Car dekho data set -Regression

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data=pd.read_csv("C:\\Users\\User\\Desktop\\Car-details-v3.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

In [5]:
```
data.shape
```

Out[5]:
```
(8128, 13)
```

In [6]:
```
data.describe()
```

Out[6]:

	year	selling_price	km_driven	seats
count	8128.000000	8.128000e+03	8.128000e+03	7907.000000
mean	2013.804011	6.382718e+05	6.981951e+04	5.416719
std	4.044249	8.062534e+05	5.655055e+04	0.959588
min	1983.000000	2.999900e+04	1.000000e+00	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	14.000000

In [7]:
```
# remove kmpl from mileage and convert it into float type from object type
data['mileage'] = data['mileage'].apply(lambda x: float(x.split()[0]) if type(x)==str else np.nan)
data['mileage'] = data['mileage'].astype("float")
```

In [8]:
```
# remove CC from engine variable
data['engine'] = data['engine'].apply(lambda x: x.replace("CC","") if type(x)==str else np.nan)
```

In [9]:
```
# remove bhp from max_power and convert it into float type from object type
data['max_power'] = data['max_power'].apply(lambda x : x.split()[0] if type(x)==str else np.nan )
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```

In [11]:
```
data.drop(['torque'],inplace=True,axis=1)
```

In [12]:
```
data.dropna(inplace=True)
```

In [13]:
```
data.shape
```

Out[13]:
```
(7907, 12)
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7907 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           7907 non-null   object
 1   year           7907 non-null   int64
 2   selling_price  7907 non-null   int64
 3   km_driven      7907 non-null   int64
 4   fuel           7907 non-null   object
 5   seller_type    7907 non-null   object
 6   transmission   7907 non-null   object
 7   owner          7907 non-null   object
 8   mileage        7907 non-null   float64
 9   engine         7907 non-null   object
 10  max_power      7907 non-null   object
 11  seats          7907 non-null   float64
dtypes: float64(2), int64(3), object(7)
memory usage: 803.1+ KB
```

In [15]:
```
data[data['max_power'].str.contains('bhp')]
```

Out[15]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
4933	Maruti Omni CNG	2000	80000	100000	CNG	Individual	Manual	Second Owner	10.9	796	bhp	8.0

In [16]:
```
# we have one row containing only bhp with no value in it, lets drop it
data = data[data['max_power'].str.contains('bhp') == False]
```

In [17]:
```
data.shape
```

Out[17]:
```
(7906, 12)
```

In [18]:
```
sns.boxplot(x=data['mileage'])
```

Out[18]:
```
<AxesSubplot:xlabel='mileage'>
```

In [241…
```
#from above boxplot we see some cars having 0 mileage, it does not makesense, need to drop them
data[data["mileage"]==0]
```

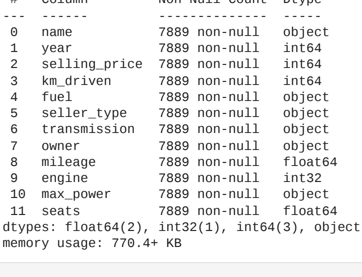
Out[241…

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
644	Tata Indica Vista Aura Safire Anniversary Edition	2009	135000	28900	Petrol	Individual	Manual	Second Owner	0.0	1172	65	5.0
785	Hyundai Santro Xing GL	2009	120000	90000	Petrol	Individual	Manual	Second Owner	0.0	1086	62	5.0
1649	Hyundai Santro Xing GL	2008	105000	128000	Petrol	Individual	Manual	First Owner	0.0	1086	62	5.0
1676	Mercedes-Benz M-Class ML 350 4Matic	2011	1700000	110000	Diesel	Individual	Automatic	Third Owner	0.0	2987	165	5.0
2137	Land Rover Freelander 2 TD4 HSE	2013	1650000	64788	Diesel	Dealer	Automatic	First Owner	0.0	2179	115	5.0
2366	Hyundai Santro Xing (Non-AC)	2010	110000	80000	Petrol	Individual	Manual	Second Owner	0.0	1086	62.1	5.0
2725	Hyundai Santro Xing (Non-AC)	2013	184000	15000	Petrol	Individual	Manual	First Owner	0.0	1086	62.1	5.0
4527	Mercedes-Benz M-Class ML 350 4Matic	2011	1700000	110000	Diesel	Individual	Automatic	Third Owner	0.0	2987	165	5.0
5276	Hyundai Santro Xing GL	2008	175000	40000	Petrol	Individual	Manual	First Owner	0.0	1086	62	5.0
5843	Volkswagen Polo GT TSI BSIV	2014	574000	28080	Petrol	Dealer	Automatic	First Owner	0.0	1197	103.25	5.0
5846	Volkswagen Polo GT TSI BSIV	2014	575000	28100	Petrol	Dealer	Automatic	First Owner	0.0	1197	103.25	5.0
5900	Mahindra Bolero Pik-Up FB 1.7T	2020	679000	5000	Diesel	Individual	Manual	First Owner	0.0	2523	70	2.0
6534	Hyundai Santro Xing GL	2010	150000	110000	Petrol	Individual	Manual	First Owner	0.0	1086	62	5.0
6629	Mahindra Bolero Pik-Up CBC 1.7T	2019	722000	80000	Diesel	Individual	Manual	First Owner	0.0	2523	70	2.0
6824	Hyundai Santro Xing GL	2011	150000	40000	Petrol	Individual	Manual	Fourth & Above Owner	0.0	1086	62	5.0
7002	Hyundai Santro Xing (Non-AC)	2010	110000	80000	Petrol	Individual	Manual	Second Owner	0.0	1086	62.1	5.0
7337	Mercedes-Benz GLC 220d 4MATIC	2017	3300000	60000	Diesel	Dealer	Automatic	First Owner	0.0	1950	194	5.0

In [19]:
```
# drop ows having 0 mileage
data = data[data["mileage"]!=0]
```

In [20]:
```
data['engine']=data['engine'].astype(int)
sns.boxplot(x=data['engine'])
```

Out[20]:
```
<AxesSubplot:xlabel='engine'>
```



In [21]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7889 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           7889 non-null   object
 1   year           7889 non-null   int64
 2   selling_price  7889 non-null   int64
 3   km_driven      7889 non-null   int64
 4   fuel           7889 non-null   object
 5   seller_type    7889 non-null   object
 6   transmission   7889 non-null   object
 7   owner          7889 non-null   object
 8   mileage        7889 non-null   float64
 9   engine         7889 non-null   int32
 10  max_power      7889 non-null   object
 11  seats          7889 non-null   float64
dtypes: float64(2), int32(1), int64(3), object(6)
memory usage: 770.4+ KB
```

In [22]:
```
data[data.duplicated()]
```

Out[22]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
291	Hyundai Grand i10 Sportz	2017	450000	35000	Petrol	Individual	Manual	First Owner	18.90	1197	82	5.0
296	Maruti Swift VXI	2012	330000	50000	Petrol	Individual	Manual	Second Owner	18.60	1197	85.8	5.0
370	Jaguar XE 2016-2019 2.0L Diesel Prestige	2017	2625000	9000	Diesel	Dealer	Automatic	First Owner	13.60	1999	177	5.0
371	Lexus ES 300h	2019	5150000	20000	Petrol	Dealer	Automatic	First Owner	22.37	2487	214.56	5.0
372	Jaguar XF 2.0 Diesel Portfolio	2017	3200000	45000	Diesel	Dealer	Automatic	First Owner	19.33	1999	177	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...
7987	Renault Captur 1.5 Diesel RXT	2018	1265000	12000	Diesel	Individual	Manual	First Owner	20.37	1461	108.45	5.0
7988	Maruti Ciaz Alpha Diesel	2019	1025000	32000	Diesel	Individual	Manual	First Owner	28.09	1248	88.50	5.0
8117	Maruti Swift Dzire VDI	2015	625000	50000	Diesel	Individual	Manual	First Owner	26.59	1248	74	5.0
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57	1396	70	5.0
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57	1396	70	5.0

1187 rows × 12 columns

In [23]:
```
data=data.drop_duplicates()
```

In [24]:
```
data.shape
```

Out[24]:
```
(6702, 12)
```

In [25]:
```
# let's extact how many years old the car is from year column and drop year column

data["old"] = 2022-data["year"]
data.drop(["year"],axis=1,inplace=True)
```

In [26]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6702 entries, 0 to 8125
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           6702 non-null   object
 1   selling_price  6702 non-null   int64
 2   km_driven      6702 non-null   int64
 3   fuel           6702 non-null   object
 4   seller_type    6702 non-null   object
 5   transmission   6702 non-null   object
 6   owner          6702 non-null   object
 7   mileage        6702 non-null   float64
 8   engine         6702 non-null   int32
 9   max_power      6702 non-null   object
 10  seats          6702 non-null   float64
 11  old            6702 non-null   int64
dtypes: float64(2), int32(1), int64(3), object(6)
memory usage: 654.5+ KB
```

In [27]:
```
data.drop('name',inplace=True,axis=1)
```

In [32]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['fuel']=le.fit_transform(data['fuel'])
data['seller_type']=le.fit_transform(data['seller_type'])
data['transmission']=le.fit_transform(data['transmission'])
data['owner']=le.fit_transform(data['owner'])
```

In [34]:
```
data.head(20)
```

Out[34]:

	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats	old
0	450000	145500	1	1	1	0	23.40	1248	74	5.0	8
1	370000	120000	1	1	1	2	21.14	1498	103.52	5.0	8
2	158000	140000	3	1	1	4	17.70	1497	78	5.0	16
3	225000	127000	1	1	1	0	23.00	1396	90	5.0	12
4	130000	120000	3	1	1	0	16.10	1298	88.2	5.0	15
5	440000	45000	3	1	1	0	20.14	1197	81.86	5.0	5
6	96000	175000	2	1	1	0	17.30	1061	57.5	5.0	15
7	45000	5000	3	1	1	2	16.10	796	37	4.0	21
8	350000	90000	1	1	1	0	23.59	1364	67.1	5.0	11
9	200000	169000	1	1	1	0	20.00	1399	68.1	5.0	9
10	500000	68000	1	1	1	2	19.01	1461	108.45	5.0	8
11	92000	100000	3	1	1	2	17.30	993	60	5.0	17
12	280000	140000	1	1	1	2	19.30	1248	73.9	5.0	13
14	180000	90000	3	1	1	2	18.90	1061	67	5.0	13
15	400000	40000	3	1	1	0	18.15	1198	82	5.0	6
16	778000	70000	1	1	1	2	24.52	1248	88.5	7.0	6
17	500000	53000	1	1	1	2	23.00	1396	90	5.0	10
18	150000	80000	3	1	1	2	19.70	796	46.3	5.0	20
19	680000	100000	1	1	1	0	22.54	1396	88.73	5.0	6
20	174000	100000	1	1	1	2	21.00	1461	64.1	5.0	11

In [37]:
```
# split data into train and test
y=data['selling_price']
x=data.drop(['selling_price'],axis=1)
```

In [261…
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
fitted=le.fit_transform(X_train['fuel'])
```

In [38]:
```
from sklearn.model_selection import train_test_split
```

In [39]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
```

In [42]:
```
from sklearn.linear_model import LinearRegression
line_reg=LinearRegression()
line_reg.fit(x_train,y_train)
#Now predicting on the test data
y_pred=line_reg.predict(x_test)
```

In [41]:
```
print(y_pred)
```

```
[320678.29636595 436426.74020904 441899.14118783 ... 491766.58907966
 333322.05451381 602850.07792214]
```

In [44]:
```
# compare the actual output values for X_test with the predicted values

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df.reset_index(inplace=True,drop=True)
df
```

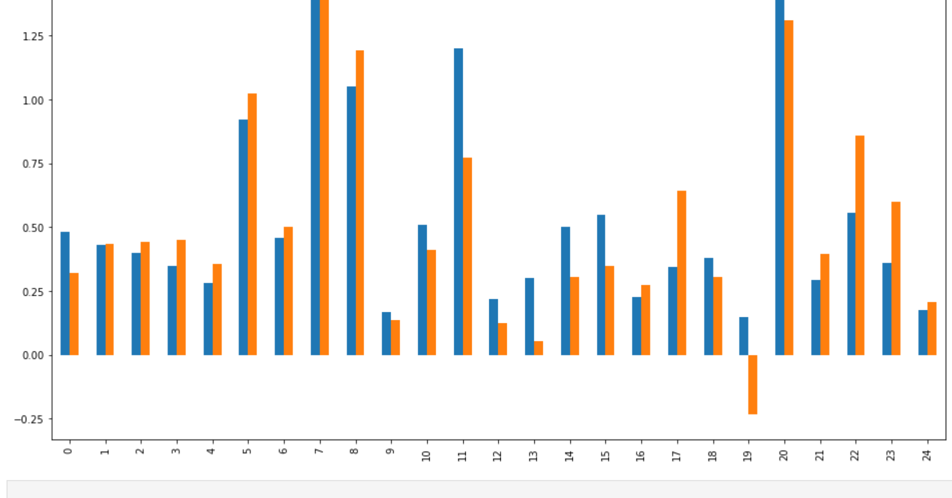
Out[44]:

	Actual	Predicted
0	480000	320678.296366
1	430000	436426.740209
2	400000	441899.141188
3	350000	452265.922270
4	280000	356332.079158
...	...	...
2006	345000	334345.154750
2007	330000	591235.334784
2008	550000	491766.589080
2009	199000	333322.054514
2010	650000	602850.077922

2011 rows × 2 columns

In [45]:
```
#Showing the difference between the actual and predicted value

df1 = df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.show()
```



In [ ]: