1. Classify the algorithms into supervised and unsupervised machine learning algorithms

[Support vector machine, Decision tree, Hierarchical clustering, K-nearest neighbor, K-mean clustering, Naïve Bayes, Principal component analysis, Linear discriminant analysis, independent component analysis]

In [ ]:
```
#Supervised:
supervised machine learning alogrithms : Decision tree,K-nearest neighbor,Naïv
Bayes,Support vector machine,Linear discriminant analysis
#Unsupervised
unsupervised machine learning alogrithms : K-mean clustering,Hierarchical
clustering,Principal component analysis,independent component analysis
```

In [ ]:
```

```

2)Given a data set, we are going to see how feature extraction can be done. Electric

motor drive data set is given. (electric_motor.csv).

Steps for feature extraction

(a) Load the data into the variable explorer

(b) Construct covariance matrix

(c) Find Eigen values and Eigen vectors of the matrix

(d) Arrange the Eigen values and Eigen vectors in descending order

(e) Select the number of Eigen values such that the ration of selected and total Eigen values

is 0.95.

(f) After doing this select as many Eigen vector as number of Eigen values selected.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn import preprocessing
```

In [5]:
```
data=pd.read_csv('electric_motor.csv')
data.head()
```

Out[5]:

	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 [6]:
```
data_columns_list=list(data.columns)
input_columns_list=list(set(data_columns_list)-
set(['pm','profile_id']))
print(input_columns_list)
output_list=list(['pm'])
print(output_list)
input_columns_list.sort()
print(input_columns_list)
```

```
['u_d', 'i_q', 'motor_speed', 'i_d', 'ambient', 'coolant', 'stator_winding',
'stator_yoke', 'torque', 'stator_tooth', 'u_q']
['pm']
['ambient', 'coolant', 'i_d', 'i_q', 'motor_speed', 'stator_tooth', 'stator_w
inding', 'stator_yoke', 'torque', 'u_d', 'u_q']
```

In [7]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
data[input_columns_list]=sc.fit_transform(data[input_columns_list])
print(data)
```

```
         ambient   coolant       u_d       u_q  motor_speed    torque  \
0      -0.753416 -1.120454  0.323842 -1.289164    -1.214600 -0.247367
1      -0.772669 -1.119032  0.325575 -1.288994    -1.214601 -0.246316
2      -0.784378 -1.118693  0.328689 -1.293119    -1.214600 -0.246615
3      -0.782408 -1.118776  0.329619 -1.293150    -1.214602 -0.245818
4      -0.775467 -1.118787  0.331128 -1.294413    -1.214601 -0.245883
...          ...       ...       ...       ...          ...       ...
998065 -0.043894  0.336101  0.327389 -1.237541    -1.214600 -0.252836
998066 -0.045245  0.314537  0.327616 -1.242071    -1.214609 -0.252836
998067 -0.038711  0.301961  0.326859 -1.238277    -1.214602 -0.252836
998068 -0.035774  0.296640  0.326900 -1.240924    -1.214604 -0.252836
998069 -0.040173  0.307199  0.326743 -1.238017    -1.214603 -0.252836

            i_d       i_q        pm  stator_yoke  stator_tooth  \
0      1.024560 -0.243174 -2.522071    -1.830112     -2.064767
1      1.024497 -0.243146 -2.522418    -1.829659     -2.063483
2      1.024435 -0.243132 -2.522673    -1.829091     -2.062697
3      1.027836 -0.244271 -2.521639    -1.829024     -2.061760
4      1.026797 -0.243925 -2.521900    -1.829189     -2.061418
...         ...       ...       ...          ...           ...
998065 1.024130 -0.243036  0.429853     1.016892      0.838630
998066 1.024135 -0.243049  0.429751     1.011746      0.836983
998067 1.024178 -0.243015  0.429439     1.001246      0.836481
998068 1.024134 -0.243041  0.429558     0.997502      0.833048
998069 1.024128 -0.243036  0.429166     0.985520      0.830588

        stator_winding  profile_id
0            -2.017442           4
1            -2.017040           4
2            -2.016752           4
3            -2.017041           4
4            -2.017555           4
...                ...         ...
998065        0.499487          72
998066        0.499041          72
998067        0.497425          72
998068        0.495337          72
998069        0.494136          72

[998070 rows x 13 columns]
```

In [8]:
```
#B:
input_data=data[input_columns_list]
cov_matrix=input_data.cov()
print(cov_matrix)
```

```
                ambient   coolant       i_d       i_q  motor_speed  \
ambient        1.000001  0.434436  0.005605 -0.260562     0.078283
coolant        0.434436  1.000001  0.108489 -0.186060    -0.033332
i_d            0.005605  0.108489  1.000001 -0.204230    -0.722915
i_q           -0.260562 -0.186060 -0.204230  1.000001     0.006323
motor_speed    0.078283 -0.033332 -0.722915  0.006323     1.000001
stator_tooth   0.396742  0.689004 -0.387712 -0.025192     0.333910
stator_winding 0.301884  0.509298 -0.539925  0.060949     0.393155
stator_yoke    0.451617  0.874038 -0.179903 -0.098753     0.182564
torque        -0.262264 -0.189617 -0.239060  0.996561     0.024517
u_d            0.194754  0.178709  0.358518 -0.796587    -0.233828
u_q            0.087430  0.027851 -0.181855 -0.026348     0.716898

                stator_tooth  stator_winding  stator_yoke    torque       u_d
\
ambient             0.396742        0.301884     0.451617 -0.262264  0.194754
coolant             0.689004        0.509298     0.874038 -0.189617  0.178709
i_d                -0.387712       -0.539925    -0.179903 -0.239060  0.358518
i_q                -0.025192        0.060949    -0.098753  0.996561 -0.796587
motor_speed         0.333910        0.393155     0.182564  0.024517 -0.233828
stator_tooth        1.000001        0.965634     0.949899 -0.011055 -0.066256
stator_winding      0.965634        1.000001     0.845145  0.080981 -0.150714
stator_yoke         0.949899        0.845145     1.000001 -0.092207  0.041428
torque             -0.011055        0.080981    -0.092207  1.000001 -0.821326
u_d                -0.066256       -0.150714     0.041428 -0.821326  1.000001
u_q                 0.149017        0.125445     0.106187 -0.037262 -0.027472

                    u_q
ambient        0.087430
coolant        0.027851
i_d           -0.181855
i_q           -0.026348
motor_speed    0.716898
stator_tooth   0.149017
stator_winding 0.125445
stator_yoke    0.106187
torque        -0.037262
u_d           -0.027472
u_q            1.000001
```

In [9]:
```
#c:
eig_val,eig_vec=np.linalg.eig(cov_matrix.to_numpy())
len(eig_val)
11
eig_pairs=[(np.abs(eig_val[i]),eig_vec[:,-1])for i in
range(len(eig_val))]
eig_pairs
```

Out[9]:
```
[(3.9553262258871866,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (3.126649594583,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (1.8151763109167807,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.876838249023958,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.6733779985624924,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.2740495129394412,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.21717093918478284,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.04950805855264753,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.008974252828338514,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.0009037779802785378,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081])),
 (0.0020361008231908955,
  array([ 0.00205195, -0.02569351,  0.02518613, -0.68136998,  0.01576494,
         -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
          0.00385081]))]
```

In [10]:
```
#D:
eig_pairs.sort(key=lambda x:x[0], reverse=True)
print(eig_pairs)
```

```
[(3.9553262258871866, array([ 0.00205195, -0.02569351,  0.02518613, -0.681369
98,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (3.126649594583, array([ 0.00205195, -0.02569351,  0.0
2518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (1.8151763109167807, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.876838249023958, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.6733779985624924, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.2740495129394412, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.21717093918478284, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.04950805855264753, array([ 0.00205195, -0.02569351,
0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.008974252828338514, array([ 0.00205195, -0.0256935
1,  0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.0020361008231908955, array([ 0.00205195, -0.0256935
1,  0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081])), (0.0009037779802785378, array([ 0.00205195, -0.0256935
1,  0.02518613, -0.68136998,  0.01576494,
       -0.02817042, -0.01324635,  0.05860679,  0.72636119,  0.04656725,
        0.00385081]))]
```

In [14]:
```
#E:
threshold=0.95
Total_variance=0.0
count=0
eigv_sum=np.sum(eig_val)
for i,j in enumerate(eig_pairs):
 variance=(j[0]/eigv_sum).real
 print('eigenvalue {}:{}'.format(i+1,(j[0]/eigv_sum).real*100))
 Total_variance=Total_variance+variance
 count=count+1
 if (Total_variance>=threshold):
     break
```

```
eigenvalue 1:35.95747511738563
eigenvalue 2:28.424058744430024
eigenvalue 3:16.50158629300368
eigenvalue 4:7.971248822637624
eigenvalue 5:6.121612035294191
eigenvalue 6:2.4913567123635443
```

In [15]:
```
print(Total_variance)
print('Total number of eigenvectors is {}'.format(len(eig_vec)))
print('Total number of important feautures are {}'.format(count))
```

```
0.9746733772511469
Total number of eigenvectors is 11
Total number of important feautures are 6
```

In [ ]:

In [ ]:

In [ ]: