1.Calculate the Eigen value and Eigen vector for the following matrices and sort the Eigenvector in decreasing order of magnitude of Eigen values.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import numpy.linalg as linalg
from matplotlib import pyplot
from sklearn.decomposition import PCA
from sklearn import preprocessing
```

In [2]:
```
data=pd.read_csv("electric_motor.csv")
data.head()
```

Out[2]:

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521

In [3]:
```
x = [[4,2,5],[2,6,2],[6,1,2]]
eigenValues, eigenVectors = linalg.eig(x)
idx = eigenValues.argsort()[::-1]
eigen_values = eigenValues[idx]
eigen_vector = eigenVectors[:,idx]
print(eigen_values)
print(eigen_vector)
```

```
[10.0862564   4.45231344 -2.53856984]
[[-0.62581291  0.3003003   0.59754478]
 [-0.56795402 -0.87594531  0.04752115]
 [-0.53458996  0.3775441  -0.80042612]]
```

In [4]:
```
y = [[4,2,4],[2,6,2],[6,1,2]]
eigenValues, eigenVectors = linalg.eig(y)
idy = eigenValues.argsort()[::-1]
eigen_values1 = eigenValues[idy]
eigen_vector2 = eigenVectors[:,idy]
print(eigen_values1)
print(eigen_vector2)
```

```
[ 9.70953904  4.23567914 -1.94521818]
[[-0.58779736  0.30479966  0.53902619]
 [-0.60592357 -0.84456255  0.07548528]
 [-0.53605121  0.44024001 -0.83889972]]
```

In [5]:
```
z = [[4,2,5],[2,5,2],[6,1,2]]
eigenValues, eigenVectors = linalg.eig(z)
idz = eigenValues.argsort()[::-1]
eigen_values3 = eigenValues[idz]
eigen_vector4 = eigenVectors[:,idz]
print(eigen_values3)
print(eigen_vector4)
```

```
[ 9.84837399  3.68599053 -2.53436452]
[[-0.65457471  0.24809842  0.59624933]
 [-0.50287543 -0.90472019  0.05434002]
 [-0.56448937  0.34630702 -0.80095811]]
```

In [ ]:

2. Now think it would be better, if you have defined a function for the above operation that takes input as matrix and give the output desired in question 5. Create a function to do that.

In [6]:
```
def Eig_val_vect():
 lst1 = list(input('Enter the first row : '))
 lst2 = list(input('Enter the second row : '))
 lst3 = list(input('Enter the third row : '))
 x= np.array([lst1]+[lst2]+[lst3])

 eigenValues, eigenVectors = linalg.eig(x)
 idx = eigenValues.argsort()[::-1]
 eigen_values = eigenValues[idx]
 eigen_vector = eigenVectors[:,idx]

print(eigen_values)
print(eigen_vector)
```

```
[10.0862564   4.45231344 -2.53856984]
[[-0.62581291  0.3003003   0.59754478]
 [-0.56795402 -0.87594531  0.04752115]
 [-0.53458996  0.3775441  -0.80042612]]
```

3. Repeat the above task using singular value decomposition. Report your observation.

In [9]:
```
#Creating a matrix A
A = np.array([[3,4,3],[1,2,3],[4,2,1]])
#Performing SVD
U, D, VT = np.linalg.svd(A)
#Checking if we can remake the original matrix using U,D,VT
A_remake = (U @ np.diag(D) @ VT)
print(A_remake)
```

```
[[3. 4. 3.]
 [1. 2. 3.]
 [4. 2. 1.]]
```

4. Use your ‘elctric_motor.csv’ and calculate covariance matrix and then calculate Eigen value and Eigen vector and sort the Eigen vector in decreasing order of magnitude of Eigen values. This could also be done with slight change in the code.

In [12]:
```
features = list(set(data.columns)-set(['motor_speed','torque']))
target = list(['motor_speed','torque'])
print(features)
print(target)
['stator_tooth', 'stator_winding', 'coolant', 'ambient', 'i_d', 'i_q',
'pm', 'profile_id', 'u_d', 'stator_yoke', 'u_q']
['motor_speed', 'torque']
x = data.loc[:,features]
y = data.loc[:,target].astype(float)
cova_matr = x.cov()
print(cova_matr)
```

```
['pm', 'ambient', 'i_d', 'stator_winding', 'profile_id', 'u_q', 'i_q', 'stato
r_tooth', 'coolant', 'stator_yoke', 'u_d']
['motor_speed', 'torque']
                      pm   ambient       i_d  stator_winding  profile_id  \
pm              0.991391  0.495901 -0.297636        0.725210    3.444704
ambient         0.495901  0.986301  0.005560        0.299312    8.430198
i_d            -0.297636  0.005560  0.997990       -0.538487    3.139344
stator_winding  0.725210  0.299312 -0.538487        0.996688    4.008615
profile_id      3.444704  8.430198  3.139344        4.008615  487.222866
u_q             0.101034  0.087031 -0.182095        0.125529   -2.704943
i_q            -0.085933 -0.258231 -0.203598        0.060721   -5.641714
stator_tooth    0.764730  0.393857 -0.387166        0.963645    6.199925
coolant         0.429730  0.432495  0.108642        0.509686   11.055985
stator_yoke     0.692742  0.448983 -0.179911        0.844629    8.794776
u_d            -0.082034  0.193005  0.357397       -0.150145    6.624865

                     u_q       i_q  stator_tooth    coolant  stator_yoke  \
pm              0.101034 -0.085933      0.764730   0.429730     0.692742
ambient         0.087031 -0.258231      0.393857   0.432495     0.448983
i_d            -0.182095 -0.203598     -0.387166   0.108642    -0.179911
stator_winding  0.125529  0.060721      0.963645   0.509686     0.844629
profile_id     -2.704943 -5.641714      6.199925  11.055985     8.794776
u_q             1.004666 -0.026355      0.149304   0.027984     0.106546
i_q            -0.026355  0.995829     -0.025129  -0.186121    -0.098650
stator_tooth    0.149304 -0.025129      0.999195   0.690395     0.950512
coolant         0.027984 -0.186121      0.690395   1.004853     0.877074
stator_yoke     0.106546 -0.098650      0.950512   0.877074     1.002099
u_d            -0.027478 -0.793236     -0.066089   0.178761     0.041384

                     u_d
pm             -0.082034
ambient         0.193005
i_d             0.357397
stator_winding -0.150145
profile_id      6.624865
u_q            -0.027478
i_q            -0.793236
stator_tooth   -0.066089
coolant         0.178761
stator_yoke     0.041384
u_d             0.995761
```

In [ ]: