gi

car-pricing-prediction

November 18, 2020

[1374]:
```
# Importing the libraries
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

from datetime import datetime as dt
```

[1375]:
```
# Fetching, exploring and checking out the data
```

[1376]:
```
# Read data from CSV file
df = pd.read_csv("car-pricing-data.csv")
```

[1377]:
```
# Show the first 5 rows
df.head()
```

[1377]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	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	
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	

```
1    5.0
2    5.0
3    5.0
4    5.0
```

[1378]:
```
# Check the rows with missing values (if any)
df.isnull().sum()
```

[1378]:
```
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
seats           221
dtype: int64
```

[1379]:
```
# Drop the empty rows
df.dropna(inplace=True)
df.isnull().sum()
```

[1379]:
```
name            0
year            0
selling_price   0
km_driven       0
fuel            0
seller_type     0
transmission    0
owner           0
mileage         0
engine          0
max_power       0
seats           0
dtype: int64
```

[1380]:
```
# Print the number of rows (instances) and columns (features)
df.shape
```

[1380]:
```
(7907, 12)
```

[1381]:
```
# Feature Engineering
```

[1382]:
```
# Convert year to age
df = df.rename(columns={"year": "age"})
df["age"] = dt.now().year - df["age"]
```

[1383]:
```
# Convert car name to brand
df['car_brand'] = df['name'].apply(lambda x:x.split(' ')[0])
```

[1384]:
```
# Drop the name column, since it is not needed anymore
df.drop(['name'], axis=1, inplace=True)
```

[1385]:
```
# Get rid of units
df['mileage'] = df['mileage'].str.extract('(\d+)', expand=False)
df['engine'] = df['engine'].str.extract('(\d+)', expand=False)
df['max_power'] = df['max_power'].str.extract('(\d+)', expand=False)

# Need to drop na liens again since there is now na cells.
df.dropna(inplace=True)
```

[1386]:
```
# One-hot encoding for categorical features
df = pd.concat([df,pd.get_dummies(df['fuel'], prefix='fuel')],axis=1)
df = pd.concat([df,pd.get_dummies(df['seller_type'], prefix='seller')],axis=1)
df = pd.concat([df,pd.get_dummies(df['owner'], prefix='owner')],axis=1)
df = pd.concat([df,pd.get_dummies(df['car_brand'], prefix='brand')],axis=1)
df = pd.concat([df,pd.get_dummies(df['transmission'], prefix='transmission')],
                axis=1)
```

[1387]:
```
# Drop the pre-one-hot columns
df.drop(['fuel'], axis=1, inplace=True)
df.drop(['seller_type'], axis=1, inplace=True)
df.drop(['owner'], axis=1, inplace=True)
df.drop(['transmission'], axis=1, inplace=True)
df.drop(['car_brand'], axis=1, inplace=True)
```

[1388]:
```
# Check the data
df.head()
```

[1388]:

	age	selling_price	km_driven	mileage	engine	max_power	seats	fuel_CNG	\
0	6	450000	145500	23	1248	74	5.0	0	
1	6	370000	120000	21	1498	103	5.0	0	
2	14	158000	140000	17	1497	78	5.0	0	
3	10	225000	127000	23	1396	90	5.0	0	
4	13	130000	120000	16	1298	88	5.0	0	

	fuel_Diesel	fuel_LPG	...	brand_Nissan	brand_Opel	brand_Renault	\
0	1	0	...	0	0	0	
1	1	0	...	0	0	0	
2	0	0	...	0	0	0	

```
3            1         0  ...              0           0              0
4            0         0  ...              0           0              0

   brand_Skoda  brand_Tata  brand_Toyota  brand_Volkswagen  brand_Volvo  \
0            0           0             0                 0            0
1            1           0             0                 0            0
2            0           0             0                 0            0
3            0           0             0                 0            0
4            0           0             0                 0            0

   transmission_Automatic  transmission_Manual
0                       0                    1
1                       0                    1
2                       0                    1
3                       0                    1
4                       0                    1

[5 rows x 52 columns]
```

[1389]:
```
# Create correlation matrix
corr_matrix = df.corr().abs()
corr_matrix
```

[1389]:

	age	selling_price	km_driven	seats	\
age	1.000000	0.412302	0.428548	0.007923	
selling_price	0.412302	1.000000	0.222158	0.041617	
km_driven	0.428548	0.222158	1.000000	0.227259	
seats	0.007923	0.041617	0.227259	1.000000	
fuel_CNG	0.029095	0.033197	0.005432	0.038587	
fuel_Diesel	0.037536	0.204831	0.271662	0.354764	
fuel_LPG	0.059887	0.035978	0.023095	0.028949	
fuel_Petrol	0.034351	0.195074	0.274460	0.345399	
seller_Dealer	0.214525	0.401803	0.178725	0.074086	
seller_Individual	0.243729	0.386151	0.202851	0.081197	
seller_Trustmark Dealer	0.100385	0.032779	0.083181	0.028106	
owner_First Owner	0.491934	0.239850	0.295470	0.035481	
owner_Fourth & Above Owner	0.205631	0.073601	0.089244	0.007845	
owner_Second Owner	0.317328	0.178786	0.209913	0.033751	
owner_Test Drive Car	0.032661	0.116081	0.024168	0.010921	
owner_Third Owner	0.271317	0.115113	0.148795	0.005175	
brand_Ambassador	0.066877	0.014542	0.005274	0.009767	
brand_Ashok	0.002864	0.004836	0.025908	0.030296	
brand_Audi	0.022907	0.172014	0.019203	0.008654	
brand_BMW	0.089073	0.530173	0.085028	0.061050	
brand_Chevrolet	0.082071	0.079992	0.032300	0.062953	
brand_Daewoo	0.068840	0.013718	0.004161	0.008458	
brand_Datsun	0.071441	0.037516	0.052661	0.001365	

brand_Fiat	0.024769	0.029520	0.017758	0.031344
brand_Force	0.023892	0.008052	0.006620	0.026346
brand_Ford	0.004582	0.036006	0.013124	0.074206
brand_Honda	0.031067	0.016272	0.040741	0.077292
brand_Hyundai	0.023322	0.098909	0.043154	0.193686
brand_Isuzu	0.026150	0.039957	0.010467	0.020552
brand_Jaguar	0.067721	0.265111	0.065443	0.041326
brand_Jeep	0.058410	0.115668	0.035416	0.027238
brand_Kia	0.032123	0.023637	0.023450	0.009767
brand_Land	0.021514	0.100222	0.015274	0.026346
brand_Lexus	0.085327	0.363541	0.056925	0.028531
brand_MG	0.026977	0.027147	0.019495	0.008458
brand_Mahindra	0.001576	0.007923	0.123206	0.552437
brand_Maruti	0.077702	0.193495	0.064701	0.192228
brand_Mercedes-Benz	0.000053	0.185590	0.024863	0.029597
brand_Mitsubishi	0.054333	0.008681	0.052862	0.028764
brand_Nissan	0.003349	0.023062	0.005738	0.044169
brand_Opel	0.046533	0.008044	0.008083	0.004883
brand_Renault	0.083663	0.039652	0.030747	0.033835
brand_Skoda	0.010152	0.005690	0.003858	0.045493
brand_Tata	0.058919	0.111956	0.064046	0.022296
brand_Toyota	0.020692	0.109607	0.138013	0.268546
brand_Volkswagen	0.004975	0.030429	0.007524	0.067200
brand_Volvo	0.102888	0.297988	0.089566	0.041574
transmission_Automatic	0.249002	0.590269	0.201186	0.072722
transmission_Manual	0.249002	0.590269	0.201186	0.072722

	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol	\
age	0.029095	0.037536	0.059887	0.034351	
selling_price	0.033197	0.204831	0.035978	0.195074	
km_driven	0.005432	0.271662	0.023095	0.274460	
seats	0.038587	0.354764	0.028949	0.345399	
fuel_CNG	1.000000	0.088831	0.005426	0.072894	
fuel_Diesel	0.088831	1.000000	0.072800	0.978020	
fuel_LPG	0.005426	0.072800	1.000000	0.059739	
fuel_Petrol	0.072894	0.978020	0.059739	1.000000	
seller_Dealer	0.032833	0.065171	0.021417	0.057113	
seller_Individual	0.036808	0.003567	0.025092	0.012914	
seller_Trustmark Dealer	0.014273	0.140763	0.011697	0.144956	
owner_First Owner	0.004295	0.040051	0.024475	0.044107	
owner_Fourth & Above Owner	0.000582	0.009021	0.003946	0.008609	
owner_Second Owner	0.009838	0.051721	0.013442	0.055230	
owner_Test Drive Car	0.002047	0.017362	0.001678	0.017957	
owner_Third Owner	0.008627	0.007567	0.021268	0.006146	
brand_Ambassador	0.001831	0.020609	0.001500	0.020156	
brand_Ashok	0.000915	0.010302	0.000750	0.010076	
brand_Audi	0.005802	0.047423	0.004755	0.045948	

brand_BMW	0.010016	0.106467	0.008208	0.103975
brand_Chevrolet	0.014085	0.005945	0.011130	0.005154
brand_Daewoo	0.001585	0.021270	0.001299	0.021748
brand_Datsun	0.007408	0.099399	0.006071	0.101633
brand_Fiat	0.005875	0.037851	0.004815	0.036335
brand_Force	0.002242	0.025244	0.001838	0.024689
brand_Ford	0.018485	0.089363	0.015149	0.084529
brand_Honda	0.020364	0.123346	0.016689	0.129158
brand_Hyundai	0.020503	0.142329	0.040282	0.140595
brand_Isuzu	0.002047	0.023043	0.001678	0.022536
brand_Jaguar	0.007746	0.087196	0.006348	0.085280
brand_Jeep	0.005105	0.012773	0.004184	0.011411
brand_Kia	0.001831	0.020609	0.001500	0.020156
brand_Land	0.002242	0.025244	0.001838	0.024689
brand_Lexus	0.005348	0.071748	0.004382	0.073360
brand_MG	0.001585	0.021270	0.001299	0.021748
brand_Mahindra	0.026497	0.270684	0.021715	0.264067
brand_Maruti	0.107389	0.236812	0.035448	0.215130
brand_Mercedes-Benz	0.006748	0.032796	0.005530	0.031032
brand_Mitsubishi	0.003427	0.038580	0.002809	0.037732
brand_Nissan	0.008279	0.009974	0.006785	0.007743
brand_Opel	0.000915	0.012279	0.000750	0.012555
brand_Renault	0.014022	0.022728	0.011491	0.026594
brand_Skoda	0.009394	0.003229	0.007699	0.000679
brand_Tata	0.020294	0.122795	0.021092	0.116946
brand_Toyota	0.020037	0.096493	0.016421	0.091252
brand_Volkswagen	0.012595	0.051074	0.010322	0.047759
brand_Volvo	0.007523	0.081913	0.006165	0.080045
transmission_Automatic	0.031686	0.025579	0.025967	0.034257
transmission_Manual	0.031686	0.025579	0.025967	0.034257

	seller_Dealer	seller_Individual	...	\
age	0.214525	0.243729	...	
selling_price	0.401803	0.386151	...	
km_driven	0.178725	0.202851	...	
seats	0.074086	0.081197	...	
fuel_CNG	0.032833	0.036808	...	
fuel_Diesel	0.065171	0.003567	...	
fuel_LPG	0.021417	0.025092	...	
fuel_Petrol	0.057113	0.012914	...	
seller_Dealer	1.000000	0.891999	...	
seller_Individual	0.891999	1.000000	...	
seller_Trustmark Dealer	0.070780	0.387768	...	
owner_First Owner	0.212159	0.229695	...	
owner_Fourth & Above Owner	0.057993	0.065014	...	
owner_Second Owner	0.159130	0.168052	...	
owner_Test Drive Car	0.062344	0.055611	...	

owner_Third Owner	0.100024	0.113304	...
brand_Ambassador	0.009078	0.010178	...
brand_Ashok	0.004538	0.005088	...
brand_Audi	0.084251	0.072186	...
brand_BMW	0.229901	0.202661	...
brand_Chevrolet	0.048159	0.058262	...
brand_Daewoo	0.007862	0.008814	...
brand_Datsun	0.008482	0.015076	...
brand_Fiat	0.024059	0.027971	...
brand_Force	0.015353	0.011996	...
brand_Ford	0.039942	0.020410	...
brand_Honda	0.036758	0.109895	...
brand_Hyundai	0.044841	0.074100	...
brand_Isuzu	0.004348	0.002018	...
brand_Jaguar	0.212734	0.189015	...
brand_Jeep	0.001987	0.006823	...
brand_Kia	0.023340	0.019779	...
brand_Land	0.055062	0.048691	...
brand_Lexus	0.162872	0.145281	...
brand_MG	0.048285	0.043070	...
brand_Mahindra	0.075688	0.095827	...
brand_Maruti	0.073561	0.072136	...
brand_Mercedes-Benz	0.081603	0.068816	...
brand_Mitsubishi	0.000344	0.003030	...
brand_Nissan	0.012096	0.015920	...
brand_Opel	0.004538	0.005088	...
brand_Renault	0.012904	0.025623	...
brand_Skoda	0.097377	0.083762	...
brand_Tata	0.084524	0.103250	...
brand_Toyota	0.005164	0.116380	...
brand_Volkswagen	0.009877	0.003178	...
brand_Volvo	0.089941	0.075763	...
transmission_Automatic	0.336591	0.378698	...
transmission_Manual	0.336591	0.378698	...

	brand_Nissan	brand_Opel	brand_Renault	\
age	0.003349	0.046533	0.083663	
selling_price	0.023062	0.008044	0.039652	
km_driven	0.005738	0.008083	0.030747	
seats	0.044169	0.004883	0.033835	
fuel_CNG	0.008279	0.000915	0.014022	
fuel_Diesel	0.009974	0.012279	0.022728	
fuel_LPG	0.006785	0.000750	0.011491	
fuel_Petrol	0.007743	0.012555	0.026594	
seller_Dealer	0.012096	0.004538	0.012904	
seller_Individual	0.015920	0.005088	0.025623	
seller_Trustmark Dealer	0.010466	0.001973	0.030227	

owner_First Owner	0.001139	0.015657	0.047223
owner_Fourth & Above Owner	0.005702	0.001616	0.024766
owner_Second Owner	0.000995	0.006580	0.031454
owner_Test Drive Car	0.002559	0.000283	0.004335
owner_Third Owner	0.003962	0.042831	0.020638
brand_Ambassador	0.002289	0.000253	0.003877
brand_Ashok	0.001144	0.000127	0.001938
brand_Audi	0.007255	0.000802	0.012288
brand_BMW	0.012524	0.001384	0.021212
brand_Chevrolet	0.017612	0.001947	0.029829
brand_Daewoo	0.001982	0.000219	0.003357
brand_Datsun	0.009263	0.001024	0.015690
brand_Fiat	0.007346	0.000812	0.012442
brand_Force	0.002804	0.000310	0.004749
brand_Ford	0.023113	0.002555	0.039148
brand_Honda	0.025463	0.002815	0.043127
brand_Hyundai	0.046375	0.005127	0.078546
brand_Isuzu	0.002559	0.000283	0.004335
brand_Jaguar	0.009685	0.001071	0.016404
brand_Jeep	0.006383	0.000706	0.010812
brand_Kia	0.002289	0.000253	0.003877
brand_Land	0.002804	0.000310	0.004749
brand_Lexus	0.006686	0.000739	0.011325
brand_MG	0.001982	0.000219	0.003357
brand_Mahindra	0.033132	0.003663	0.056116
brand_Maruti	0.066510	0.007352	0.112649
brand_Mercedes-Benz	0.008437	0.000933	0.014291
brand_Mitsubishi	0.004285	0.000474	0.007258
brand_Nissan	1.000000	0.001144	0.017532
brand_Opel	0.001144	1.000000	0.001938
brand_Renault	0.017532	0.001938	1.000000
brand_Skoda	0.011747	0.001299	0.019896
brand_Tata	0.032180	0.003557	0.054505
brand_Toyota	0.025054	0.002770	0.042434
brand_Volkswagen	0.015749	0.001741	0.026674
brand_Volvo	0.009406	0.001040	0.015931
transmission_Automatic	0.001243	0.004380	0.038046
transmission_Manual	0.001243	0.004380	0.038046

	brand_Skoda	brand_Tata	brand_Toyota	\
age	0.010152	0.058919	0.020692	
selling_price	0.005690	0.111956	0.109607	
km_driven	0.003858	0.064046	0.138013	
seats	0.045493	0.022296	0.268546	
fuel_CNG	0.009394	0.020294	0.020037	
fuel_Diesel	0.003229	0.122795	0.096493	
fuel_LPG	0.007699	0.021092	0.016421	

fuel_Petrol	0.000679	0.116946	0.091252
seller_Dealer	0.097377	0.084524	0.005164
seller_Individual	0.083762	0.103250	0.116380
seller_Trustmark Dealer	0.013729	0.055482	0.267340
owner_First Owner	0.001408	0.009535	0.028917
owner_Fourth & Above Owner	0.016593	0.004845	0.000571
owner_Second Owner	0.003771	0.009747	0.040925
owner_Test Drive Car	0.002904	0.007957	0.006195
owner_Third Owner	0.005835	0.004690	0.015874
brand_Ambassador	0.002598	0.007116	0.005540
brand_Ashok	0.001299	0.003557	0.002770
brand_Audi	0.008233	0.022555	0.017560
brand_BMW	0.014212	0.038933	0.030311
brand_Chevrolet	0.019985	0.054750	0.042626
brand_Daewoo	0.002249	0.006162	0.004798
brand_Datsun	0.010512	0.028798	0.022421
brand_Fiat	0.008336	0.022837	0.017779
brand_Force	0.003182	0.008717	0.006786
brand_Ford	0.026229	0.071855	0.055942
brand_Honda	0.028895	0.079158	0.061628
brand_Hyundai	0.052625	0.144169	0.112242
brand_Isuzu	0.002904	0.007957	0.006195
brand_Jaguar	0.010991	0.030109	0.023441
brand_Jeep	0.007244	0.019845	0.015450
brand_Kia	0.002598	0.007116	0.005540
brand_Land	0.003182	0.008717	0.006786
brand_Lexus	0.007588	0.020787	0.016183
brand_MG	0.002249	0.006162	0.004798
brand_Mahindra	0.037597	0.102999	0.080189
brand_Maruti	0.075474	0.206764	0.160975
brand_Mercedes-Benz	0.009575	0.026230	0.020421
brand_Mitsubishi	0.004863	0.013322	0.010372
brand_Nissan	0.011747	0.032180	0.025054
brand_Opel	0.001299	0.003557	0.002770
brand_Renault	0.019896	0.054505	0.042434
brand_Skoda	1.000000	0.036518	0.028431
brand_Tata	0.036518	1.000000	0.077887
brand_Toyota	0.028431	0.077887	1.000000
brand_Volkswagen	0.017872	0.048960	0.038117
brand_Volvo	0.010674	0.029241	0.022766
transmission_Automatic	0.119199	0.098449	0.053949
transmission_Manual	0.119199	0.098449	0.053949

	brand_Volkswagen	brand_Volvo	\
age	0.004975	0.102888	
selling_price	0.030429	0.297988	
km_driven	0.007524	0.089566	

seats	0.067200	0.041574
fuel_CNG	0.012595	0.007523
fuel_Diesel	0.051074	0.081913
fuel_LPG	0.010322	0.006165
fuel_Petrol	0.047759	0.080045
seller_Dealer	0.009877	0.089941
seller_Individual	0.003178	0.075763
seller_Trustmark Dealer	0.027152	0.016217
owner_First Owner	0.024776	0.063498
owner_Fourth & Above Owner	0.013405	0.013287
owner_Second Owner	0.016943	0.050921
owner_Test Drive Car	0.029388	0.002326
owner_Third Owner	0.007037	0.024277
brand_Ambassador	0.003483	0.002080
brand_Ashok	0.001741	0.001040
brand_Audi	0.011038	0.006593
brand_BMW	0.019054	0.011380
brand_Chevrolet	0.026794	0.016003
brand_Daewoo	0.003016	0.001801
brand_Datsun	0.014094	0.008417
brand_Fiat	0.011176	0.006675
brand_Force	0.004266	0.002548
brand_Ford	0.035165	0.021003
brand_Honda	0.038740	0.023137
brand_Hyundai	0.070555	0.042139
brand_Isuzu	0.003894	0.002326
brand_Jaguar	0.014735	0.008801
brand_Jeep	0.009712	0.005800
brand_Kia	0.003483	0.002080
brand_Land	0.004266	0.002548
brand_Lexus	0.010173	0.006076
brand_MG	0.003016	0.001801
brand_Mahindra	0.050407	0.030106
brand_Maruti	0.101189	0.060435
brand_Mercedes-Benz	0.012837	0.007667
brand_Mitsubishi	0.006520	0.003894
brand_Nissan	0.015749	0.009406
brand_Opel	0.001741	0.001040
brand_Renault	0.026674	0.015931
brand_Skoda	0.017872	0.010674
brand_Tata	0.048960	0.029241
brand_Toyota	0.038117	0.022766
brand_Volkswagen	1.000000	0.014311
brand_Volvo	0.014311	1.000000
transmission_Automatic	0.004060	0.237412
transmission_Manual	0.004060	0.237412

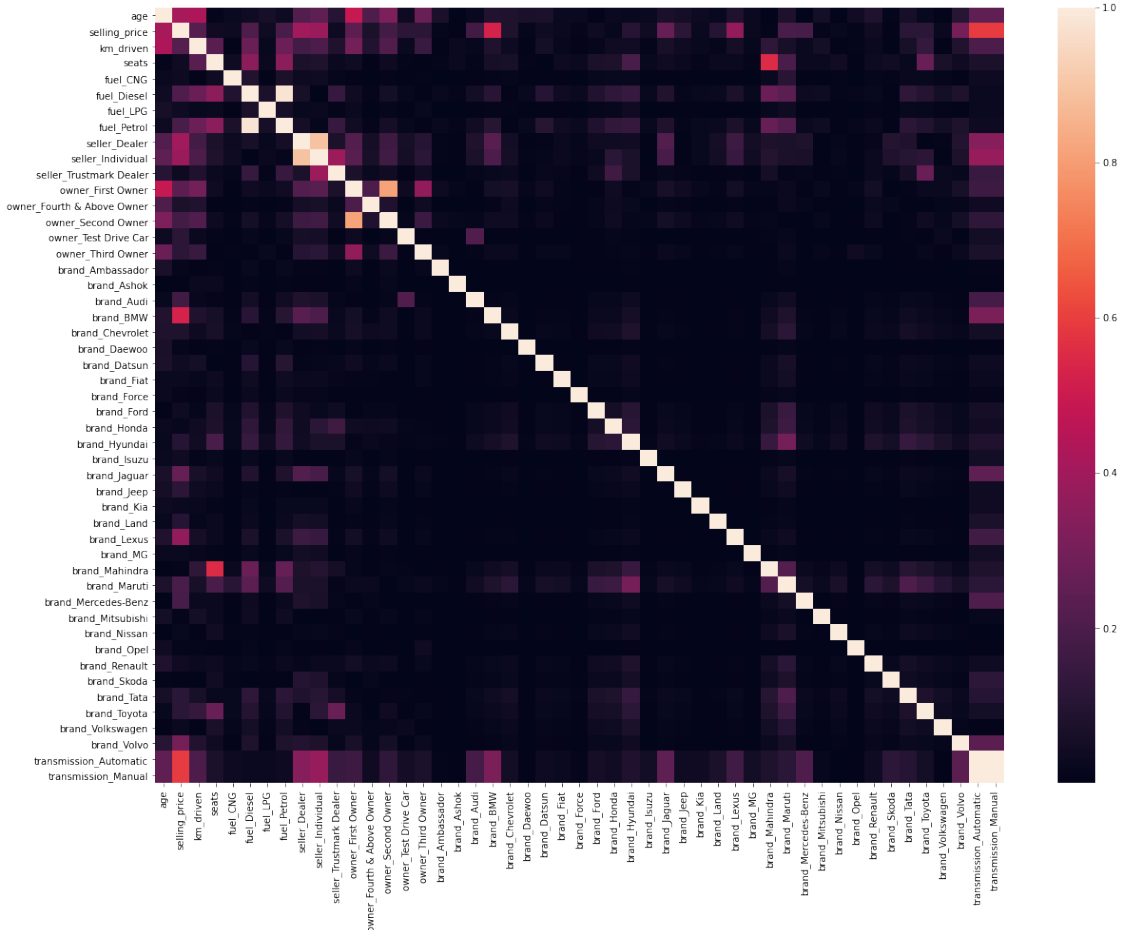
	transmission_Automatic	transmission_Manual
age	0.249002	0.249002
selling_price	0.590269	0.590269
km_driven	0.201186	0.201186
seats	0.072722	0.072722
fuel_CNG	0.031686	0.031686
fuel_Diesel	0.025579	0.025579
fuel_LPG	0.025967	0.025967
fuel_Petrol	0.034257	0.034257
seller_Dealer	0.336591	0.336591
seller_Individual	0.378698	0.378698
seller_Trustmark Dealer	0.149310	0.149310
owner_First Owner	0.158941	0.158941
owner_Fourth & Above Owner	0.040027	0.040027
owner_Second Owner	0.122257	0.122257
owner_Test Drive Car	0.049722	0.049722
owner_Third Owner	0.071802	0.071802
brand_Ambassador	0.008761	0.008761
brand_Ashok	0.004380	0.004380
brand_Audi	0.183125	0.183125
brand_BMW	0.316099	0.316099
brand_Chevrolet	0.049600	0.049600
brand_Daewoo	0.007587	0.007587
brand_Datsun	0.031312	0.031312
brand_Fiat	0.028116	0.028116
brand_Force	0.010732	0.010732
brand_Ford	0.057296	0.057296
brand_Honda	0.043902	0.043902
brand_Hyundai	0.084324	0.084324
brand_Isuzu	0.049722	0.049722
brand_Jaguar	0.244458	0.244458
brand_Jeep	0.035424	0.035424
brand_Kia	0.041143	0.041143
brand_Land	0.070771	0.070771
brand_Lexus	0.168769	0.168769
brand_MG	0.050033	0.050033
brand_Mahindra	0.078528	0.078528
brand_Maruti	0.113259	0.113259
brand_Mercedes-Benz	0.203878	0.203878
brand_Mitsubishi	0.016401	0.016401
brand_Nissan	0.001243	0.001243
brand_Opel	0.004380	0.004380
brand_Renault	0.038046	0.038046
brand_Skoda	0.119199	0.119199
brand_Tata	0.098449	0.098449
brand_Toyota	0.053949	0.053949
brand_Volkswagen	0.004060	0.004060

```
brand_Volvo                    0.237412              0.237412
transmission_Automatic         1.000000              1.000000
transmission_Manual            1.000000              1.000000

[49 rows x 49 columns]
```

[1390]:
```
# Visualize the correlation using seaborn
plt.subplots(figsize=(20,15))
sns.heatmap(corr_matrix)
```

[1390]:
```
<AxesSubplot:>
```



[1391]:
```
# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
 ↪bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.98)] +\
```

```
            [column for column in upper.columns if any(upper[column] < 0.01)]

# to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]
# Print what will be dropped
print("The columns to be dropped: " + str(to_drop))
```

```
The columns to be dropped: ['transmission_Manual', 'seats', 'fuel_CNG',
'fuel_LPG', 'seller_Individual', 'owner_First Owner', 'owner_Fourth & Above
Owner', 'owner_Second Owner', 'owner_Test Drive Car', 'owner_Third Owner',
'brand_Ambassador', 'brand_Ashok', 'brand_Audi', 'brand_BMW', 'brand_Chevrolet',
'brand_Daewoo', 'brand_Datsun', 'brand_Fiat', 'brand_Force', 'brand_Ford',
'brand_Honda', 'brand_Hyundai', 'brand_Isuzu', 'brand_Jaguar', 'brand_Jeep',
'brand_Kia', 'brand_Land', 'brand_Lexus', 'brand_MG', 'brand_Mahindra',
'brand_Maruti', 'brand_Mercedes-Benz', 'brand_Mitsubishi', 'brand_Nissan',
'brand_Opel', 'brand_Renault', 'brand_Skoda', 'brand_Tata', 'brand_Toyota',
'brand_Volkswagen', 'brand_Volvo', 'transmission_Automatic',
'transmission_Manual']
```

[1392]:
```
# Drop features, this lowers model complexity, and aids in generalizing the
# model.
df.drop(df[to_drop], axis=1, inplace=True)
```

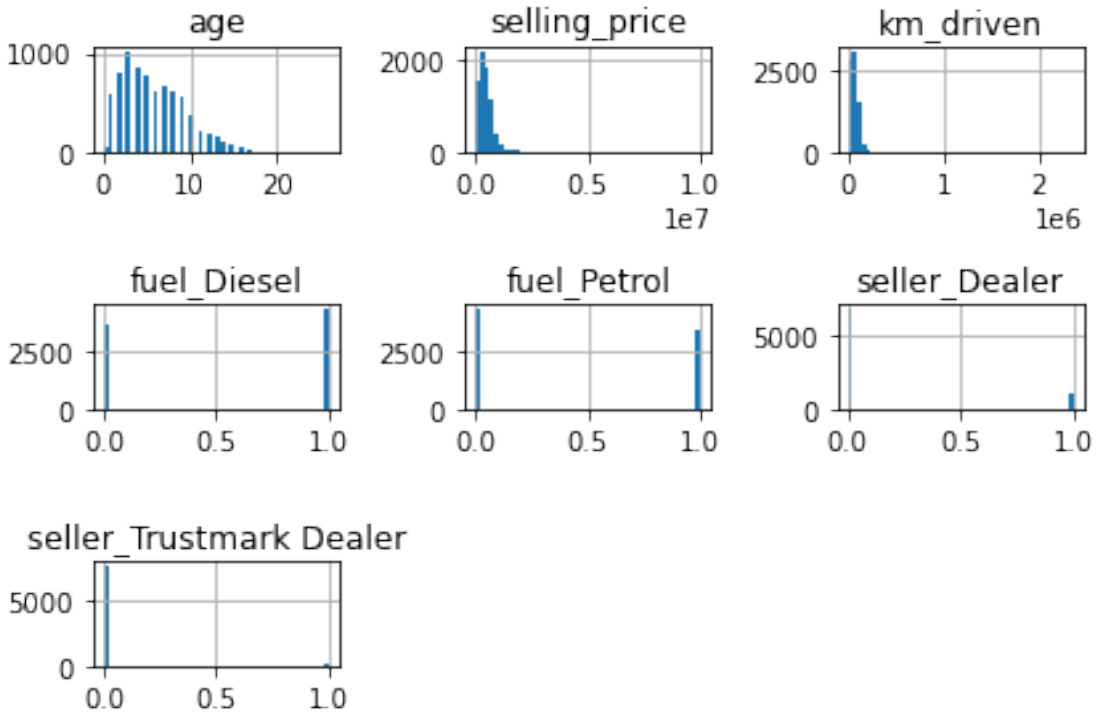
[1393]:
```
# Check the data
df.head()
```

[1393]:

	age	selling_price	km_driven	mileage	engine	max_power	fuel_Diesel	\
0	6	450000	145500	23	1248	74	1	
1	6	370000	120000	21	1498	103	1	
2	14	158000	140000	17	1497	78	0	
3	10	225000	127000	23	1396	90	1	
4	13	130000	120000	16	1298	88	0	

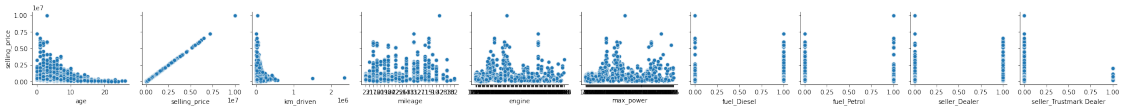
	fuel_Petrol	seller_Dealer	seller_Trustmark Dealer
0	0	0	0
1	0	0	0
2	1	0	0
3	0	0	0
4	1	0	0

[1394]:
```
# Draw histogram of the data for visualization
hist = df.hist(bins=50)
plt.tight_layout()
```



[1395]:
```
# Visualizing the data for each dependent feature with independent feature
pos_of_col = df.columns.get_loc("selling_price") + 1
x_vars = df.iloc[:,np.r_[range(pos_of_col-1),range(pos_of_col,len(df.columns)
                                                   )]].columns
x_vars = df.columns

sns.pairplot(data=df,
             y_vars=['selling_price'],
             x_vars=x_vars,
             diag_kind=None
             )
plt.tight_layout()
```



[1396]:
```
# Training the data
```

[1397]:
```
df.head()
```

[1397]:
```
   age  selling_price  km_driven mileage engine max_power  fuel_Diesel  \
0    6         450000     145500      23   1248        74            1
1    6         370000     120000      21   1498       103            1
2   14         158000     140000      17   1497        78            0
3   10         225000     127000      23   1396        90            1
4   13         130000     120000      16   1298        88            0

   fuel_Petrol  seller_Dealer  seller_Trustmark Dealer
0            0              0                       0
1            0              0                       0
2            1              0                       0
3            0              0                       0
4            1              0                       0
```

[1398]:
```
# Generate different featuresets and save the used columns to a variable for
# later pickleing.
feature_set = df[['age', 'max_power', 'fuel_Diesel', 'km_driven']]
# feature_set = df[['age', 'max_power', 'fuel_Diesel', 'engine']]
# feature_set = df[['age', 'mileage', 'fuel_Diesel', 'km_driven']]
feature_set_cols = '-'.join(list(feature_set.columns))
```

[1399]:
```
# x = df.iloc[:,0:-1].values.astype(float)
y = df['selling_price'].astype(float)
```

[1400]:
```
# Split the data to training and test sets
x_train, x_test, y_train, y_test = train_test_split(feature_set, y, test_size=0.
 ↪2,
                                                    random_state=147)
```

[1401]:
```
# Initialize min-max scaler and transform each feature by using min-max scaler
# You need to put the feature values to a certain range (in general: (0, 1)) in␣
 ↪order to stabilize the model
scaler = MinMaxScaler(feature_range=(0, 1))
x_train = scaler.fit_transform(x_train)
x_train
```

[1401]:
```
array([[0.26923077, 0.19836957, 1.        , 0.03600957],
       [0.26923077, 0.11413043, 1.        , 0.03473863],
       [0.03846154, 0.13315217, 0.        , 0.0023809 ],
       ...,
       [0.15384615, 0.05706522, 0.        , 0.00423605],
       [0.19230769, 0.20652174, 1.        , 0.10167485],
       [0.34615385, 0.11141304, 0.        , 0.02118192]])
```

[1402]:
```
# Print the number of instances in training & test set
print(x_train.shape)
print(x_test.shape)
```

```
(6324, 4)
(1582, 4)
```

[1403]:
```
# Initialize the linear regression model
model = LinearRegression()
```

[1404]:
```
# Fit the training data to the model
model.fit(x_train, y_train)
model.coef_
```

[1404]:
```
array([-1202121.48764798,  5896958.92350053,     8632.78488789,
       -3350152.1377382 ])
```

[1405]:
```
# Print the general formula of our linear regression model
_str = "y = "
for i, m in enumerate(model.coef_):
    _str += "x_{}*{}+".format(i+1, m)
_str += str(model.intercept_)
print(_str)
```

```
y = x_1*-1202121.4876479814+x_2*5896958.923500527+x_3*8632.784887891676+x_4*-335
0152.137738197+73752.16816291772
```

[1406]:
```
# Scale each feature to range(0, 1)
x_test = scaler.transform(x_test)
x_test
```

[1406]:
```
array([[3.46153846e-01, 1.90217391e-01, 1.00000000e+00, 5.16595099e-02],
       [5.76923077e-01, 1.38586957e-01, 0.00000000e+00, 3.60095676e-02],
       [3.84615385e-01, 1.57608696e-01, 1.00000000e+00, 3.81278024e-02],
       ...,
       [7.69230769e-02, 3.20652174e-01, 1.00000000e+00, 8.46870266e-04],
       [2.69230769e-01, 1.14130435e-01, 1.00000000e+00, 1.86307222e-02],
       [3.46153846e-01, 2.55434783e-01, 1.00000000e+00, 6.48175607e-02]])
```

[1407]:
```
x_test[0]
```

[1407]:
```
array([0.34615385, 0.19021739, 1.        , 0.05165951])
```

[1408]:
```
# Predict the values by using all test data
y_pred = model.predict(x_test)
```

[1409]:
```
# Print the predicted and the actual value of the first row in test set
"Predicted: {}, Actual: {}".format(y_pred[1], y_test[1])
```

[1409]:
```
'Predicted: 76824.60059758362, Actual: 370000.0'
```

[1410]:
```
# Calculate the score of the model in test data
score = model.score(x_test, y_test)
print(score)
```

```
0.6417537167824829
```

[1411]:
```
# Calculate mean squared error of predicted values
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
195623662859.41428
```

[1412]:
```
# Calculate absolute squared error of predicted values
mae = mean_absolute_error(y_test, y_pred)
print(mae)
```

```
275135.5069473646
```

[1413]:
```
# Calculate  rsquared error of predicted values
r2e = r2_score(y_test, y_pred)
print(r2e)
```

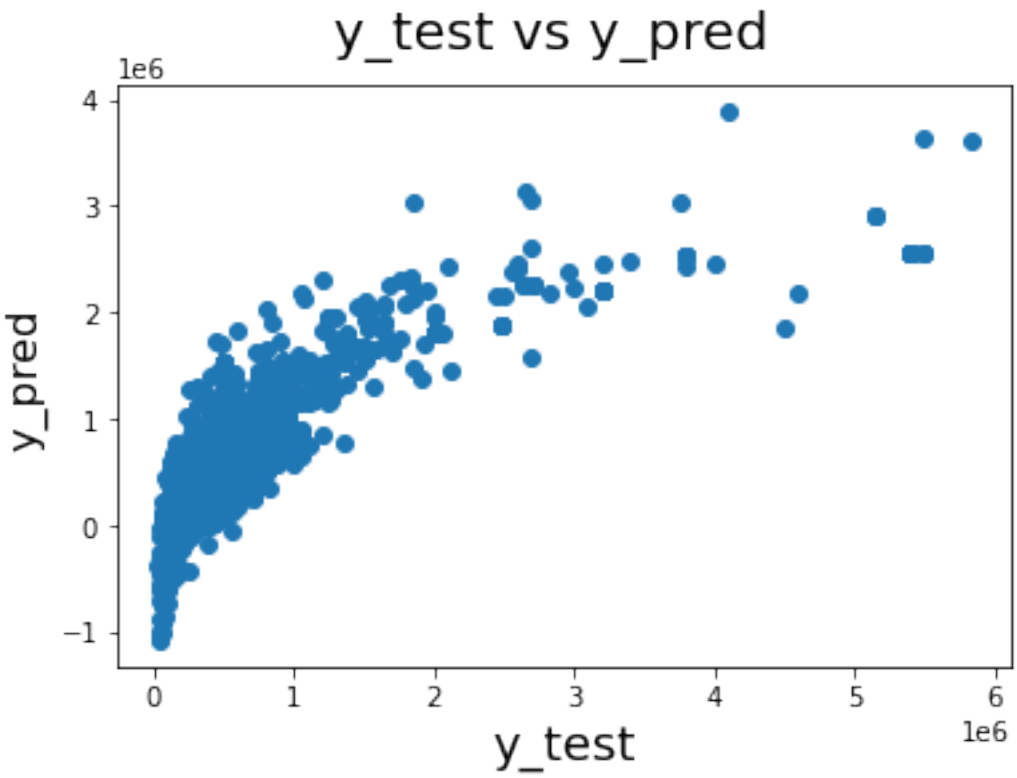
```
0.6417537167824829
```

[1414]:
```
# Calculate  root mean squared error of predicted values
rmse = mean_squared_error(y_test, y_pred, squared=True)
print(rmse)
```

```
195623662859.41428
```

[1415]:
```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```

[1415]:
```
Text(0, 0.5, 'y_pred')
```



[1416]:
```
# Save the model
import pickle
with open("./lr_model_columns{}_score{}_mse{}_mae{}_r2e{}_rmse{}.pkl".format
              (feature_set_cols,score, mse, mae, r2e, rmse), "wb") as f:
    pickle.dump(model, f)
f.close()
```

[1416]:
```

```