```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
from sklearn.preprocessing import Imputer
from sklearn.metrics import accuracy_score
import seaborn as sns
import os
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib as mp
import seaborn as sns
%matplotlib inline
sns.set(style="ticks")

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
from sklearn.preprocessing import Imputer
from sklearn.metrics import accuracy_score
import seaborn as sns
import os
%matplotlib inline
from sklearn import metrics

from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
data = pd.read_csv("/content/drive/My Drive/6oct/cars-dataset.csv")
```

```
print(data.shape)

data.head()
```

(398, 8)

	car name	cyl	disp	hp	wt	acc	yr	mpg
0	chevrolet chevelle malibu	8	307.0	130	3504	12.0	70	18.0
1	buick skylark 320	8	350.0	165	3693	11.5	70	15.0
2	plymouth satellite	8	318.0	150	3436	11.0	70	18.0
3	amc rebel sst	8	304.0	150	3433	12.0	70	16.0
4	ford torino	8	302.0	140	3449	10.5	70	17.0

```
print(data.head())
print(data.index)
print(data.columns)
```

```
                    car name  cyl   disp   hp    wt   acc  yr   mpg
0  chevrolet chevelle malibu    8  307.0  130  3504  12.0  70  18.0
1          buick skylark 320    8  350.0  165  3693  11.5  70  15.0
2         plymouth satellite    8  318.0  150  3436  11.0  70  18.0
3              amc rebel sst    8  304.0  150  3433  12.0  70  16.0
4                ford torino    8  302.0  140  3449  10.5  70  17.0
RangeIndex(start=0, stop=398, step=1)
Index(['car name', 'cyl', 'disp', 'hp', 'wt', 'acc', 'yr', 'mpg'], dtype='object')
```

```
data.isnull().any()
```

```
car name    False
cyl         False
disp        False
hp          False
wt          False
acc         False
yr          False
mpg         False
dtype: bool
```

```
data.dtypes
```

```
car name     object
cyl           int64
disp        float64
hp           object
wt            int64
acc         float64
yr            int64
mpg         float64
dtype: object
```

```
data.describe().transpose()
```

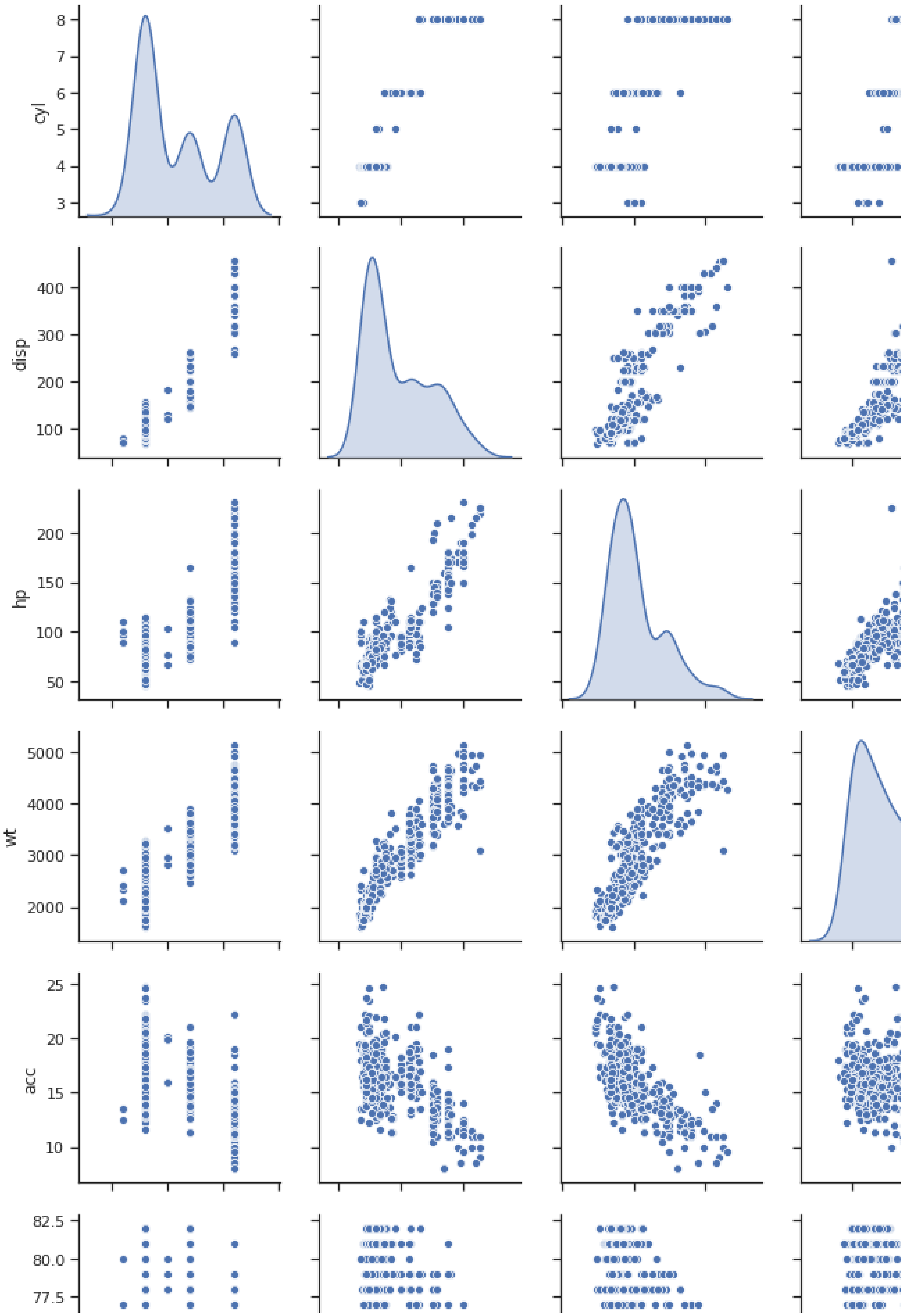
	count	mean	std	min	25%	50%	75%	max
cyl	398.0	5.454774	1.701004	3.0	4.000	4.0	8.000	8.0
disp	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0
wt	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0
acc	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8
yr	398.0	76.010050	3.697627	70.0	73.000	76.0	79.000	82.0
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6

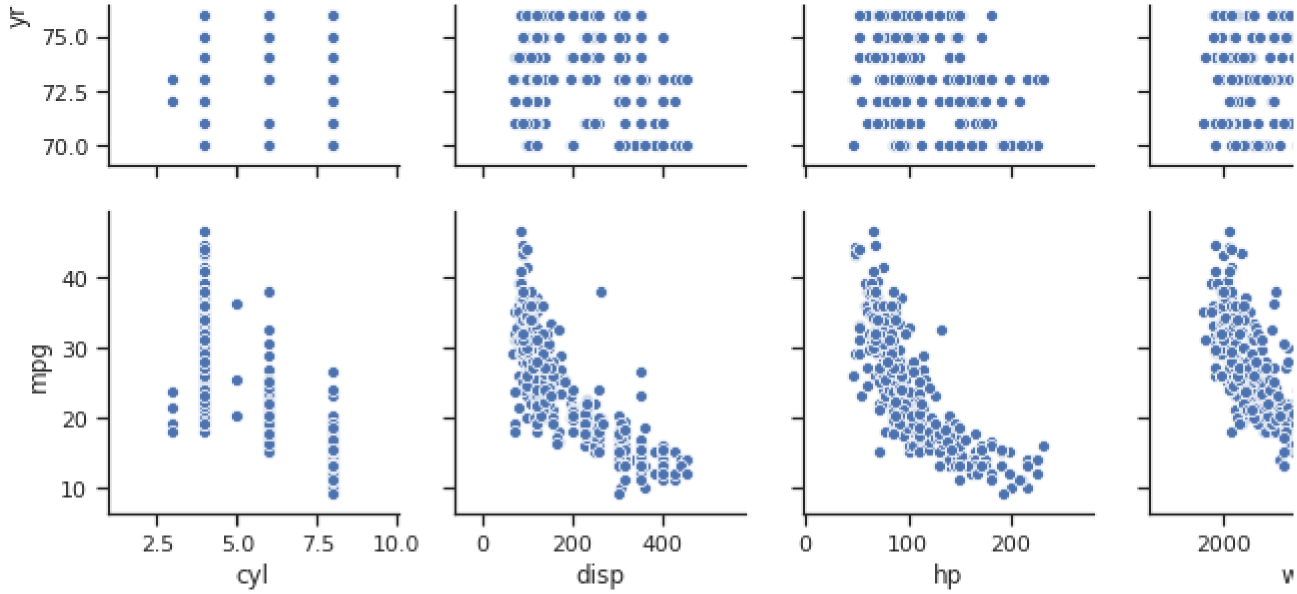
```
data = data.replace('?', np.nan)
```

```
data.hp = data.hp.astype('float64')
```

```
sns.pairplot(data, diag_kind ='kde')
```

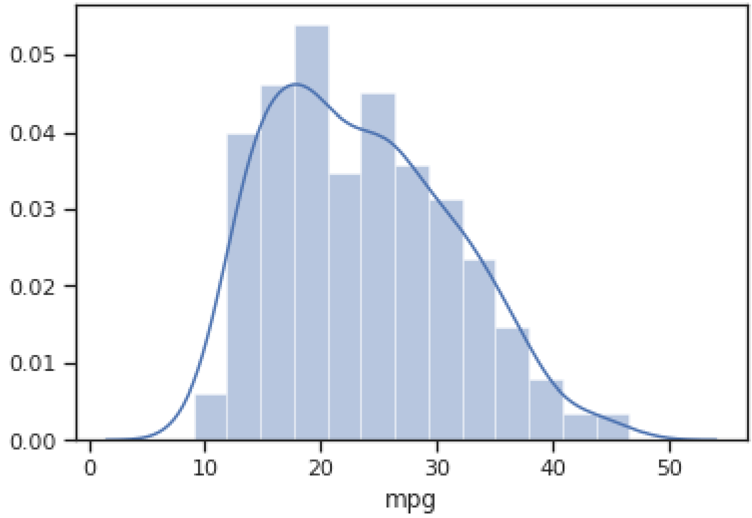
```
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kde.py:447: RuntimeW
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kde.py:447: RuntimeW
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
<seaborn.axisgrid.PairGrid at 0x7fdf5e9ad400>
```






```
sns.distplot(data['mpg'])
```

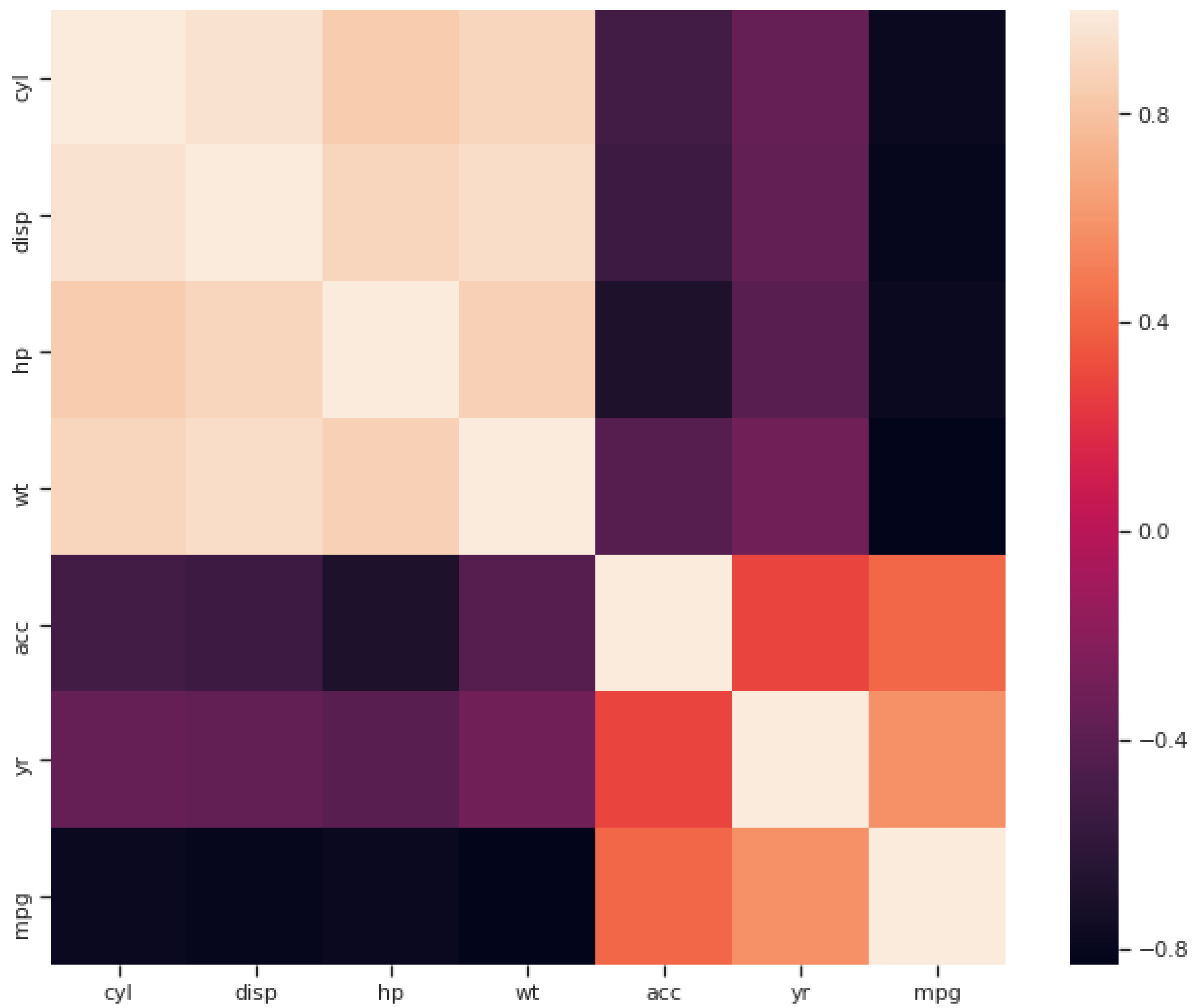
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf5b246a20>




```
print("Skewness: %f" % data['mpg'].skew())
print("Kurtosis: %f" % data['mpg'].kurt())
```

```
Skewness: 0.457066
Kurtosis: -0.510781
```

```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, square=True);
```



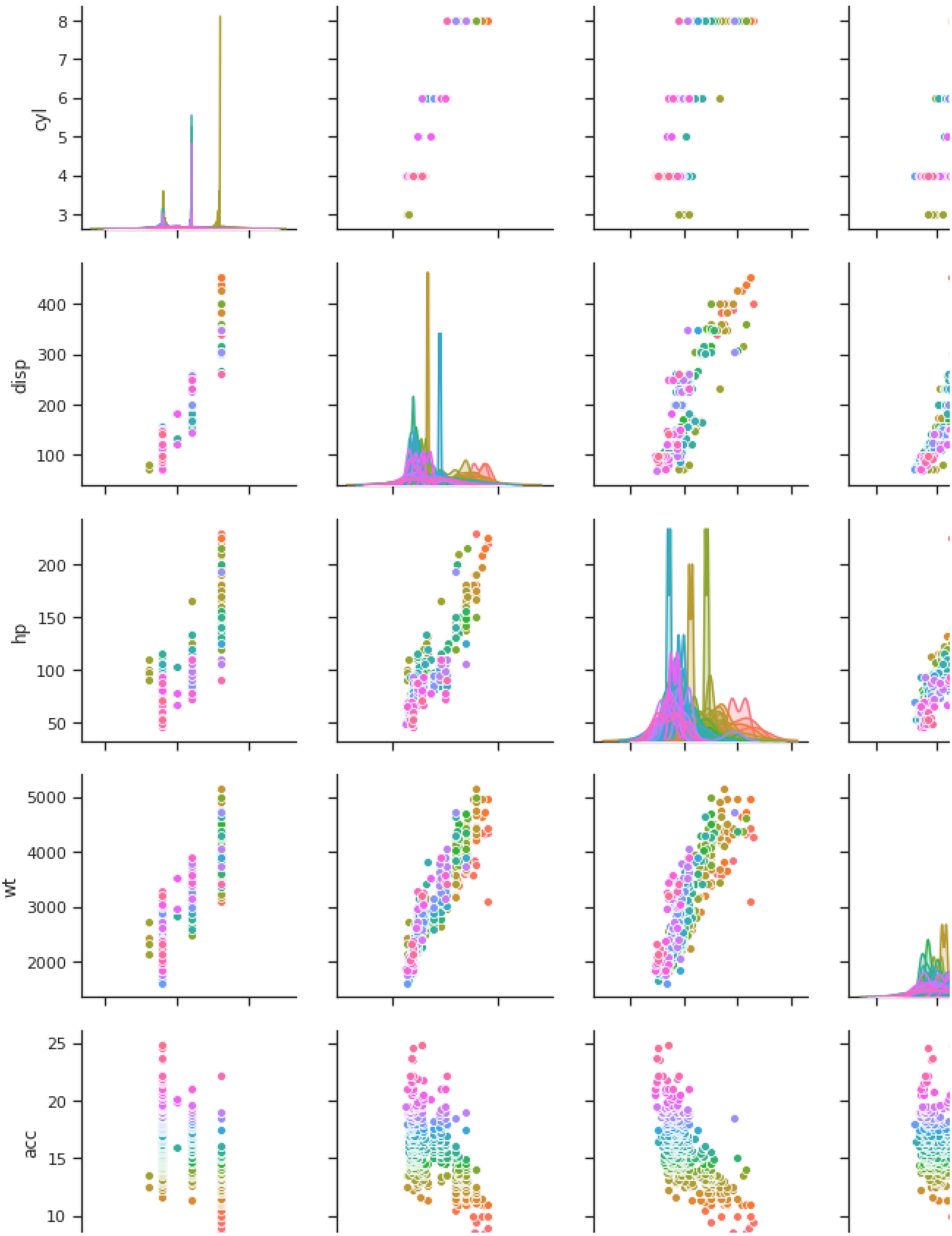

```
#Accelation of a vehicle models is an independent of other.
#As number of Cylinder/Horsepower increase, we can positive impact/increase in Horsepower/Cylinde
#Mileage/Weight is inversly proprtional to Cylinder/Horsepower.
```

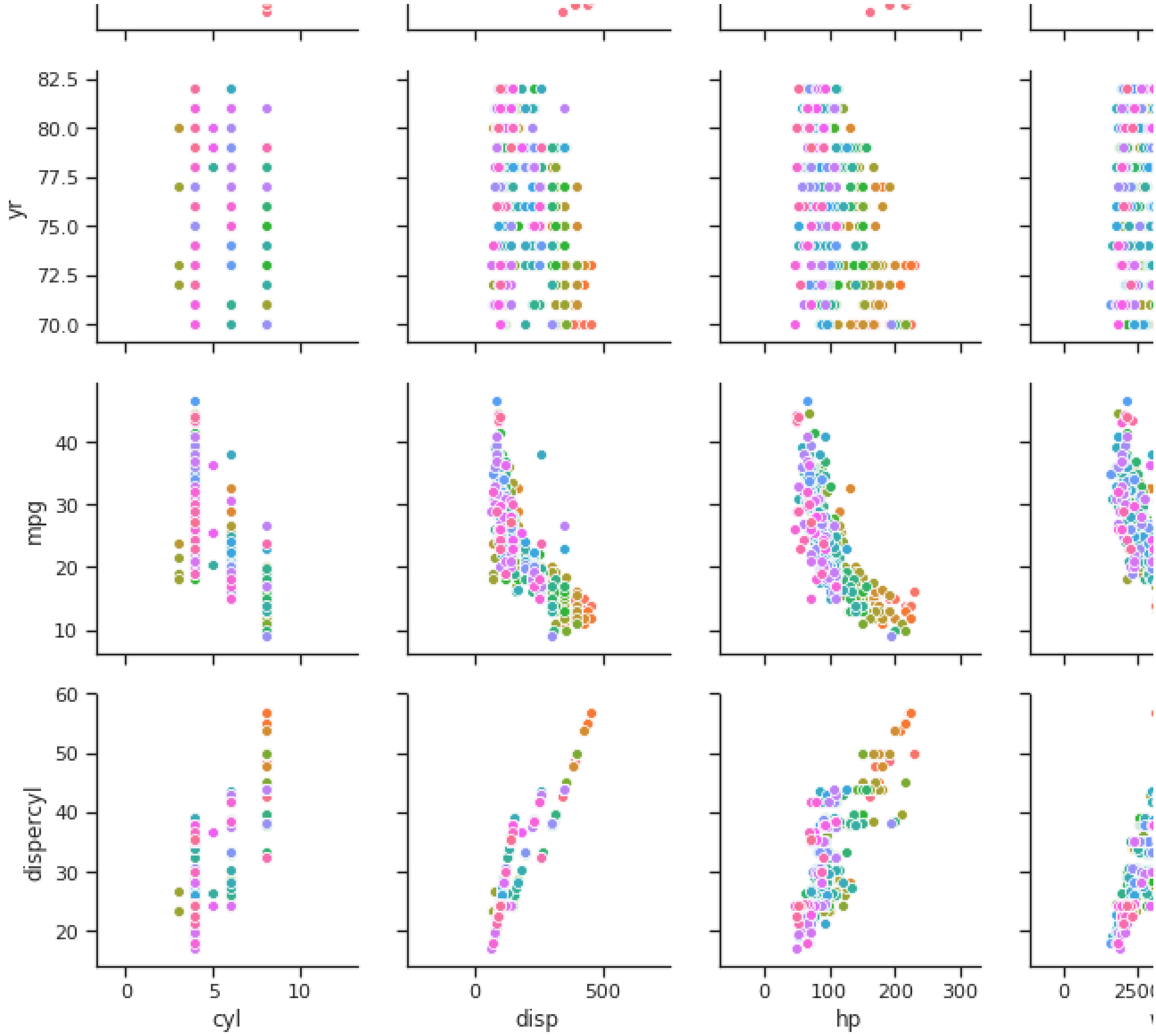
```
numeric_cols = data.drop('car name', axis=1)

car_names = pd.DataFrame(data[['car name']])


numeric_cols = numeric_cols.apply(lambda x: x.fillna(x.median()),axis=0)
data = numeric_cols.join(car_names)

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
cyl         398 non-null int64
disp        398 non-null float64
hp          398 non-null float64
wt          398 non-null int64
acc         398 non-null float64
yr          398 non-null int64
mpg         398 non-null float64
car name    398 non-null object
dtypes: float64(4), int64(3), object(1)
memory usage: 25.0+ KB
```

```
cars_df_attr = data.iloc[:, 0:7]
cars_df_attr['dispercyl'] = cars_df_attr['disp'] / cars_df_attr['cyl']
sns.pairplot(cars_df_attr, diag_kind='kde', hue = 'acc')
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:140: RuntimeWarning: De
  keepdims=keepdims)
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:132: RuntimeWarning: in
  ret = ret.dtype.type(ret / rcount)
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kde.py:487: RuntimeW
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kdetools.py:34: Runt
  FAC1 = 2*(np.pi*bw/RANGE)**2
<seaborn.axisgrid.PairGrid at 0x7fdf599289b0>
```






```
from scipy.stats import zscore

cars_df_attr = data.loc[:, 'cyl':'mpg']
cars_df_attr
```

	cyl	disp	hp	wt	acc	yr	mpg
0	8	307.0	130.0	3504	12.0	70	18.0
1	8	350.0	165.0	3693	11.5	70	15.0
2	8	318.0	150.0	3436	11.0	70	18.0
3	8	304.0	150.0	3433	12.0	70	16.0
4	8	302.0	140.0	3449	10.5	70	17.0
5	8	429.0	198.0	4341	10.0	70	15.0
6	8	454.0	220.0	4354	9.0	70	14.0
7	8	440.0	215.0	4312	8.5	70	14.0
8	8	455.0	225.0	4425	10.0	70	14.0
9	8	390.0	190.0	3850	8.5	70	15.0
10	8	383.0	170.0	3563	10.0	70	15.0
11	8	340.0	160.0	3609	8.0	70	14.0
12	8	400.0	150.0	3761	9.5	70	15.0
13	8	455.0	225.0	3086	10.0	70	14.0
14	4	113.0	95.0	2372	15.0	70	24.0
15	6	198.0	95.0	2833	15.5	70	22.0
16	6	199.0	97.0	2774	15.5	70	18.0
17	6	200.0	85.0	2587	16.0	70	21.0
18	4	97.0	88.0	2130	14.5	70	27.0
19	4	97.0	46.0	1835	20.5	70	26.0
20	4	110.0	87.0	2672	17.5	70	25.0
21	4	107.0	90.0	2430	14.5	70	24.0
22	4	104.0	95.0	2375	17.5	70	25.0
23	4	121.0	113.0	2234	12.5	70	26.0
24	6	199.0	90.0	2648	15.0	70	21.0
25	8	360.0	215.0	4615	14.0	70	10.0
26	8	307.0	200.0	4376	15.0	70	10.0
27	8	318.0	210.0	4382	13.5	70	11.0
28	8	304.0	193.0	4732	18.5	70	9.0
29	4	97.0	88.0	2130	14.5	71	27.0
...	...	...	...	...	...	...	...
368	4	112.0	88.0	2640	18.6	82	27.0
369	4	112.0	88.0	2395	18.0	82	34.0

369	4	112.0	88.0	2395	18.0	82	34.0
370	4	112.0	85.0	2575	16.2	82	31.0
371	4	135.0	84.0	2525	16.0	82	29.0
372	4	151.0	90.0	2735	18.0	82	27.0
373	4	140.0	92.0	2865	16.4	82	24.0
374	4	151.0	93.5	3035	20.5	82	23.0
375	4	105.0	74.0	1980	15.3	82	36.0
376	4	91.0	68.0	2025	18.2	82	37.0
377	4	91.0	68.0	1970	17.6	82	31.0
378	4	105.0	63.0	2125	14.7	82	38.0
379	4	98.0	70.0	2125	17.3	82	36.0
380	4	120.0	88.0	2160	14.5	82	36.0
381	4	107.0	75.0	2205	14.5	82	36.0
382	4	108.0	70.0	2245	16.9	82	34.0
383	4	91.0	67.0	1965	15.0	82	38.0
384	4	91.0	67.0	1965	15.7	82	32.0
385	4	91.0	67.0	1995	16.2	82	38.0
386	6	181.0	110.0	2945	16.4	82	25.0
387	6	262.0	85.0	3015	17.0	82	38.0
388	4	156.0	92.0	2585	14.5	82	26.0
389	6	232.0	112.0	2835	14.7	82	22.0
390	4	144.0	96.0	2665	13.9	82	32.0
391	4	135.0	84.0	2370	13.0	82	36.0
392	4	151.0	90.0	2950	17.3	82	27.0
393	4	140.0	86.0	2790	15.6	82	27.0
394	4	97.0	52.0	2130	24.6	82	44.0
395	4	135.0	84.0	2295	11.6	82	32.0
396	4	120.0	79.0	2625	18.6	82	28.0

```
cars_df_attr_z = cars_df_attr.apply(zscore)
     # Removing  year column
cars_df_attr_z.pop('yr')
array = cars_df_attr_z.values
```

```
#KMeans Clustering
```

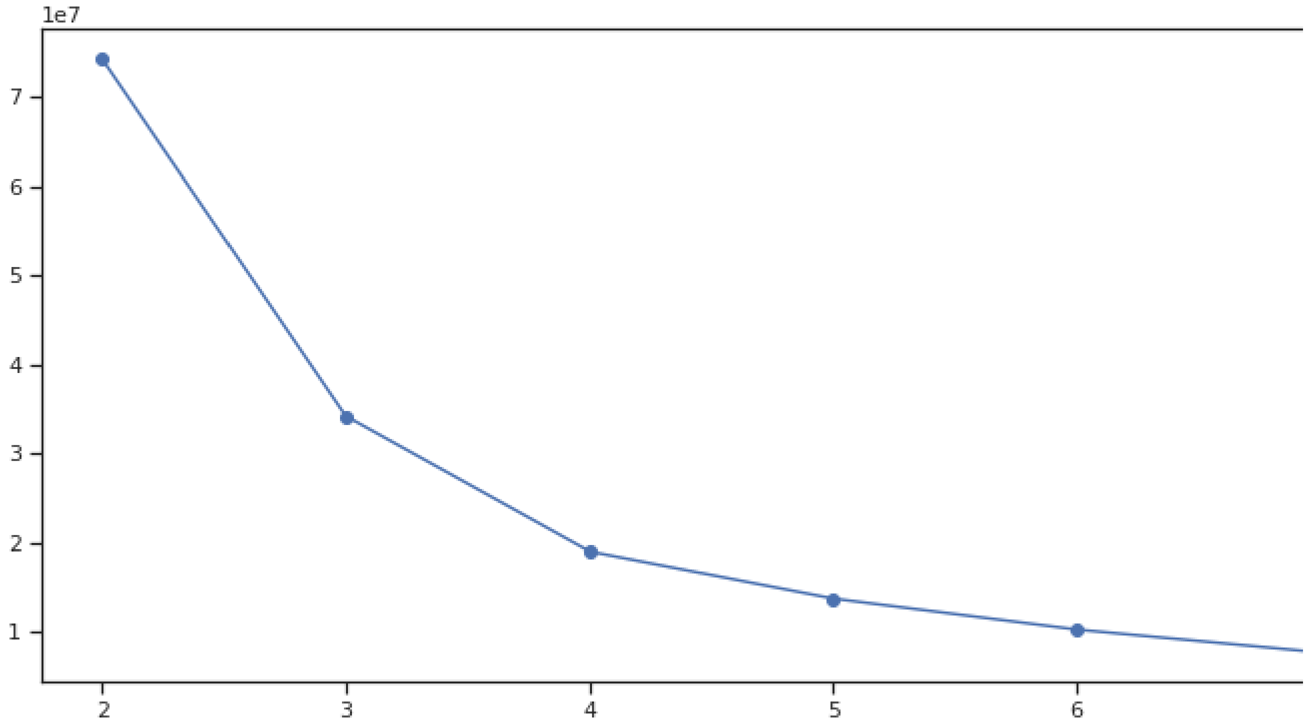
```
cluster_range = range( 2, 8)   # expect 4 to 5 clusters from the plot showing 2 to 8
cluster_errors = []
```

```
cluster_sil_scores = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 5)
  clusters.fit(cars_df_attr)
  labels = clusters.labels_
  centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
  cluster_sil_scores.append(metrics.silhouette_score(cars_df_attr_z, labels, metric='euclidean'))
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors": cluster_errors,"Avg
clusters_df[0:15]
```

	num_clusters	cluster_errors	Avg Sil Score
0	2	7.429910e+07	0.469829
1	3	3.420799e+07	0.335123
2	4	1.905160e+07	0.199671
3	5	1.376961e+07	0.153402
4	6	1.029191e+07	0.098656
5	7	7.718966e+06	0.054446

```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

```
[<matplotlib.lines.Line2D at 0x7fdf52d494e0>]
```




```
#the elbow plot shows there are likely 3 to 4 culusters



#taking 3 clusters


cluster = KMeans( n_clusters = 3, random_state = 2354 )
cluster.fit(cars_df_attr_z)
```

```
cars_df_attr_z_copy = cars_df_attr_z.copy(deep = True)
```

```
centroids = cluster.cluster_centers_
centroids
```

```
array([[ 1.4860546 ,  1.48450715,  1.50624078,  1.38753374, -1.06267868,
        -1.15110476],
       [-0.85347696, -0.80321374, -0.67506194, -0.78549879,  0.36133415,
         0.75394661],
       [ 0.34598334,  0.23689416, -0.06773972,  0.29795187,  0.30089004,
        -0.47244453]])
```

```
centroid_df = pd.DataFrame(centroids, columns = list(cars_df_attr_z) )
centroid_df
```

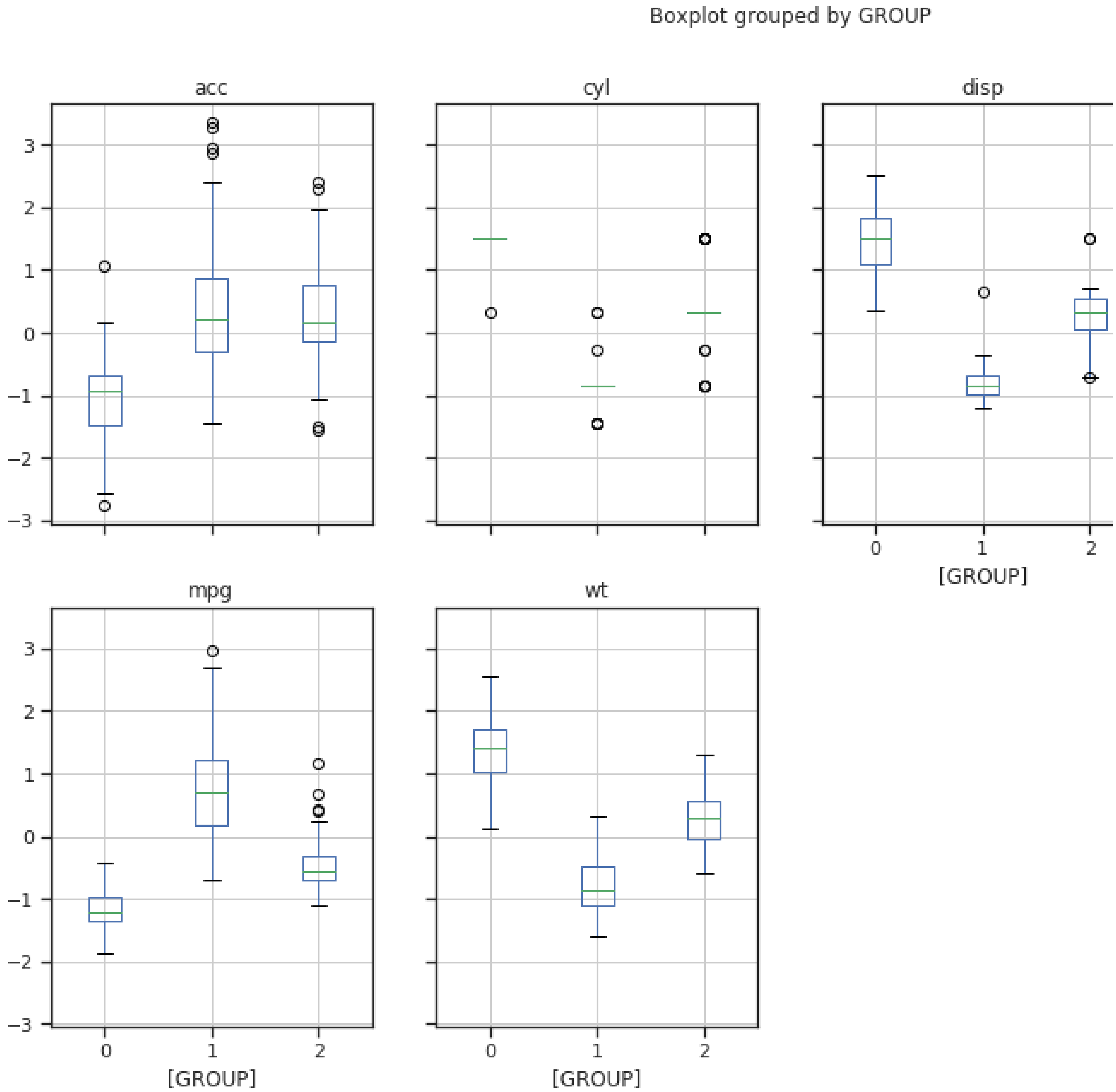
	cyl	disp	hp	wt	acc	mpg
0	1.486055	1.484507	1.506241	1.387534	-1.062679	-1.151105
1	-0.853477	-0.803214	-0.675062	-0.785499	0.361334	0.753947
2	0.345983	0.236894	-0.067740	0.297952	0.300890	-0.472445

```
# create column "GROUP" to hold  the cluster id of each record
```

```
prediction=cluster.predict(cars_df_attr_z)
cars_df_attr_z["GROUP"] = prediction
```

```
import matplotlib.pylab as plt

cars_df_attr_z.boxplot(by = 'GROUP',  layout=(2,4), figsize=(15, 10))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fdf5250cf98>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf5349ce48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf51b5e940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf529a4d68>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fdf51ac4c88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf517e8400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf523ac898>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf52bb5e48>]],
      dtype=object)
```

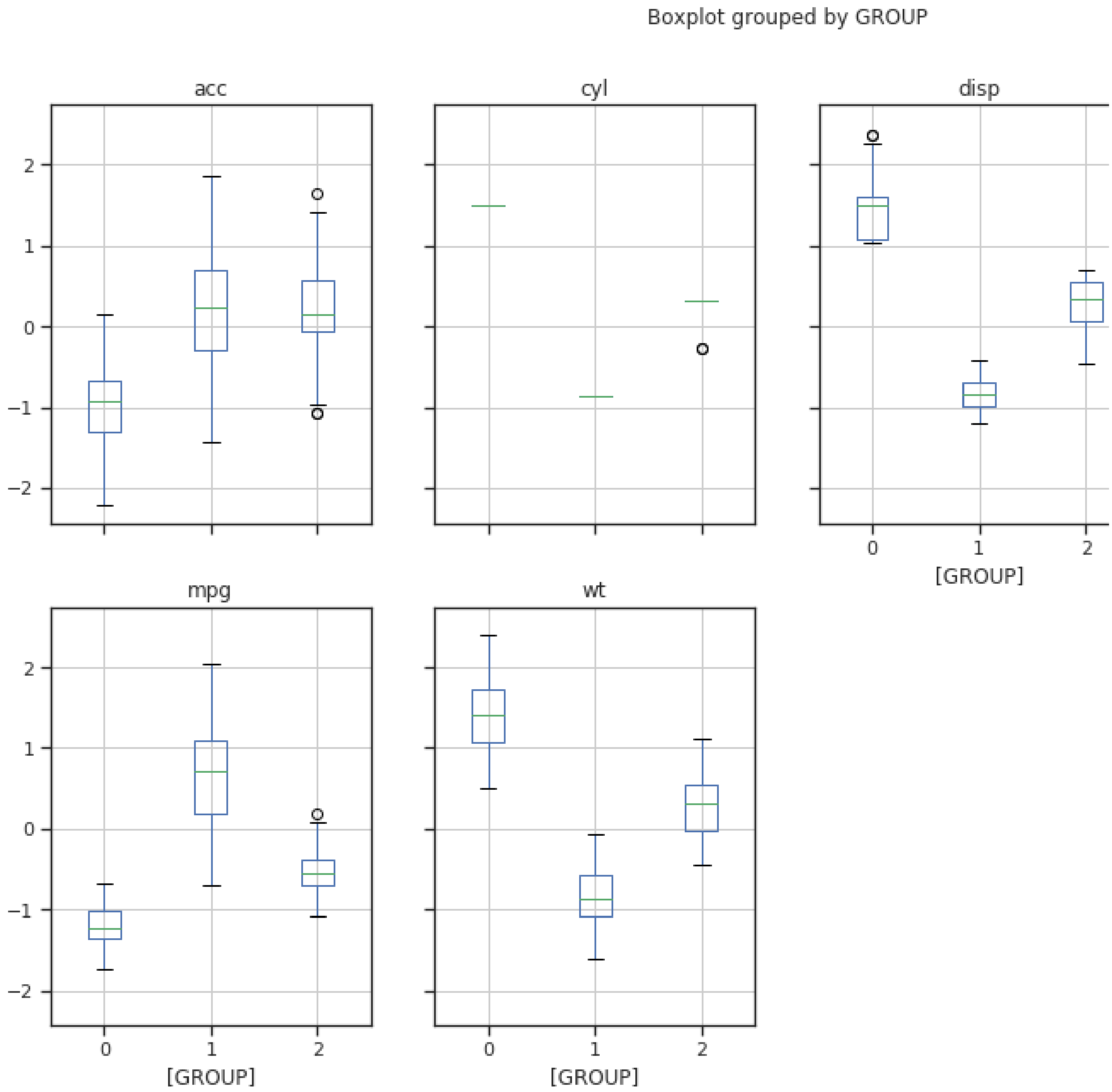



```
data1 = cars_df_attr_z

def replace(group):
    median, std = group.median(), group.std()
    outliers = (group - median).abs() > 2*std
    group[outliers] = group.median()
    return group

data_corrected = (data1.groupby('GROUP').transform(replace))
concat_data = data_corrected.join(pd.DataFrame(cars_df_attr_z['GROUP']))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/ind
```

```
concat_data.boxplot(by = 'GROUP', layout=(2,4), figsize=(15, 10))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fdf527f1a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf52aa3f98>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf54348d30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf560de748>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fdf52ffdb38>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf54bfe860>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf566c5eb8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fdf57857dd8>]],
      dtype=object)
```




```
#The new outliers would be much closer to the centre
```

```
data
```

	cyl	disp	hp	wt	acc	yr	mpg	car name
0	8	307.0	130.0	3504	12.0	70	18.0	chevrolet chevelle malibu
1	8	350.0	165.0	3693	11.5	70	15.0	buick skylark 320
2	8	318.0	150.0	3436	11.0	70	18.0	plymouth satellite
3	8	304.0	150.0	3433	12.0	70	16.0	amc rebel sst
4	8	302.0	140.0	3449	10.5	70	17.0	ford torino
5	8	429.0	198.0	4341	10.0	70	15.0	ford galaxie 500
6	8	454.0	220.0	4354	9.0	70	14.0	chevrolet impala
7	8	440.0	215.0	4312	8.5	70	14.0	plymouth fury iii
8	8	455.0	225.0	4425	10.0	70	14.0	pontiac catalina
9	8	390.0	190.0	3850	8.5	70	15.0	amc ambassador dpl
10	8	383.0	170.0	3563	10.0	70	15.0	dodge challenger se
11	8	340.0	160.0	3609	8.0	70	14.0	plymouth 'cuda 340
12	8	400.0	150.0	3761	9.5	70	15.0	chevrolet monte carlo
13	8	455.0	225.0	3086	10.0	70	14.0	buick estate wagon (sw)
14	4	113.0	95.0	2372	15.0	70	24.0	toyota corona mark ii
15	6	198.0	95.0	2833	15.5	70	22.0	plymouth duster
16	6	199.0	97.0	2774	15.5	70	18.0	amc hornet
17	6	200.0	85.0	2587	16.0	70	21.0	ford maverick
18	4	97.0	88.0	2130	14.5	70	27.0	datsun pl510
19	4	97.0	46.0	1835	20.5	70	26.0	volkswagen 1131 deluxe sedan
20	4	110.0	87.0	2672	17.5	70	25.0	peugeot 504
21	4	107.0	90.0	2430	14.5	70	24.0	audi 100 ls
22	4	104.0	95.0	2375	17.5	70	25.0	saab 99e
23	4	121.0	113.0	2234	12.5	70	26.0	bmw 2002
24	6	199.0	90.0	2648	15.0	70	21.0	amc gremlin
25	8	360.0	215.0	4615	14.0	70	10.0	ford f250
26	8	307.0	200.0	4376	15.0	70	10.0	chevy c20
27	8	318.0	210.0	4382	13.5	70	11.0	dodge d200
28	8	304.0	193.0	4732	18.5	70	9.0	hi 1200d
29	4	97.0	88.0	2130	14.5	71	27.0	datsun pl510
...	...	...	...	...	...	...	...	...
368	4	112.0	88.0	2640	18.6	82	27.0	chevrolet cavalier wagon
369	4	112.0	88.0	2395	18.0	82	34.0	chevrolet cavalier 2-door

369	4	112.0	88.0	2395	18.0	82	34.0	chevrolet cavalier 2-door
370	4	112.0	85.0	2575	16.2	82	31.0	pontiac j2000 se hatchback
371	4	135.0	84.0	2525	16.0	82	29.0	dodge aries se
372	4	151.0	90.0	2735	18.0	82	27.0	pontiac phoenix
373	4	140.0	92.0	2865	16.4	82	24.0	ford fairmont futura
374	4	151.0	93.5	3035	20.5	82	23.0	amc concord dl
375	4	105.0	74.0	1980	15.3	82	36.0	volkswagen rabbit l
376	4	91.0	68.0	2025	18.2	82	37.0	mazda glc custom l
377	4	91.0	68.0	1970	17.6	82	31.0	mazda glc custom
378	4	105.0	63.0	2125	14.7	82	38.0	plymouth horizon miser
379	4	98.0	70.0	2125	17.3	82	36.0	mercury lynx l
380	4	120.0	88.0	2160	14.5	82	36.0	nissan stanza xe
381	4	107.0	75.0	2205	14.5	82	36.0	honda accord
382	4	108.0	70.0	2245	16.9	82	34.0	toyota corolla
383	4	91.0	67.0	1965	15.0	82	38.0	honda civic
384	4	91.0	67.0	1965	15.7	82	32.0	honda civic (auto)
385	4	91.0	67.0	1995	16.2	82	38.0	datsun 310 gx
386	6	181.0	110.0	2945	16.4	82	25.0	buick century limited
387	6	262.0	85.0	3015	17.0	82	38.0	oldsmobile cutlass ciera (diesel)
388	4	156.0	92.0	2585	14.5	82	26.0	chrysler lebaron medallion
389	6	232.0	112.0	2835	14.7	82	22.0	ford granada l
390	4	144.0	96.0	2665	13.9	82	32.0	toyota celica gt
391	4	135.0	84.0	2370	13.0	82	36.0	dodge charger 2.2
392	4	151.0	90.0	2950	17.3	82	27.0	chevrolet camaro
393	4	140.0	86.0	2790	15.6	82	27.0	ford mustang gl
394	4	97.0	52.0	2130	24.6	82	44.0	vw pickup
395	4	135.0	84.0	2295	11.6	82	32.0	dodge rampage
396	4	120.0	79.0	2625	18.6	82	28.0	ford ranger

```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
```

```
factors = ['cyl','disp','hp','acc','wt','yr']
X = pd.DataFrame(data[factors].copy())
y = data['mpg'].copy()
```

```
X = StandardScaler().fit_transform(X)
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.33,random_state=324)
X_train.shape[0] == y_train.shape[0]
```

```
True
```

```
regressor = LinearRegression()
```

```
regressor.get_params()
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': False}
```

```
regressor.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_predicted = regressor.predict(X_test)
```

```
rmse = sqrt(mean_squared_error(y_true=y_test,y_pred=y_predicted))
rmse
```

```
3.433500527518434
```

```
gb_regressor = GradientBoostingRegressor(n_estimators=4000)
gb_regressor.fit(X_train,y_train)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=4000,
                          n_iter_no_change=None, presort='auto',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

```
gb_regressor.get_params()
```

```
{'alpha': 0.9,
 'criterion': 'friedman_mse',
 'init': None,
 'learning_rate': 0.1,
 'loss': 'ls',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 4000,
 'n_iter_no_change': None,
 'presort': 'auto',
 'random_state': None,
 'subsample': 1.0,
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': 0,
 'warm_start': False}
```

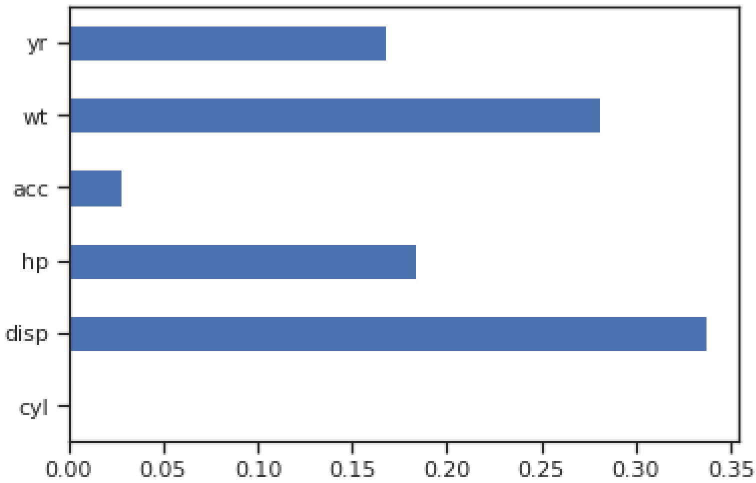
```
y_predicted_gbr = gb_regressor.predict(X_test)
```

```
rmse_bgr = sqrt(mean_squared_error(y_true=y_test,y_pred=y_predicted_gbr))
rmse_bgr
```

```
2.7052785799211354
```

```
fi= pd.Series(gb_regressor.feature_importances_,index=factors)
fi.plot.barh()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf52f02390>
```




```
labels = cluster.predict(cars_df_attr_z)
labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 1, 1, 1, 1,
       1, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0,
       0, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 2, 1, 1,
       1, 1, 1, 2, 1, 0, 0, 1, 1, 1, 2, 0, 1, 2, 0, 2, 2, 2, 2, 1, 1, 1,
       1, 2, 2, 2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1,
       2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 2, 2, 2, 1, 1, 1,
       1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 0, 2, 2, 2, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 2, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 2, 2, 0,
       0, 0, 0, 0, 0, 2, 0, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 2, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int32)
```

```
cars_df_attr_z['label_cluster'] = labels
```

```
car_df_z_0 = cars_df_attr_z[cars_df_attr_z['label_cluster'] == 0]
```

```
car_df_z_1 = cars_df_attr_z[cars_df_attr_z['label_cluster'] == 1]
car_df_z_2 = cars_df_attr_z[cars_df_attr_z['label_cluster'] == 2]
```

```
from sklearn.model_selection import train_test_split

X1 = car_df_z_0.drop(['mpg', 'label_cluster'], axis = 1)
Y1 = car_df_z_0['mpg']
X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X1, Y1, test_size = 0.3, random_state = 1

X2 = car_df_z_1.drop(['mpg', 'label_cluster'], axis = 1)
Y2 = car_df_z_1['mpg']
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X2, Y2, test_size = 0.3, random_state = 1

X3 = car_df_z_2.drop(['mpg', 'label_cluster'], axis = 1)
Y3 = car_df_z_2['mpg']
X_train3, X_test3, Y_train3, Y_test3 = train_test_split(X3, Y3, test_size = 0.3, random_state = 1
```

```
from sklearn.linear_model import LinearRegression
car_df_z_LRModel = LinearRegression()
```

```
car_df_z_LRModel.fit(X_train1, Y_train1)
Y_pred_1 = car_df_z_LRModel.predict(X_test1)

for index_of_col, col_name in enumerate(X_train1.columns):
    print("The coefficient for", col_name, "is", car_df_z_LRModel.coef_[index_of_col])
```

```
The coefficient for cyl is -0.36927619313129295
The coefficient for disp is 0.11869853545278303
The coefficient for hp is -0.2298154239559793
The coefficient for wt is -0.17735365412194048
The coefficient for acc is -0.07574909146667422
```

```
car_df_z_LRModel.fit(X_train2, Y_train2)
```

```
Y_pred_2 = car_df_z_LRModel.predict(X_test2)

for index_of_col, col_name in enumerate(X_train2.columns):
    print("The coefficient for", col_name, "is", car_df_z_LRModel.coef_[index_of_col])
```

```
The coefficient for cyl is 1.045233660667113
The coefficient for disp is -0.012949533636438817
The coefficient for hp is -1.147916495160105
The coefficient for wt is -0.27071394462230675
The coefficient for acc is -0.22572069086589092
```

```
car_df_z_LRModel.fit(X_train3, Y_train3)
Y_pred_3 = car_df_z_LRModel.predict(X_test3)

for index_of_col, col_name in enumerate(X_train3.columns):
    print("The coefficient for", col_name, "is", car_df_z_LRModel.coef_[index_of_col])
```

```
The coefficient for cyl is 0.3835753541852739
The coefficient for disp is -0.28509137265571
The coefficient for hp is -0.019941841619295624
The coefficient for wt is -0.40120686168553854
The coefficient for acc is 0.08532654774160968
```