipe.fit(x_train, y_train)
y_pred=pipe.predict(x_test)
r2_score(y_test, y_pred)
```

Out[36]:

```
0.889770931767991
```

In [37]:

```
import pickle
```

In [38]:

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

In [39]:

```
pipe.predict(pd.DataFrame([['Maruti Suzuki Swift','Maruti', 2019,100,'Petrol']], colum
```

Out[39]:

```
array([400795.54550476])
```

In [40]:

```
pipe.predict(pd.DataFrame([['Maruti Suzuki Swift','Maruti', 2017,100,'Petrol']], colum
```

Out[40]:

```
array([363365.76708183])
```

In [41]:

```
pipe.predict(pd.DataFrame([['Maruti Suzuki Swift','Maruti', 2015,1000,'Petrol']], colur
```

Out[41]:

```
array([325857.22134978])
```

In [42]:

```
pipe.predict(pd.DataFrame([['Maruti Suzuki Swift','Maruti', 2015,500000,'Petrol']], col
```

Out[42]:

```
array([282185.12441023])
```

In [ ]:

```

```