eu0s1feia

March 30, 2023

[9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib.inline
```

```
UsageError: Line magic function `%matplotlib.inline` not found.
```

[2]:
```
data = pd.read_csv('CarPrice_Assignment.csv')
```

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

[3]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	\
0	1	3	alfa-romero giulia	gas	std	two	
1	2	3	alfa-romero stelvio	gas	std	two	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	
3	4	2	audi 100 ls	gas	std	four	
4	5	2	audi 100ls	gas	std	four	

	carbody	drivewheel	enginelocation	wheelbase	…	enginesize	\
0	convertible	rwd	front	88.6	…	130	
1	convertible	rwd	front	88.6	…	130	
2	hatchback	rwd	front	94.5	…	152	
3	sedan	fwd	front	99.8	…	109	
4	sedan	4wd	front	99.4	…	136	

	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	\
0	mpfi	3.47	2.68	9.0	111	5000	21	
1	mpfi	3.47	2.68	9.0	111	5000	21	
2	mpfi	2.68	3.47	9.0	154	5000	19	
3	mpfi	3.19	3.40	10.0	102	5500	24	
4	mpfi	3.19	3.40	8.0	115	5500	18	

	highwaympg	price
0	27	13495.0
1	27	16500.0
2	26	16500.0
3	30	13950.0

```
4          22  17450.0

[5 rows x 26 columns]
```

[4]:
```
data.columns
```

[4]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

[5]:
```
data.describe()
```

[5]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight
count	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
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000

	curbweight	enginesize	boreratio	stroke	compressionratio
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	2555.565854	126.907317	3.329756	3.255415	10.142537
std	520.680204	41.642693	0.270844	0.313597	3.972040
min	1488.000000	61.000000	2.540000	2.070000	7.000000
25%	2145.000000	97.000000	3.150000	3.110000	8.600000
50%	2414.000000	120.000000	3.310000	3.290000	9.000000
75%	2935.000000	141.000000	3.580000	3.410000	9.400000
max	4066.000000	326.000000	3.940000	4.170000	23.000000

	horsepower	peakrpm	citympg	highwaympg	price
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	104.117073	5125.121951	25.219512	30.751220	13276.710571
std	39.544167	476.985643	6.542142	6.886443	7988.852332
min	48.000000	4150.000000	13.000000	16.000000	5118.000000
25%	70.000000	4800.000000	19.000000	25.000000	7788.000000
50%	95.000000	5200.000000	24.000000	30.000000	10295.000000
75%	116.000000	5500.000000	30.000000	34.000000	16503.000000
max	288.000000	6600.000000	49.000000	54.000000	45400.000000

[6]:
```
data.info()
```

```
<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
```

[7]:
```
data.isnull().sum()
```

[7]:
```
car_ID              0
symboling           0
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
```

[10]:
```
# Feature Engineering

data['mpg']=(data['citympg']+data['highwaympg'])/2
data.drop(['citympg','highwaympg'],axis=1,inplace=True)
```

[11]:
```
data.head(1)
```

[11]:
```
   car_ID  symboling             CarName fueltype aspiration doornumber  \
0       1          3  alfa-romero giulia      gas        std        two

       carbody drivewheel enginelocation  wheelbase  …  cylindernumber  \
0  convertible        rwd          front       88.6  …            four

   enginesize  fuelsystem  boreratio stroke compressionratio  horsepower  \
0         130        mpfi       3.47   2.68              9.0         111

  peakrpm    price   mpg
0    5000  13495.0  24.0

[1 rows x 25 columns]
```

[12]:
```
data['vol'] = (data['carwidth']*data['carlength']*data['carheight'])/(12.54**3)
data.drop(['carwidth','carlength','carheight'],axis=1,inplace=True)
```

[13]:
```
data.head(1)
```

[13]:
```
   car_ID  symboling             CarName fueltype aspiration doornumber  \
0       1          3  alfa-romero giulia      gas        std        two

       carbody drivewheel enginelocation  wheelbase  …  enginesize  \
```

```
0  convertible        rwd          front       88.6  …         130

  fuelsystem boreratio  stroke compressionratio  horsepower  peakrpm    price  \
0       mpfi      3.47    2.68              9.0         111     5000  13495.0

    mpg         vol
0  24.0  267.767389

[1 rows x 23 columns]
```

[14]:
```
data.select_dtypes(include='number').head()
```

[14]:
```
   car_ID  symboling  wheelbase  curbweight  enginesize  boreratio  stroke  \
0       1          3       88.6        2548         130       3.47    2.68
1       2          3       88.6        2548         130       3.47    2.68
2       3          1       94.5        2823         152       2.68    3.47
3       4          2       99.8        2337         109       3.19    3.40
4       5          2       99.4        2824         136       3.19    3.40

   compressionratio  horsepower  peakrpm    price   mpg         vol
0               9.0         111     5000  13495.0  24.0  267.767389
1               9.0         111     5000  16500.0  24.0  267.767389
2               9.0         154     5000  16500.0  22.5  297.977682
3              10.0         102     5500  13950.0  27.0  321.925893
4               8.0         115     5500  17450.0  20.0  322.898479
```

[15]:
```
data['weights']=(data['wheelbase']*data['curbweight']*data['enginesize'])/3
data.drop(['wheelbase','curbweight','enginesize'],axis=1,inplace=True)
```

[16]:
```
data.select_dtypes(include='number').head()
```

[16]:
```
   car_ID  symboling  boreratio  stroke  compressionratio  horsepower  \
0       1          3       3.47    2.68               9.0         111
1       2          3       3.47    2.68               9.0         111
2       3          1       2.68    3.47               9.0         154
3       4          2       3.19    3.40              10.0         102
4       5          2       3.19    3.40               8.0         115

   peakrpm    price   mpg         vol       weights
0     5000  13495.0  24.0  267.767389  9.782621e+06
1     5000  16500.0  24.0  267.767389  9.782621e+06
2     5000  16500.0  22.5  297.977682  1.351652e+07
3     5500  13950.0  27.0  321.925893  8.474118e+06
4     5500  17450.0  20.0  322.898479  1.272532e+07
```

[17]:
```
data['size'] = (data['vol']*data['weights'])/9.81
data.drop(['vol','weights'],axis=1,inplace=True)
```

[18]:
```
data.select_dtypes(include='number').head(1)
```

[18]:
```
   car_ID  symboling  boreratio  stroke  compressionratio  horsepower  \
0       1          3       3.47    2.68               9.0         111

   peakrpm    price   mpg          size
0     5000  13495.0  24.0  2.670201e+08
```

[19]:
```
data['force'] = (data['horsepower']/(data['boreratio']*data['stroke']))
data.drop(['horsepower','boreratio','stroke'],axis=1,inplace=True)
```

[20]:
```
data.select_dtypes(include='number').head(1)
```

[20]:
```
   car_ID  symboling  compressionratio  peakrpm    price   mpg          size  \
0       1          3               9.0     5000  13495.0  24.0  2.670201e+08

       force
0  11.935997
```

[21]:
```
data.columns
```

[21]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'enginetype',
       'cylindernumber', 'fuelsystem', 'compressionratio', 'peakrpm', 'price',
       'mpg', 'size', 'force'],
      dtype='object')
```

[22]:
```
data.drop(['symboling','compressionratio','peakrpm'],axis=1,inplace=True)
```

[23]:
```
data.head(1)
```

[23]:
```
   car_ID            CarName fueltype aspiration doornumber      carbody  \
0       1  alfa-romero giulia      gas        std        two  convertible

  drivewheel enginelocation enginetype cylindernumber fuelsystem    price  \
0        rwd          front       dohc           four       mpfi  13495.0

    mpg          size      force
0  24.0  2.670201e+08  11.935997
```

[31]:
```
data.select_dtypes(exclude='number').head(1)
```

[31]:
```
  fueltype aspiration drivewheel enginetype fuelsystem
0      gas        std        rwd       dohc       mpfi
```

[32]:
```
data.head()
```

[32]:
```
  fueltype aspiration drivewheel enginetype fuelsystem    price   mpg  \
0      gas        std        rwd       dohc       mpfi  13495.0  24.0
1      gas        std        rwd       dohc       mpfi  16500.0  24.0
2      gas        std        rwd       ohcv       mpfi  16500.0  22.5
3      gas        std        fwd        ohc       mpfi  13950.0  27.0
4      gas        std        4wd        ohc       mpfi  17450.0  20.0

           size      force
0  2.670201e+08  11.935997
1  2.670201e+08  11.935997
2  4.105629e+08  16.559852
3  2.780875e+08   9.404389
4  4.188569e+08  10.602987
```

[33]:
```
target_data = pd.DataFrame(data[['price','mpg']])
data.drop(['price','mpg'],axis=1,inplace=True)
```

[34]:
```
target_data.head(1)
```

[34]:
```
     price   mpg
0  13495.0  24.0
```

[35]:
```
feature_name = []
```

[36]:
```
from sklearn.preprocessing import OneHotEncoder
```

[37]:
```
ohe = OneHotEncoder()
```

[38]:
```
car_data = ohe.fit_transform(data[data.select_dtypes(exclude='number').columns].
 ↪values)
feature_name.extend(ohe.get_feature_names())
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function get_feature_names is deprecated; get_feature_names is
deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

[39]:
```
from sklearn.preprocessing import StandardScaler
```

[40]:
```
sc = StandardScaler()
```

[42]:
```
num_data = sc.fit_transform(data[data.select_dtypes(include='number').columns].
 ↪values)
feature_name.extend(['size','force'])
```

[43]:
```
from scipy.sparse import hstack
```

[44]:
```
data_new = hstack((car_data,num_data))
car_data = pd.DataFrame(data_new.toarray(),columns=feature_name)
```

[46]:
```
car_data.head()
```

[46]:

	x0_diesel	x0_gas	x1_std	x1_turbo	x2_4wd	x2_fwd	x2_rwd	x3_dohc	\
0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	
1	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	
2	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	
3	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	
4	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	

	x3_dohcv	x3_l	…	x4_1bbl	x4_2bbl	x4_4bbl	x4_idi	x4_mfi	x4_mpfi	\
0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	1.0	
1	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	1.0	
4	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	1.0	

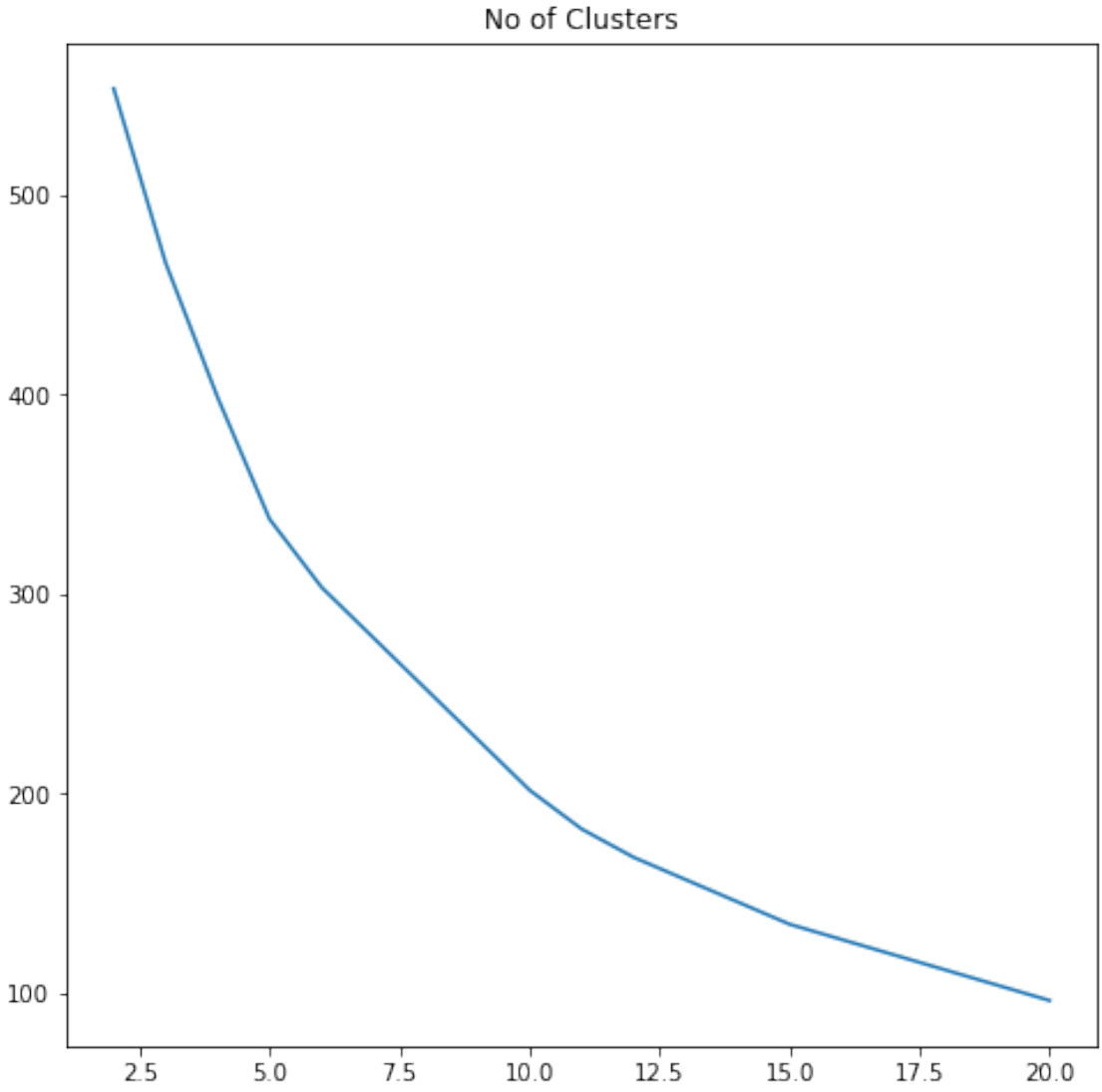
	x4_spdi	x4_spfi	size	force
0	0.0	0.0	-0.411308	0.702606
1	0.0	0.0	-0.411308	0.702606
2	0.0	0.0	0.070485	2.077343
3	0.0	0.0	-0.374161	-0.050077
4	0.0	0.0	0.098323	0.306283

[5 rows x 24 columns]

[49]:
```
# Defining the best no of clusters
from collections import defaultdict
from sklearn.cluster import KMeans
no_of_cluster = [2,3,4,5,6,10,11,12,15,20]

wcss = defaultdict(int)

for Clusters in no_of_cluster:
    cluster_algo = KMeans(n_clusters=Clusters,random_state=100)
    cluster_algo.fit(car_data.values)
    wcss[Clusters] = cluster_algo.inertia_

plt.figure(figsize=(8,8))
sns.lineplot(list(wcss.keys()),list(wcss.values()))
plt.title('No of Clusters')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
```

```
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[51]:
```
# Defining the best cluster

best_cluster = 10
cluster_algo = KMeans(n_clusters=best_cluster,random_state=0)
cluster_algo.fit(car_data.values)
label_array = cluster_algo.labels_
car_data['cluster'] = label_array
```

[66]:
```
from sklearn.manifold import TSNE
```

[54]:
```
transform = TSNE #PCA

trans = transform(n_components=2)
data2 = trans.fit(car_data.values)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random' to
'pca' in 1.2.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790:
FutureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
```

[57]:
```
from sklearn.linear_model import LinearRegression
```

[59]:
```
y_true = defaultdict(int)
y_pred = defaultdict(int)

for cluster in range(5):
    lin_reg = LinearRegression()

    index = car_data.index[car_data['cluster'] == cluster].tolist()

    lin_reg.fit(car_data.loc[index],target_data['price'][index])

    pred = lin_reg.predict(car_data.loc[index])

    y_true[cluster] = target_data['price'][index]

    y_pred = pred
```

[60]:
```
from sklearn.metrics import mean_squared_error
```

[ ]:
```
# Checking for Error

for cluster in range (best_cluster):
    true = y_true[cluster]
    pred = y_pred[cluster]
    print("Mean Squared Error"," ",cluster, " : ",mean_squared_error(true,pred))
```

[64]:
```
y_true = defaultdict(int)
y_pred = defaultdict(int)

for cluster in range(5):
```

```
    lin_reg = LinearRegression()

    index = car_data.index[car_data['cluster'] == cluster].tolist()

    lin_reg.fit(car_data.loc[index],target_data['mpg'][index])

    pred = lin_reg.predict(car_data.loc[index])

    y_true[cluster] = target_data['mpg'][index]

    y_pred = pred
```

[ ]:
```
for cluster in range (best_cluster):
    true = y_true[cluster]
    pred = y_pred[cluster]
    print("Mean Squared Error"," ",cluster, " : ",mean_squared_error(true,pred))
```