```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, GridSearchCV

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv("C:/Users/shank/Desktop/Linear Regression Project/Car details v3.csv")

df.head()
```

	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

```
df.shape
```

```
(8128, 13)
```

```
df.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
```

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

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

# remove bhp from max_power and convert it into float type from object
type
df['max_power'] = df['max_power'].apply(lambda x : x.split()[0] if
type(x)==str else np.nan)
df = df[df['max_power'].str.contains('bhp') == False]
df["max_power"] = df["max_power"].astype(float)

# As seats is categorical column, let's first convert it to
categorical from float
df.seats = df.seats.astype('category')
```

```
df.seats.value_counts()
```

```
5.0     6254
7.0     1120
```

```
8.0      235
4.0      133
9.0       80
6.0       62
10.0      19
2.0        2
14.0       1
Name: seats, dtype: int64
```

```
df.torque
```

```
0                   190Nm@ 2000rpm
1              250Nm@ 1500-2500rpm
2            12.7@ 2,700(kgm@ rpm)
3         22.4 kgm at 1750-2750rpm
4            11.5@ 4,500(kgm@ rpm)
                   ...
8123              113.7Nm@ 4000rpm
8124    24@ 1,900-2,750(kgm@ rpm)
8125                190Nm@ 2000rpm
8126           140Nm@ 1800-3000rpm
8127           140Nm@ 1800-3000rpm
Name: torque, Length: 7912, dtype: object
```

```
# we will drop the torque as it doesn't have proper format
df.drop(["torque"],axis=1,inplace=True)

# check missing values number in each column
df.isna().sum()
```

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
mileage          6
engine           6
max_power        0
seats            6
dtype: int64
```

```
df[df["mileage"].isnull() & df["engine"].isnull() &
df["seats"].isnull()]
```

```
                                name  year  selling_price  km_driven
fuel  \
575              Maruti Alto K10 LXI  2011         204999      97500
Petrol
576              Maruti Alto K10 LXI  2011         204999      97500
```

```
Petrol
1442  Maruti Swift Dzire VDI Optional  2017         589000      41232
Diesel
1443  Maruti Swift Dzire VDI Optional  2017         589000      41232
Diesel
2549   Tata Indica Vista Quadrajet LS  2012         240000      70000
Diesel
2550   Tata Indica Vista Quadrajet LS  2012         240000      70000
Diesel

     seller_type transmission        owner  mileage engine  max_power
seats
575   Individual       Manual  First Owner      NaN    NaN        0.0
NaN
576   Individual       Manual  First Owner      NaN    NaN        0.0
NaN
1442      Dealer       Manual  First Owner      NaN    NaN        0.0
NaN
1443      Dealer       Manual  First Owner      NaN    NaN        0.0
NaN
2549  Individual       Manual  First Owner      NaN    NaN        0.0
NaN
2550  Individual       Manual  First Owner      NaN    NaN        0.0
NaN
```

```
df.dropna(inplace=True)
```

```
df.shape
```

```
(7906, 12)
```

```
df["engine"] = df["engine"].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7906 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           7906 non-null   object
 1   year           7906 non-null   int64
 2   selling_price  7906 non-null   int64
 3   km_driven      7906 non-null   int64
 4   fuel           7906 non-null   object
 5   seller_type    7906 non-null   object
 6   transmission   7906 non-null   object
 7   owner          7906 non-null   object
 8   mileage        7906 non-null   float64
 9   engine         7906 non-null   int32
 10  max_power      7906 non-null   float64
```

```
 11  seats          7906 non-null   category
dtypes: category(1), float64(2), int32(1), int64(3), object(5)
memory usage: 718.4+ KB
```

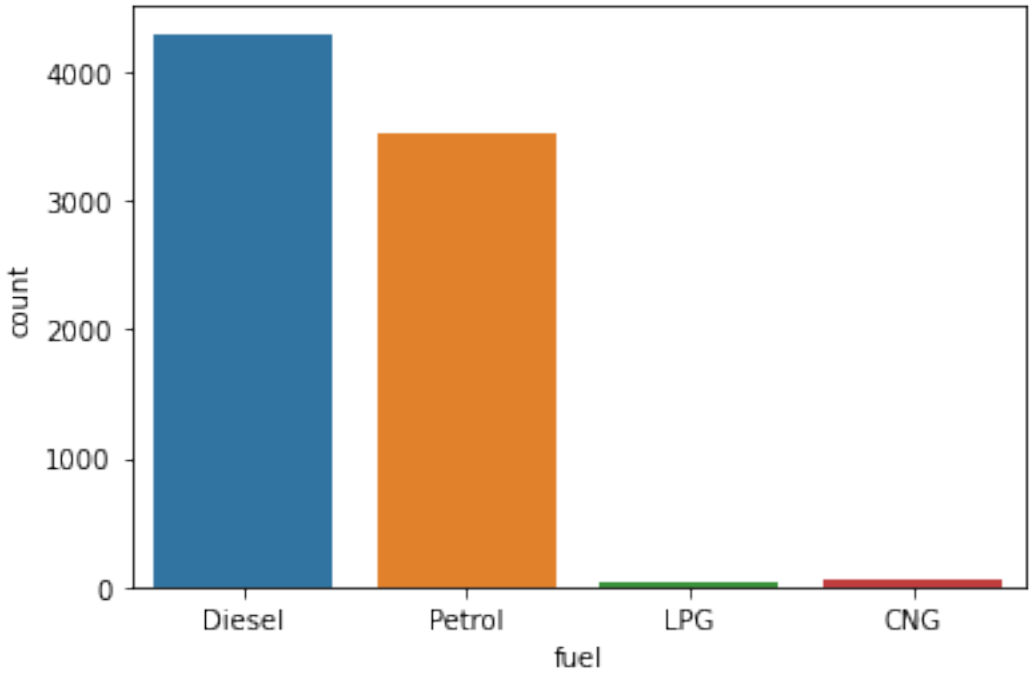
```
df["old"] = 2022-df["year"]
df.drop(["year"],axis=1,inplace=True)
```

```
#EDA
```

```
df["fuel"].value_counts()
```

```
Diesel    4299
Petrol    3520
CNG         52
LPG         35
Name: fuel, dtype: int64
```

```
sns.countplot(x="fuel",data = df)
plt.show()
```

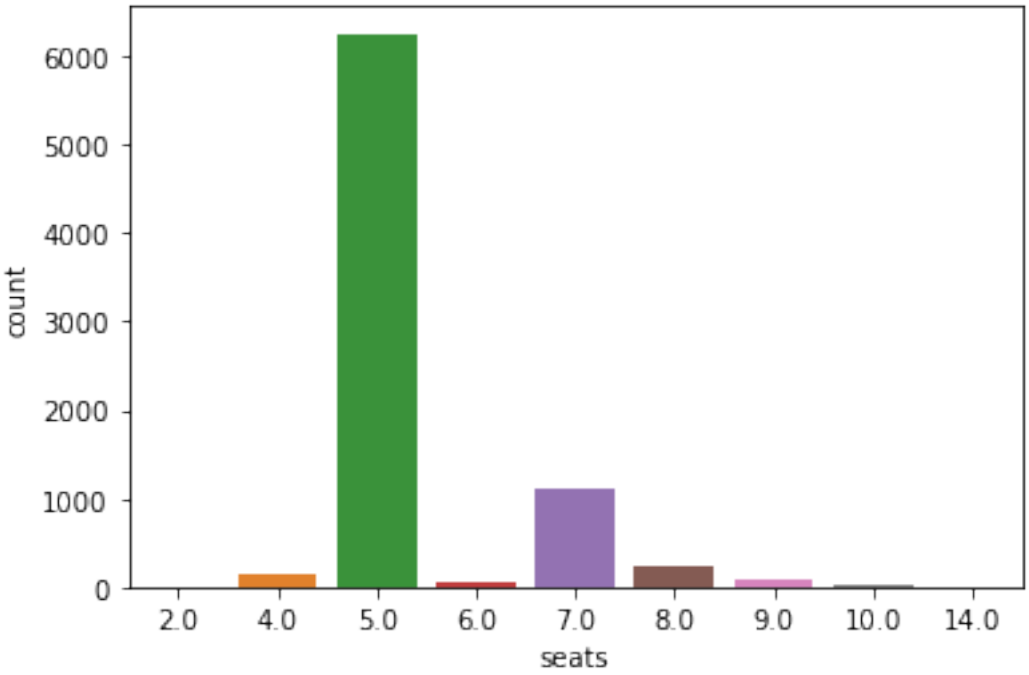



```
df["seats"].value_counts()
```

```
5.0     6254
7.0     1120
8.0      235
4.0      133
9.0       80
6.0       62
10.0      19
2.0        2
```

```
14.0       1
Name: seats, dtype: int64
```

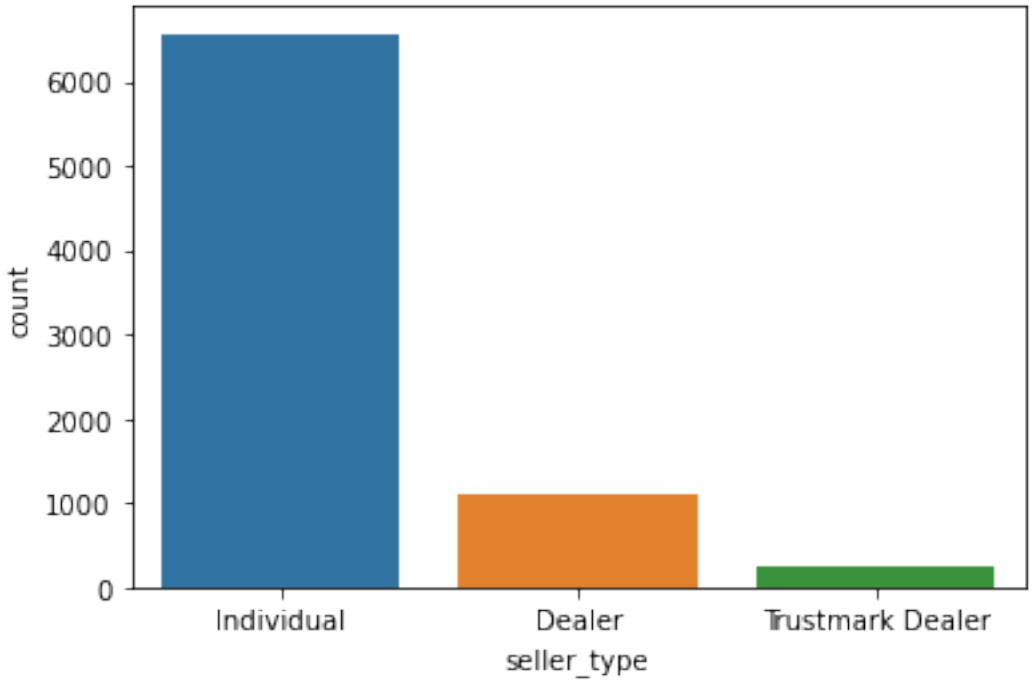
```
sns.countplot(x="seats",data = df)
plt.show()
```




```
df["seller_type"].value_counts()
```

```
Individual          6563
Dealer              1107
Trustmark Dealer     236
Name: seller_type, dtype: int64
```

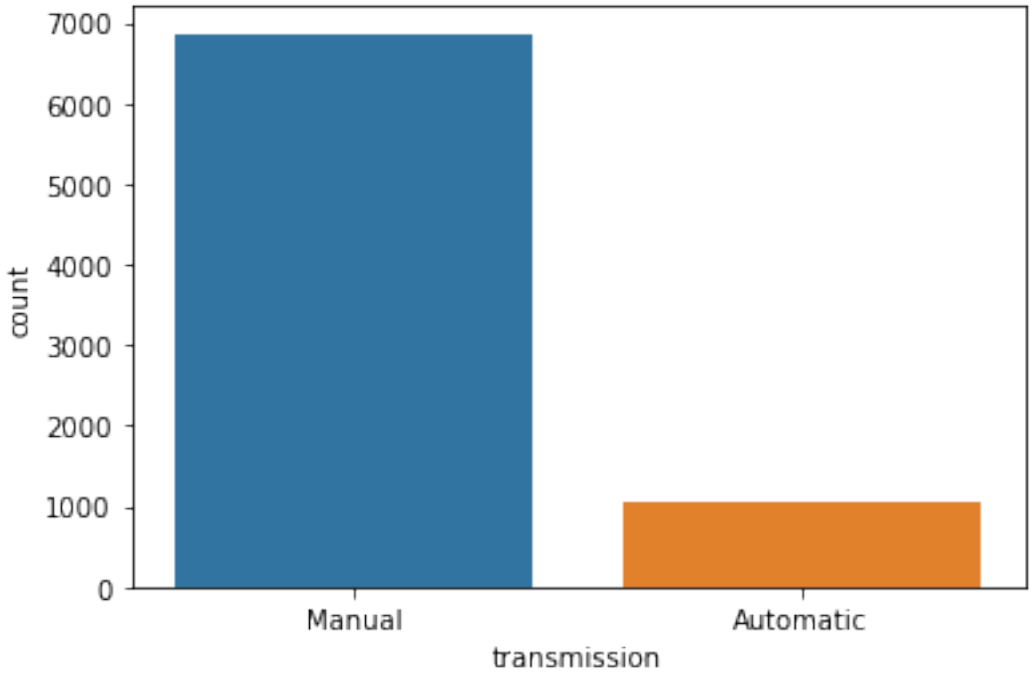
```
sns.countplot(x="seller_type",data = df)
plt.show()
```




```
df["transmission"].value_counts()
```

```
Manual       6865
Automatic    1041
Name: transmission, dtype: int64
```

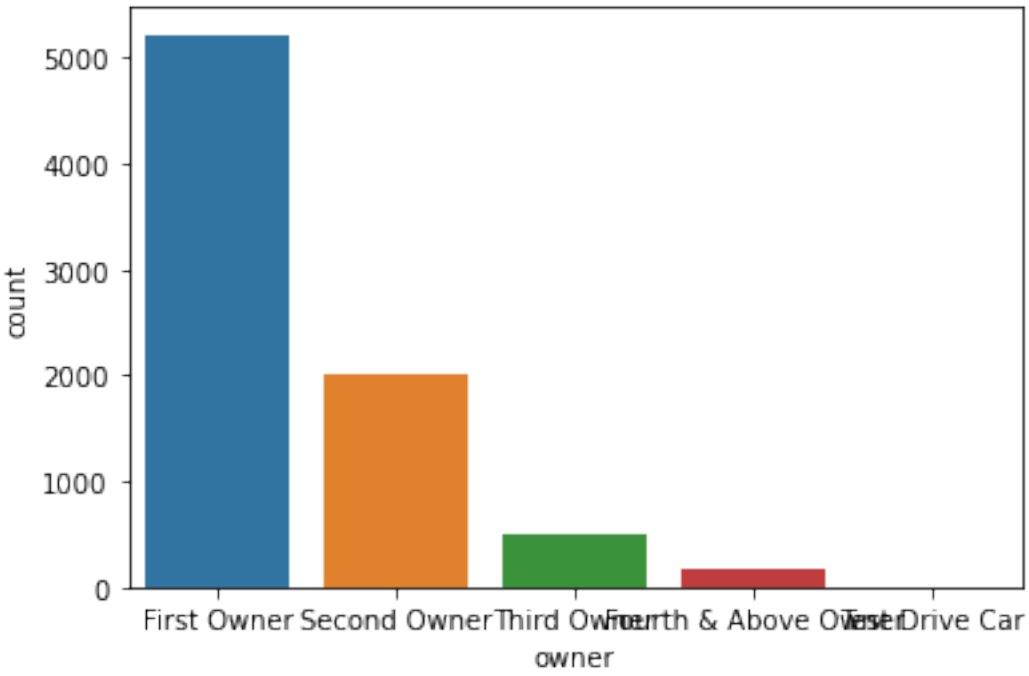
```
sns.countplot(x="transmission",data = df)
plt.show()
```



```
df["owner"].value_counts()
```

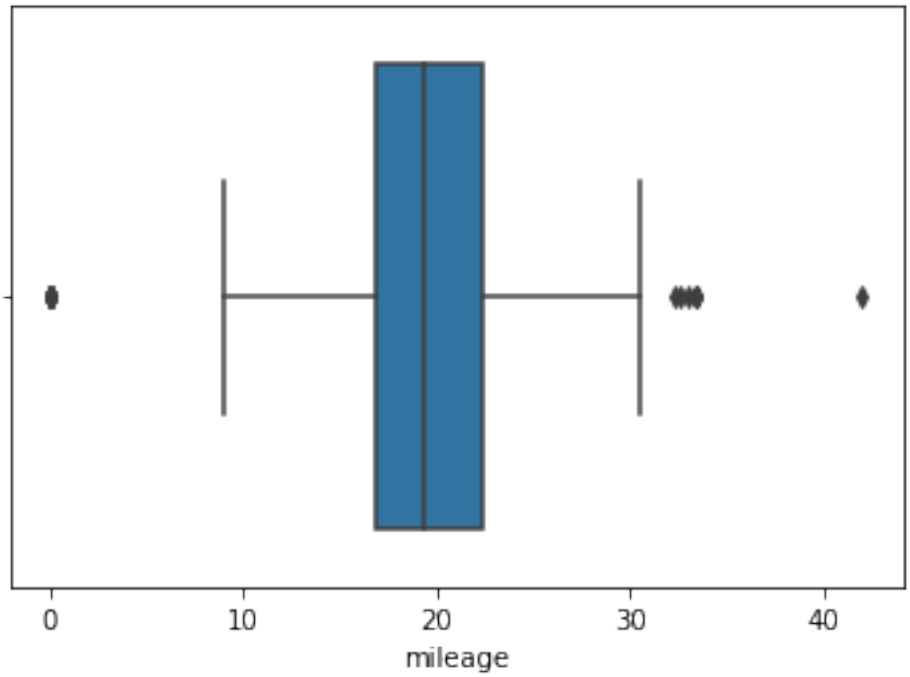
```
First Owner             5215
Second Owner            2016
Third Owner              510
Fourth & Above Owner     160
Test Drive Car             5
Name: owner, dtype: int64
```

```
sns.countplot(x="owner",data = df)
plt.show()
```




```
sns.boxplot(x=df['mileage'])
```

```
<AxesSubplot:xlabel='mileage'>
```




```
df[df["mileage"]==0]
```

	name	selling_price
644	Tata Indica Vista Aura Safire Anniversary Edition	135000
785	Hyundai Santro Xing GL	120000
1649	Hyundai Santro Xing GL	105000
1676	Mercedes-Benz M-Class ML 350 4Matic	1700000
2137	Land Rover Freelander 2 TD4 HSE	1650000
2366	Hyundai Santro Xing (Non-AC)	110000
2725	Hyundai Santro Xing (Non-AC)	184000
4527	Mercedes-Benz M-Class ML 350 4Matic	1700000
5276	Hyundai Santro Xing GL	175000
5843	Volkswagen Polo GT TSI BSIV	574000
5846	Volkswagen Polo GT TSI BSIV	575000
5900	Mahindra Bolero Pik-Up FB 1.7T	679000

6534	Hyundai Santro Xing GL	150000
6629	Mahindra Bolero Pik-Up CBC 1.7T	722000
6824	Hyundai Santro Xing GL	150000
7002	Hyundai Santro Xing (Non-AC)	110000
7337	Mercedes-Benz GLC 220d 4MATIC	3300000

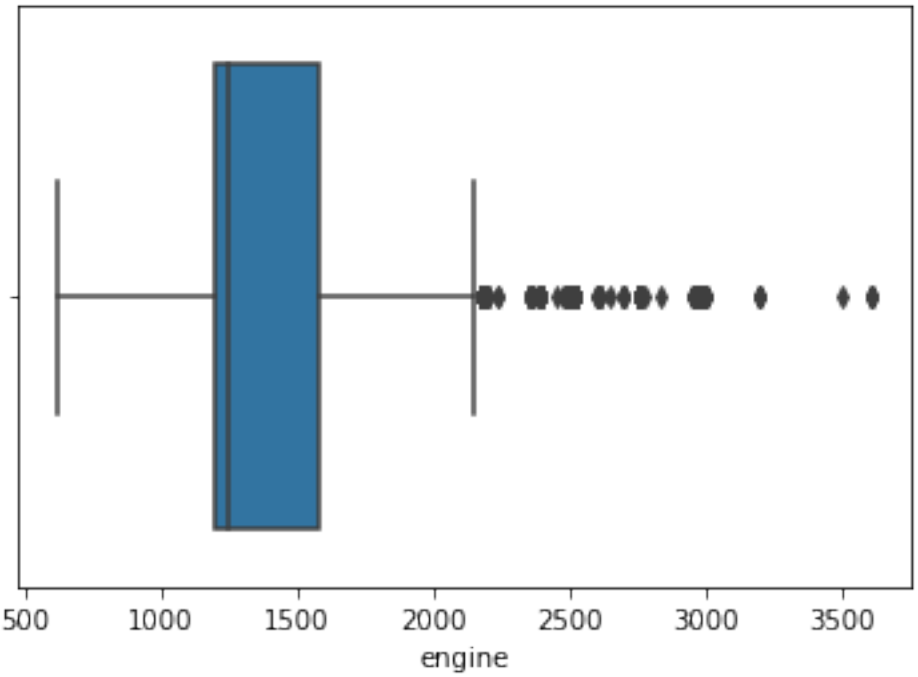
	km_driven	fuel	seller_type	transmission	owner
644	28900	Petrol	Individual	Manual	Second Owner
785	90000	Petrol	Individual	Manual	Second Owner
1649	128000	Petrol	Individual	Manual	First Owner
1676	110000	Diesel	Individual	Automatic	Third Owner
2137	64788	Diesel	Dealer	Automatic	First Owner
2366	80000	Petrol	Individual	Manual	Second Owner
2725	15000	Petrol	Individual	Manual	First Owner
4527	110000	Diesel	Individual	Automatic	Third Owner
5276	40000	Petrol	Individual	Manual	First Owner
5843	28080	Petrol	Dealer	Automatic	First Owner
5846	28100	Petrol	Dealer	Automatic	First Owner
5900	5000	Diesel	Individual	Manual	First Owner
6534	110000	Petrol	Individual	Manual	First Owner
6629	80000	Diesel	Individual	Manual	First Owner
6824	40000	Petrol	Individual	Manual	Fourth & Above Owner
7002	80000	Petrol	Individual	Manual	Second Owner
7337	60000	Diesel	Dealer	Automatic	First Owner

```
      mileage  engine  max_power seats  old
644       0.0    1172      65.00   5.0   13
785       0.0    1086      62.00   5.0   13
1649      0.0    1086      62.00   5.0   14
1676      0.0    2987     165.00   5.0   11
2137      0.0    2179     115.00   5.0    9
2366      0.0    1086      62.10   5.0   12
2725      0.0    1086      62.10   5.0    9
4527      0.0    2987     165.00   5.0   11
5276      0.0    1086      62.00   5.0   14
5843      0.0    1197     103.25   5.0    8
5846      0.0    1197     103.25   5.0    8
5900      0.0    2523      70.00   2.0    2
6534      0.0    1086      62.00   5.0   12
6629      0.0    2523      70.00   2.0    3
6824      0.0    1086      62.00   5.0   11
7002      0.0    1086      62.10   5.0   12
7337      0.0    1950     194.00   5.0    5
```

```
df=df[df["mileage"]!=0]
```

```
sns.boxplot(x=df['engine'])
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
df[df["engine"]>3500]
```

```
                                name  selling_price  km_driven
fuel  \
```

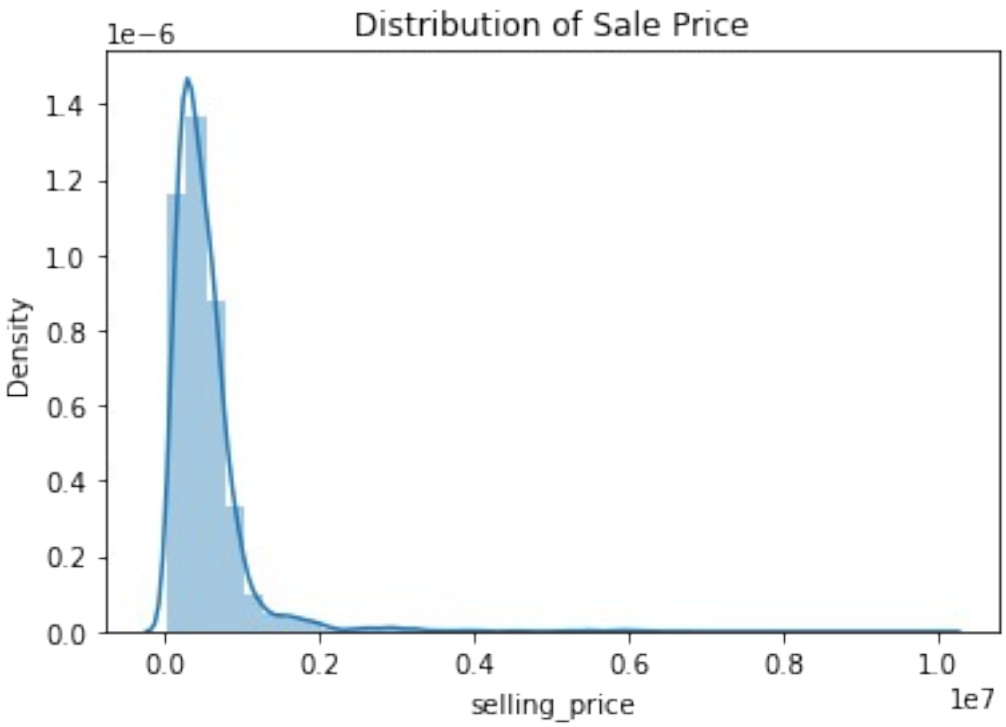
```
134   Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol
1564  Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol
1860  Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol
3239  Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol
5248  Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol
7703  Jeep Wrangler 2016-2019 3.6 4X4        4100000      17000
Petrol

     seller_type transmission        owner  mileage  engine  max_power
seats  \
134   Individual    Automatic  First Owner      9.5    3604      280.0
5.0
1564  Individual    Automatic  First Owner      9.5    3604      280.0
5.0
1860  Individual    Automatic  First Owner      9.5    3604      280.0
5.0
3239  Individual    Automatic  First Owner      9.5    3604      280.0
5.0
5248  Individual    Automatic  First Owner      9.5    3604      280.0
5.0
7703  Individual    Automatic  First Owner      9.5    3604      280.0
5.0

      old
134     5
1564    5
1860    5
3239    5
5248    5
7703    5
```

```
df=df.drop_duplicates()
```

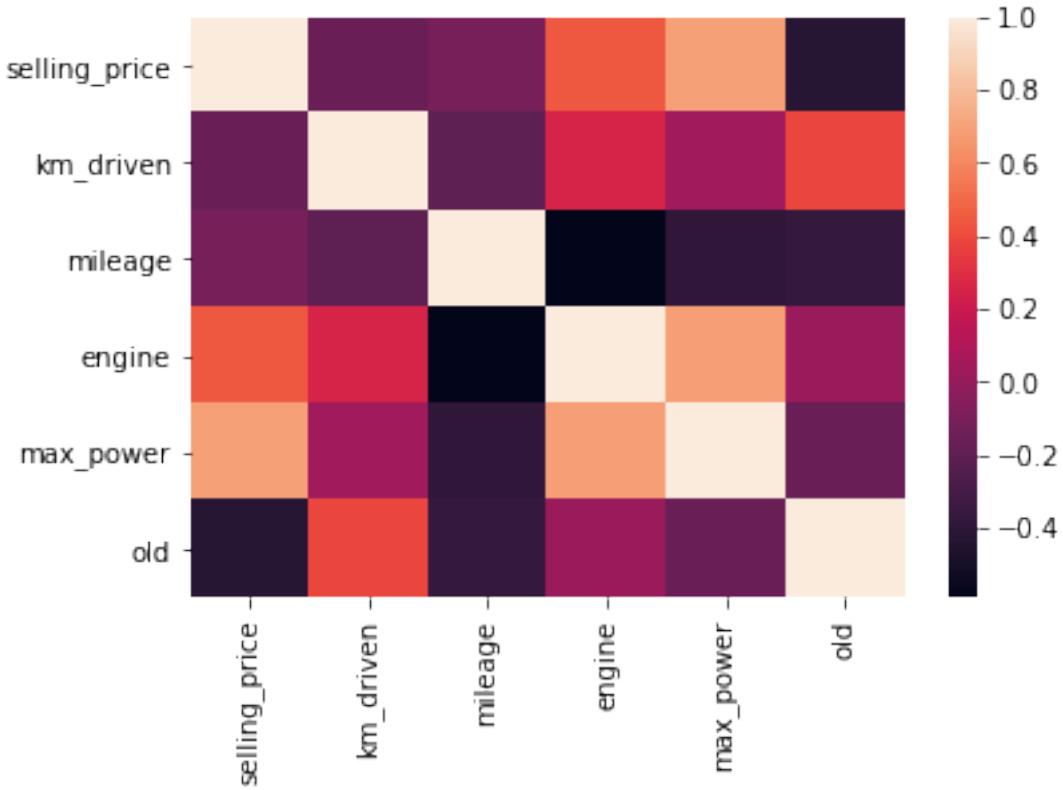
```
df.shape
```

```
(6702, 12)
```

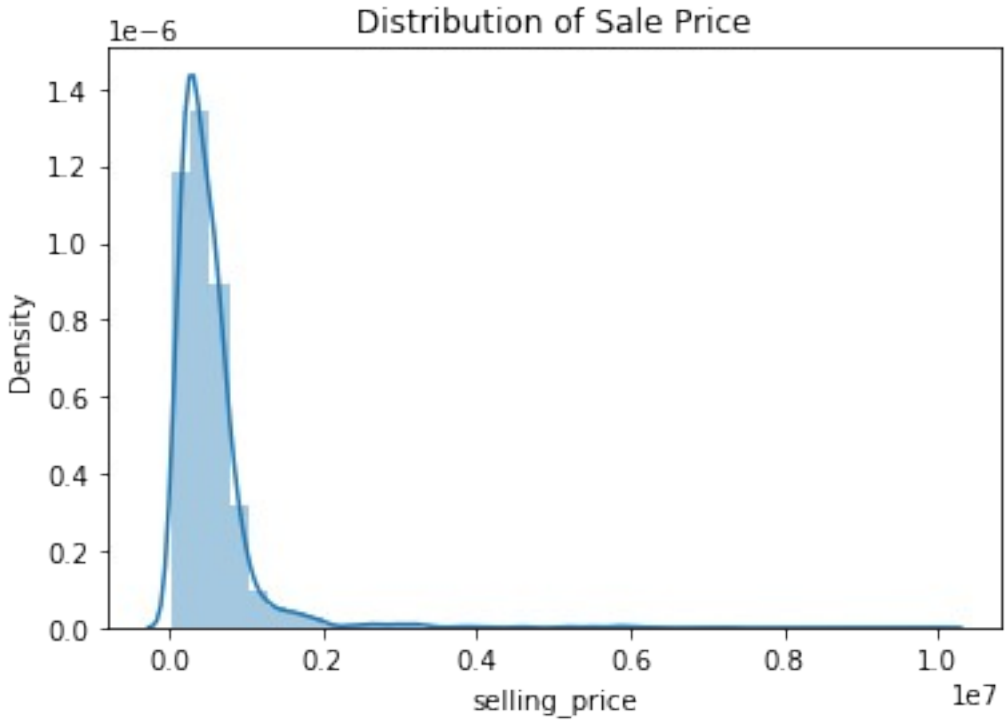
```
sns.distplot(df["selling_price"],hist=True, kde=True,
bins=40).set_title('Distribution of Sale Price')
plt.show()
```



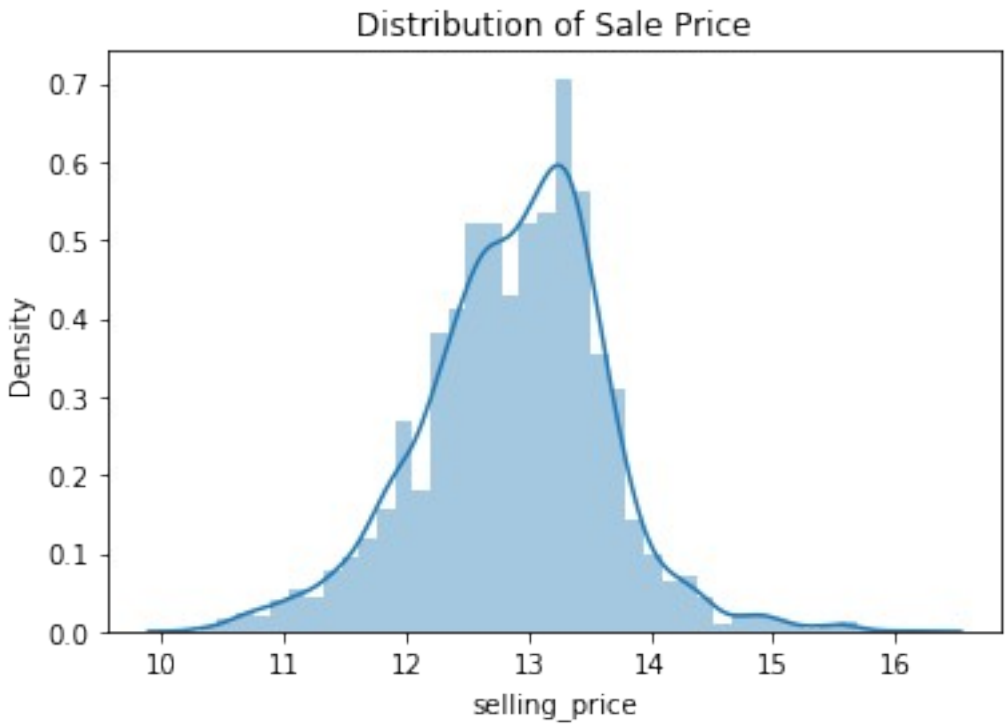

```
sns.heatmap(df.corr())
plt.show()
```



```
#selling price is negatively correlated with old variable, km_driven
as expected, means as number of years increases or km_driven increses
sp decreases
#selling price is positevly correlated with max_power and engine

#Drop car name as it is insignificant to our regression analysis.
df.drop(["name"],axis=1,inplace=True)

#REGRESSION ANALYSIS

# split the data into X and y
y = df.pop("selling_price")
X = df

# split data into train and test
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.3,random_state=1)

sns.distplot(y_train, hist=True, kde=True,
bins=40).set_title('Distribution of Sale Price')
plt.show()
```




```
# apply log transformation on y_train
y_train=np.log(y_train)

sns.distplot(y_train, hist=True, kde=True,
bins=40).set_title('Distribution of Sale Price')
plt.show()
```




```
# select categorical data and apply get_dummies function on them
categorical_df = X_train.select_dtypes(include=['object',"category"])
dummies = pd.get_dummies(categorical_df, drop_first=True)

# drop categories for which we created dummy variables
X_train = X_train.drop(list(categorical_df.columns), axis=1)

# concat both dummy vars df and original df
X_train = pd.concat([X_train,dummies], axis=1)

scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train),columns =
X_train.columns,index=X_train.index)
X_train.head()
```

```
      km_driven   mileage    engine  max_power       old  fuel_Diesel
\
4868   0.025418  0.133333  0.392617   0.251089  0.461538          0.0

5967   0.016945  0.416364  0.057718   0.039488  0.192308          0.0

2512   0.050837  0.333333  0.260067   0.095861  0.384615          1.0

6869   0.002118  0.490000  0.058725   0.055828  0.115385          0.0

4556   0.025418  0.451515  0.000000   0.012745  0.192308          0.0
```

```
      fuel_LPG  fuel_Petrol  seller_type_Individual  \
4868       0.0          1.0                     1.0
5967       0.0          1.0                     1.0
2512       0.0          0.0                     1.0
6869       0.0          1.0                     1.0
4556       0.0          1.0                     1.0

      seller_type_Trustmark Dealer  ...  owner_Test Drive Car  \
4868                          0.0  ...                   0.0
5967                          0.0  ...                   0.0
2512                          0.0  ...                   0.0
6869                          0.0  ...                   0.0
4556                          0.0  ...                   0.0

      owner_Third Owner  seats_4.0  seats_5.0  seats_6.0  seats_7.0  \
4868                0.0        0.0        1.0        0.0        0.0
5967                0.0        0.0        1.0        0.0        0.0
2512                0.0        0.0        1.0        0.0        0.0
6869                0.0        0.0        1.0        0.0        0.0
4556                0.0        1.0        0.0        0.0        0.0

      seats_8.0  seats_9.0  seats_10.0  seats_14.0
4868        0.0        0.0         0.0         0.0
5967        0.0        0.0         0.0         0.0
2512        0.0        0.0         0.0         0.0
6869        0.0        0.0         0.0         0.0
4556        0.0        0.0         0.0         0.0

[5 rows x 23 columns]
```

```
reg = LinearRegression()
reg.fit(X_train,y_train)
```

```
LinearRegression()
```

```
y_pred_train = reg.predict(X_train)
```

```
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
```

```
rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
```

```
mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
```

```
0.8497821934534971
401.6445664119136
0.08562024438540047
```

```
# Apply same steps on test data as performed for training data

# apply log transformation on y_train
y_test = np.log(y_test)
# select categorical data and apply get_dummies function on them
categorical_df = X_test.select_dtypes(include=['object',"category"])
dummies = pd.get_dummies(categorical_df, drop_first=True)

# drop categories for which we created dummy variables
X_test = X_test.drop(list(categorical_df.columns), axis=1)

# concat both dummy vars df and original df
X_test = pd.concat([X_test,dummies], axis=1)

# Apply min max scaling to bring down the values to range from 0-1
X_test = pd.DataFrame(scaler.transform(X_test),columns =
X_test.columns,index=X_test.index)
X_test.head()
```

```
      km_driven   mileage    engine  max_power       old  fuel_Diesel
\
2844   0.044483  0.436364  0.209396   0.112200  0.269231          1.0

1691   0.002118  0.352424  0.192617   0.093137  0.076923          0.0

2939   0.004236  0.193030  0.191946   0.109477  0.115385          0.0

675    0.033891  0.306061  0.192282   0.133987  0.153846          0.0

3701   0.029655  0.290909  0.192282   0.144336  0.307692          0.0


      fuel_LPG  fuel_Petrol  seller_type_Individual  \
2844       0.0          0.0                     1.0
1691       0.0          1.0                     1.0
2939       0.0          1.0                     1.0
675        0.0          1.0                     1.0
3701       0.0          1.0                     1.0

      seller_type_Trustmark Dealer  ...  owner_Test Drive Car  \
2844                           0.0  ...                   0.0
1691                           0.0  ...                   0.0
2939                           0.0  ...                   0.0
675                            0.0  ...                   0.0
3701                           0.0  ...                   0.0
```

```
      owner_Third Owner  seats_4.0  seats_5.0  seats_6.0  seats_7.0  \
2844                1.0        0.0        1.0        0.0        0.0
1691                0.0        0.0        1.0        0.0        0.0
2939                0.0        0.0        1.0        0.0        0.0
675                 0.0        0.0        1.0        0.0        0.0
3701                0.0        0.0        1.0        0.0        0.0

      seats_8.0  seats_9.0  seats_10.0  seats_14.0
2844        0.0        0.0         0.0         0.0
1691        0.0        0.0         0.0         0.0
2939        0.0        0.0         0.0         0.0
675         0.0        0.0         0.0         0.0
3701        0.0        0.0         0.0         0.0

[5 rows x 23 columns]
```

```
y_pred_test = reg.predict(X_test)
```

```
r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
```

```
rss1_lr = np.sum(np.square(y_test - y_pred_test))
print(rss1_lr)
```

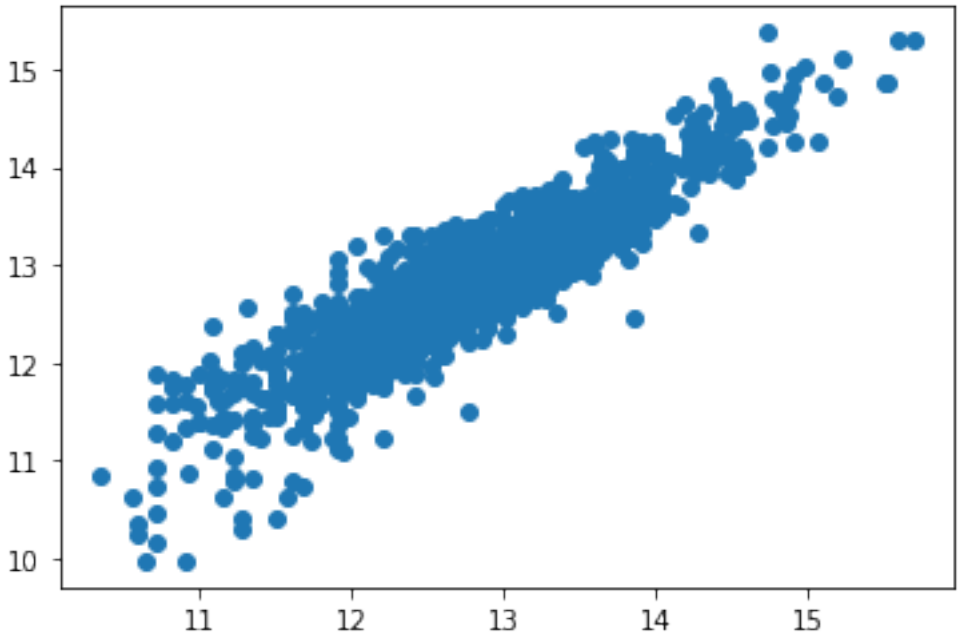
```
mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
```

```
0.8321875624037562
182.5951442486128
0.09079818212263192
```

```
#we got 84.9% accuacy on train data and 83.2% accuracy on test data.
```

```
plt.scatter(y_test,y_pred_test)
```

```
<matplotlib.collections.PathCollection at 0x20748909400>
```




```
from sklearn.metrics import confusion_matrix
```