In [88]:

```
import numpy as np
from numpy import nan
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
#from sklearn.feature_extraction.text import CountVectorizer  #DT does not take strings as input for the model fit step....
from IPython.display import Image
#import pydotplus as pydot
from scipy.stats import zscore
from sklearn import tree
from os import system
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier,RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
from sklearn.decomposition import PCA
```

PART A marks 30

- DOMAIN: Automobile
- CONTEXT: The data concerns city-cycle fuel consumption in miles per gallon to be predicted in terms of 3 multivalued discrete and 5 continuous attributes
- DATA DESCRIPTION:

- cylinders: multi-valued discrete

- acceleration: continuous
- displacement: continuous
- model year: multi-valued discrete
- horsepower: continuous
- origin: multi-valued discrete
- weight: continuous car
- name: string (unique for each instance)
- mpg: continuous

- PROJECT OBJECTIVE: To understand K-means Clustering by applying on the Car Dataset to segment the cars into various categories.

• STEPS AND TASK [30 Marks]:

1. Data Understanding & Exploration: [5 Marks]

1 A. Read 'Car name.csv' as a DataFrame and assign it to a variable. [1 Mark]

In [29]:
```
df_carname=pd.read_csv("C:\\Users\\CHIRAG\\Desktop\\Greatlearning- Projects\\AIML project -unsupervised learning\\Car name.csv")
```

In [30]:
```
df_carname.head()
```

Out[30]:

	car_name
0	chevrolet chevelle malibu
1	buick skylark 320
2	plymouth satellite
3	amc rebel sst
4	ford torino

In [31]:
```
df_carname.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   car_name  398 non-null    object
dtypes: object(1)
memory usage: 3.2+ KB
```

1 B. Read 'Car-Attributes.json as a DataFrame and assign it to a variable. [1 Mark]

In [32]:
```
df_attribute=pd.read_json("C:\\Users\\CHIRAG\\Desktop\\Greatlearning- Projects\\AIML project -unsupervised learning\\Car-Attribut
```

In [33]:
```
df_attribute
```

Out[33]:

	mpg	cyl	disp	hp	wt	acc	yr	origin
0	18.0	8	307.0	130	3504	12.0	70	1
1	15.0	8	350.0	165	3693	11.5	70	1
2	18.0	8	318.0	150	3436	11.0	70	1
3	16.0	8	304.0	150	3433	12.0	70	1
4	17.0	8	302.0	140	3449	10.5	70	1
...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1
394	44.0	4	97.0	52	2130	24.6	82	2
395	32.0	4	135.0	84	2295	11.6	82	1
396	28.0	4	120.0	79	2625	18.6	82	1
397	31.0	4	119.0	82	2720	19.4	82	1

398 rows × 8 columns

In [34]:
```
df_attribute.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   mpg     398 non-null    float64
 1   cyl     398 non-null    int64  
 2   disp    398 non-null    float64
 3   hp      398 non-null    object 
 4   wt      398 non-null    int64  
 5   acc     398 non-null    float64
 6   yr      398 non-null    int64  
 7   origin  398 non-null    int64  
dtypes: float64(3), int64(4), object(1)
memory usage: 25.0+ KB
```

1 C. Merge both the DataFrames together to form a single DataFrame [2 Mark]

In [35]:
```
df=pd.concat([df_carname,df_attribute],axis=1)
df
```

Out[35]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin
0	chevrolet chevelle malibu	18.0	8	307.0	130	3504	12.0	70	1
1	buick skylark 320	15.0	8	350.0	165	3693	11.5	70	1
2	plymouth satellite	18.0	8	318.0	150	3436	11.0	70	1
3	amc rebel sst	16.0	8	304.0	150	3433	12.0	70	1
4	ford torino	17.0	8	302.0	140	3449	10.5	70	1
...	...	...	...	...	...	...	...	...	...
393	ford mustang gl	27.0	4	140.0	86	2790	15.6	82	1
394	vw pickup	44.0	4	97.0	52	2130	24.6	82	2
395	dodge rampage	32.0	4	135.0	84	2295	11.6	82	1
396	ford ranger	28.0	4	120.0	79	2625	18.6	82	1
397	chevy s-10	31.0	4	119.0	82	2720	19.4	82	1

398 rows × 9 columns

In [36]:
```
df.shape
```

Out[36]:
```
(398, 9)
```

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   car_name  398 non-null    object
 1   mpg       398 non-null    float64
 2   cyl       398 non-null    int64
 3   disp      398 non-null    float64
 4   hp        398 non-null    object
 5   wt        398 non-null    int64
 6   acc       398 non-null    float64
 7   yr        398 non-null    int64
 8   origin    398 non-null    int64
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

1D. Print 5 point summary of the numerical features and share insights. [1 Marks]

In [38]:
```
df.describe().T
```

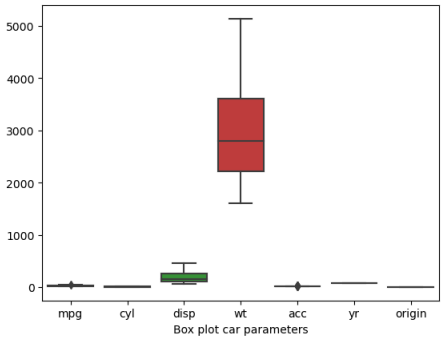
Out[38]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
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
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

In [39]:
```
df['origin'].value_counts()
```

Out[39]:
```
origin
1    249
3     79
2     70
Name: count, dtype: int64
```

In [40]:
```
sns.boxplot(data=df)
plt.xlabel('Box plot car parameters')
plt.text(x=30,y=.4,s='3rd quartyile')
plt.show()
```



3rd quartyile

In [41]:
```
df.describe().T
```

Out[41]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
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
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

insight

- the 5 pont summary describes that 'wt' which is continious weight. dominates all other features its median is somewhere between 2700-2900.
- maximum number of cylinders are 8 and minimum is 3 in dataset.
- vehicle with minimum mpg is 9 and maximum mpg is 46. 50% of the vehicle has mpg between 9 - 23 25% of the vehcle has mpg betwenn 9 - 17 75% of the vehicle has mpg betwen 9 - 29
- vehicle with minimum disp (displacement) is 68 and maximum disp is 262. 50% of the vehicle has disp between 68 - 148 25% of the vehcle has disp betwenn 68 - 104 75% of the vehicle has disp betwen 68 - 262
- vehicle with minimum hp (horsepower) is 46 and maximum hp is 230. 50% of the vehicle has disp between 46 - 93.5 25% of the vehcle has disp betwenn 46 - 75 75% of the vehicle has disp betwen 46 - 126
- vehicle with minimum acc (acceleration) is 8 and maximum is 24.8 50% of the vehicle has disp between 8 - 15.5 25% of the vehcle has disp betwenn 8 - 13.825 75% of the vehicle has disp betwen 8 - 17.125
- average weight of the vehicle is arround 3000kg

2. Data Preparation & Analysis: [10 Marks]

2 A. Check and print feature-wise percentage of missing values present in the

In [42]:
```
#dropping/ignoring car_name  and origin
#df = df.drop(['car_name','origin'], axis=1)
#df.head()
```

In [43]:
```
def missing_value(func_df):
    total=func_df.isnull().sum().sort_values()
    percentage=(total/func_df.isnull().count().sum()).sort_values()*100
    null_df=pd.concat([total,percentage],keys=['total','percentage'],axis=1)
    return(null_df)
```

In [44]:
```
missing_value(df)
```

Out[44]:

	total	percentage
car_name	0	0.0
mpg	0	0.0
cyl	0	0.0
disp	0	0.0
hp	0	0.0
wt	0	0.0
acc	0	0.0
yr	0	0.0
origin	0	0.0

In [45]:
```
# isdigit()? on 'horsepower'
hpIsDigit = pd.DataFrame(df.hp.str.isdigit())  # if the string is made of digits store True else False

#print isDigit = False!
df[hpIsDigit['hp'] == False]   # from temp take only those rows where hp has false
```

Out[45]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin
32	ford pinto	25.0	4	98.0	?	2046	19.0	71	1
126	ford maverick	21.0	6	200.0	?	2875	17.0	74	1
330	renault lecar deluxe	40.9	4	85.0	?	1835	17.3	80	2
336	ford mustang cobra	23.6	4	140.0	?	2905	14.3	80	1
354	renault 18i	34.5	4	100.0	?	2320	15.8	81	2
374	amc concord dl	23.0	4	151.0	?	3035	20.5	82	1

There are various ways to handle missing values. Drop the rows, replace missing values with median values etc. of the 398 rows 6 have NAN in the hp column. We could drop those 6 rows - which might not be a good idea under all situations. Here, we will replace them with their median values. First replace '?' with NaN and then replace NaN with median

In [46]:
```
df=df.replace('?',np.nan)
df[hpIsDigit['hp'] == False]
```

Out[46]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin
32	ford pinto	25.0	4	98.0	NaN	2046	19.0	71	1
126	ford maverick	21.0	6	200.0	NaN	2875	17.0	74	1
330	renault lecar deluxe	40.9	4	85.0	NaN	1835	17.3	80	2
336	ford mustang cobra	23.6	4	140.0	NaN	2905	14.3	80	1
354	renault 18i	34.5	4	100.0	NaN	2320	15.8	81	2
374	amc concord dl	23.0	4	151.0	NaN	3035	20.5	82	1

In [47]:
```
#instead of dropping the rows, lets replace the missing values with median value.
df.iloc[:,1:9].median()
```

Out[47]:
```
mpg          23.0
cyl           4.0
disp        148.5
hp           93.5
wt         2803.5
acc          15.5
yr           76.0
origin        1.0
dtype: float64
```

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   car_name  398 non-null    object
 1   mpg       398 non-null    float64
 2   cyl       398 non-null    int64
 3   disp      398 non-null    float64
 4   hp        392 non-null    float64
 5   wt        398 non-null    int64
 6   acc       398 non-null    float64
 7   yr        398 non-null    int64
 8   origin    398 non-null    int64
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

In [49]:
```
df.describe().T
```

Out[49]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
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
hp	392.0	104.469388	38.491160	46.0	75.000	93.5	126.000	230.0
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
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

2 B. Check for duplicate values in the data and impute with the best suitable approach. [1 Mark]

here we check if any duplicated row is created ?

In [50]:
```
print("So Total number of duplicated rows is = ",df.duplicated().sum())
```

```
So Total number of duplicated rows is =  0
```

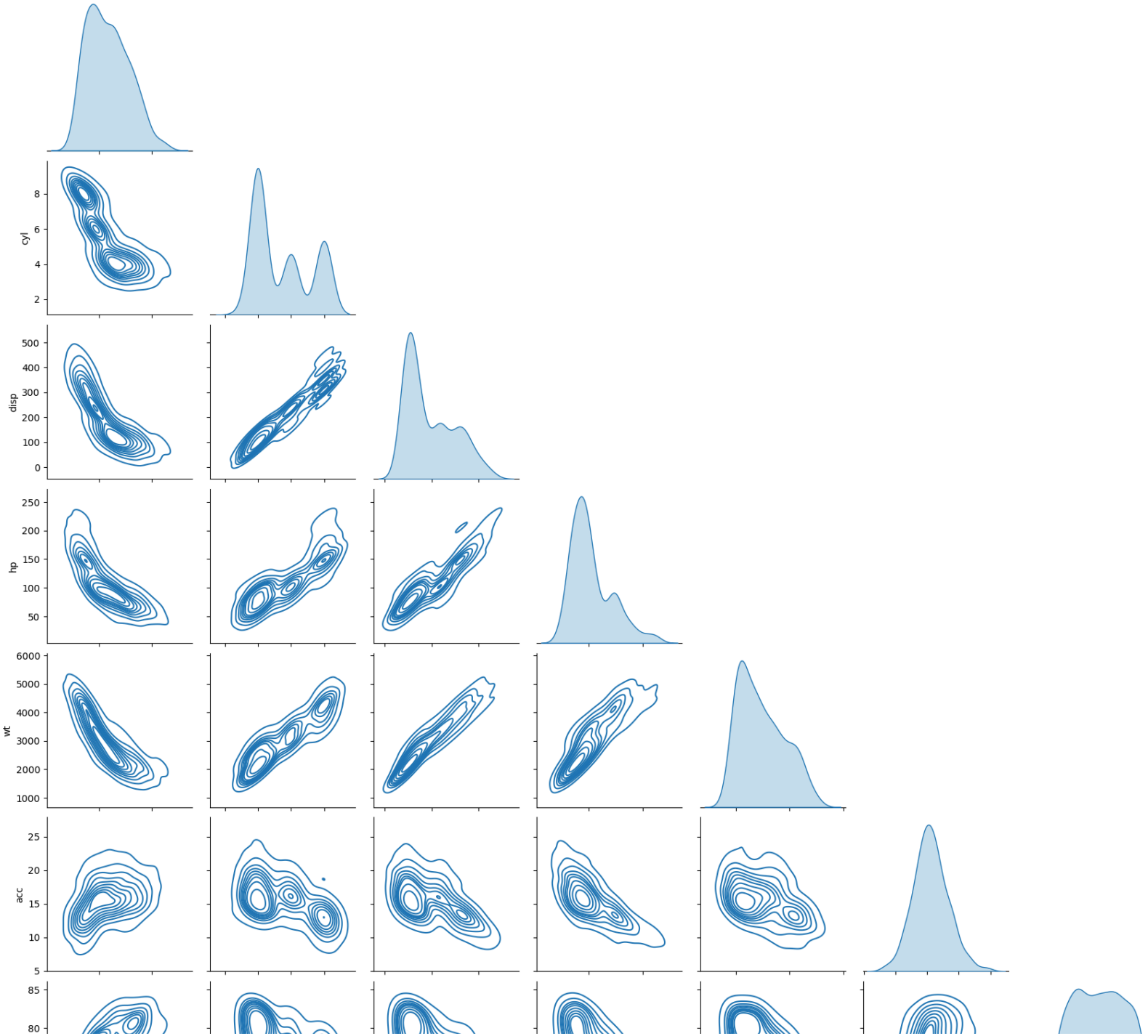
In [51]:
```
print("searching if any duplicated value rawvise = ",df.iloc[:].duplicated().sum())
```

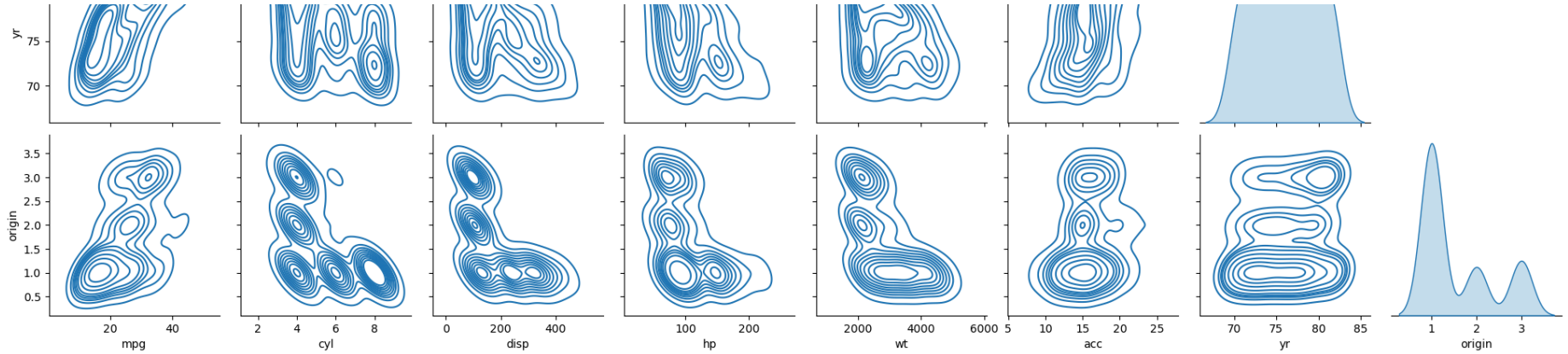
```
searching if any duplicated value rawvise =  0
```

2 C. Plot a pairplot for all features. [1 Marks]

In [52]:
```
sns.pairplot(data=df,corner=True,kind='kde')
```

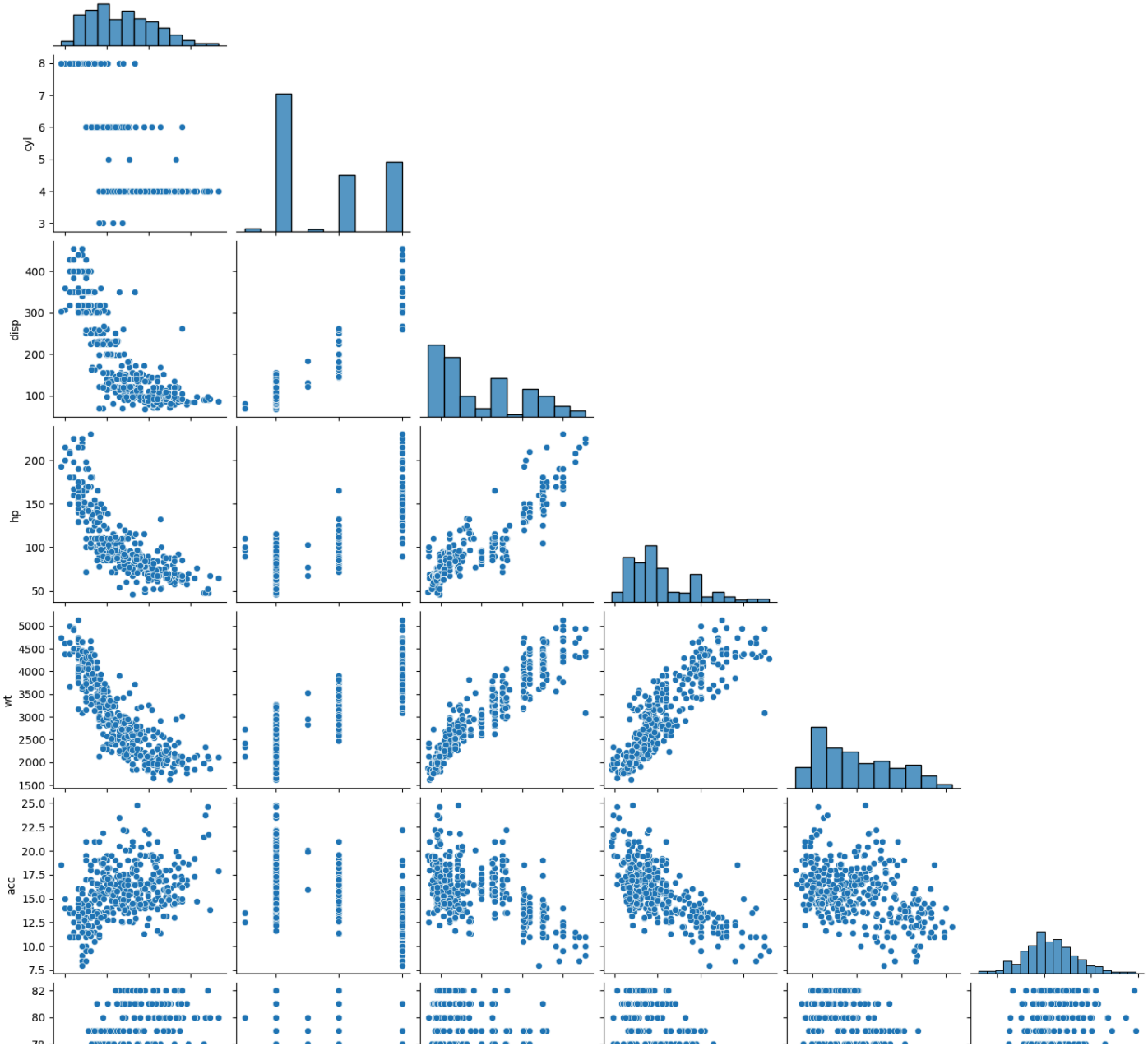
Out[52]:
```
<seaborn.axisgrid.PairGrid at 0x134c5906a90>
```

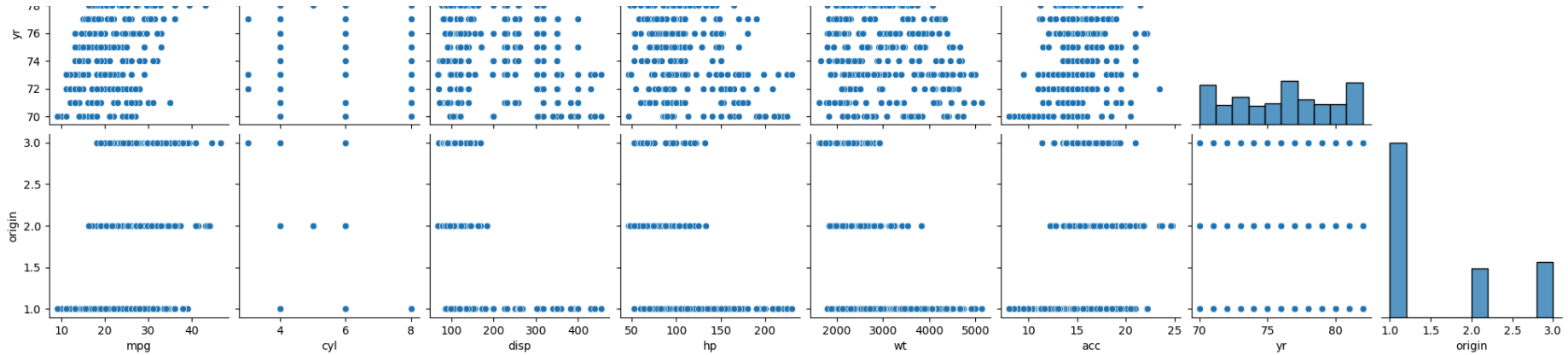

cyl
disp
hp
wt
acc
8
6
4
2
500
400
300
200
100
0
250
200
150
100
50
6000
5000
4000
3000
2000
1000
25
20
15
10
5
85
80



In [53]:
```
sns.pairplot(data=df,corner=True,kind='scatter')
```

Out[53]:
```
<seaborn.axisgrid.PairGrid at 0x134c7e0c4d0>
```


cyl
disp
hp
wt
acc



2 D. Visualize a scatterplot for ‘wt’ and ‘disp’. Datapoints should be distinguishable by ‘cyl’. [1 Marks]

solution:

before separting or visualising sctere plot with respecto cyl lets see what are different categories are there in cyl

In [54]:
```
print("Different categories are",df['cyl'].unique(),"cylinders in engine")
```

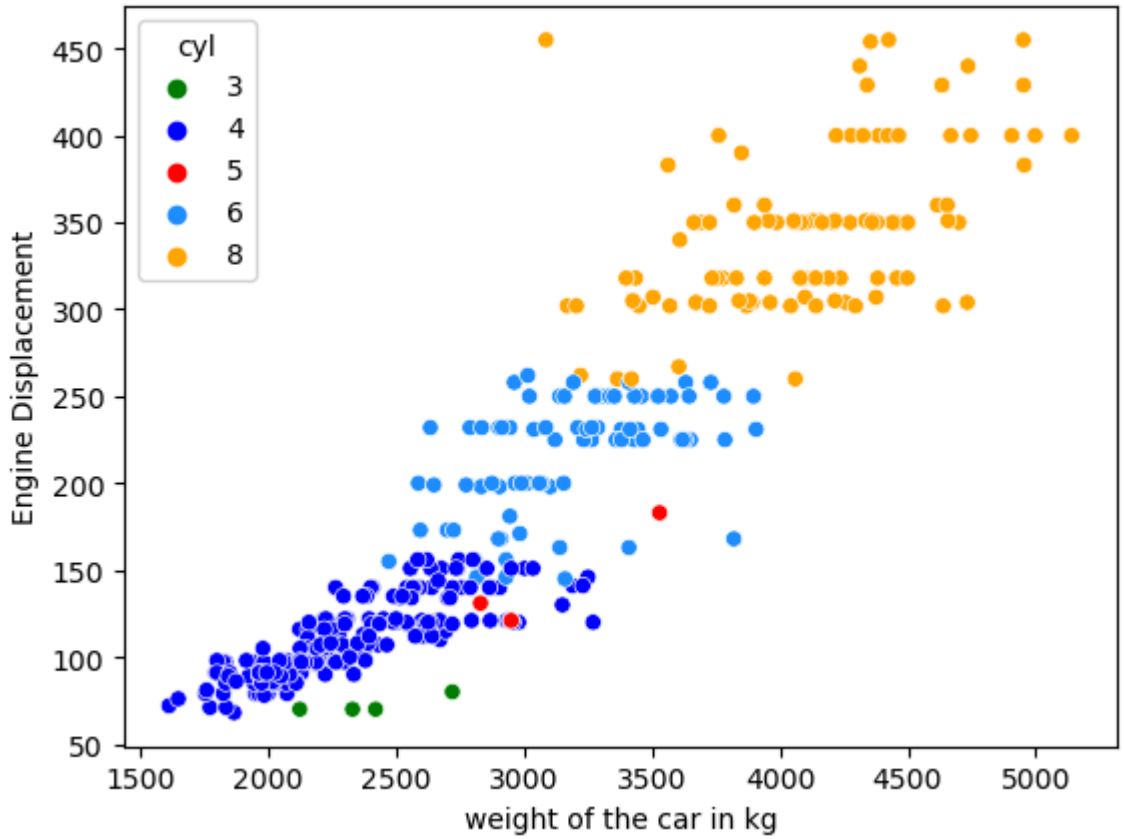
```
Different categories are [8 4 6 3 5] cylinders in engine
```

In [55]:
```
print("The quantity for every type of cylinder car is are \n ",df['cyl'].value_counts())
```

```
The quantity for every type of cylinder car is are
  cyl
4    204
8    103
6     84
3      4
5      3
Name: count, dtype: int64
```

In [56]:
```
plt.ylabel("Engine Displacement")
plt.xlabel("weight of the car in kg")
#s.linewidth=0.7
#s.MarkerEdgeColor='g'
```

```
sns.scatterplot(data=df,x='wt',y='disp',hue='cyl',legend='full',palette=['green','blue','red','dodgerblue','orange'])
plt.show()
```



2 E. Share insights for Q2.d. [1 Marks]

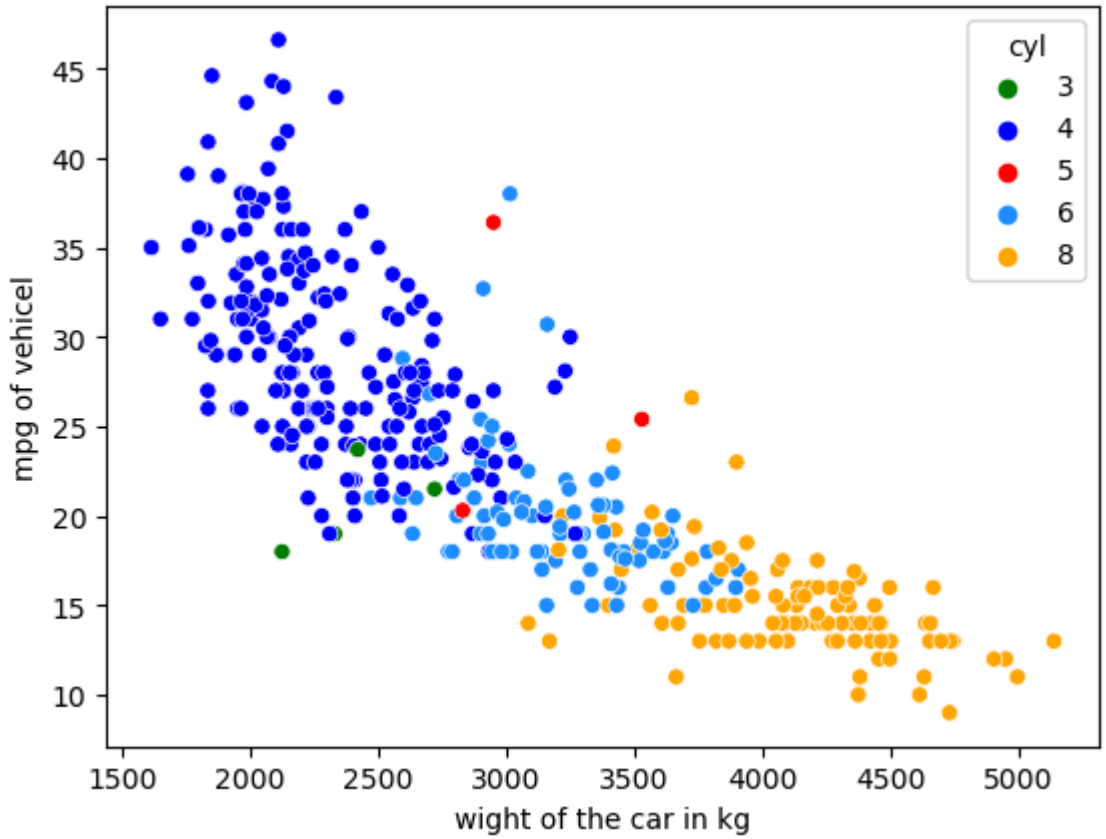
insights

- The scatter plot between car weight and engine displcement is shown above.
- Its is clear that there is positice correlation between 'wt' and 'disp' that is 'wt' increases there is also rise in 'disp'
- but for different 'cyl' observatio is as follows
- with increse in 'wt' and 'disp' the number of 'cyl' is also more spread of scatter plot increses (variece is more)
- as per scatter plot if 'cyl' is 8 the 'wt' ranges between 3200 to 5000 and more with 'disp' value ranges betwenn 250 to 500
- for 'cyl' value 6 the 'wt' ranges 2500-4000, 'disp' ranges 120-280,

- the number of observation for 'cyl' = 5 and 3 is very less
- for 'cyl' value 5 the 'wt' ranges 2800-3700, 'disp' ranges 120-200,
- for 'cyl' value 4 the 'wt' ranges 1500-3400, 'disp' ranges 40 -160, data points are more closely attached (we can say varience is less)compare to other

2 F. Visualize a scatterplot for ‘wt’ and ‘mpg’. Datapoints should be distinguishable by ‘cyl’. [1 Marks]

In [58]:
```
plt.ylabel("mpg of vehicel")
plt.xlabel("wight of the car in kg")
sns.scatterplot(data=df,x='wt',y='mpg',hue='cyl',legend='full',palette=['green','blue','red','dodgerblue','orange'])
plt.show()
```



2 G. Share insights for Q2.f. [1 Marks]

insights:

- There is some-negative correlation is seen between 'wt' and 'mpg datapoint as 'wt' increses 'mpg' decreases
- For different 'cyl' values observations are as follows
- for 'cyl' = 8, the 'wt' range between 3000-6000 'mpg' ranges between 7-25
- for 'cyl' = 6, the 'wt' range between 2200-3900 'mpg'ranges between 14-40 this is very wide range variation for 'mpg' with 'wt'
- there are very less obervation for 'cyl' =5 however oberservation is as follows
- for 'cyl' = 5, the 'wt' range between 2900-3500 'mpg'ranges between 20-36
- for 'cyl' = 4, the 'wt' range between 1500-3200 'mpg' ranges between18-50
- for 'cyl'=3 very limited number of obersvations however there ranges are as follows for 'cyl' = 3, the 'wt' range between 2000-2700 'mpg' ranges between17-24
- for 'cyl' = 2, the 'wt' range between 1500-3200 'mpg' ranges between19-45

- overall it has some-negative correlation

2 H. Check for unexpected values in all the features and datapoints with such values. [2 Marks]

[Hint: '?' is present in 'hp']

In [59]:
```
df['hp'].dtype
```

Out[59]:
```
dtype('float64')
```

In [60]:
```
df['hp'].unique()
```

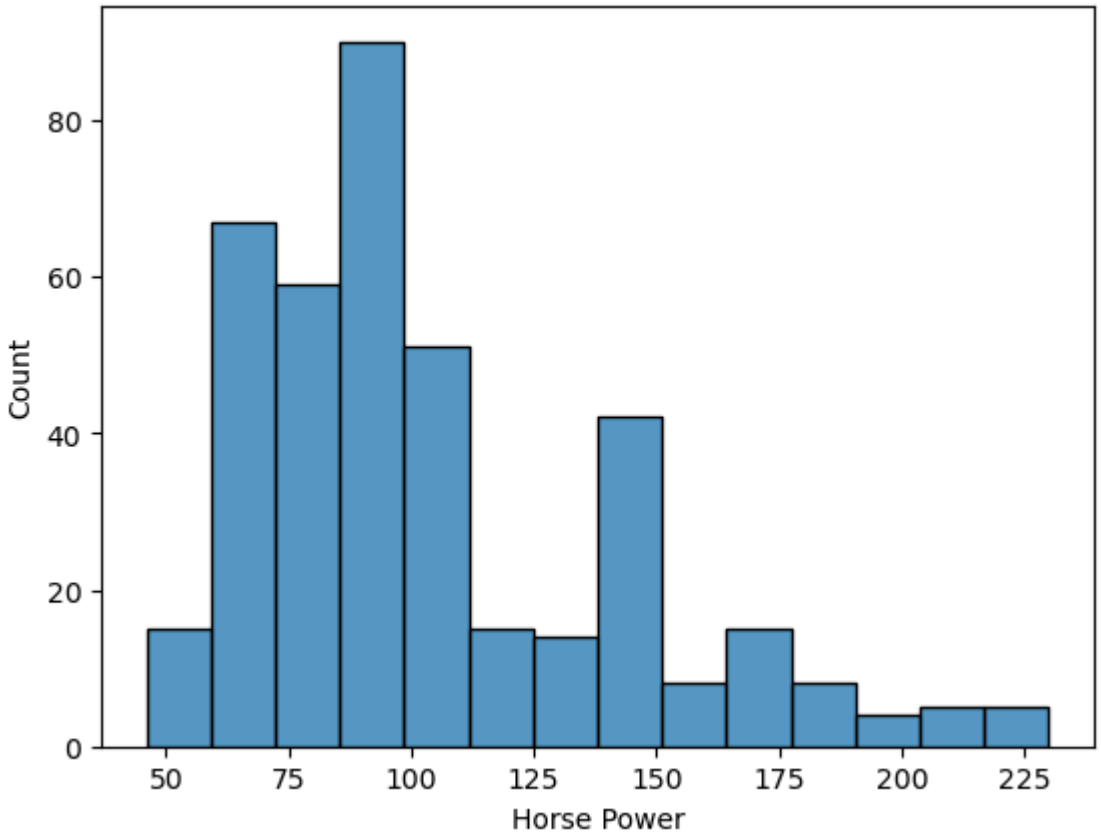
Out[60]:
```
array([130., 165., 150., 140., 198., 220., 215., 225., 190., 170., 160.,
        95.,  97.,  85.,  88.,  46.,  87.,  90., 113., 200., 210., 193.,
        nan, 100., 105., 175., 153., 180., 110.,  72.,  86.,  70.,  76.,
        65.,  69.,  60.,  80.,  54., 208., 155., 112.,  92., 145., 137.,
       158., 167.,  94., 107., 230.,  49.,  75.,  91., 122.,  67.,  83.,
        78.,  52.,  61.,  93., 148., 129.,  96.,  71.,  98., 115.,  53.,
        81.,  79., 120., 152., 102., 108.,  68.,  58., 149.,  89.,  63.,
        48.,  66., 139., 103., 125., 133., 138., 135., 142.,  77.,  62.,
       132.,  84.,  64.,  74., 116.,  82.])
```

In [61]:
```
df['hp']=df['hp'].replace(nan,df['hp'].median())
df['hp']=df['hp'].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   car_name  398 non-null    object 
 1   mpg       398 non-null    float64
 2   cyl       398 non-null    int64  
 3   disp      398 non-null    float64
 4   hp        398 non-null    float64
 5   wt        398 non-null    int64  
 6   acc       398 non-null    float64
 7   yr        398 non-null    int64  
 8   origin    398 non-null    int64  
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

In [62]:
```
sns.histplot(data=df,x='hp')
plt.xlabel("Horse Power")
```

Out[62]:
```
Text(0.5, 0, 'Horse Power')
```



as distribution of horse power is right skewned lets take median and replace ? values in data

In [63]:
```
df.describe().T
```

Out[63]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
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
hp	398.0	104.304020	38.222625	46.0	76.000	93.5	125.000	230.0
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
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

3. Clustering: [15 Marks]

3A. Apply K-Means clustering for 2 to 10 clusters. [3 Marks]

In [64]:
```
# before Applying K-means clustering lets see how features are
# dominating each other by simply see there graphs
```

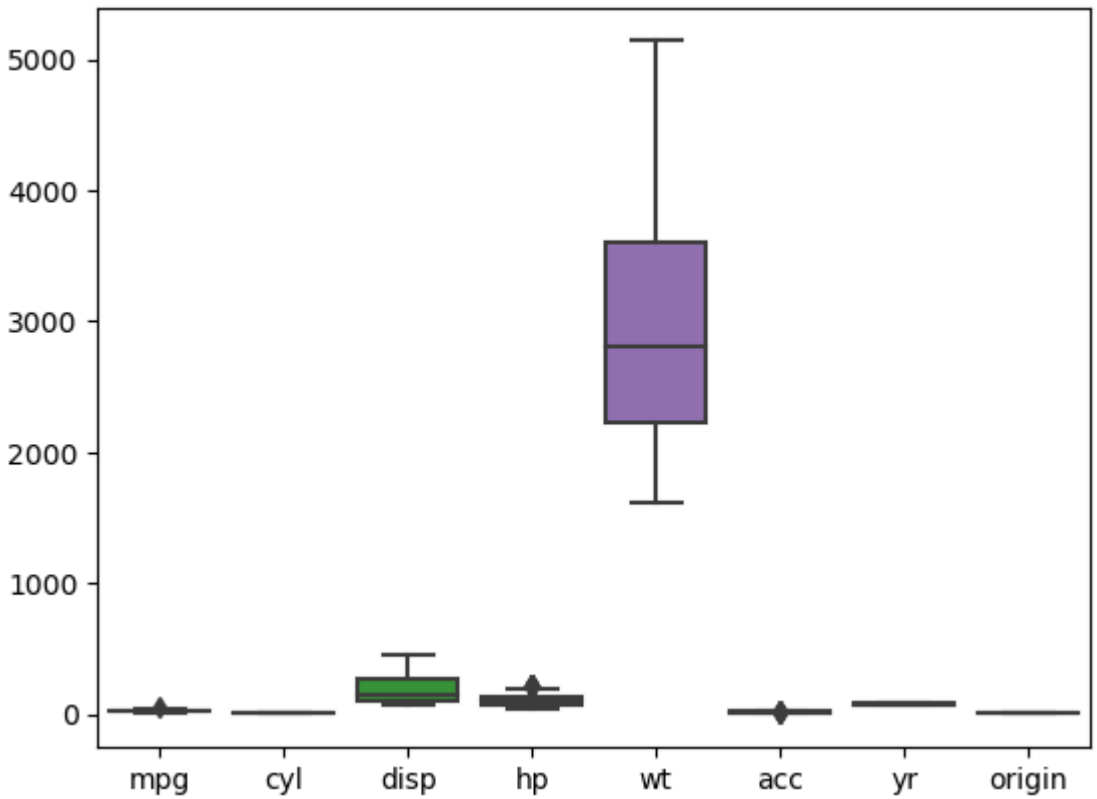
In [65]:
```
df.head()
```

Out[65]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	70	1
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70	1
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	70	1
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	70	1
4	ford torino	17.0	8	302.0	140.0	3449	10.5	70	1

In [66]:
```
sns.boxplot(data=df)
```

Out[66]: <Axes: >

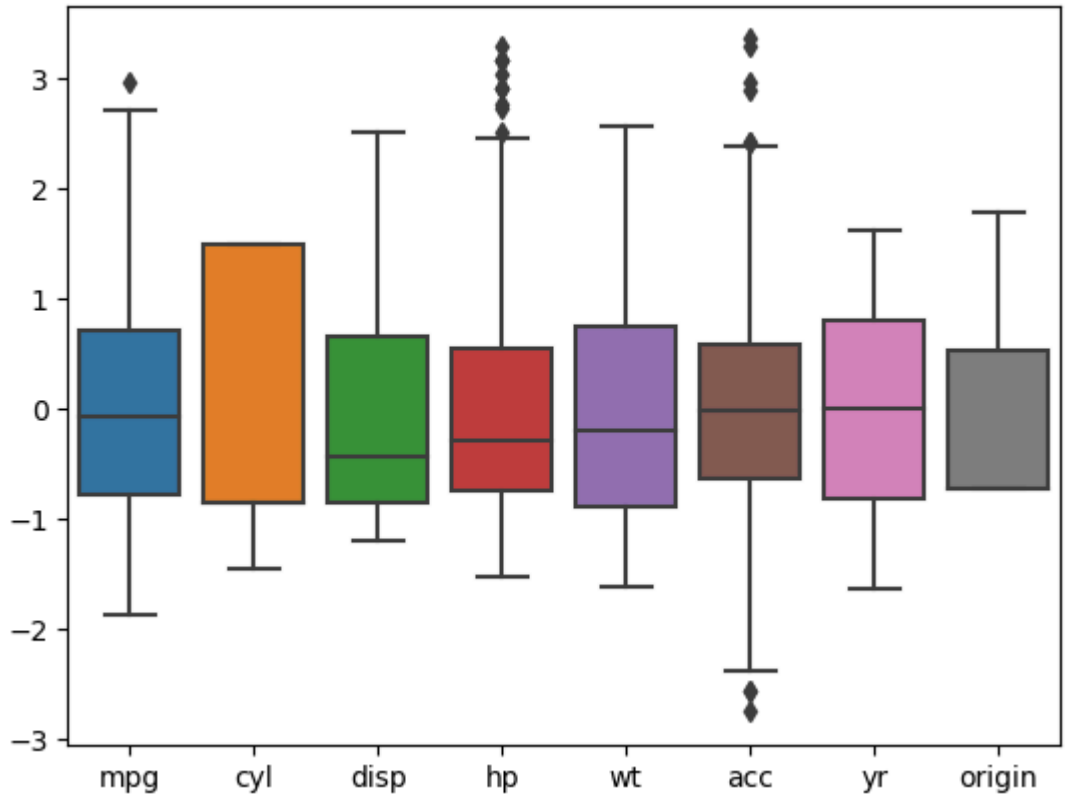


- from graph we can see from box plot
- that collumn 'wt' dominates to all other features
- we will need to apply scaling of data

- performing scaling values

In [67]:
```
scaled_df=df.iloc[:,1:]
scaled_df1=scaled_df.apply(zscore)
sns.boxplot(data=scaled_df1)
```

Out[67]: <Axes: >

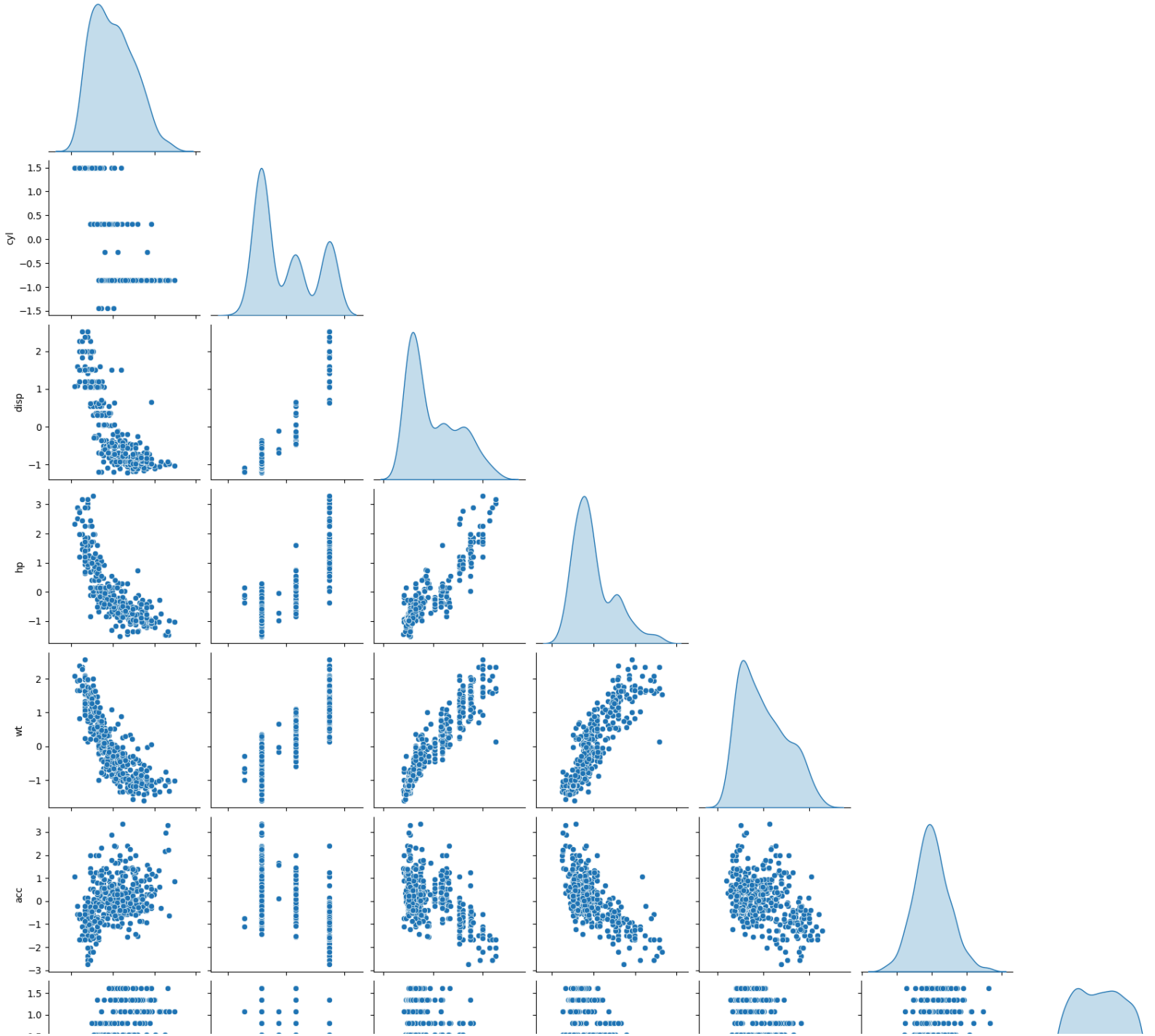


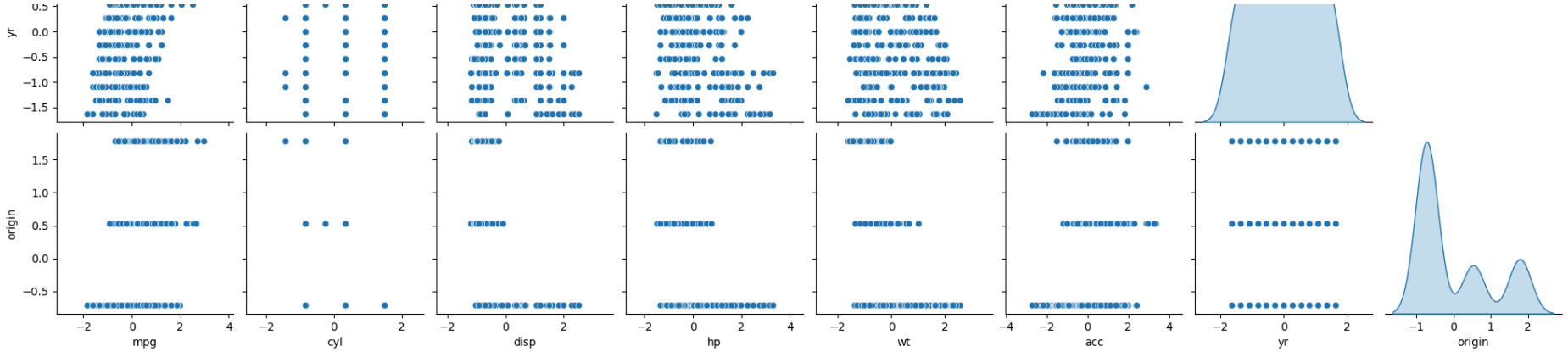
- we are ignoring outliers if we treat than data may tend to overfitted adn we may loose some information

- From the box plot we can see that now dominance of of wt has reduce
- And now we can apply k-mean clustering

In [68]:
```
sns.pairplot(data=scaled_df1,diag_kind='kde',corner=True)
```

Out[68]:
```
<seaborn.axisgrid.PairGrid at 0x134ccf4d350>
```


cyl
disp
hp
wt
acc

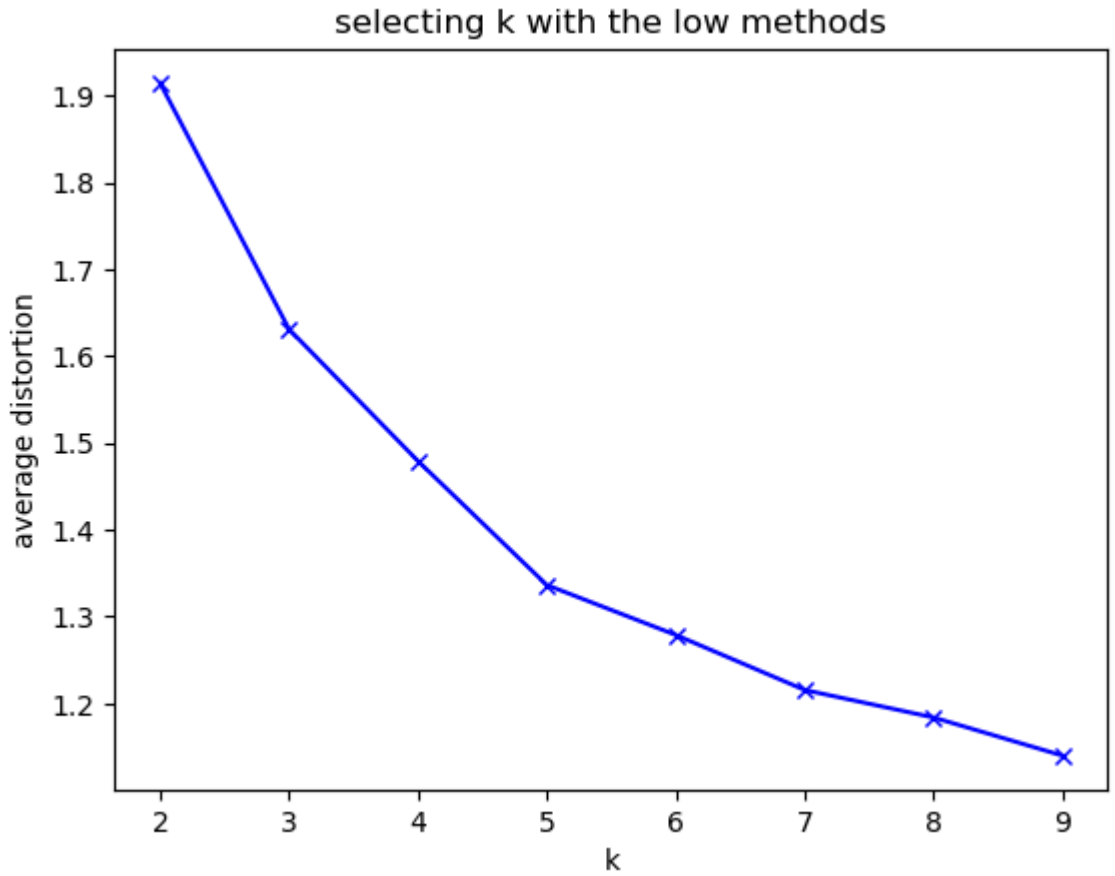


In [69]:
```
## importing libraary to perform cluring methods for unsupervised modling
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
clusters=range(2,10)
meanDistortion=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(scaled_df1)
    predict=model.predict(scaled_df1)
    meanDistortion.append(sum(np.min(cdist(scaled_df1,model.cluster_centers_,'euclidean'),axis=1))/scaled_df1.shape[0])
```

3B. Plot a visual and find elbow point. [2 Marks]

In [70]:
```
plt.plot(clusters,meanDistortion,'bx-')
plt.xlabel('k')
plt.ylabel('average distortion')
plt.title('selecting k with the low methods')
```

Out[70]:
```
Text(0.5, 1.0, 'selecting k with the low methods')
```



- visually k=3 and 5 where there is pivot view point

3 C. On the above visual, highlight which are the possible Elbow points. [1 Marks]

- from visual we see there is clear lbow at 3,5 we have lbow

3 D. Train a K-means clustering model once again on the optimal number of clusters. [3 Marks]

In [71]:
```
# let us first start with k =5
final_model = KMeans(5)
final_model.fit(scaled_df1)
prediction=final_model.predict(scaled_df1)
```

- Analyze the distribution of the data among the two groups(k=

In [72]:
```
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [73]:
```
# Calculating silhouette_score
labels=final_model.labels_
silhouette_score(scaled_df1,labels)
```

Out[73]:
```
0.33296973781495653
```

now with k == 3

In [74]:
```
# let us first start with k =3
final_model = KMeans(3)
final_model.fit(scaled_df1)
prediction=final_model.predict(scaled_df1)
```

In [75]:
```
# Calculating silhouette_score
labels=final_model.labels_
silhouette_score(scaled_df1,labels)
```

Out[75]:
```
0.3246673901601845
```

- no much difference in silhoutte score k =3 and 5 let go for k=3 cluster model

3 E. Add a new feature in the DataFrame which will have labels based upon cluster value. [2 Mark]

In [76]:
```
#Append the prediction
df['GROUP'] = prediction
scaled_df1['GROUP'] = prediction
```

```
print("group Asiigned : \n")
df.head(15)
```

```
group Asiigned :
```

Out[76]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin	GROUP
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	70	1	1
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70	1	1
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	70	1	1
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	70	1	1
4	ford torino	17.0	8	302.0	140.0	3449	10.5	70	1	1
5	ford galaxie 500	15.0	8	429.0	198.0	4341	10.0	70	1	1
6	chevrolet impala	14.0	8	454.0	220.0	4354	9.0	70	1	1
7	plymouth fury iii	14.0	8	440.0	215.0	4312	8.5	70	1	1
8	pontiac catalina	14.0	8	455.0	225.0	4425	10.0	70	1	1
9	amc ambassador dpl	15.0	8	390.0	190.0	3850	8.5	70	1	1
10	dodge challenger se	15.0	8	383.0	170.0	3563	10.0	70	1	1
11	plymouth 'cuda 340	14.0	8	340.0	160.0	3609	8.0	70	1	1
12	chevrolet monte carlo	15.0	8	400.0	150.0	3761	9.5	70	1	1
13	buick estate wagon (sw)	14.0	8	455.0	225.0	3086	10.0	70	1	1
14	toyota corona mark ii	24.0	4	113.0	95.0	2372	15.0	70	3	0

In [77]:

```
df_supcluster=scaled_df1.groupby(['GROUP'])
df_supcluster.mean()
```

Out[77]:

GROUP	mpg	cyl	disp	hp	wt	acc	yr	origin
0	0.898442	-0.816104	-0.864265	-0.705479	-0.869856	0.270026	0.310110	0.959683
1	-1.127260	1.486419	1.468657	1.473388	1.367364	-1.033718	-0.611986	-0.715145
2	-0.233015	-0.125906	-0.056343	-0.246400	0.024163	0.437210	0.082269	-0.605799

In [78]:
```
#Append the prediction
df['GROUP'] = prediction
scaled_df['GROUP'] = prediction
print("group Asiigned : \n")
df.head(15)

df_Orgcluster=scaled_df.groupby(['GROUP'])
df_Orgcluster.mean()
```

```
group Asiigned :
```

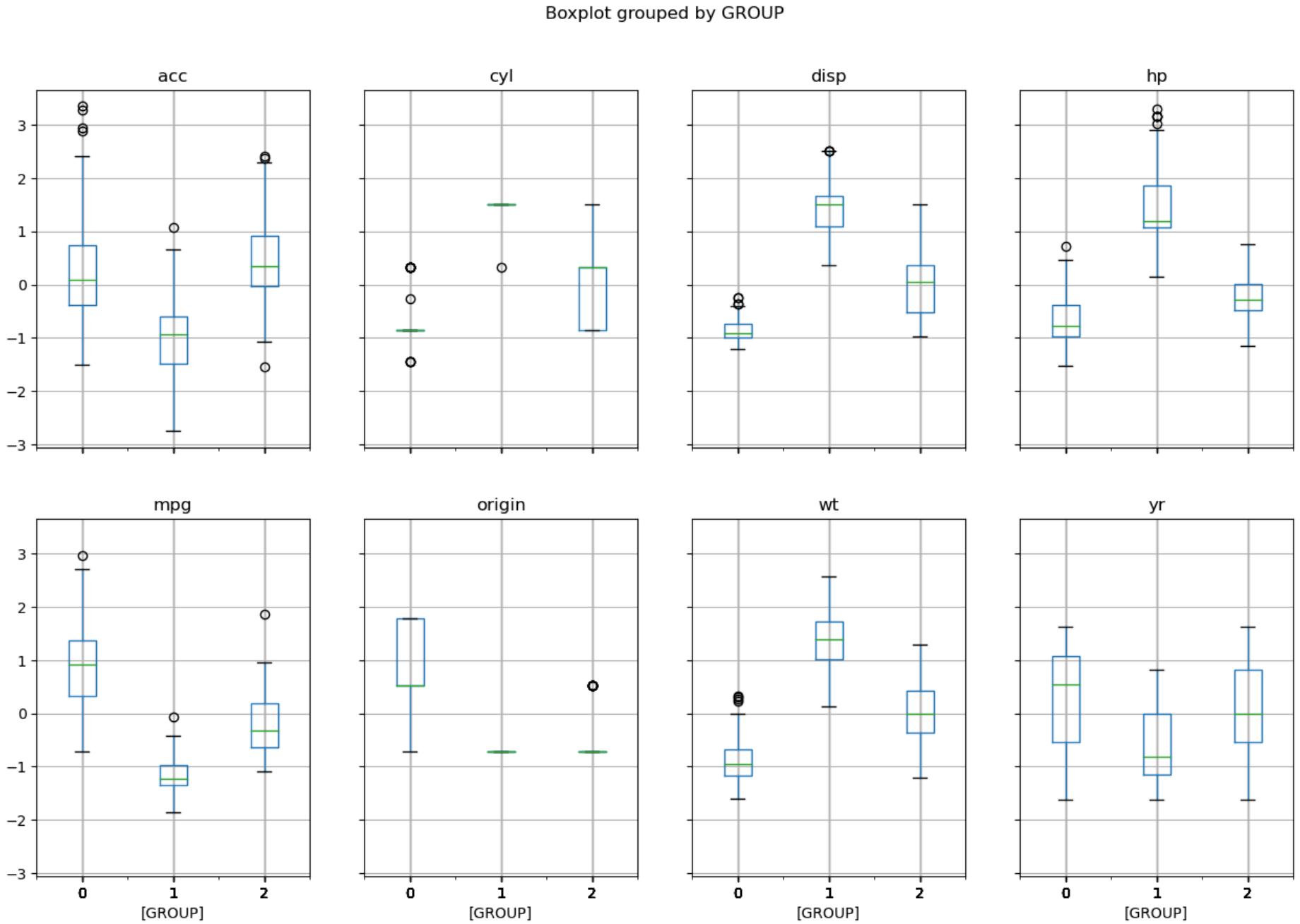
Out[78]:

GROUP	mpg	cyl	disp	hp	wt	acc	yr	origin
0	30.52795	4.068323	103.422360	77.372671	2234.720497	16.311801	77.155280	2.341615
1	14.71500	7.980000	346.370000	160.550000	4126.910000	12.721000	73.750000	1.000000
2	21.69562	5.240876	187.558394	94.897810	2990.861314	16.772263	76.313869	1.087591

3 F. Plot a visual and color the datapoints based upon clusters. [2 Marks]

In [79]:
```
scaled_df1.boxplot(by='GROUP',layout=(2,4),figsize=(15,10))
```

Out[79]:
```
array([[<Axes: title={'center': 'acc'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'cyl'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'disp'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'hp'}, xlabel='[GROUP]'>],
       [<Axes: title={'center': 'mpg'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'origin'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'wt'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'yr'}, xlabel='[GROUP]'>]], dtype=object)
```



In [80]:
```
# let us first start with k =5
final_model = KMeans(5)
```

```
kmeans=final_model.fit(scaled_df1)
prediction=final_model.predict(scaled_df1)

#Append the prediction
df['GROUP'] = prediction
scaled_df1['GROUP'] = prediction
print("group Asiigned : \n")
df.head()
```

```
group Asiigned :
```

Out[80]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin	GROUP
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	70	1	4
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70	1	4
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	70	1	4
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	70	1	4
4	ford torino	17.0	8	302.0	140.0	3449	10.5	70	1	4

In [81]:

```
df_supcluster=scaled_df1.groupby(['GROUP'])
df_supcluster.mean()
```

Out[81]:

GROUP	mpg	cyl	disp	hp	wt	acc	yr	origin
0	1.401399	-0.793995	-0.852053	-0.822796	-0.880751	0.392605	1.099529	0.797582
1	0.169421	-0.742393	-0.537884	-0.392234	-0.376869	0.440839	0.346677	-0.533930
2	0.335923	-0.840830	-0.877924	-0.574268	-0.857671	0.132930	-0.572793	1.140979
3	-0.557641	0.426587	0.365363	-0.105189	0.369511	0.413765	-0.113812	-0.667131
4	-1.151105	1.486055	1.484507	1.506241	1.387534	-1.062679	-0.644787	-0.715145

In [82]:

```
# let us first start with k =3
final_model = KMeans(3)
final_model.fit(scaled_df1)
prediction=final_model.predict(scaled_df1)
```

```
#Append the prediction
df['GROUP'] = prediction
scaled_df1['GROUP'] = prediction
print("group Asiigned : \n")
df.head()
```

```
group Asiigned :
```

Out[82]:

	car_name	mpg	cyl	disp	hp	wt	acc	yr	origin	GROUP
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	70	1	1
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70	1	1
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	70	1	1
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	70	1	1
4	ford torino	17.0	8	302.0	140.0	3449	10.5	70	1	1

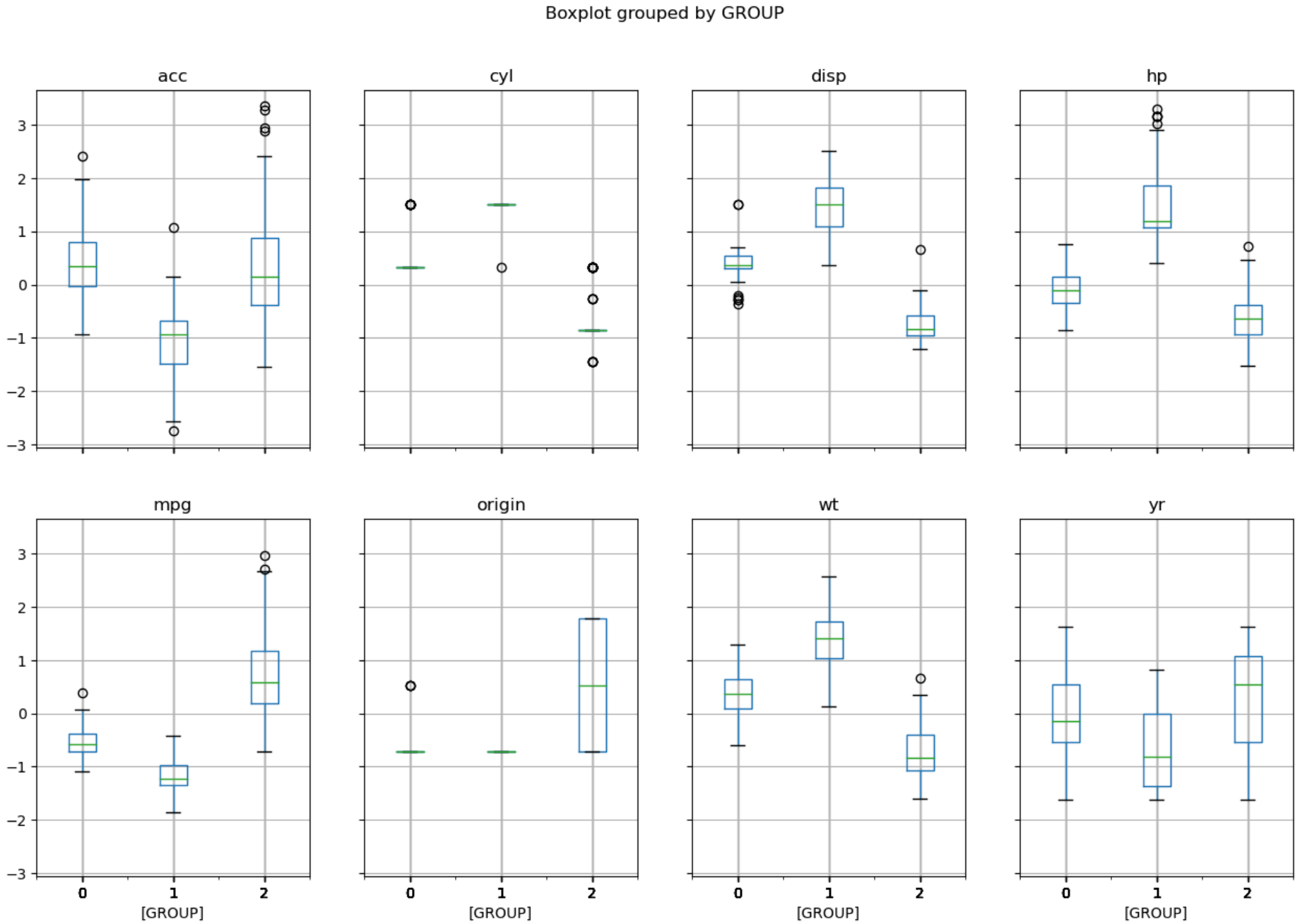
In [83]:
```
df_orgclust=scaled_df1.groupby(['GROUP'])
df_final=df_orgclust.mean()
df_final
```

Out[83]:

GROUP	mpg	cyl	disp	hp	wt	acc	yr	origin
0	-0.557641	0.426587	0.365363	-0.105189	0.369511	0.413765	-0.113812	-0.667131
1	-1.151105	1.486055	1.484507	1.506241	1.387534	-1.062679	-0.644787	-0.715145
2	0.695754	-0.795610	-0.773523	-0.618388	-0.732792	0.317516	0.320277	0.544418

In [84]:
```
scaled_df1.boxplot(by='GROUP',layout=(2,4),figsize=(15,10))
```

Out[84]:
```
array([[<Axes: title={'center': 'acc'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'cyl'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'disp'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'hp'}, xlabel='[GROUP]'>],
       [<Axes: title={'center': 'mpg'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'origin'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'wt'}, xlabel='[GROUP]'>,
        <Axes: title={'center': 'yr'}, xlabel='[GROUP]'>]], dtype=object)
```



In [89]:
```
# Instantiate transformer
pca = PCA(n_components=2,random_state=42)
```

```
# Transform `X`
X_t = pca.fit_transform(scaled_df1)

# Put `X_t` into DataFrame
X_pca = pd.DataFrame(X_t,columns=["PC1","PC2"])

print("X_pca type:", type(X_pca))
print("X_pca shape:", X_pca.shape)
X_pca.head()
```

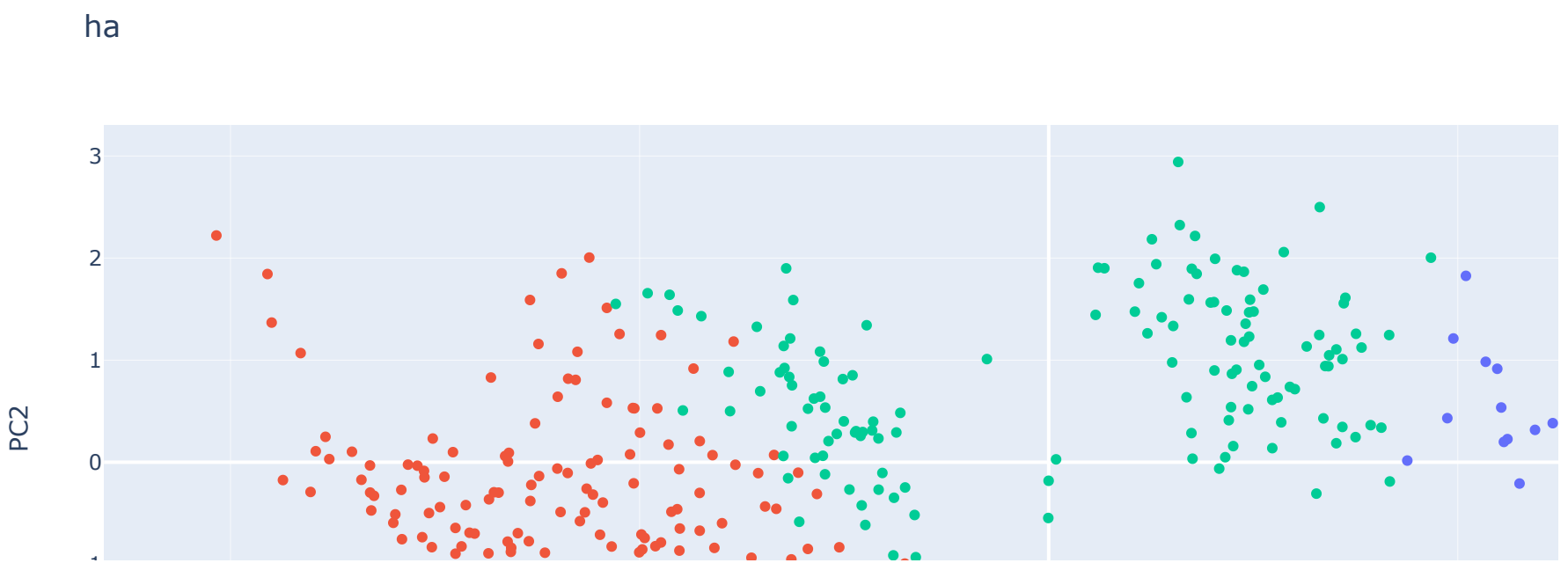
```
X_pca type: <class 'pandas.core.frame.DataFrame'>
X_pca shape: (398, 2)
```

Out[89]:

	PC1	PC2
0	2.749821	-0.882889
1	3.550919	-1.144452
2	3.053785	-1.225098
3	3.004395	-1.016351
4	2.989667	-1.279075

In [90]:
```
labels=labels.astype(str)
```

In [91]:
```
import plotly.express as px
fig=px.scatter(
    data_frame=X_pca,
    x="PC1",
    y="PC2",
    color=labels,
    title="ha"

)
fig.update_layout(xaxis_title="PC1",yaxis_title="PC2")
fig.show()
```



- from diagram it it devided into three groups 0 ,1,and 2 distribution of data points are as shown in figure

3 G. Pass a new DataPoint and predict which cluster it belongs to. [2 Marks]

In [92]:
```
# let us first start with k =3
final_model = KMeans(3)
final_model.fit(scaled_df1)
prediction=final_model.predict(scaled_df)
```

In [93]:
```
scaled_df.columns
```

Out[93]:
```
Index(['mpg', 'cyl', 'disp', 'hp', 'wt', 'acc', 'yr', 'origin', 'GROUP'], dtype='object')
```

In [94]:
```
list=[[40,15,350.5,2000, 2000,9,75,6,3]]

table=['mpg','cyl','disp','hp','wt','acc','yr','origin','GROUP']
```

In [95]:
```
new_points=pd.DataFrame(data=list,columns = table)
new_points
```

Out[95]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	GROUP
0	40	15	350.5	2000	2000	9	75	6	3

In [96]:
```
new_cluster=final_model.predict(new_points)
print("the new datapoint is belong to cluster number",new_cluster[0])
```

```
the new datapoint is belong to cluster number 1
```

==========END OF PART

==========

PART B

DOMAIN: Automobile

• **CONTEXT:** The purpose is to classify a given silhouette as one of three types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles.

• **DATA DESCRIPTION:** The data contains features extracted from the silhouette of vehicles in different angles. Four "Corgie" model vehicles were used for the experiment: a double decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400 cars. This particular combination of vehicles was chosen with the expectation that the bus, van and either one of the cars would be readily distinguishable, but it would be more difficult to distinguish between the cars.

• All the features are numeric i.e. geometric features extracted from the silhouette.

• **PROJECT OBJECTIVE:** Apply dimensionality reduction technique – PCA and train a model and compare relative results.

• STEPS AND TASK [30 Marks]:

1. Data Understanding & Cleaning: [5 Marks]

1 A. Read 'vehicle.csv' and save as DataFrame. [1 Marks]

In [97]:
```
vehicle_df=pd.read_csv("C:\\Users\\CHIRAG\\Desktop\\Greatlearning- Projects\\AIML project -unsupervised learning\\vehicle.csv")
```

In [98]:
```
vehicle_df.head()
```

Out[98]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangular
0	95	48.0	83.0	178.0	72.0	10	162.0	42.0	2
1	91	41.0	84.0	141.0	57.0	9	149.0	45.0	1
2	104	50.0	106.0	209.0	66.0	10	207.0	32.0	2
3	93	41.0	82.0	159.0	63.0	9	144.0	46.0	1
4	85	44.0	70.0	205.0	103.0	52	149.0	45.0	1

In [99]:
```
vehicle_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 846 entries, 0 to 845
Data columns (total 19 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   compactness                  846 non-null    int64
 1   circularity                  841 non-null    float64
 2   distance_circularity         842 non-null    float64
 3   radius_ratio                 840 non-null    float64
 4   pr.axis_aspect_ratio         844 non-null    float64
 5   max.length_aspect_ratio      846 non-null    int64
 6   scatter_ratio                845 non-null    float64
 7   elongatedness                845 non-null    float64
 8   pr.axis_rectangularity       843 non-null    float64
 9   max.length_rectangularity    846 non-null    int64
 10  scaled_variance              843 non-null    float64
 11  scaled_variance.1            844 non-null    float64
 12  scaled_radius_of_gyration    844 non-null    float64
 13  scaled_radius_of_gyration.1  842 non-null    float64
 14  skewness_about               840 non-null    float64
 15  skewness_about.1             845 non-null    float64
 16  skewness_about.2             845 non-null    float64
 17  hollows_ratio                846 non-null    int64
 18  class                        846 non-null    object
dtypes: float64(14), int64(4), object(1)
memory usage: 125.7+ KB
```

1 B. Check percentage of missing values and impute with correct approach. [1 Marks]

In [100…

```
def missing_values_vehicle_df (df):
    total=df.isnull().sum().sort_values()
    percent=((total/df.isnull().count().sum()).sort_values())*100
    missing_value_df=pd.concat([total,percent],keys=['Total','percent'],axis=1)
    return (missing_value_df)
```

In [101…

```
missing_values_vehicle_df(vehicle_df)
```

Out[101]:

	Total	percent
compactness	0	0.000000
hollows_ratio	0	0.000000
max.length_aspect_ratio	0	0.000000
max.length_rectangularity	0	0.000000
class	0	0.000000
scatter_ratio	1	0.006221
elongatedness	1	0.006221
skewness_about.1	1	0.006221
skewness_about.2	1	0.006221
pr.axis_aspect_ratio	2	0.012442
scaled_variance.1	2	0.012442
scaled_radius_of_gyration	2	0.012442
pr.axis_rectangularity	3	0.018664
scaled_variance	3	0.018664
distance_circularity	4	0.024885
scaled_radius_of_gyration.1	4	0.024885
circularity	5	0.031106
skewness_about	6	0.037327
radius_ratio	6	0.037327

In [102…

```
missing_values_cols= vehicle_df.columns[vehicle_df.isnull().any()]
vehicle_df[missing_values_cols].isnull().sum()
vehicle_df[vehicle_df.isnull().any(axis=1)][missing_values_cols].shape
```

Out[102]:

```
(33, 14)
```

- it shows that it has total 33 rows that has missing values from 14 columns. which has to be treated

In [103…
```
# Label encoder
from sklearn.preprocessing import LabelEncoder
#Label encode the target calss
labelencoder = LabelEncoder()
vehicle_df['class']=labelencoder.fit_transform(vehicle_df['class'])
vehicle_df['class'].value_counts()
```

Out[103]:
```
class
1    429
0    218
2    199
Name: count, dtype: int64
```

Missing Values Treatment:¶

Find individual row with missing values in each of the columns and then we will make decision on whether to drop or not

Missing Treatment Values for 'circularity'

In [104…
```
vehicle_df[vehicle_df['circularity'].isnull()][missing_values_cols]
```

Out[104]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
5	NaN	106.0	172.0	50.0	255.0	26.0	28.0	280.0	957.(
105	NaN	103.0	202.0	64.0	220.0	30.0	25.0	NaN	711.(
118	NaN	NaN	128.0	56.0	150.0	46.0	19.0	168.0	324.(
266	NaN	65.0	116.0	53.0	152.0	45.0	19.0	175.0	335.(
396	NaN	106.0	177.0	51.0	256.0	26.0	28.0	285.0	966.(

- 5 rows have missing vales for circularity. one of the 5 rows alsq has missing value for distance_circularity.
- Another row has missing values for scaled valiance and skewness_about.1. One of the row has missing value for scaled_radius_of_gyration.

- We will drop those rows which has missing value in any other coulmn as well apart from circularity which is 3.
- will impute missing value in rest 2 rows.

In [105…
```
# Row 105,118,266 has missing value in more than 1 colum.drop those
vehicle_df.drop([105,118,266],inplace=True)
```

In [106…
```
# Now lets Check the class level of remaining 2 rows- we will replace the value with median value of the corresponding class
vehicle_df.loc[5].loc['class'],vehicle_df.loc[396].loc['class']
```

Out[106]:
```
(0.0, 0.0)
```

In [107…
```
# Belong to Bus Class
Median_circularity_bus=vehicle_df['circularity'][vehicle_df['class']==0].median()
Median_circularity_bus
```

Out[107]:
```
44.0
```

In [108…
```
vehicle_df['circularity'].fillna(Median_circularity_bus, inplace=True)
```

In [109…
```
# Douple Check if missing values have been teated for curcularity
vehicle_df[vehicle_df['circularity'].isnull()][missing_values_cols]
```

Out[109]:

circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing values for circularity has been treated successfully

Missing Treatment Values for distance_circularity

In [110…
```
vehicle_df[vehicle_df['distance_circularity'].isnull()][missing_values_cols]
```

Out[110]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
35	46.0	NaN	172.0	67.0	157.0	43.0	20.0	170.0	363.(
207	42.0	NaN	121.0	55.0	149.0	46.0	19.0	167.0	323.(
319	51.0	NaN	194.0	60.0	220.0	30.0	25.0	247.0	731.(

In [111…
```
# 3 rows have missing values. row 207 has missing  values in more than 1 column- we will drop this
# row 35, 319 have missing values in just one column, We will fill it woth median of the corresponding class
```

In [112…
```
vehicle_df.drop(207, inplace=True)
```

In [113…
```
vehicle_df.shape
```

Out[113]:
```
(842, 19)
```

In [114…
```
# Now lets Check the class lavel of remeining 2 rows- we will replace the value with median value of the corresponding class
vehicle_df.loc[35].loc['class'],vehicle_df.loc[319].loc['class']
```

Out[114]:
```
(2.0, 0.0)
```

In [115…
```
Median_distance_circularity_van=vehicle_df['distance_circularity'][vehicle_df['class']==2].median()
Median_distance_circularity_bus=vehicle_df['distance_circularity'][vehicle_df['class']==0].median()
Median_distance_circularity_van,Median_distance_circularity_bus
```

Out[115]:
```
(75.0, 72.5)
```

In [116…
```
vehicle_df.loc[35]=vehicle_df.loc[35].replace(np.nan,Median_distance_circularity_van)
vehicle_df.loc[319]=vehicle_df.loc[319].replace(np.nan,Median_distance_circularity_bus)
```

In [117…
```
vehicle_df.loc[[35,319]]
```

Out[117]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangu
35	100	46.0	75.0	172.0	67.0	9	157.0	43.0	
319	102	51.0	72.5	194.0	60.0	6	220.0	30.0	

In [118…

```
# double check if missing values have been handled for 'distance cicularity'
vehicle_df[vehicle_df['distance_circularity'].isnull()][missing_values_cols]
```

Out[118]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing values handled for dian_distance_circularity is sucessfull

Missing Treatment Values for radius_ratio

In [119…

```
vehicle_df[vehicle_df['radius_ratio'].isnull()][missing_values_cols]
```

Out[119]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
9	44.0	98.0	NaN	62.0	183.0	36.0	22.0	202.0	505.(
78	52.0	94.0	NaN	66.0	208.0	31.0	24.0	227.0	666.(
159	45.0	75.0	NaN	57.0	150.0	44.0	19.0	170.0	335.(
287	43.0	84.0	NaN	55.0	154.0	44.0	19.0	174.0	350.(
345	54.0	106.0	NaN	57.0	236.0	28.0	26.0	256.0	833.(
467	54.0	104.0	NaN	58.0	215.0	31.0	24.0	221.0	682.(

In [120…

```
# For all the rows with missing radius_ratio only radius ratio is having missing values all the other columns have values.
# We will not drop any rather replace with median of corresponding class.
```

In [121…

```
vehicle_df.loc[[9,78,159,287,345,467]]['class']
```

Out[121]:
```
9      1
78     0
159    1
287    2
345    0
467    1
Name: class, dtype: int32
```

In [122…
```
# Lets find median value for car, bus,van
Median_distance_radius_ratio_van=vehicle_df['radius_ratio'][vehicle_df['class']==2].median()
Median_distance_radius_ratio_bus=vehicle_df['radius_ratio'][vehicle_df['class']==0].median()
Median_distance_radius_ratio_car=vehicle_df['radius_ratio'][vehicle_df['class']==1].median()
Median_distance_radius_ratio_van,Median_distance_radius_ratio_bus,Median_distance_radius_ratio_car
```

Out[122]:
```
(144.0, 169.0, 186.0)
```

In [123…
```
# replace rows 9,159 and 467 with car median, 78,345 with bus median and 287 with  van
```

In [124…
```
vehicle_df.loc[[9,159,467]]=vehicle_df.loc[[9,159,467]].replace(np.nan,Median_distance_radius_ratio_car)
```

In [125…
```
vehicle_df.loc[[9,159,467]]
```

Out[125]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangu
9	93	44.0	98.0	186.0	62.0	11	183.0	36.0	
159	91	45.0	75.0	186.0	57.0	6	150.0	44.0	
467	96	54.0	104.0	186.0	58.0	10	215.0	31.0	

In [126…
```
vehicle_df.loc[[78,345 ]]=vehicle_df.loc[[ 78,345 ]].replace(np.nan,Median_distance_radius_ratio_bus)
```

In [127…
```
vehicle_df.loc[[78,345 ]]
```

Out[127]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangu
78	104	52.0	94.0	169.0	66.0	5	208.0	31.0	
345	101	54.0	106.0	169.0	57.0	7	236.0	28.0	

In [128…
```
vehicle_df.loc[287]=vehicle_df.loc[287].replace(np.nan,Median_distance_radius_ratio_van)
```

In [129…
```
vehicle_df.loc[[287]]
```

Out[129]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangu
287	88	43.0	84.0	144.0	55.0	11	154.0	44.0	

Missing treatment value for pr.axis_aspect_ratio

In [130…
```
vehicle_df[vehicle_df['pr.axis_aspect_ratio'].isnull()][missing_values_cols]
```

Out[130]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
19	56.0	100.0	215.0	NaN	208.0	32.0	24.0	227.0	651.(
222	50.0	81.0	197.0	NaN	186.0	34.0	22.0	206.0	531.(

In [131…
```
# There are 2 rows with missing values. One row has missing value in one more column in addityion to pr.axis_aspect_ratio
# We will drop that row but treat the missing value in pr.axis_aspect_ratio with median of corresponding class
```

In [132…
```
# drop row 222
vehicle_df.drop(222, inplace=True)
```

In [133…
```
vehicle_df.loc[19]['class']
```

Out[133]:
```
1.0
```

In [134…
```
Median_distance_pr_axis_aspect_ratio_car=vehicle_df['pr.axis_aspect_ratio'][vehicle_df['class']==1].median()
Median_distance_pr_axis_aspect_ratio_car
```

Out[134]:
```
61.0
```

In [135…
```
vehicle_df.loc[19]=vehicle_df.loc[19].replace(np.nan,Median_distance_pr_axis_aspect_ratio_car)
```

In [136…
```
vehicle_df[vehicle_df['pr.axis_aspect_ratio'].isnull()][missing_values_cols]
```

Out[136]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing Treatment values for scateer_ratio

In [137…
```
vehicle_df[vehicle_df['scatter_ratio'].isnull()][missing_values_cols]
```

Out[137]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
249	34.0	53.0	127.0	58.0	NaN	58.0	17.0	137.0	197.(

In [138…
```
# Only one row and 2 cols have missing value in that row including scatter_ratio
# we will drop this row
```

In [139…
```
vehicle_df.drop(249, inplace=True)
```

missing treatment valus for elongatednes

In [140…
```
vehicle_df[vehicle_df['elongatedness'].isnull()][missing_values_cols]
```

Out[140]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
215	39.0	86.0	169.0	62.0	162.0	NaN	20.0	194.0	388.(

In [141…
```
vehicle_df.loc[215]['class']
```

Out[141]:
```
1.0
```

In [142…
```
Median_distance_elongatedness_car=vehicle_df['elongatedness'][vehicle_df['class']==1].median()
Median_distance_elongatedness_car
```

Out[142]:
```
36.0
```

In [143…
```
vehicle_df.loc[215]=vehicle_df.loc[215].replace(np.nan,Median_distance_elongatedness_car)
```

In [144…
```
vehicle_df[vehicle_df['elongatedness'].isnull()][missing_values_cols]
```

Out[144]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

missing Treatment values for pr.axis_rectanularity

In [145…
```
vehicle_df[vehicle_df['pr.axis_rectangularity'].isnull()][missing_values_cols]
```

Out[145]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
70	55.0	98.0	161.0	54.0	215.0	31.0	NaN	226.0	683.(
237	45.0	65.0	128.0	56.0	151.0	45.0	NaN	170.0	332.(
273	45.0	80.0	162.0	63.0	146.0	46.0	NaN	161.0	316.(

In [146…
```
# 3 rows have missing values for pr.axis_rectangularity and only this column has missing value
# We will replace this with median value of the corresponding class
```

In [147…
```
#lets loom at class level of the missing rows
vehicle_df.loc[[70,237,273]]['class']
```

Out[147]:
```
70     1
237    0
273    2
Name: class, dtype: int32
```

In [148…
```
Median_distance_pr_axis_rectangularity_van=vehicle_df['pr.axis_rectangularity'][vehicle_df['class']==2].median()
Median_distance_pr_axis_rectangularity_car=vehicle_df['pr.axis_rectangularity'][vehicle_df['class']==1].median()
Median_distance_pr_axis_rectangularity_bus=vehicle_df['pr.axis_rectangularity'][vehicle_df['class']==0].median()
Median_distance_pr_axis_rectangularity_van,Median_distance_pr_axis_rectangularity_car,Median_distance_pr_axis_rectangularity_bus
```

Out[148]:
```
(18.0, 22.0, 19.0)
```

In [149…
```
vehicle_df.loc[70]=vehicle_df.loc[70].replace(np.nan,Median_distance_pr_axis_rectangularity_car)
vehicle_df.loc[237]=vehicle_df.loc[237].replace(np.nan,Median_distance_pr_axis_rectangularity_bus)
vehicle_df.loc[273]=vehicle_df.loc[273].replace(np.nan,Median_distance_pr_axis_rectangularity_van)
```

In [150…
```
vehicle_df[vehicle_df['pr.axis_rectangularity'].isnull()][missing_values_cols]
```

Out[150]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

missing values treatment for scaled variance

In [151…
```
vehicle_df[vehicle_df['scaled_variance'].isnull()][missing_values_cols]
```

Out[151]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
372	47.0	87.0	164.0	64.0	156.0	43.0	20.0	NaN	359.(
522	36.0	69.0	162.0	63.0	140.0	48.0	18.0	NaN	291.(

In [152…
```
# 2 rows have missing values for scaled_variance, no other columns have missing values for these rows. We will replace with media
# of corresponding class
```

In [153…
```
vehicle_df.loc[[372,522]]['class']
```

Out[153]:
```
372    2
522    1
Name: class, dtype: int32
```

In [154…
```
Median_distance_scaled_variance_van=vehicle_df['scaled_variance'][vehicle_df['class']==2].median()
Median_distance_scaled_variance_car=vehicle_df['scaled_variance'][vehicle_df['class']==1].median()
Median_distance_scaled_variance_van,Median_distance_scaled_variance_car
```

Out[154]:
```
(164.0, 206.5)
```

In [155…
```
vehicle_df.loc[372]=vehicle_df.loc[372].replace(np.nan,Median_distance_scaled_variance_van)
vehicle_df.loc[522]=vehicle_df.loc[522].replace(np.nan,Median_distance_scaled_variance_car)
```

In [156…
```
vehicle_df[vehicle_df['scaled_variance'].isnull()][missing_values_cols]
```

Out[156]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing Treatment values for scaled_variance.1

In [157…
```
vehicle_df[vehicle_df['scaled_variance.1'].isnull()][missing_values_cols]
```

Out[157]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
308	51.0	100.0	197.0	59.0	192.0	34.0	22.0	210.0	NaN
496	55.0	98.0	224.0	68.0	215.0	31.0	24.0	222.0	NaN

In [158…
```
# 2 rows have missing values for scaled_variance, no other columns have missing values for these rows. We will replace with media
# of corresponding class
```

In [159…
```
vehicle_df.loc[[308,496]]['class']
```

Out[159]:
```
308    1
496    1
Name: class, dtype: int32
```

In [160…
```
Median_distance_scaled_variance1_car=vehicle_df['scaled_variance.1'][vehicle_df['class']==1].median()
Median_distance_scaled_variance1_car
```

Out[160]:
```
512.0
```

In [161…
```
vehicle_df.loc[[308,496]]=vehicle_df.loc[[ 308,496]].replace(np.nan,Median_distance_scaled_variance1_car)
```

In [162…
```
vehicle_df[vehicle_df['scaled_variance.1'].isnull()][missing_values_cols]
```

Out[162]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing Treatment value for scaled_radius_of_gyration.1

In [163…
```
vehicle_df[vehicle_df['scaled_radius_of_gyration.1'].isnull()][missing_values_cols]
```

Out[163]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
66	43.0	68.0	125.0	57.0	149.0	46.0	19.0	169.0	323.(
77	40.0	62.0	140.0	62.0	150.0	45.0	19.0	165.0	330.(
192	43.0	76.0	149.0	57.0	149.0	44.0	19.0	172.0	335.(
329	38.0	80.0	169.0	59.0	161.0	41.0	20.0	186.0	389.(

In [164…
```
# there are 4  rows with scaled_radius_of_gyration.1 as missing values
# row with index 66 has missing values in 2 columns- will be dropped
# Other 3 rows missing values will be replaced with median value of cotresponding class
```

In [165…
```
# Drop row 66
vehicle_df.drop(66, inplace=True)
```

In [166…
```
vehicle_df.loc[[77,192,329]]['class']
```

Out[166]:
```
77     1
192    1
329    1
Name: class, dtype: int32
```

In [167…
```
Median_distance_radius_gyr1_car=vehicle_df['scaled_radius_of_gyration.1'][vehicle_df['class']==1].median()
Median_distance_radius_gyr1_car
```

Out[167]:
```
70.0
```

In [168…
```
vehicle_df.loc[[77,192,329]]=vehicle_df.loc[[ 77,192,329]].replace(np.nan,Median_distance_radius_gyr1_car)
```

In [169…
```
vehicle_df[vehicle_df['scaled_radius_of_gyration.1'].isnull()][missing_values_cols]
```

Out[169]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

missing Values Treatment for skewness_about

In [170…
```
vehicle_df[vehicle_df['skewness_about'].isnull()][missing_values_cols]
```

Out[170]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
141	42.0	63.0	125.0	55.0	149.0	46.0	19.0	166.0	320.(
177	44.0	72.0	160.0	66.0	144.0	46.0	19.0	166.0	312.(
285	48.0	85.0	189.0	64.0	169.0	39.0	20.0	188.0	427.(

In [171…
```
# 3 rows have missing values  in skewness_about column , no other column has missing value for these rows.
# we will replace these values with median of the corresponding class
```

In [172…
```
vehicle_df.loc[[141,177,285]]['class']
```

Out[172]:
```
141    0
177    0
285    1
Name: class, dtype: int32
```

In [173…
```
Median_distance_skewness_about_car=vehicle_df['skewness_about'][vehicle_df['class']==1].median()
Median_distance_skewness_about_bus=vehicle_df['skewness_about'][vehicle_df['class']==0].median()
Median_distance_skewness_about_car,Median_distance_skewness_about_bus
```

Out[173]: (6.0, 5.0)

In [174…
```
vehicle_df.loc[[141,177]]=vehicle_df.loc[[141,177]].replace(np.nan,Median_distance_skewness_about_bus)
```

In [175…
```
vehicle_df.loc[[285]]=vehicle_df.loc[[285]].replace(np.nan,Median_distance_skewness_about_car)
```

In [176…
```
vehicle_df[vehicle_df['skewness_about'].isnull()][missing_values_cols]
```

Out[176]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Missing values Treatment for skewness_about.1

In [177…
```
vehicle_df[vehicle_df['skewness_about.1'].isnull()][missing_values_cols]
```

Out[177]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

In [178…
```
#  No longer Missing values- corresponding row
#has been dropped while treating other missing values
```

Missing Values Treatment for skewness_about.2

In [179…
```
vehicle_df[vehicle_df['skewness_about.2'].isnull()][missing_values_cols]
```

Out[179]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1
419	34.0	72.0	144.0	56.0	133.0	50.0	18.0	158.0	263.(

In [180…
```
# One row has missing value for skewness_about.2 and no other value is missing for that row
# Lets replace that value with median of the corresponding class
```

In [181…
```
vehicle_df.loc[419]['class']
```

Out[181]:
```
1.0
```

In [182…
```
Median_distance_skewness_about2_car=vehicle_df['skewness_about.2'][vehicle_df['class']==1].median()
Median_distance_skewness_about2_car
```

Out[182]:
```
189.0
```

In [183…
```
vehicle_df.loc[[419]]=vehicle_df.loc[[419]].replace(np.nan,Median_distance_skewness_about2_car)
```

In [184…
```
vehicle_df[vehicle_df['skewness_about.2'].isnull()][missing_values_cols]
```

Out[184]:

	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularity	scaled_variance	scaled_variance.1	s

Data Frame Summary statistics after missing values treatment

In [185…
```
vehicle_df.describe().T
```

Out[185]:

	count	mean	std	min	25%	50%	75%	max
compactness	839.0	93.709178	8.218746	73.0	87.5	93.0	100.0	119.0
circularity	839.0	44.839094	6.144567	33.0	40.0	44.0	49.0	59.0
distance_circularity	839.0	82.138856	15.744684	40.0	70.0	80.0	98.0	112.0
radius_ratio	839.0	169.117998	33.346151	104.0	141.0	168.0	195.0	333.0
pr.axis_aspect_ratio	839.0	61.710369	7.900381	47.0	57.0	61.0	65.0	138.0
max.length_aspect_ratio	839.0	8.578069	4.617162	2.0	7.0	8.0	10.0	55.0
scatter_ratio	839.0	168.910608	33.255794	112.0	146.0	157.0	198.0	265.0
elongatedness	839.0	40.905840	7.803796	26.0	33.0	43.0	46.0	61.0
pr.axis_rectangularity	839.0	20.584029	2.591483	17.0	19.0	20.0	23.0	29.0
max.length_rectangularity	839.0	148.013111	14.522752	118.0	137.0	146.0	159.5	188.0
scaled_variance	839.0	188.753874	31.419128	130.0	167.0	179.0	217.0	320.0
scaled_variance.1	839.0	440.063170	176.579093	184.0	318.0	365.0	587.0	1018.0
scaled_radius_of_gyration	839.0	174.697259	32.601944	109.0	149.0	174.0	198.0	268.0
scaled_radius_of_gyration.1	839.0	72.398093	7.467754	59.0	67.0	71.0	75.0	135.0
skewness_about	839.0	6.359952	4.916886	0.0	2.0	6.0	9.0	22.0
skewness_about.1	839.0	12.617402	8.945485	0.0	5.0	11.0	19.0	41.0
skewness_about.2	839.0	188.961859	6.133439	176.0	184.0	189.0	193.0	206.0
hollows_ratio	839.0	195.692491	7.415286	181.0	191.0	197.0	201.0	211.0
class	839.0	0.983313	0.700976	0.0	0.0	1.0	1.0	2.0

In [186…

```
vehicle_df.isnull().sum()
```

Out[186]:
```
compactness                  0
circularity                  0
distance_circularity         0
radius_ratio                 0
pr.axis_aspect_ratio         0
max.length_aspect_ratio      0
scatter_ratio                0
elongatedness                0
pr.axis_rectangularity       0
max.length_rectangularity    0
scaled_variance              0
scaled_variance.1            0
scaled_radius_of_gyration    0
scaled_radius_of_gyration.1  0
skewness_about               0
skewness_about.1             0
skewness_about.2             0
hollows_ratio                0
class                        0
dtype: int64
```

All missing values are treated succefully.

1 C. Visualize a Pie-chart and print percentage of values for variable ‘class’. [2 Marks]

In [187…
```
df6=vehicle_df['class'].value_counts()
df6
```

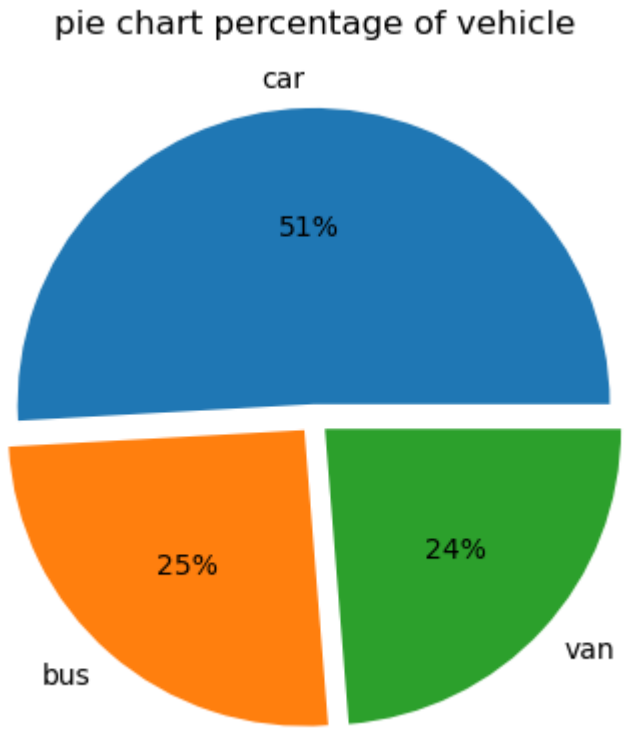
Out[187]:
```
class
1    427
0    213
2    199
Name: count, dtype: int64
```

In [188…
```
keys=['car','bus','van']
```

In [189…
```
vehicle_df.columns
```

Out[189]:
```
Index(['compactness', 'circularity', 'distance_circularity', 'radius_ratio',
       'pr.axis_aspect_ratio', 'max.length_aspect_ratio', 'scatter_ratio',
       'elongatedness', 'pr.axis_rectangularity', 'max.length_rectangularity',
       'scaled_variance', 'scaled_variance.1', 'scaled_radius_of_gyration',
       'scaled_radius_of_gyration.1', 'skewness_about', 'skewness_about.1',
       'skewness_about.2', 'hollows_ratio', 'class'],
      dtype='object')
```

In [190…
```
explode=[0.05,0.05,0.05]
plt.pie(df6,labels=keys, autopct='%.0f%%',explode=explode)
plt.title("pie chart percentage of vehicle")
plt.show()
```



insight:

- data has observations for

- from the data there are 25% bus
- from data there are 51% car
- from data there are 24% van

1 D. Check for duplicate rows in the data and impute with correct approach. [1 Marks]

In [191…
```
vehicle_df.iloc[:].duplicated().value_counts()
```

Out[191]:
```
False    839
Name: count, dtype: int64
```

In [192…
```
duplicateRows = vehicle_df[vehicle_df.duplicated()]
duplicateRows
```

Out[192]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangularit

no duplicate values are present in data

2. Data Preparation: [2 Marks]

2 A. Split data into X and Y. [Train and Test optional] [1 Marks]

splitting and also let treat ouliner

In [193…
```
vehicle_df.shape
```

Out[193]:
```
(839, 19)
```

In [194…
```
vehicle_df.columns.unique()
```

Out[194]:
```
Index(['compactness', 'circularity', 'distance_circularity', 'radius_ratio',
       'pr.axis_aspect_ratio', 'max.length_aspect_ratio', 'scatter_ratio',
       'elongatedness', 'pr.axis_rectangularity', 'max.length_rectangularity',
       'scaled_variance', 'scaled_variance.1', 'scaled_radius_of_gyration',
       'scaled_radius_of_gyration.1', 'skewness_about', 'skewness_about.1',
       'skewness_about.2', 'hollows_ratio', 'class'],
      dtype='object')
```

In [195…
```
vehicle_df.head()
```

Out[195]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ratio	scatter_ratio	elongatedness	pr.axis_rectangular
0	95	48.0	83.0	178.0	72.0	10	162.0	42.0	2
1	91	41.0	84.0	141.0	57.0	9	149.0	45.0	1
2	104	50.0	106.0	209.0	66.0	10	207.0	32.0	2
3	93	41.0	82.0	159.0	63.0	9	144.0	46.0	1
4	85	44.0	70.0	205.0	103.0	52	149.0	45.0	1

- here we will not do any data treatment for outlier because it may tend to loose some vital information and data may become overfitted

In [196…
```
# we separate the target variable (close ) and save it in the y variable
# also X contain independent variables

X=vehicle_df.iloc[:,0:18].values
y=vehicle_df.iloc[:,18].values
```

In [197…
```
#Splitting data into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.2,random_state=10)
```

2 B. Standardize the Data. [1 Marks]

In [198…
```
from sklearn.preprocessing import StandardScaler
scaler =StandardScaler()
X_train_sd =scaler.fit_transform(X_train)
X_test_sd=scaler.fit_transform(X_test)
```

In [199…
```
X_train_sd
```

Out[199]:
```
array([[-1.03990585, -0.43651283, -0.99038017, ..., -0.95569408,
        -1.45364507, -1.84579066],
       [-1.1620229 , -0.92278861,  0.5167384 , ...,  2.79249308,
         0.17147296,  0.30466062],
       [ 0.0591476 , -1.24697246, -0.61360053, ...,  0.14671391,
         1.796591  ,  1.11107984],
       ...,
       [-0.4293206 , -0.92278861, -1.55554963, ..., -0.51473088,
        -0.96610966, -1.84579066],
       [ 0.42549875,  0.04976295,  0.39114518, ..., -0.07376769,
         0.49649657,  0.57346703],
       [-0.30720355, -0.27442091,  0.39114518, ...,  1.4696035 ,
         1.30905559,  1.51428946]])
```

3. Model Building: [13 Marks]

3 A. Train a base Classification model using SVM. [1 Marks]

In [200…
```
from sklearn.svm import SVC
clf=SVC()
clf.fit(X_train_sd,y_train)
print('Before PCA Score on train data',clf.score(X_test_sd,y_test))
```

```
Before PCA Score on train data 0.9404761904761905
```

3 B. Print Classification metrics for train data. [1 Marks]

In [201…
```
cov_matrix=np.cov(X_train_sd.T)
print('cobariance matrix \n%s',cov_matrix)
```

```
cobariance matrix
%s [[ 1.00149254  0.68199023  0.78598666  0.67241601  0.06255222  0.14902094
   0.80990539 -0.78662321  0.81069109  0.68047984  0.75570365  0.81482178
   0.57871489 -0.25920701  0.22099533  0.16219987  0.30354905  0.3751529 ]
 [ 0.68199023  1.00149254  0.7970607   0.61342612  0.13061489  0.26735336
   0.85082014 -0.82196625  0.84477612  0.96604921  0.78993299  0.841309
   0.92768937  0.03243     0.13884032 -0.02078035 -0.10294392  0.05872527]
 [ 0.78598666  0.7970607   1.00149254  0.76315081  0.1379579   0.26603854
   0.9076308  -0.91301961  0.89443051  0.78335892  0.86108202  0.89052669
   0.70892588 -0.23652908  0.09129505  0.26742694  0.1512049   0.34247834]
 [ 0.67241601  0.61342612  0.76315081  1.00149254  0.65794874  0.4842287
   0.72602263 -0.7812329   0.69884848  0.57022108  0.80020158  0.71197786
   0.52606702 -0.15282361  0.02441252  0.18985323  0.37772359  0.47717808]
 [ 0.06255222  0.13061489  0.1379579   0.65794874  1.00149254  0.68234892
   0.07922202 -0.15746823  0.05281961  0.10590495  0.27753578  0.06480941
   0.10025493  0.21123438 -0.08215465 -0.02816693  0.22365788  0.25441289]
 [ 0.14902094  0.26735336  0.26603854  0.4842287   0.68234892  1.00149254
   0.17587687 -0.18822456  0.16787235  0.30829489  0.36086025  0.15549379
   0.2128645   0.3546438  -0.00451068  0.02275043 -0.02929582  0.12911191]
 [ 0.80990539  0.85082014  0.9076308   0.72602263  0.07922202  0.17587687
   1.00149254 -0.97466787  0.99312322  0.82151528  0.94460817  0.99784558
   0.79859496 -0.04545376  0.06302426  0.21228633  0.0082374   0.13236012]
 [-0.78662321 -0.82196625 -0.91301961 -0.7812329  -0.15746823 -0.18822456
  -0.97466787  1.00149254 -0.95143574 -0.78373833 -0.93373556 -0.95768403
  -0.76346935  0.11666742 -0.03331472 -0.18990721 -0.11434209 -0.22556675]
 [ 0.81069109  0.84477612  0.89443051  0.69884848  0.05281961  0.16787235
   0.99312322 -0.95143574  1.00149254  0.82150175  0.92928361  0.99374222
   0.79323747 -0.03656163  0.07515661  0.21256953 -0.01469137  0.1121825 ]
 [ 0.68047984  0.96604921  0.78335892  0.57022108  0.10590495  0.30829489
   0.82151528 -0.78373833  0.82150175  1.00149254  0.75321607  0.81055926
   0.8723586   0.019634    0.11955997 -0.00334929 -0.09489883  0.09097041]
 [ 0.75570365  0.78993299  0.86108202  0.80020158  0.27753578  0.36086025
   0.94460817 -0.93373556  0.92928361  0.75321607  1.00149254  0.94162818
   0.77127262  0.11303906  0.02641209  0.20284116  0.02035601  0.10440164]
 [ 0.81482178  0.841309    0.89052669  0.71197786  0.06480941  0.15549379
   0.99784558 -0.95768403  0.99374222  0.81055926  0.94162818  1.00149254
   0.79508241 -0.03757458  0.06819956  0.20632426  0.00916865  0.11930221]
 [ 0.57871489  0.92768937  0.70892588  0.52606702  0.10025493  0.2128645
   0.79859496 -0.76346935  0.79323747  0.8723586   0.77127262  0.79508241
   1.00149254  0.17217047  0.17256544 -0.06595561 -0.2274899  -0.10985808]
 [-0.25920701  0.03243    -0.23652908 -0.15282361  0.21123438  0.3546438
  -0.04545376  0.11666742 -0.03656163  0.019634    0.11303906 -0.03757458
   0.17217047  1.00149254 -0.06777877 -0.14287133 -0.73328523 -0.78268706]
 [ 0.22099533  0.13884032  0.09129505  0.02441252 -0.08215465 -0.00451068
```

```
  0.06302426 -0.03331472  0.07515661  0.11955997  0.02641209  0.06819956
  0.17256544 -0.06777877  1.00149254 -0.02319522  0.08448609  0.06335045]
[ 0.16219987 -0.02078035  0.26742694  0.18985323 -0.02816693  0.02275043
  0.21228633 -0.18990721  0.21256953 -0.00334929  0.20284116  0.20632426
 -0.06595561 -0.14287133 -0.02319522  1.00149254  0.11823539  0.2325512 ]
[ 0.30354905 -0.10294392  0.1512049   0.37772359  0.22365788 -0.02929582
  0.0082374  -0.11434209 -0.01469137 -0.09489883  0.02035601  0.00916865
 -0.2274899  -0.73328523  0.08448609  0.11823539  1.00149254  0.89601962]
[ 0.3751529   0.05872527  0.34247834  0.47717808  0.25441289  0.12911191
  0.13236012 -0.22556675  0.1121825   0.09097041  0.10440164  0.11930221
 -0.10985808 -0.78268706  0.06335045  0.2325512   0.89601962  1.00149254]]
```

In [202…

```
# generating the eigen values and the eigen vectors
```

In [203…

```
e_vals,e_vecs = np.linalg.eig(cov_matrix)
print("eigenvecter \n%s"%e_vecs)
print("eigenvalues \n%s"%e_vals)
```

```
eigenvecter
[[ 2.73855842e-01  1.30434067e-01  1.12081052e-01 -8.67246700e-02
  -7.20343855e-02  1.90454202e-01  5.06729625e-01  4.94950291e-01
   4.94683881e-01  2.72027254e-01 -9.77706964e-03 -2.14151580e-02
  -1.41957596e-01 -1.70366344e-03 -3.84027644e-03  3.88011926e-02
   5.89607539e-02  1.66571072e-02]
 [ 2.93213109e-01 -1.28084188e-01  3.90053044e-02 -1.89608629e-01
   1.01100042e-01 -3.17644620e-01 -2.00165126e-01  1.99264014e-01
   3.79623021e-02 -1.07147577e-01 -3.07766893e-02  1.84261357e-01
  -9.30871409e-02  4.62954630e-03 -7.77495321e-02  4.38281164e-01
  -3.64271317e-01 -5.30960750e-01]
 [ 3.05087490e-01  7.49686947e-02  5.06458066e-02  6.73641890e-02
  -2.50636745e-02 -1.37046727e-01  4.97357739e-02 -4.54818129e-01
   1.57656220e-01  2.81017780e-01 -6.77238220e-01 -2.73872733e-02
   2.32613490e-01 -7.90578975e-03  2.81645801e-02 -5.12498639e-02
   1.24604131e-01 -1.70615213e-01]
 [ 2.65599763e-01  1.76871238e-01 -2.93270942e-01  3.14244911e-02
   1.85950130e-02  2.32213842e-01 -2.18432662e-01 -9.88142760e-02
   2.29154762e-01  3.86147129e-02  1.26473908e-01  1.53333460e-01
   6.36028656e-02 -2.75086491e-02  4.48882112e-03 -4.88095465e-01
  -6.01460446e-01  3.32866890e-02]
 [ 7.30500925e-02  9.00123184e-02 -6.39991890e-01 -4.91748591e-02
   1.85752908e-02  1.89925761e-01 -4.01998029e-01  1.04003991e-01
   2.96863916e-01 -1.02150449e-01 -5.26493234e-02 -9.66068232e-02
   2.92411334e-02  2.44779954e-02  7.22244625e-03  2.72149626e-01
   4.23550232e-01  9.12540699e-03]
 [ 1.01691438e-01 -4.49460152e-02 -5.79490328e-01 -6.01192670e-02
  -1.62751454e-01 -3.92491108e-01  5.16345316e-01 -2.19953546e-01
  -1.57297010e-01  8.61953619e-02  2.78454666e-01  1.53454988e-01
  -5.32035034e-02 -1.14295962e-02  3.20851496e-02 -4.03187870e-02
   6.90244200e-02 -9.51203592e-02]
 [ 3.17346113e-01 -4.40734096e-02  9.71392160e-02  1.01540921e-01
   9.35658534e-03  1.26753395e-01  3.80151433e-02 -9.78959539e-02
  -6.61426674e-02 -1.85918997e-01  1.39544121e-01 -9.18300787e-02
  -3.86122651e-02  7.96985091e-01  3.64414920e-01 -2.01590989e-02
   7.91940737e-02 -8.61460262e-02]
 [-3.14049574e-01 -1.42782245e-02 -5.53520904e-02 -9.06972702e-02
  -6.91216651e-02 -1.60617629e-01  5.68440076e-02  1.97917438e-01
   1.54173252e-01  1.13772056e-01  8.26978575e-02 -1.12591501e-01
   8.14553416e-01  2.21897368e-01  8.38453475e-02  8.98202938e-02
  -1.80421444e-01 -1.69168410e-02]
 [ 3.13897013e-01 -5.49197353e-02  1.11619351e-01  9.76300288e-02
  -6.87135078e-03  1.13755037e-01  7.83395671e-02 -7.41704187e-02
  -1.89465456e-02 -2.41258713e-01  2.56403700e-01 -1.99127636e-01
```

```
  2.89376021e-01 -1.56930524e-02 -7.12092427e-01 -1.30860414e-01
  2.02328979e-01 -1.96862821e-01]
[ 2.85216584e-01 -1.16286226e-01  3.53849194e-02 -1.80767375e-01
  9.55044151e-02 -4.50972558e-01 -3.53410132e-02  2.53401519e-01
  7.18648968e-02 -4.43698110e-01 -1.97584727e-01  1.50355551e-01
  1.07681410e-01 -2.45891408e-02  4.25664183e-02 -2.69163616e-01
  1.18522392e-01  4.78595764e-01]
[ 3.08898901e-01 -6.08956345e-02 -7.37373689e-02  1.25763666e-01
 -2.65760932e-02  2.38701970e-01  8.37860444e-02 -4.53811082e-02
 -3.08496312e-01  1.35511783e-01 -6.40413431e-02  1.71205352e-01
  1.57161028e-01  2.97988439e-02 -1.56705738e-01  5.23266050e-01
 -2.30066890e-01  5.37611998e-01]
[ 3.14684769e-01 -4.79365837e-02  1.08141391e-01  9.89721784e-02
  6.70077185e-03  1.62173945e-01  5.42884003e-02 -3.55913671e-02
 -8.14958967e-02 -1.82795284e-01  2.27161781e-01 -1.42640434e-01
  2.75245121e-01 -5.57226239e-01  5.63984112e-01  7.80416297e-02
  8.30670603e-02 -1.31499540e-01]
[ 2.71089386e-01 -2.14081370e-01  4.80053568e-02 -2.00143184e-01
  5.99765595e-02 -1.68196704e-01 -3.56590866e-01  1.62870583e-01
 -2.37563285e-01  6.57618046e-01  2.31651940e-01 -1.92278871e-01
 -1.71649134e-03  1.20532283e-02  1.66044617e-02 -1.92717335e-01
  1.73820595e-01  7.37401690e-02]
[-2.44669049e-02 -4.88168916e-01 -2.85371098e-01  8.19675190e-02
 -1.66934200e-01  2.21061002e-01  1.27441427e-01  3.38829350e-01
 -2.97764746e-01 -9.31407882e-02 -4.18698016e-01 -3.10593535e-01
 -3.78087454e-02 -8.11985576e-03  2.59075883e-03 -2.00619817e-01
 -1.69021973e-01 -1.60378288e-01]
[ 3.59042839e-02  3.15297716e-02  1.25712247e-01 -6.12748470e-01
 -7.31484909e-01  1.55750554e-01 -9.42674462e-02 -1.55486225e-01
 -2.51155403e-02 -1.08039249e-01 -4.87527679e-03  3.21852243e-03
 -1.38034368e-02 -3.53707154e-03  1.55770513e-03  2.27237829e-02
 -5.59867776e-03  3.11639288e-02]
[ 5.92570037e-02  1.54860100e-01  5.59987831e-02  6.45234366e-01
 -6.05503705e-01 -2.74031244e-01 -2.19255650e-01  2.26111829e-01
  1.30669322e-02  3.77923984e-02  3.46872943e-02  8.33851665e-02
 -3.45645211e-02 -8.12982556e-03  6.50085192e-04 -3.16455857e-03
  3.06403809e-02 -1.22830639e-02]
[ 3.16109214e-02  5.43032880e-01 -4.02344944e-02 -1.20993372e-01
  8.26563014e-02  1.58910048e-01  2.34752205e-02  3.17307051e-01
 -5.08289031e-01  6.92940694e-03 -1.56858625e-01  3.64309085e-01
  1.68899269e-01  3.64088176e-02  7.17325885e-03 -1.52521654e-01
  1.84319975e-01 -2.27302197e-01]
[ 7.86229615e-02  5.39317988e-01 -6.55963669e-02 -7.35460641e-02
  4.46850534e-02 -2.38288006e-01  4.26496656e-02  1.96145575e-02
```

```
  -1.54499294e-01 -8.68756048e-02 -4.89989157e-02 -7.08728984e-01
  -1.31212611e-01 -1.21483571e-02 -1.89197435e-02  1.03568248e-01
  -2.34573804e-01  1.12571618e-01]]
eigenvalues
[9.40156057e+00 2.98053621e+00 2.01672486e+00 1.16303743e+00
 9.26898619e-01 5.09202110e-01 3.46158297e-01 2.17888451e-01
 1.56915892e-01 9.19487826e-02 6.30278114e-02 4.55670994e-02
 3.56908215e-02 3.93931926e-04 6.76169783e-03 1.61294975e-02
 2.63755438e-02 2.20480420e-02]
```

In [ ]:

3 C. Apply PCA on the data with 10 components. [3 Marks]

In [204…

```
from sklearn.decomposition import PCA
pca=PCA(n_components=10,random_state=1)
X_train_pca=pca.fit_transform(X_train_sd)
X_test_pca=pca.transform(X_test_sd)
```

In [205…

```
X_train_pca
```

Out[205]:

```
array([[-2.5263488 ,  3.02644998, -0.52775023, ...,  0.02021522,
         0.35770967,  0.27905095],
       [-0.20143868, -1.08754436, -0.16106677, ...,  0.88583509,
         0.67947539,  0.09334759],
       [-1.27773546, -2.59409881,  1.21980383, ..., -0.55442483,
         0.07829334,  0.34986176],
       ...,
       [-3.67766156,  2.42789732, -1.01968664, ..., -0.52644324,
         0.24552596,  0.11205707],
       [-0.03406317, -1.17890515,  0.45960172, ..., -0.10227303,
        -0.72600468,  0.30803013],
       [-0.95105203, -2.48278878,  0.01109797, ..., -0.44312495,
         0.0465501 ,  0.43366734]])
```

3 D. Visualize Cumulative Variance Explained with Number of Components. [2 Marks]

In [206…

```
pca.explained_variance_
```

Out[206]:

```
array([9.40156057, 2.98053621, 2.01672486, 1.16303743, 0.92689862,
       0.50920211, 0.3461583 , 0.21788845, 0.15691589, 0.09194878])
```

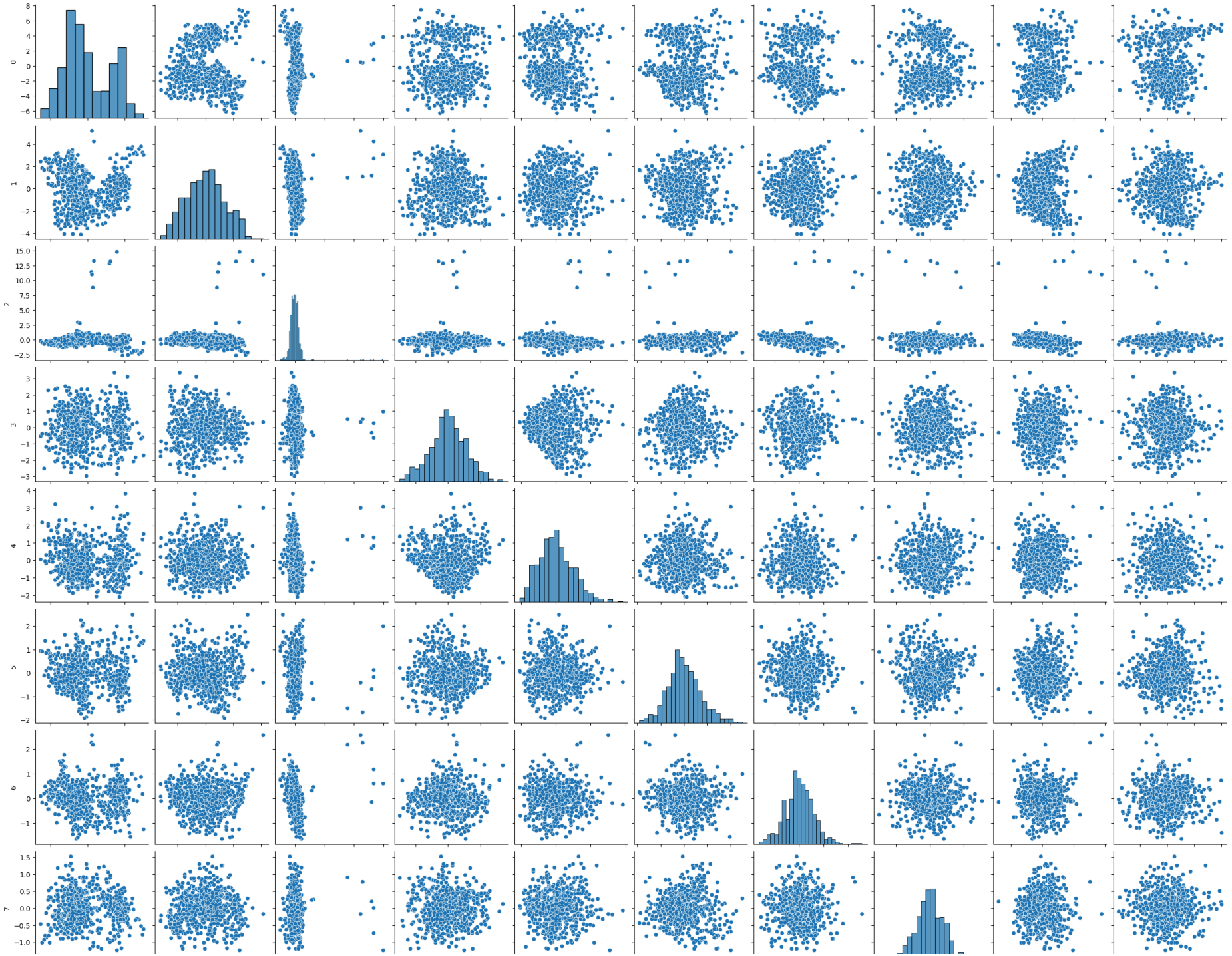
In [207…
```
total = pca.explained_variance_.sum()
total
```

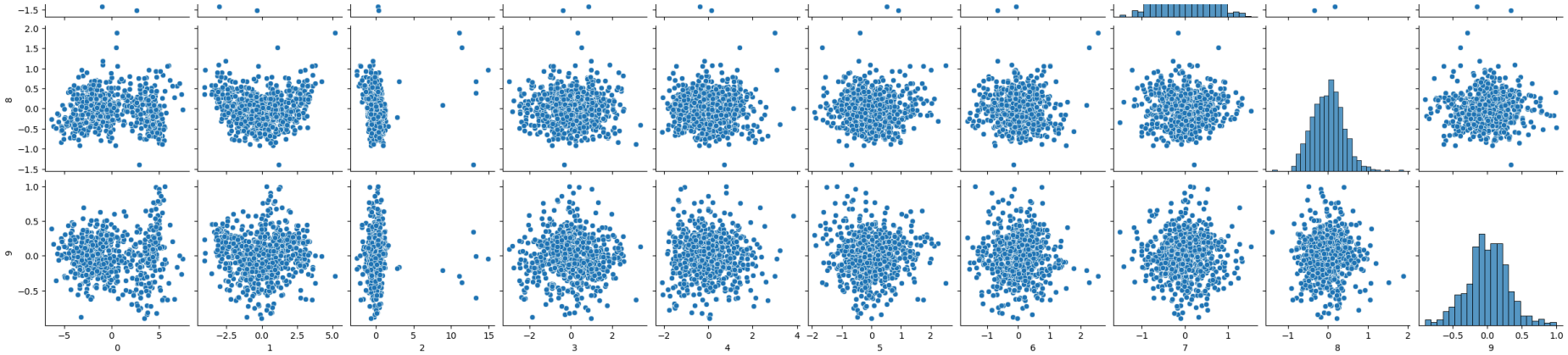
Out[207]:
```
17.810871226260183
```

In [208…
```
pca.explained_variance_/total
```

Out[208]:
```
array([0.52785518, 0.16734365, 0.11322999, 0.0652993 , 0.05204117,
       0.0285894 , 0.01943523, 0.01223345, 0.00881012, 0.00516251])
```

In [209…
```
data=pd.DataFrame(X_train_pca)
sns.pairplot(data)
plt.show()
```


0
1
2
3
4
5
6
7



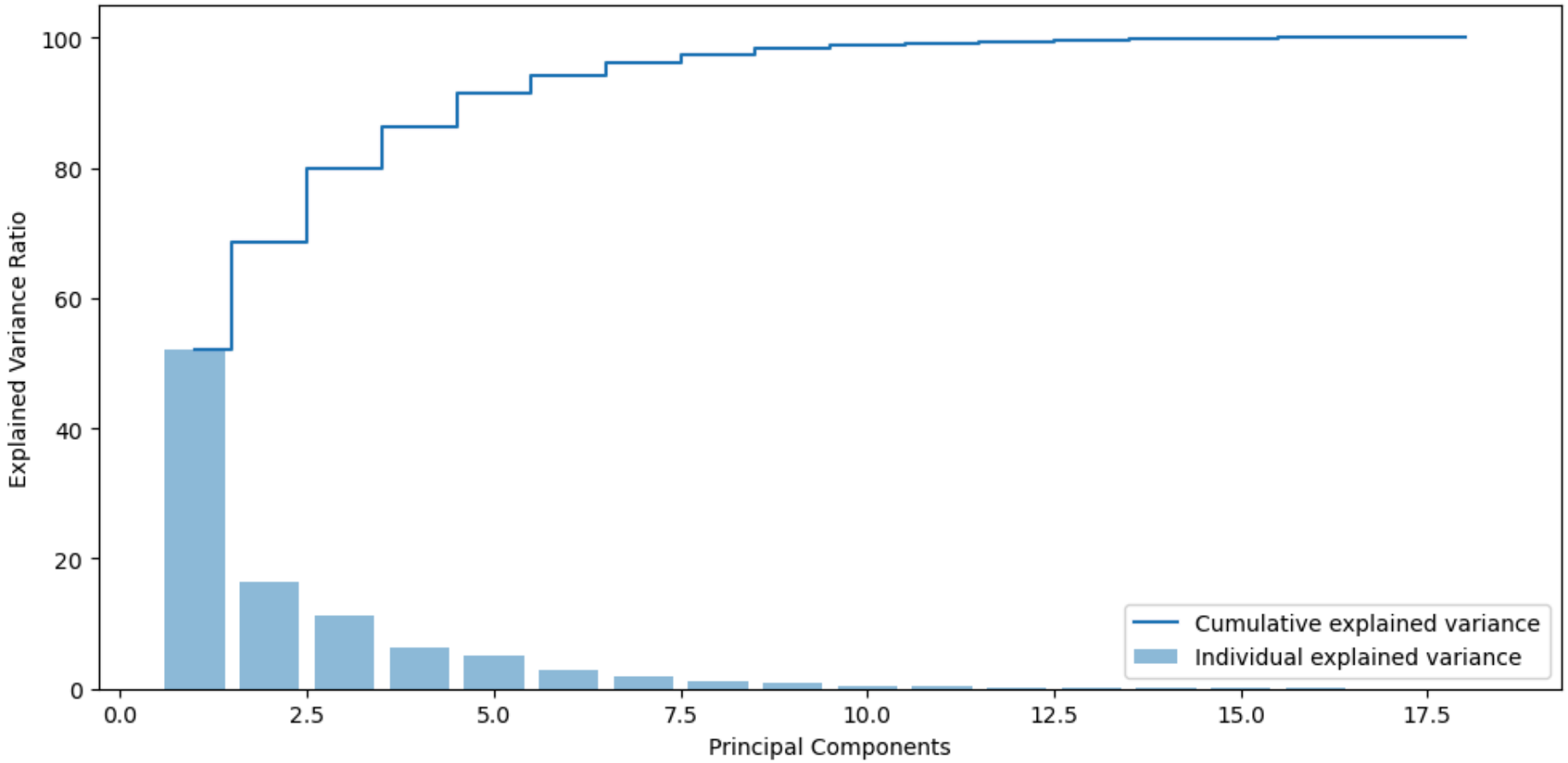
In [210…
```
from sklearn.svm import SVC

clf = SVC()
clf.fit(X_train_pca, y_train)
print ('After PCA score', clf.score(X_test_pca, y_test))
```

```
After PCA score 0.9285714285714286
```

In [211…
```
# the "cumulative variance explained" analysis
tot = sum(e_vals)
var_exp = [( i /tot ) * 100 for i in sorted(e_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained", cum_var_exp)
```

```
Cumulative Variance Explained [ 52.15305171  68.68690883  79.87423831  86.32592797  91.46769047
  94.29237513  96.21261075  97.42129814  98.29175391  98.80181919
  99.15145186  99.40422514  99.60221197  99.7485244   99.87083097
  99.96030575  99.99781475 100.        ]
```

In [212…
```
# Plotting the variance expalained by the principal components and the cumulative variance explained.
plt.figure(figsize=(10 , 5))
plt.bar(range(1, e_vals.size + 1), var_exp, alpha = 0.5, align = 'center', label = 'Individual explained variance')
plt.step(range(1, e_vals.size + 1), cum_var_exp, where='mid', label = 'Cumulative explained variance')
#plt.axvline(x = 5, color = 'b', label = 'choosing 90%')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc = 'best')
plt.tight_layout()
plt.show()
```

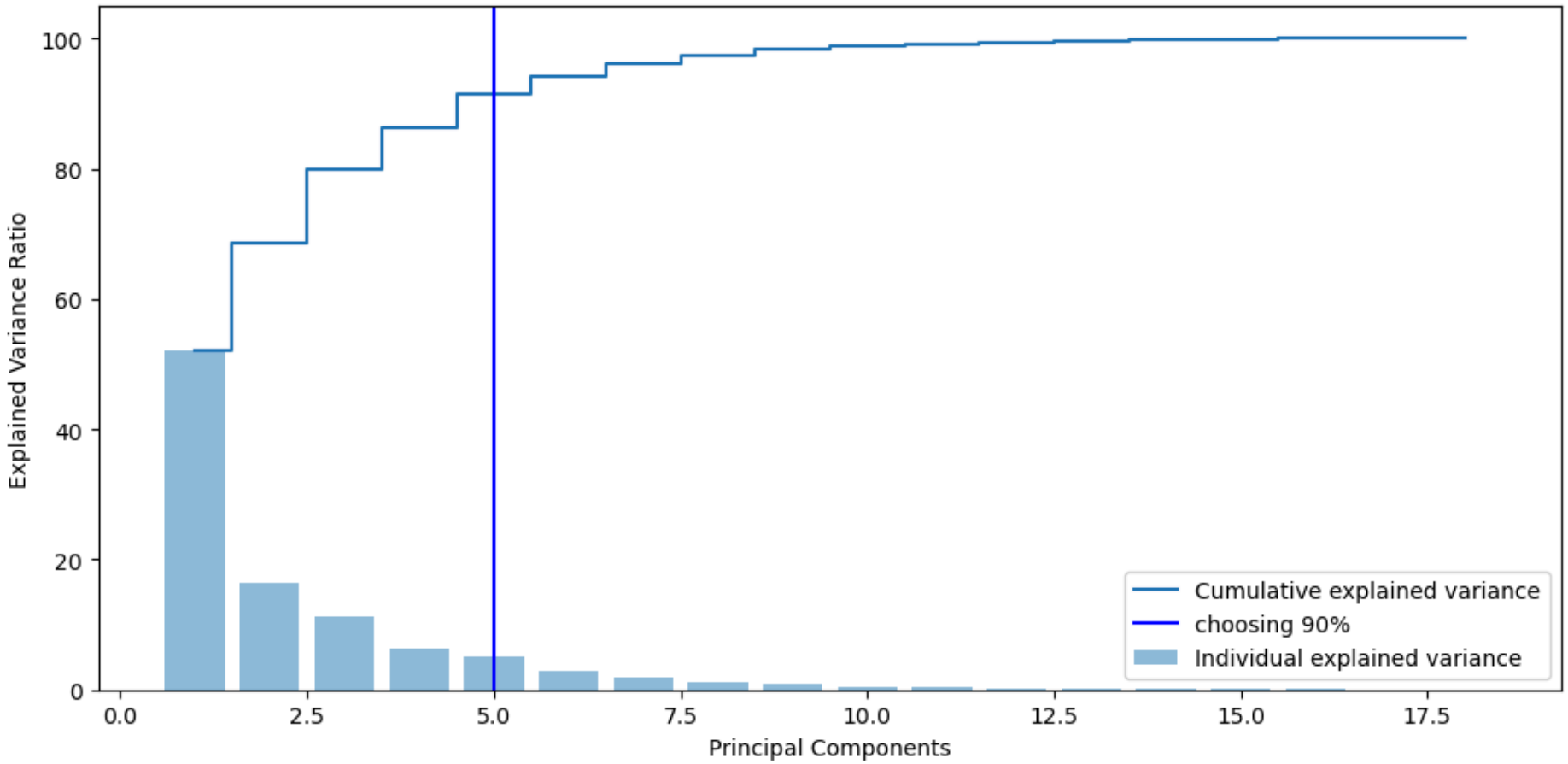


3 E. Draw a horizontal line on the above plot to highlight the threshold of 90%. [1 Marks]

In [213…

```
# Plotting the variance expalained by the principal components and the cumulative variance explained.
plt.figure(figsize=(10 , 5))
plt.bar(range(1, e_vals.size + 1), var_exp, alpha = 0.5, align = 'center', label = 'Individual explained variance')
plt.step(range(1, e_vals.size + 1), cum_var_exp, where='mid', label = 'Cumulative explained variance')
plt.axvline(x = 5, color = 'b', label = 'choosing 90%')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc = 'best')
```

```
plt.tight_layout()
plt.show()
```



3 F. Apply PCA on the data. This time Select Minimum Components with 90% or above variance explained. [2 Marks]

- so from cumlative varience explained we can choose component valuee 7 for more thwn 90% varience explaned around 96%

In [214…
```
from sklearn.decomposition import PCA
pca=PCA(n_components=7,random_state=1)
X_train_pca=pca.fit_transform(X_train_sd)
X_test_pca=pca.transform(X_test_sd)
```

In [215…
```
pca.explained_variance_
```

Out[215]:
```
array([9.40156057, 2.98053621, 2.01672486, 1.16303743, 0.92689862,
       0.50920211, 0.3461583 ])
```

In [216…
```
total=pca.explained_variance_.sum()
total
```

Out[216]:
```
17.3441180999673
```

In [217…
```
pca.explained_variance_/total
```

Out[217]:
```
array([0.54206046, 0.17184709, 0.11627716, 0.06705659, 0.05344167,
       0.02935878, 0.01995825])
```

In [ ]:

3 G. Train SVM model on components selected from above step. [1 Marks]

In [218…
```
from sklearn.svm import SVC

clf = SVC()
clf.fit(X_train_pca, y_train)
print ('After PCA score', clf.score(X_test_pca, y_test))
```

```
After PCA score 0.9047619047619048
```

3 H Print Classification metrics for train data of above model and share insights. [2 Marks]

In [219…
```
# generating the covariance matrix and the eigen values for the PCA analysis
cov_matrix = np.cov(X_train_pca.T) # the relevanat covariance matrix
print('Covariance Matrix \n%s', cov_matrix)

#generating the eigen values and the eigen vectors
e_vals, e_vecs = np.linalg.eig(cov_matrix)
```

```
print('Eigenvectors \n%s' %e_vecs)
print('\nEigenvalues \n%s' %e_vals)
```

```
Covariance Matrix
%s [[ 9.40156057e+00 -1.06051155e-17  7.15845293e-17  2.65127886e-18
  -5.83281350e-17 -9.54460391e-17  1.59076732e-17]
 [-1.06051155e-17  2.98053621e+00  9.86275738e-16 -7.15845293e-17
  -1.00748597e-16 -7.42358082e-17  1.59076732e-17]
 [ 7.15845293e-17  9.86275738e-16  2.01672486e+00 -1.20633188e-16
  -6.62819716e-17  1.52448535e-17 -5.30255773e-17]
 [ 2.65127886e-18 -7.15845293e-17 -1.20633188e-16  1.16303743e+00
   1.06051155e-16 -2.65542149e-16 -1.67693388e-16]
 [-5.83281350e-17 -1.00748597e-16 -6.62819716e-17  1.06051155e-16
   9.26898619e-01 -6.62819716e-18 -1.06051155e-17]
 [-9.54460391e-17 -7.42358082e-17  1.52448535e-17 -2.65542149e-16
  -6.62819716e-18  5.09202110e-01  1.52080877e-16]
 [ 1.59076732e-17  1.59076732e-17 -5.30255773e-17 -1.67693388e-16
  -1.06051155e-17  1.52080877e-16  3.46158297e-01]]
Eigenvectors
[[ 1.00000000e+00  1.65162361e-18  9.69344913e-18 -1.75670530e-18
  -1.07334898e-17  6.88265034e-18  3.21814823e-19]
 [ 0.00000000e+00  1.00000000e+00  1.84743671e-15 -2.54115774e-17
   7.35144887e-17  1.45127131e-16 -4.85366984e-17]
 [ 0.00000000e+00  1.08847459e-15 -1.00000000e+00  3.22021925e-18
  -1.46841381e-16 -1.15981258e-16 -3.85530231e-17]
 [ 0.00000000e+00  2.14564156e-16 -1.07084568e-16 -7.71477196e-18
  -6.53671161e-16 -2.63144808e-15 -1.00000000e+00]
 [ 0.00000000e+00  2.16506252e-16 -1.59519801e-16 -4.83312780e-17
  -2.84935438e-16  1.00000000e+00 -2.25041137e-15]
 [ 0.00000000e+00  1.09266094e-16 -3.85742977e-16  4.96135320e-17
  -1.00000000e+00 -1.09394871e-15  5.94173012e-16]
 [ 0.00000000e+00  2.49560880e-17  9.70762808e-17  1.00000000e+00
   7.99618555e-17  1.52156548e-16 -1.47784755e-16]]

Eigenvalues
[9.40156057 2.98053621 2.01672486 0.3461583  0.50920211 0.92689862
 1.16303743]
```

In [220…
```
len(e_vecs)
```

Out[220]:
```
7
```

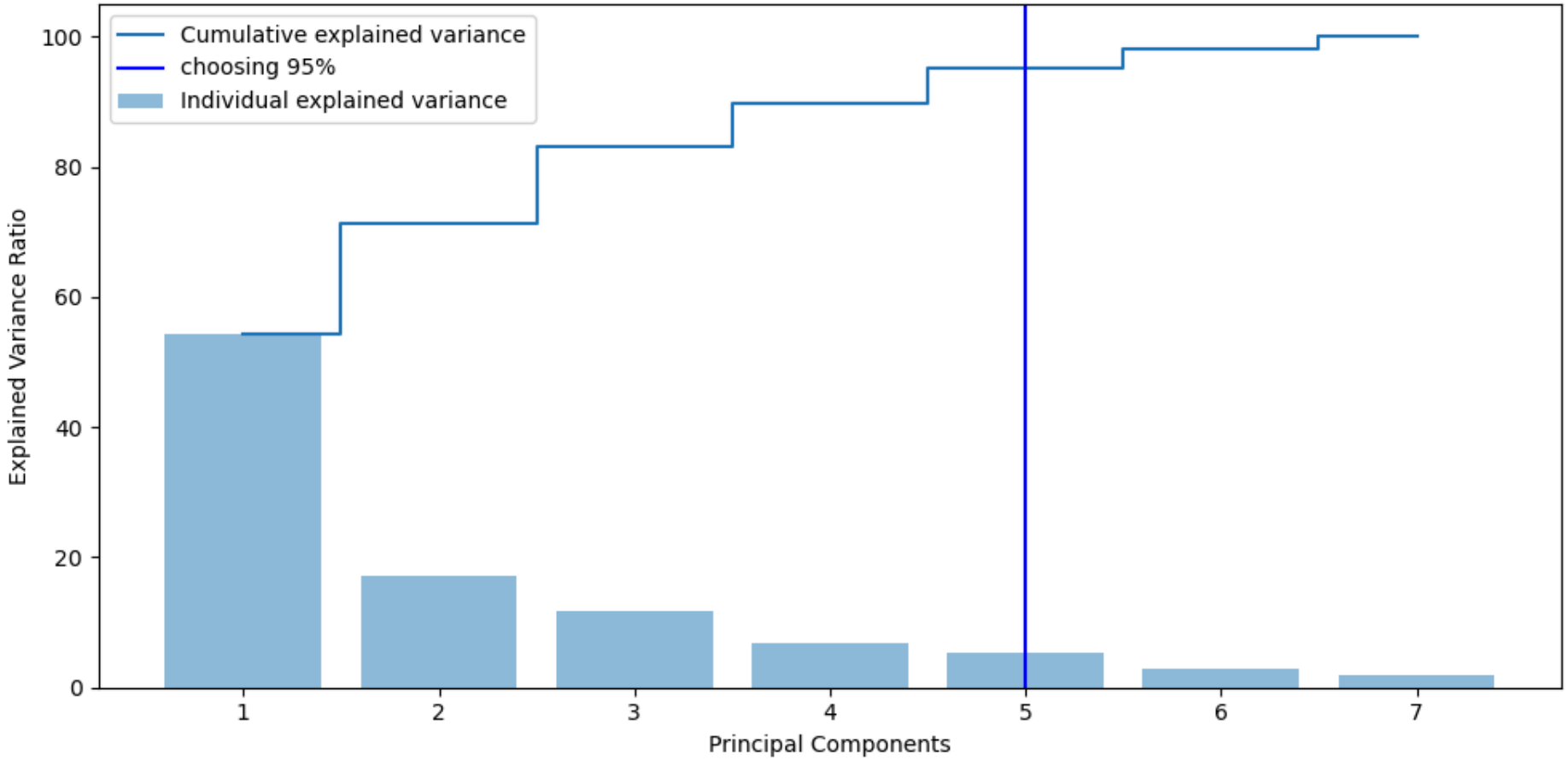
In [221…
```
# the "cumulative variance explained" analysis
tot = sum(e_vals)
```

```
var_exp = [( i /tot ) * 100 for i in sorted(e_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained", cum_var_exp)
```

```
Cumulative Variance Explained [ 54.20604565  71.39075459  83.01847095  89.72413002  95.06829692
  98.0041747  100.        ]
```

In [222…

```
# Plotting the variance expalained by the principal components and the cumulative variance explained.
plt.figure(figsize=(10 , 5))
plt.bar(range(1, e_vals.size + 1), var_exp, alpha = 0.5, align = 'center', label = 'Individual explained variance')
plt.step(range(1, e_vals.size + 1), cum_var_exp, where='mid', label = 'Cumulative explained variance')
plt.axvline(x = 5, color = 'b', label = 'choosing 95%')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc = 'best')
plt.tight_layout()
plt.show()
```



insight

- so from the cummulative variance graph it is shown that 90% of varience is explained upto minimum 5 principle components
- if we want more than 90% of vareince in data we wolud choose principle component 6 ot 7
- higher is better because with that sense lineariyy and correlation bewtweeen diffeerent features can be elemenated and behaviour of feature can be projected to other dimensions as much as possible.
- clearly eigen values represent magnitude of varience. and eigen vectors are principle components.

4. Performance Improvement: [5 Marks]

4 A. Train another SVM on the components out of PCA. Tune the parameters to improve performance. [2 Marks]

In [223…

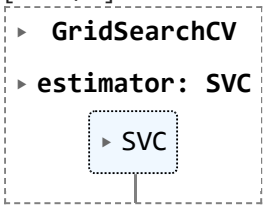
```
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10,100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train_pca, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.522 total time=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.530 total time=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.545 total time=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.545 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.785 total time=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.791 total time=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.784 total time=   0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.799 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.515 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.515 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.522 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.874 total time=   0.0s
[CV 2/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.791 total time=   0.0s
[CV 3/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.866 total time=   0.0s
[CV 4/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.821 total time=   0.0s
[CV 5/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.843 total time=   0.0s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.919 total time=   0.0s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.933 total time=   0.0s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.948 total time=   0.0s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.933 total time=   0.0s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.837 total time=   0.0s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.836 total time=   0.0s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.888 total time=   0.0s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.843 total time=   0.0s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.813 total time=   0.0s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.522 total time=   0.0s
```

```
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.530 total time=   0.0s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.515 total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.889 total time=   0.0s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.791 total time=   0.0s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.851 total time=   0.0s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.919 total time=   0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.888 total time=   0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.933 total time=   0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.933 total time=   0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.955 total time=   0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.889 total time=   0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.903 total time=   0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.933 total time=   0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.933 total time=   0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.918 total time=   0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.815 total time=   0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.754 total time=   0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.836 total time=   0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.799 total time=   0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.799 total time=   0.0s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.511 total time=   0.0s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.530 total time=   0.0s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.522 total time=   0.0s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.889 total time=   0.0s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.791 total time=   0.0s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.851 total time=   0.0s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.858 total time=   0.0s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.904 total time=   0.0s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.910 total time=   0.0s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.903 total time=   0.0s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.933 total time=   0.0s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.955 total time=   0.0s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.911 total time=   0.0s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.888 total time=   0.0s
```

```
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.933 total time=   0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.948 total time=   0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.948 total time=   0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.844 total time=   0.0s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.858 total time=   0.0s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.896 total time=   0.0s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.873 total time=   0.0s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.843 total time=   0.0s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.778 total time=   0.0s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.716 total time=   0.0s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.806 total time=   0.0s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.784 total time=   0.0s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.813 total time=   0.0s
```

Out[223]:



4 B B. Share best Parameters observed from above step. [1 Marks]

In [224…

```python
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
SVC(C=100, gamma=0.01)
```

4 C. Print Classification metrics for train data of above model and share relative improvement in performance in all the models along with insights. [2 Marks]

In [225…

```python
from sklearn.metrics import classification_report, confusion_matrix

grid_predictions = grid.predict(X_train_pca)

# print classification report
print(classification_report(y_train, grid_predictions))
```

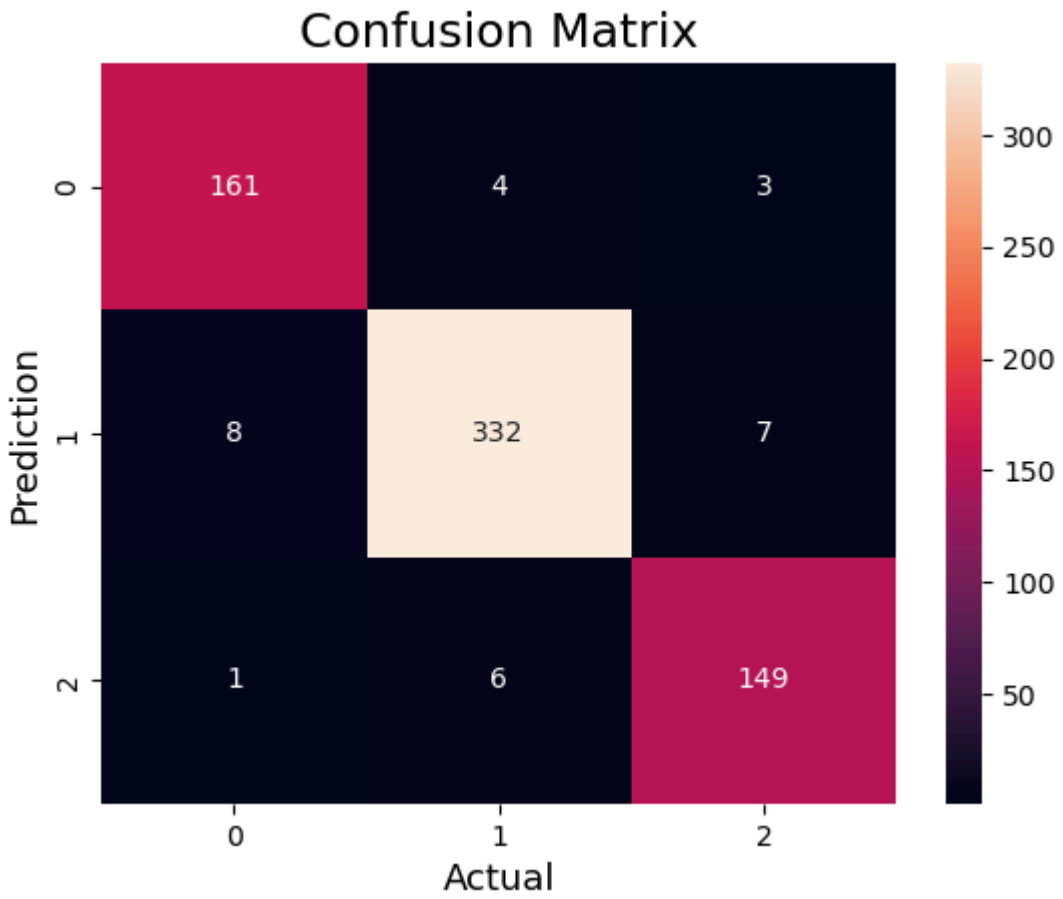
```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       168
           1       0.97      0.96      0.96       347
           2       0.94      0.96      0.95       156

    accuracy                           0.96       671
   macro avg       0.95      0.96      0.95       671
weighted avg       0.96      0.96      0.96       671
```

In [226…

```
cm = confusion_matrix(y_train,grid_predictions)
```

In [227…

```
   sns.heatmap(cm,
               annot=True,
               fmt='g',
               xticklabels=['0','1','2'],
               yticklabels=['0','1','2'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



insights

- recall is 95% that is we able to predict 95% indetify datapoints after principel componenet analysys from total actual datapoints
- precision is 94%. that is out of all prediction 94% is identfied correctly.
- accuracy 96% that is 96% times prediction got explained correctly in new dimension data set.

5. Data Understanding & Cleaning: [5 Marks]

5 A. Explain pre-requisite/assumptions of PCA. [2 Marks]

- **PCA assumes a correlation between features**. If the features (or dimensions or columns, in tabular data) are not correlated, PCA will be unable to determine principal components.
- **PCA is sensitive to the scale of the features.** Imagine we have two features - one takes values between 0 and 1000, while the other takes values between 0 and 1. PCA will be extremely biased towards the first feature being the first principle component, regardless of the actual maximum variance within the data. This is why it's so important to standardize the values first.
- **PCA is not robust against outliers.** Similar to the point above, the algorithm will be biased in datasets with strong outliers. This is why it is recommended to remove outliers before performing PCA.
- **PCA assumes a linear relationship between features.** The algorithm is not well suited to capturing non-linear relationships. That's why it's advised to turn non-linear features or relationships between features into linear, using the standard methods such as log transforms.
- **Technical implementations often assume no missing values.** When computing PCA using statistical software tools, they often assume that the feature set has no missing values (no empty rows). Be sure to remove those rows and/or columns with missing values, or impute missing values with a close approximation (e.g. the mean of the column).

- Assumption: The eigenvalue is measures the "importance" of the i th principal component

- It is intuitively reasonable that lower variability components describe the data less, but it is not always true
- PCA assumes that the intrinsic dimensions are orthogonal.When this assumption fails, we need to assume non-orthogonal components which are not compatible with PCA. PCA assumes that data lie on a lower dimensional linear manifold.
- When the data lie on a nonlinear manifold in the predictor space,then linear methods are likely to be ineffective.

5 B. Explain advantages and limitations of PCA. [3 Marks]

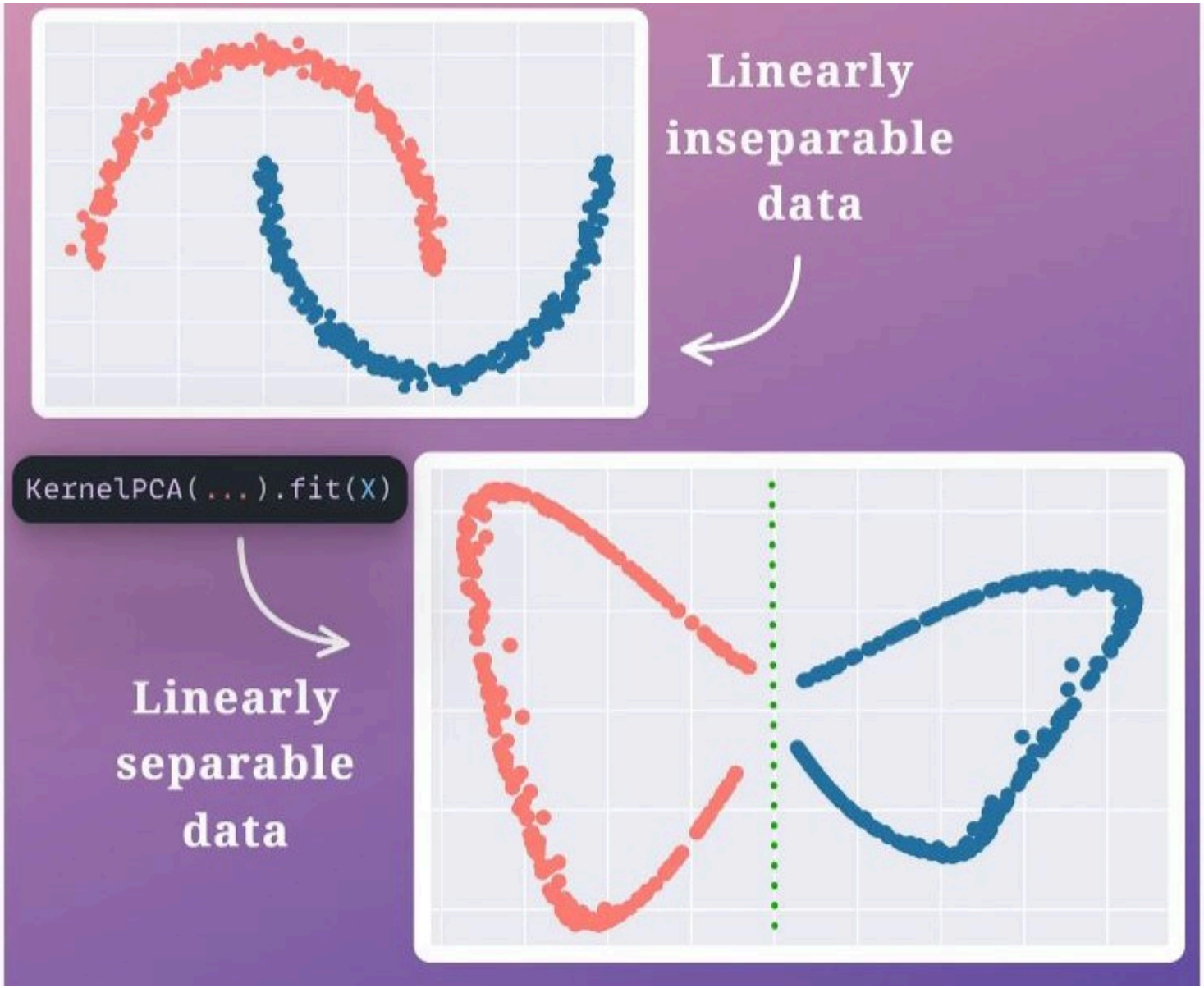
- Low interpretability of principal components. Principal components are linear combinations of the features from the original data, but they are not as easy to interpret. For example, it is difficult to tell which are the most important features in the dataset after computing principal components.

- The trade-off between information loss and dimensionality reduction. Although dimensionality reduction is useful, it comes at a cost. Information loss is a necessary part of PCA. Balancing the trade-off between dimensionality reduction and information loss is unfortunately a necessary compromise that we have to make when using PCA.

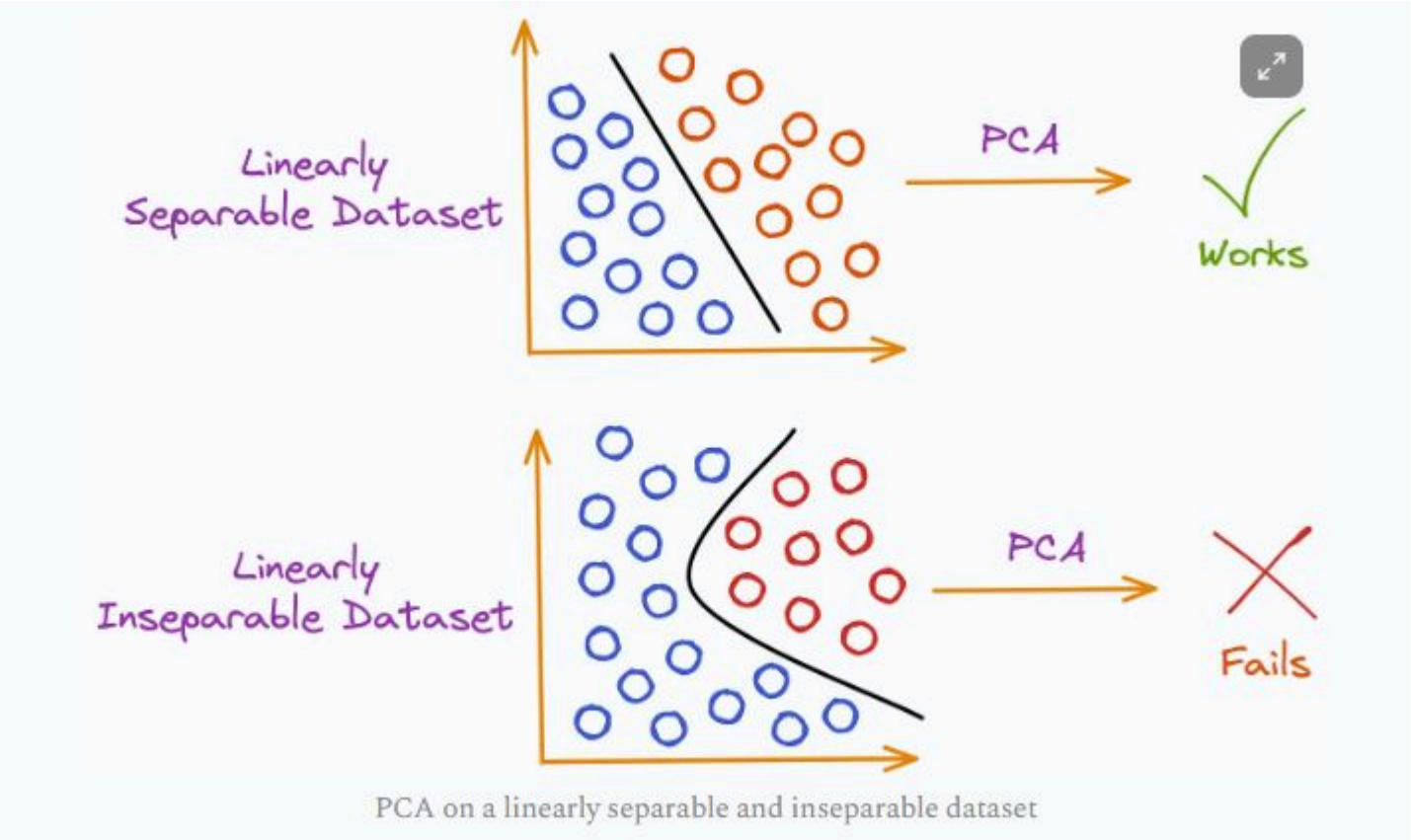
- To start PCA on the right foot, you will need to have the right tools that help you collect data from multiple sources and prepare it for machine learning models. Keboola covers all the steps, so you won't have to think about the infrastructure, only about the added-value your

machine learning models will bring.

- PCA transforms the original input variables into new principal components (or dimensions). The new dimensions offer no interpretability.
- While PCA simplifies the data and removes noise, it always leads to some loss of information when we reduce dimensions.
- PCA is a linear dimensionality reduction technique, but not all real-world datasets may be linear. Read more about this in my previous post here: The Limitation of PCA Which Many Folks Often Ignore.

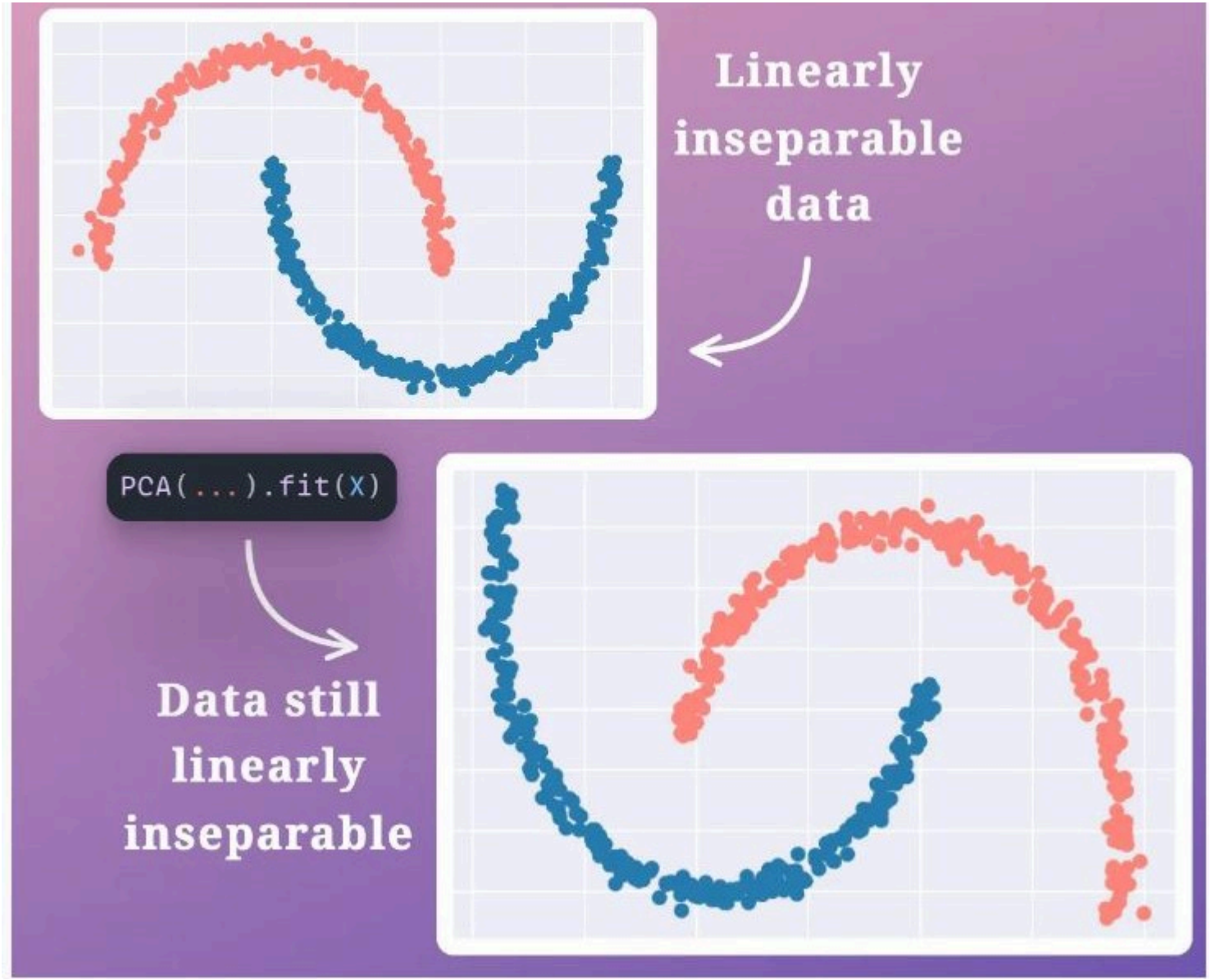


- Imagine you have a classification dataset. If you use PCA to reduce dimensions, it is inherently assumed that your data is linearly separable.
- But it may not be the case always. Thus, PCA will fail in such cases.



PCA on a linearly separable and inseparable dataset

- To resolve this, we use the kernel trick (or the KernelPCA). The idea is to:
- Project the data to another space using a kernel function, where the data becomes linearly separable.
- Apply the standard PCA algorithm to the transformed data.
- For instance, in the image below, the original data is linearly inseparable. Using PCA directly does not produce any desirable results.



- But as mentioned above, KernelPCA first transforms the data to a linearly separable space and then applies PCA,
- resulting in a linearly separable dataset.

- Sklearn provides a KernelPCA wrapper, supporting many popularly used kernel functions.
- Having said that, it is also worth noting that the run-time of PCA is cubic in relation to the number of dimensions of the data.

- PCA run-time

- When we use a KernelPCA, typically, the original data (in n dimensions) is projected to a new higher dimensional space (in m dimensions; m>n). Therefore, it increases the overall run-time of PCA.

In [ ]:

==========END PART B==========

In [ ]: