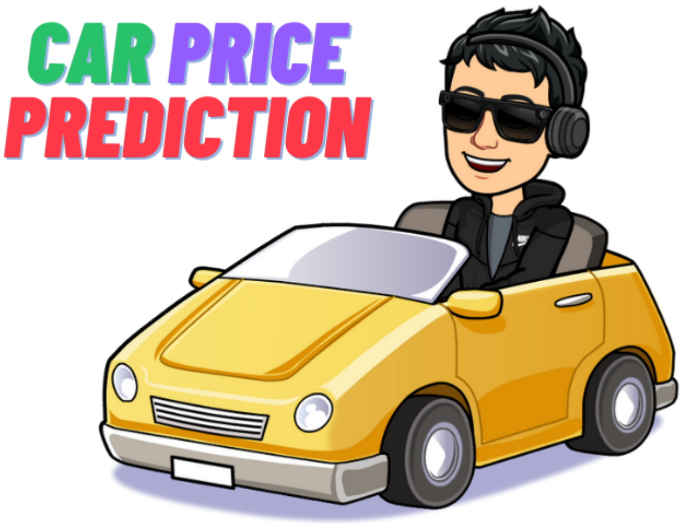
Name of the Intern : SHAHNAWAZ ALAM

Data Science Internship

Oasis Infobyte

Task3:Car price prediction with Machine Learning

Batch- April Phase 1 OIBSIP



Importing Libraries¶

In [3]:

```
# Importing libraries

import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import shapiro
from scipy.stats import anderson
from scipy.stats import normaltest
from scipy.stats import norm
from scipy.stats import skew
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from scipy import stats
import re
import warnings
from pandas.api.types import is_string_dtype
from pandas.api.types import is_numeric_dtype
from wordcloud import WordCloud, STOPWORDS
warnings.filterwarnings('ignore')
%matplotlib inline
```

Importing Dataset

In [9]:

```
print("Importing data...")
df=pd.read_csv(r"C:\Users\md naiyer azam\Desktop\CarPrice.csv")
print("Sucessfully imported.")
```

```
Importing data...
Sucessfully imported.
```

In [10]:

```
df.head()
```

Out[10]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fu
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [11]:

```
df.tail()
```

Out[11]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fue
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	

5 rows × 26 columns

In [12]:

```
df.shape    ##to get no. of rows and column(rows,column)
```

Out[12]:

```
(205, 26)
```

In [13]:

```
df.info()    #info of data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [14]:

```
df.describe()  #description of data
```

Out[14]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compr
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	

In [15]:

```
#Dropping car_ID and symboling columns

df = df.drop('car_ID', 1)
df = df.drop('symboling', 1)
```

In [16]:

```
#check null values

df.isnull().sum()
```

Out[16]:

```
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

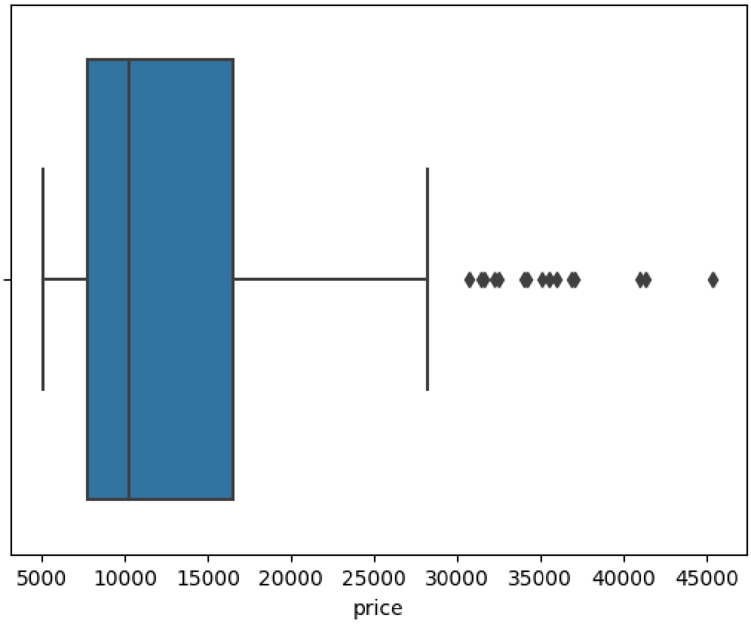
Exploratory data analysis

In [17]:

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

Out[17]:

<AxesSubplot: xlabel='price'>



It looks like there are some outliers. Let's use z-score to check which obsrvation we can call 'outlier'. The data with z-score > 3 or < -3 should be considered as outliers:

In [18]:

```
z_score = stats.zscore(df['price'])
outlier = df[np.abs(z_score) > 3]
print(outlier)
```

```
                          CarName fueltype aspiration doornumber  carbody  \
16                         bmw x5      gas        std        two    sedan
73           buick century special      gas        std       four    sedan
74  buick regal sport coupe (turbo)      gas        std        two  hardtop

   drivewheel enginelocation  wheelbase  carlength  carwidth  ...  enginesize  \
16        rwd          front      103.5      193.8      67.9  ...         209
73        rwd          front      120.9      208.1      71.7  ...         308
74        rwd          front      112.0      199.2      72.0  ...         304

    fuelsystem boreratio stroke  compressionratio horsepower  peakrpm  \
16        mpfi      3.62   3.39               8.0        182     5400
73        mpfi      3.80   3.35               8.0        184     4500
74        mpfi      3.80   3.35               8.0        184     4500

    citympg  highwaympg    price
16       16          22  41315.0
73       14          16  40960.0
74       14          16  45400.0

[3 rows x 24 columns]
```

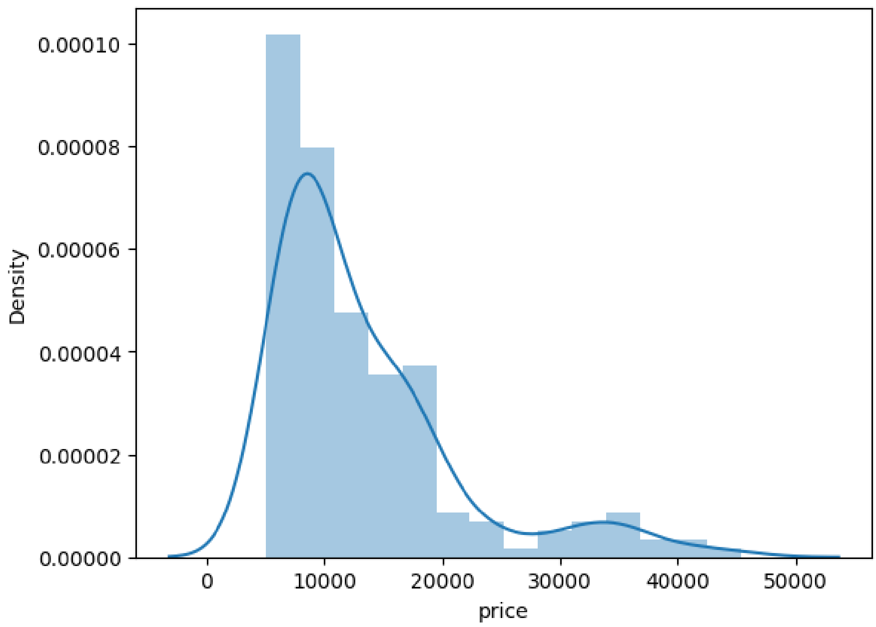
We have some outliers. Maybe we would need to remove it

In [19]:

```
sns.distplot(df['price'])
```

Out[19]:

```
<AxesSubplot: xlabel='price', ylabel='Density'>
```



The plot above shows that the distribution is skewed, so we will apply log-transformation

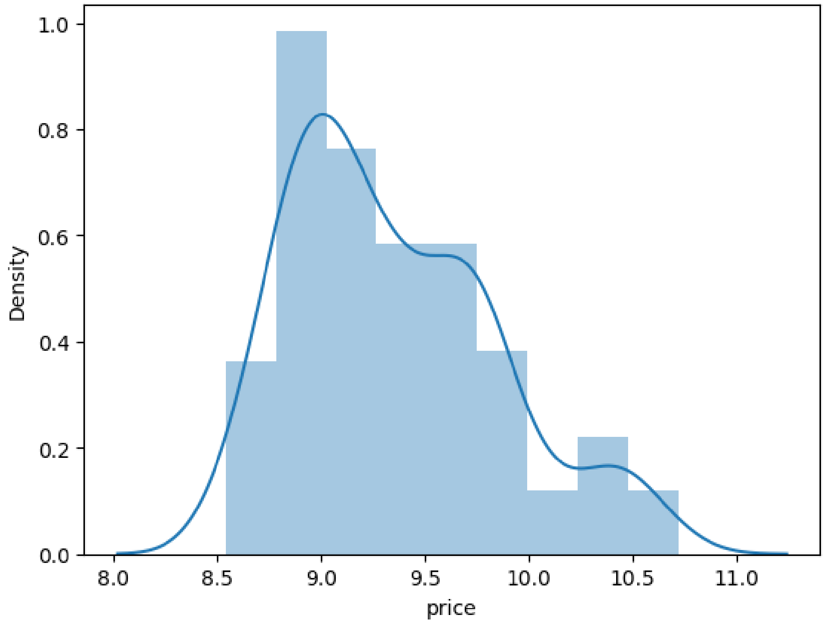
In [20]:

```
#apply log-transformation
df['price'] = np.log1p(df['price'])

sns.distplot(df['price'])
```

Out[20]:

```
<AxesSubplot: xlabel='price', ylabel='Density'>
```

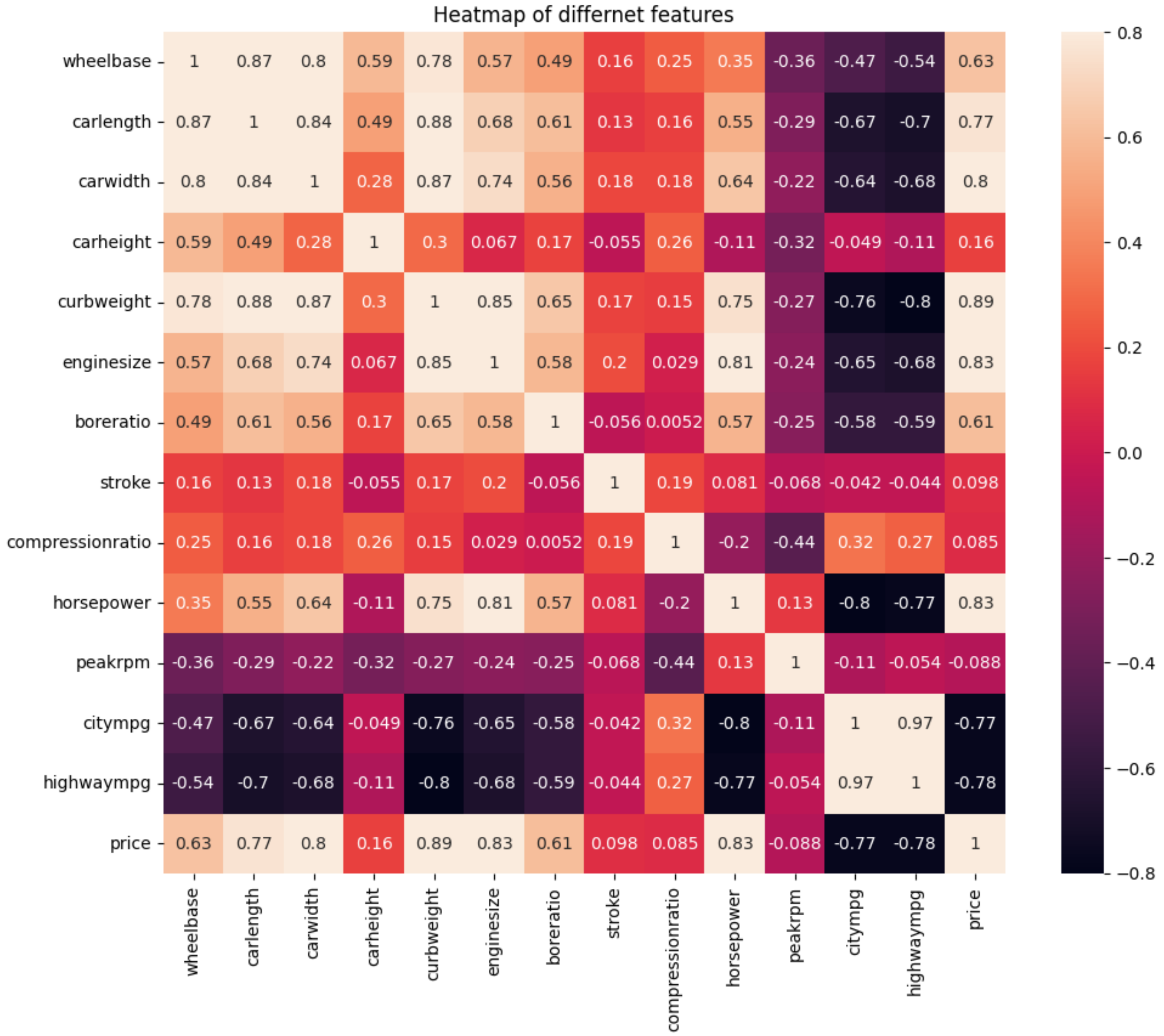


Now the distribution looks much more normal.

In [21]:

```
#correlation matrix
corrmat = df.corr()

f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, annot_kws={'size': 10}, annot = True, square=True).set(title='Heatmap of differnet features')
```

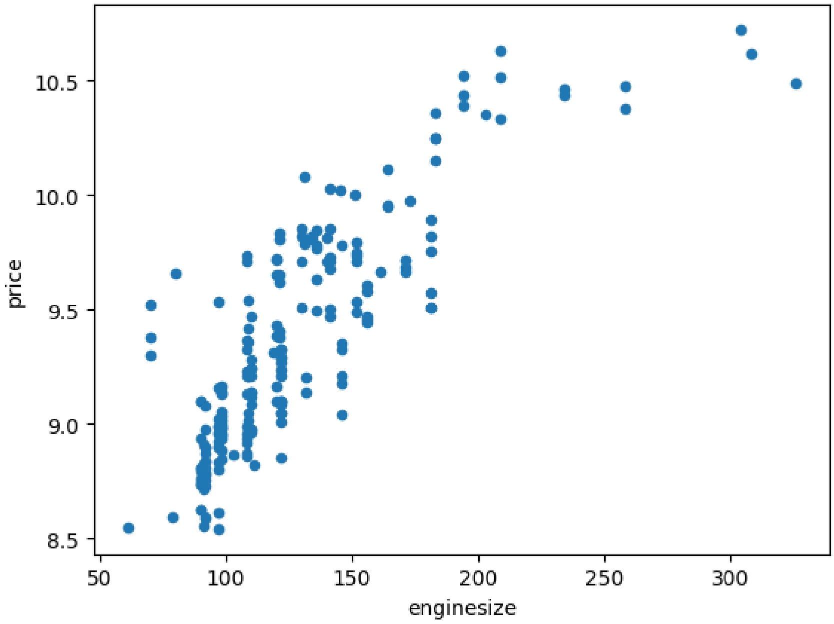


Observation

We can see that many features have strong positive correlation with each other.

In [22]:

```
df.plot.scatter(x='enginesize', y='price');
```



Looking at the plot above we can see strong relashionship between variables.

PCA

In [23]:

```
pca_columns = ["enginesize", "curbweight", "horsepower",
               "carwidth", "carlength", "wheelbase",
               "boreratio", "citympg", "highwaympg"]
pca_columns
data_pca = df[pca_columns]
#data_train.head()
#standardizing data
StandardScaler().fit_transform(data_pca)
data_pca.head()
```

Out[23]:

	enginesize	curbweight	horsepower	carwidth	carlength	wheelbase	boreratio	citympg	highwaympg
0	130	2548	111	64.1	168.8	88.6	3.47	21	27
1	130	2548	111	64.1	168.8	88.6	3.47	21	27
2	152	2823	154	65.5	171.2	94.5	2.68	19	26
3	109	2337	102	66.2	176.6	99.8	3.19	24	30
4	136	2824	115	66.4	176.6	99.4	3.19	18	22

In [24]:

```
#PCA

from sklearn.decomposition import PCA
pca_train = PCA(n_components=2)
principal_components = pca_train.fit_transform(data_pca)
pca_train.explained_variance_ratio_
```

Out[24]:

```
array([0.99548045, 0.00323041])
```

In [25]:

```
principal_data = pd.DataFrame(data = principal_components, columns = ['pca_1', 'pca_2'])
principal_data.head()
```

Out[25]:

	pca_1	pca_2
0	-7.057267	9.466968
1	-7.057267	9.466968
2	270.841223	33.378201
3	-218.880370	6.379076
4	268.738449	-7.936346

Observation

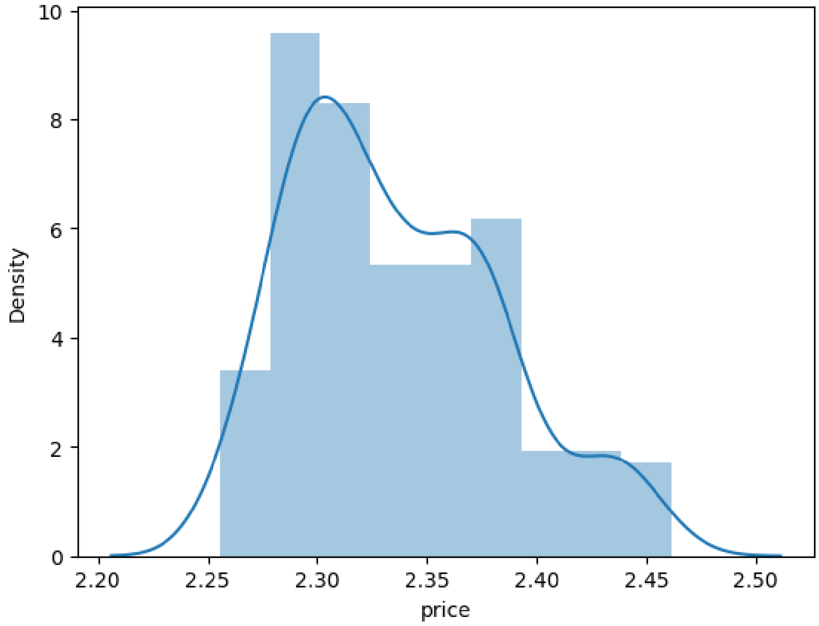
PCA allows us to reduce the dimension of variables which were correlated to price and to each other.

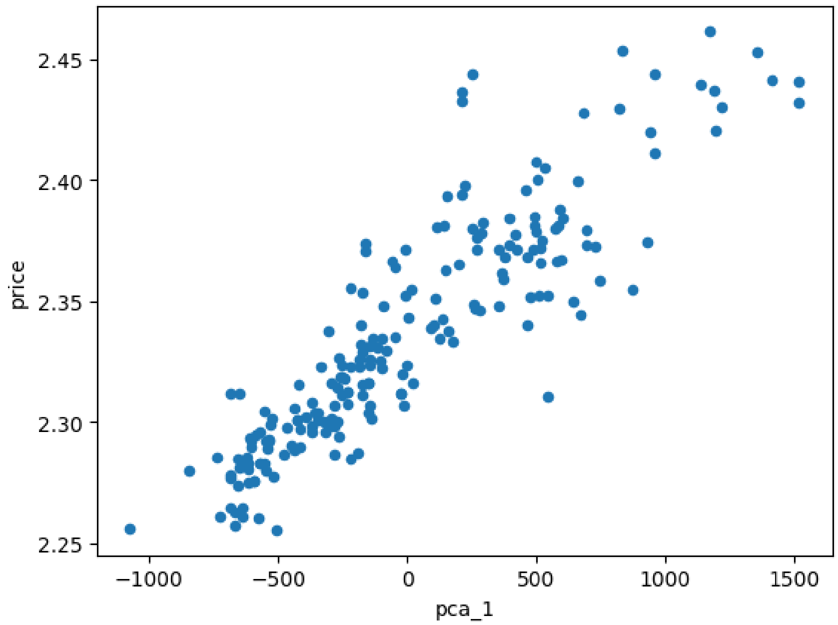
In [26]:

```
df['price'] = np.log1p(df['price'])
print(sns.distplot(df['price']))

principal_data['price'] = df['price']
principal_data.head()
principal_data.plot.scatter(x='pca_1', y='price');
```

```
AxesSubplot(0.125,0.11;0.775x0.77)
```





Observation

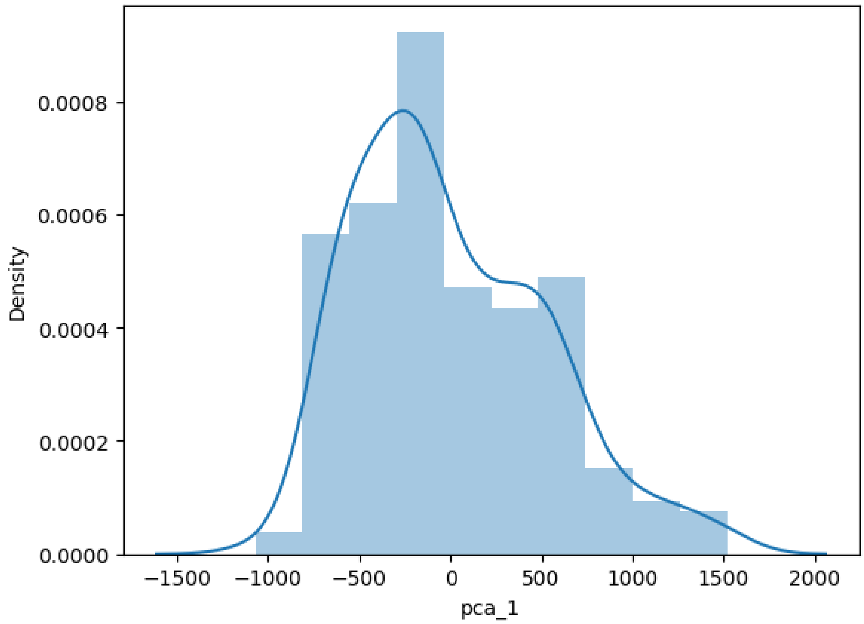
We can see few outliers in the data but still we will keep it for the further use.

In [27]:

```
sns.distplot(principal_data['pca_1'])
```

Out[27]:

```
<AxesSubplot: xlabel='pca_1', ylabel='Density'>
```



Observation

The rest of numeric variables don't have linear relationship with car price, but how knows may be they have non-linear relatioship with the price? Let's take a look at their plots.

In [28]:

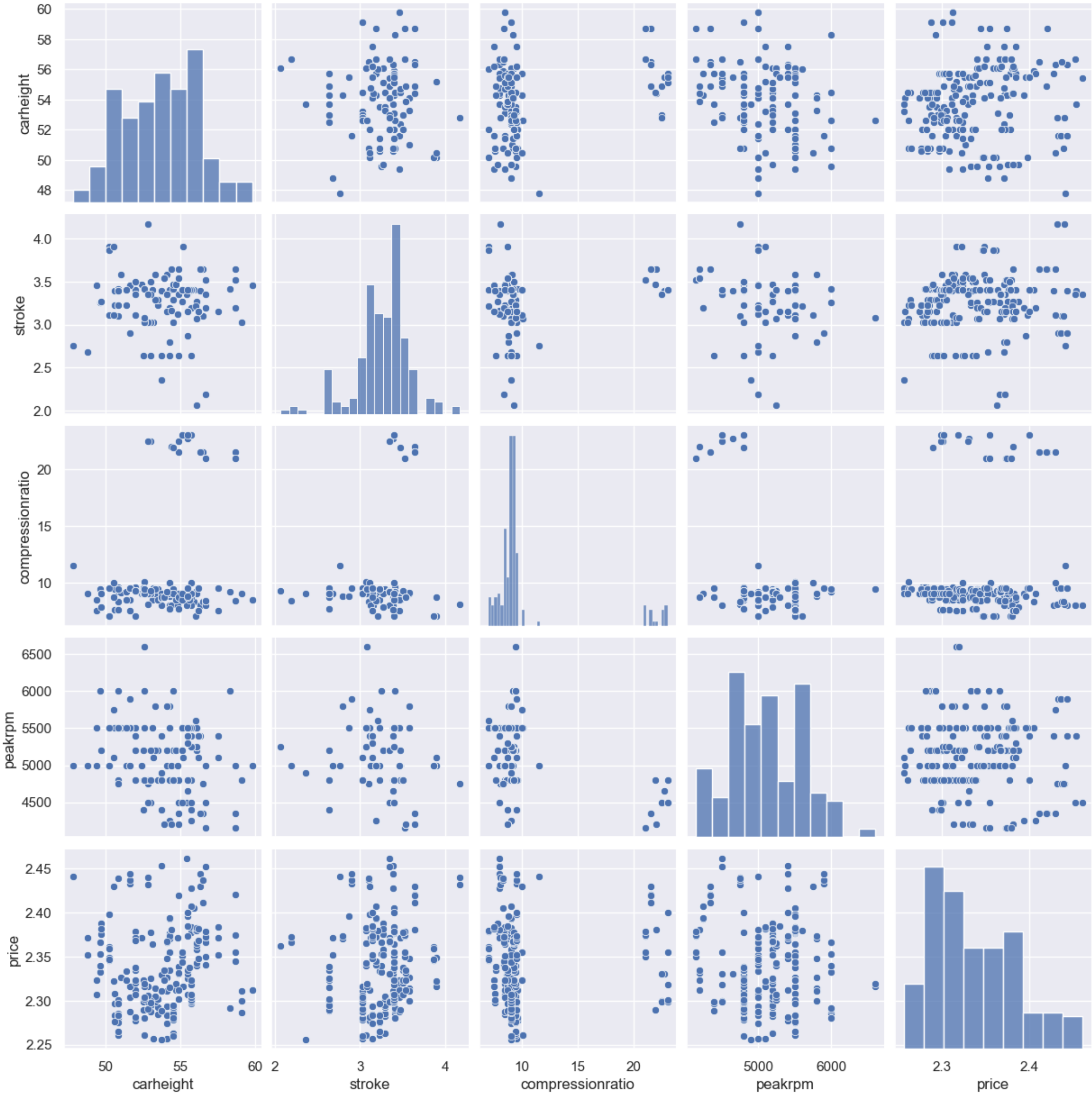
```
#remove numeric variables which we already used in PCA
data_rest = df._get_numeric_data()
data_rest.drop(pca_columns, axis=1, inplace = True)
data_rest.head()
```

Out[28]:

	carheight	stroke	compressionratio	peakrpm	price
0	48.8	2.68	9.0	5000	2.352341
1	48.8	2.68	9.0	5000	2.371288
2	52.4	3.47	9.0	5000	2.371288
3	54.3	3.40	10.0	5500	2.355491
4	54.3	3.40	8.0	5500	2.376500

In [29]:

```
#scatterplot
sns.set()
sns.pairplot(data_rest, size = 2.5)
plt.show()
```



Observation

There is no such a relationship between all the variables and the price variable.

In [30]:

```
dummy_data = df.select_dtypes(include=['object'])
dummy_data = dummy_data.drop('CarName', 1)
dummy_data.head()
```

Out[30]:

	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype	cylindernumber	fuelsystem
0	gas	std	two	convertible	rwd	front	dohc	four	mpfi
1	gas	std	two	convertible	rwd	front	dohc	four	mpfi
2	gas	std	two	hatchback	rwd	front	ohcv	six	mpfi
3	gas	std	four	sedan	fwd	front	ohc	four	mpfi
4	gas	std	four	sedan	4wd	front	ohc	five	mpfi

In [31]:

```
dummy_data = pd.get_dummies(dummy_data)
dummy_data.shape
```

Out[31]:

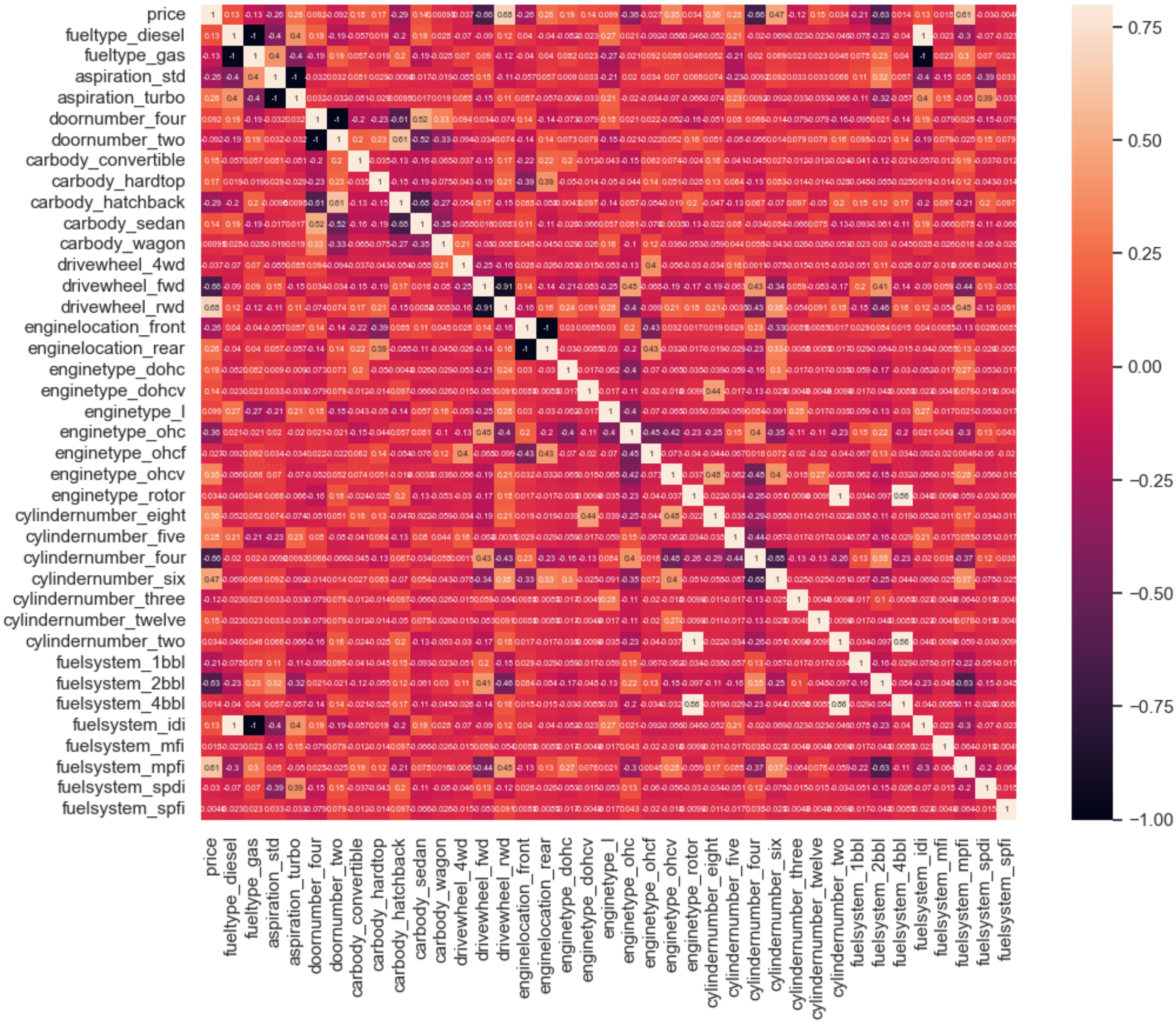
```
(205, 38)
```

In [32]:

```
dummy_data = pd.concat([df['price'], dummy_data], axis=1)
```

In [33]:

```
corrmat_dummy = dummy_data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat_dummy, vmax=.8, annot_kws={'size': 5}, annot = True, square=True);
```

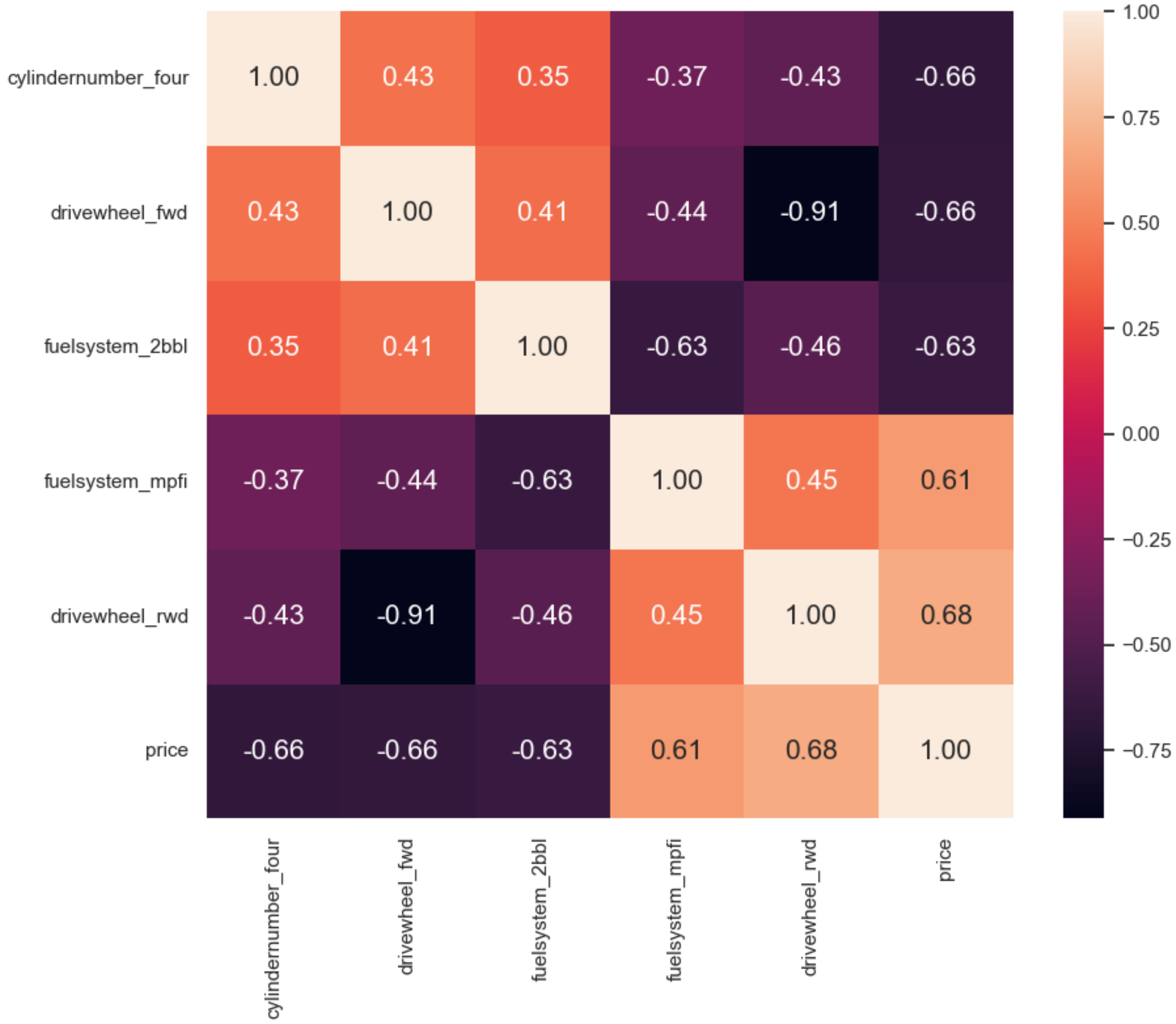


Observation

It is hard to get any idea from the above plot.

In [34]:

```
positive_corr = corrmat_dummy.sort_values('price', )[corrmat_dummy['price']>0.5]['price']
negatie_corr =  corrmat_dummy.sort_values('price', )[corrmat_dummy['price']<-0.5]['price']
correlated_dummy_cols = pd.concat([negatie_corr,positive_corr], axis=0).index
cm = np.corrcoef(dummy_data[correlated_dummy_cols].values.T)
f, ax = plt.subplots(figsize=(10, 8))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 15},
                 yticklabels=correlated_dummy_cols.values, xticklabels=correlated_dummy_cols.values)
plt.show()
```



Observation

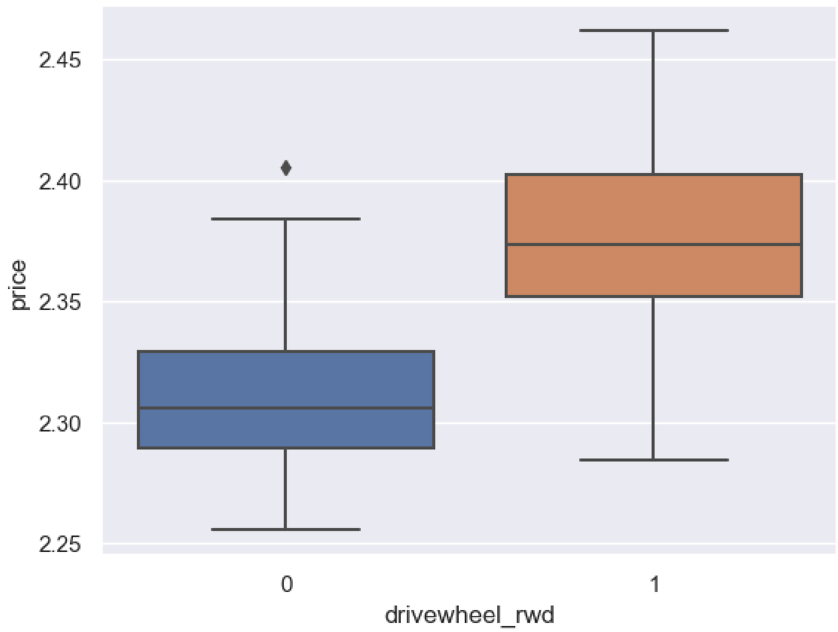
It looks like few variables and the price have strong linear relationship.

In [35]:

```
sns.boxplot(x='drivewheel_rwd', y="price", data = dummy_data)
```

Out[35]:

```
<AxesSubplot: xlabel='drivewheel_rwd', ylabel='price'>
```



In [36]:

```
correlated_dummy_cols = correlated_dummy_cols.drop('drivewheel_fwd')
```

In [37]:

```
final_data = dummy_data[correlated_dummy_cols]
final_data.head()
```

Out[37]:

	cylindernumber_four	fuelsystem_2bbl	fuelsystem_mpfi	drivewheel_rwd	price
0	1	0	1	1	2.352341
1	1	0	1	1	2.371288
2	0	0	1	1	2.371288
3	1	0	1	0	2.355491
4	0	0	1	0	2.376500

Linear Regression without PCA

In [38]:

```
pred = final_data['price']
data = final_data.drop(['price'], axis=1)
```

In [39]:

```
import sklearn.model_selection as model_selection


X_train,X_test, y_train, y_test = model_selection.train_test_split(data, pred,train_size=0.8,test_size=0.2,random_state=42)
```

In [40]:

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[40]:

```
((164, 4), (41, 4), (164,), (41,))
```

In [41]:

```
from sklearn.metrics import r2_score

def adjusted_r2(r2_score, n, p):
    len_score = (n-1)/(n-p-1)
    score = (1 - r2_score) * len_score
    return 1- score
```

In [42]:

```
from sklearn.linear_model import LinearRegression

LR1 = LinearRegression()
LR1.fit(X_train,y_train)

print(LR1.intercept_)
print(LR1.coef_)
```

```
2.3571884979278037
[-0.0426978  -0.02435599  0.01754407  0.03185343]
```

In [43]:

```
pred = LR1.predict(X_test)

print(r2_score(y_test,pred))
```

```
0.7082321088094313
```

In [44]:

```
print(adjusted_r2(r2_score(y_test,pred), len(y_test), len(X_test.columns)))
```

```
0.6758134542327014
```

Linear Regression with PCA

In [45]:

```
final_data2 = pd.concat([principal_data['pca_1'], dummy_data[correlated_dummy_cols]], axis=1)
final_data2.head()
```

Out[45]:

	pca_1	cylindernumber_four	fuelsystem_2bbl	fuelsystem_mpfi	drivewheel_rwd	price
0	-7.057267	1	0	1	1	2.352341
1	-7.057267	1	0	1	1	2.371288
2	270.841223	0	0	1	1	2.371288
3	-218.880370	1	0	1	0	2.355491
4	268.738449	0	0	1	0	2.376500

In [46]:

```
pred = final_data2['price']
data = final_data2.drop(['price'], axis=1)
```

In [47]:

```
import sklearn.model_selection as model_selection


X_train,X_test, y_train, y_test = model_selection.train_test_split(data, pred,train_size=0.8,test_size=0.2,random_state=42)
```

In [48]:

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[48]:

```
((164, 5), (41, 5), (164,), (41,))
```

In [49]:

```
from sklearn.metrics import r2_score

def adjusted_r2(r2_score, n, p):
    len_score = (n-1)/(n-p-1)
    score = (1 - r2_score) * len_score
    return 1- score
```

In [50]:

```
from sklearn.linear_model import LinearRegression

LR2 = LinearRegression()
LR2.fit(X_train,y_train)

print(LR2.intercept_)
print(LR2.coef_)
```

```
2.349555447496153
[ 5.17781714e-05 -2.42571284e-02 -1.19845333e-02  1.28167978e-02
  9.39145763e-03]
```

In [51]:

```
pred = LR2.predict(X_test)

print(r2_score(y_test,pred))
```

```
0.8916809573461777
```

In [52]:

```
print(adjusted_r2(r2_score(y_test,pred), len(y_test), len(X_test.columns)))
```

```
0.8762068083956317
```

Conclusion

.Without PCA data, Adjusted R2 score is 0.67. Whereas with PCA data, Adjusted R2 score is 0.87.
.With PCA data our model gives better result.

```
# Thank you !! 🥰
------ Shahnawaz
```