Importing Data

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
```

In [2]:

```
raw_data= pd.read_csv("vehicle-1.csv")
raw_data.head()
```

Out[2]:

compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_aspect_ra
95	48.0	83.0	178.0	72.0	
91	41.0	84.0	141.0	57.0	
104	50.0	106.0	209.0	66.0	
93	41.0	82.0	159.0	63.0	
85	44.0	70.0	205.0	103.0	

Data Preprocessing ¶

In [3]:
```
raw_data.corr()
```

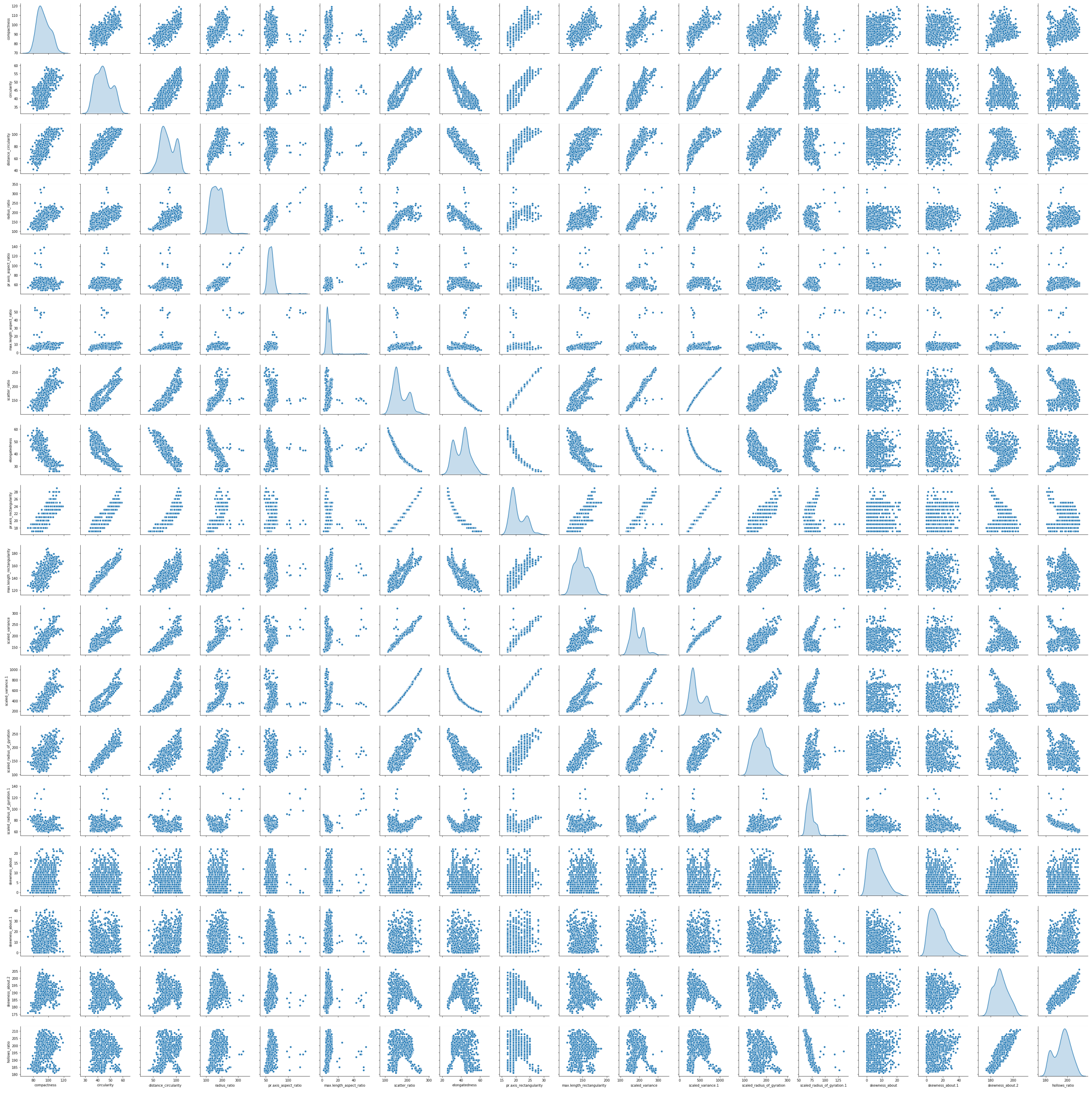
Out[3]:
```
                              compactness  circularity  distance_circularity  radius_ratio  pr.axis_aspect_
                 compactness     1.000000     0.689786              0.791707      0.691081             0.09
                 circularity     0.689786     1.000000              0.797180      0.625051             0.15
        distance_circularity     0.791707     0.797180              1.000000      0.771748             0.15
                radius_ratio     0.691081     0.625051              0.771748      1.000000             0.66
       pr.axis_aspect_ratio     0.091779     0.154283              0.158684      0.665363             1.00
     max.length_aspect_ratio     0.148249     0.251407              0.264621      0.450486             0.64
               scatter_ratio     0.812770     0.858265              0.907949      0.738480             0.10
               elongatedness    -0.788736    -0.827246             -0.913020     -0.792946            -0.18
       pr.axis_rectangularity     0.814248     0.856603              0.896273      0.712744             0.07
   max.length_rectangularity     0.676143     0.965729              0.775149      0.571083             0.12
             scaled_variance     0.764361     0.806791              0.865710      0.798294             0.27
           scaled_variance.1     0.818674     0.850863              0.890541      0.725598             0.08
    scaled_radius_of_gyration     0.585845     0.935950              0.706950      0.541325             0.12
  scaled_radius_of_gyration.1    -0.250603     0.053080             -0.227001     -0.181520             0.15
              skewness_about     0.236685     0.144968              0.114665      0.049112            -0.05
            skewness_about.1     0.157670    -0.011869              0.266049      0.174469            -0.03
            skewness_about.2     0.298528    -0.106339              0.146027      0.382912             0.24
                hollows_ratio     0.365552     0.045652              0.333648      0.472339             0.26
```

In [4]:
```
raw_data.describe()
```

Out[4]:
```
        compactness  circularity  distance_circularity  radius_ratio  pr.axis_aspect_ratio  max.length_
count    846.000000   841.000000            842.000000    840.000000            844.000000
 mean     93.678487    44.828775             82.110451    168.888095             61.678910
  std      8.234474     6.152172             15.778292     33.520198              7.891463
  min     73.000000    33.000000             40.000000    104.000000             47.000000
  25%     87.000000    40.000000             70.000000    141.000000             57.000000
  50%     93.000000    44.000000             80.000000    167.000000             61.000000
  75%    100.000000    49.000000             98.000000    195.000000             65.000000
  max    119.000000    59.000000            112.000000    333.000000            138.000000
```

In [5]:
```
sn.pairplot(raw_data, diag_kind = 'kde')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:44
7: RuntimeWarning: invalid value encountered in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:44
7: RuntimeWarning: invalid value encountered in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
```



In [6]:
```
raw_data['class']= raw_data['class'].replace({'van': 1, 'bus': 2, 'car': 3})
```

In [7]:
```
raw_data.describe()
```

Out[7]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_
count	846.000000	841.000000	842.000000	840.000000	844.000000	
mean	93.678487	44.828775	82.110451	168.888095	61.678910	
std	8.234474	6.152172	15.778292	33.520198	7.891463	
min	73.000000	33.000000	40.000000	104.000000	47.000000	
25%	87.000000	40.000000	70.000000	141.000000	57.000000	
50%	93.000000	44.000000	80.000000	167.000000	61.000000	
75%	100.000000	49.000000	98.000000	195.000000	65.000000	
max	119.000000	59.000000	112.000000	333.000000	138.000000	

In [8]:
```
data= raw_data.drop(['class'], axis= 1)
data.median()
```

Out[8]:
```
compactness                     93.0
circularity                     44.0
distance_circularity            80.0
radius_ratio                   167.0
pr.axis_aspect_ratio            61.0
max.length_aspect_ratio          8.0
scatter_ratio                  157.0
elongatedness                   43.0
pr.axis_rectangularity          20.0
max.length_rectangularity      146.0
scaled_variance                179.0
scaled_variance.1              363.5
scaled_radius_of_gyration      173.5
scaled_radius_of_gyration.1     71.5
skewness_about                   6.0
skewness_about.1                11.0
skewness_about.2               188.0
hollows_ratio                  197.0
dtype: float64
```

Dealing with outliers and empty values

In [9]:
```
data.fillna(data.median(), inplace= True)
```

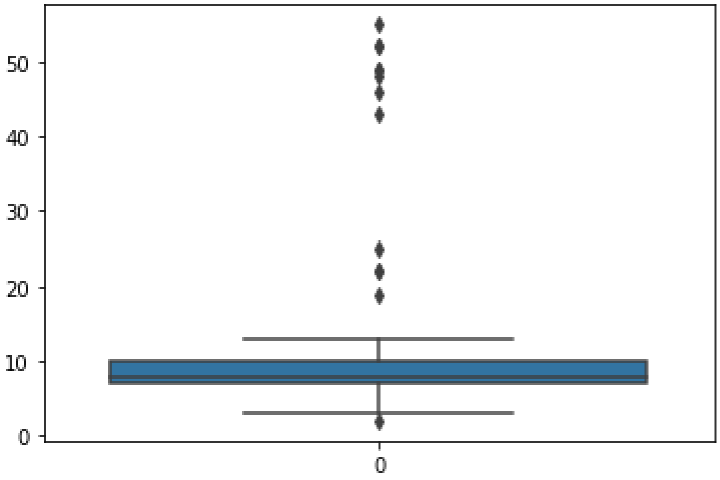
In [10]:
```
max_length_aspect_ratio= data['max.length_aspect_ratio']
sn.boxplot(data= max_length_aspect_ratio)

for i in range(846):
    if max_length_aspect_ratio[i]> 13:
        max_length_aspect_ratio[i]= max_length_aspect_ratio.median()

data['max.length_aspect_ratio']= max_length_aspect_ratio
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydat
a.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y)
```



In [11]:

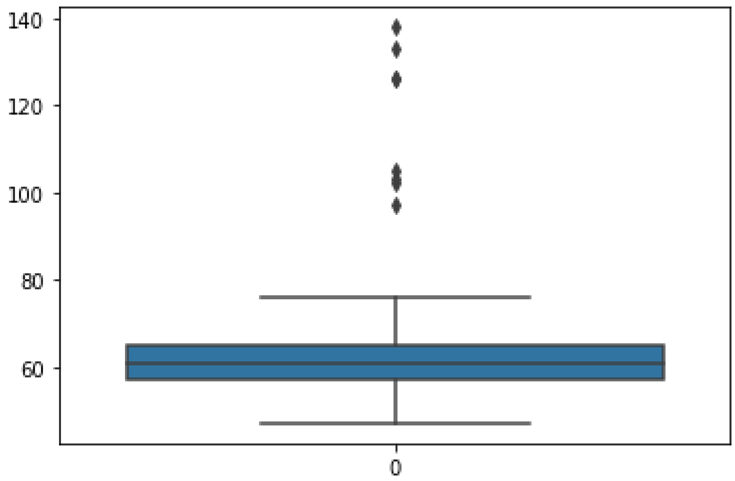
```
pr_access_aspect_ratio= data['pr.axis_aspect_ratio']
sn.boxplot(data= pr_access_aspect_ratio)

for i in range(846):
    if pr_access_aspect_ratio[i]> 80:
        pr_access_aspect_ratio[i]= pr_access_aspect_ratio.median()

data['pr.axis_aspect_ratio']= pr_access_aspect_ratio
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydat
a.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y)
```



In [12]:

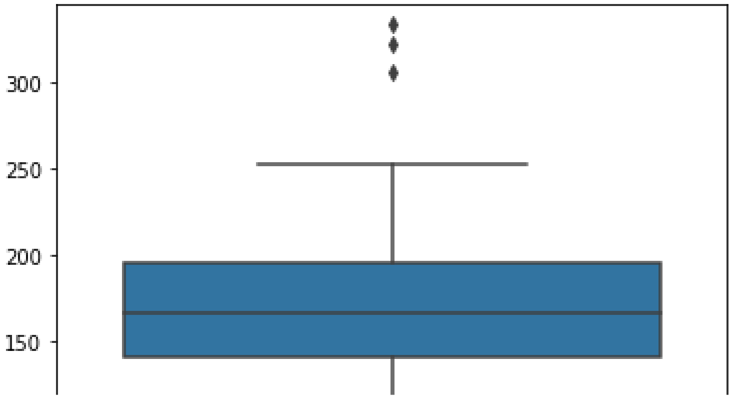
```
radius_ratio= data['radius_ratio']
sn.boxplot(data= radius_ratio)

for i in range(846):
    if radius_ratio[i]> 250:
        radius_ratio[i]= radius_ratio.median()


data['radius_ratio']= radius_ratio
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



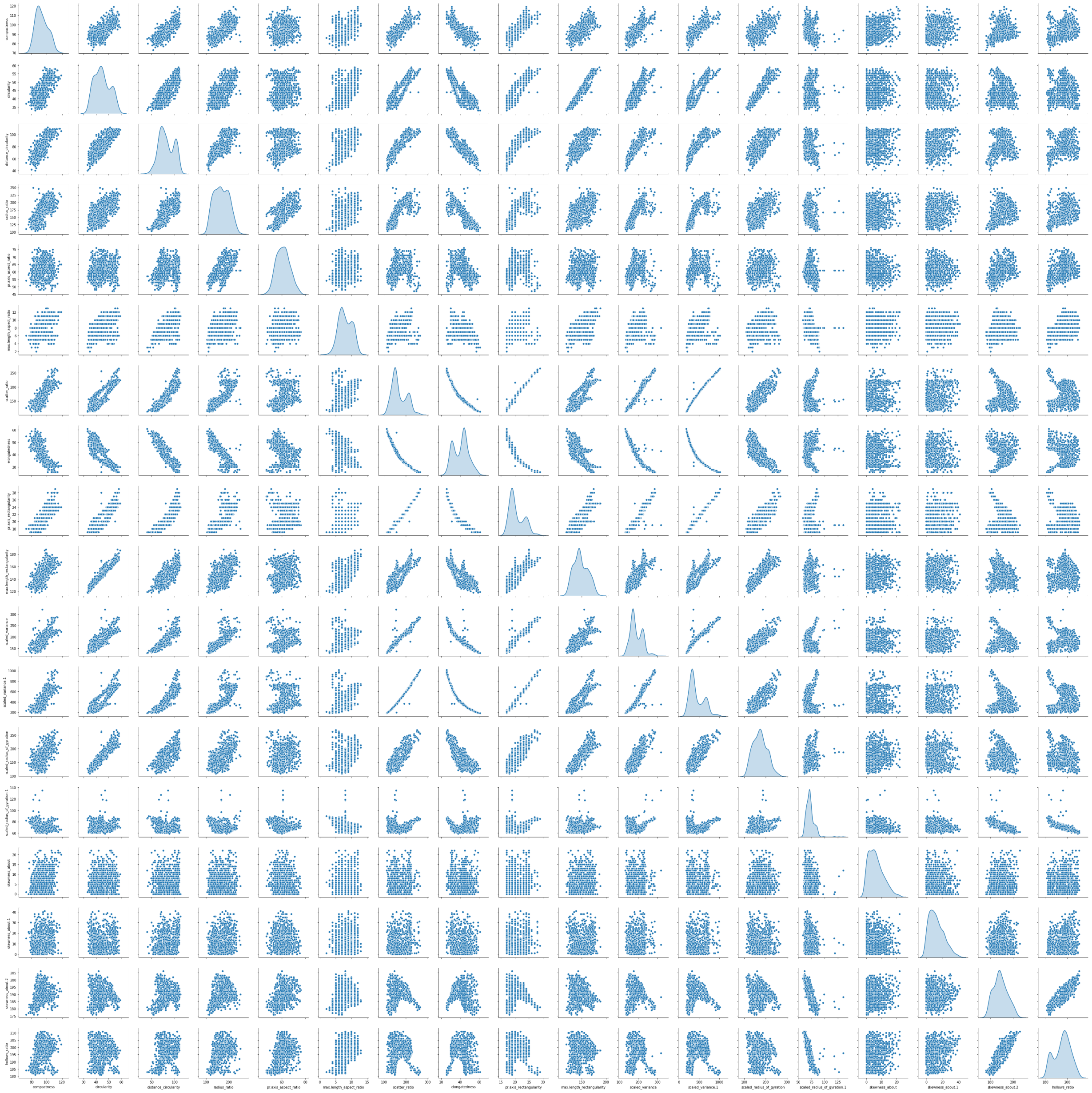
In [15]:

```
data.describe()
```

Out[15]:

	compactness	circularity	distance_circularity	radius_ratio	pr.axis_aspect_ratio	max.length_
count	846.000000	846.000000	846.000000	846.000000	846.000000	
mean	93.678487	44.823877	82.100473	168.230496	61.154846	
std	8.234474	6.134272	15.741569	32.018672	5.613458	
min	73.000000	33.000000	40.000000	104.000000	47.000000	
25%	87.000000	40.000000	70.000000	141.000000	57.000000	
50%	93.000000	44.000000	80.000000	167.000000	61.000000	
75%	100.000000	49.000000	98.000000	194.000000	65.000000	
max	119.000000	59.000000	112.000000	250.000000	76.000000	

In [16]:
```
sn.pairplot(data, diag_kind = 'kde')
plt.show()
```



Standardizing Data

In [17]:
```
from sklearn.preprocessing import StandardScaler
data = StandardScaler().fit_transform(data)
```

Splitting Data

```
In [18]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test= train_test_split(data, raw_data['class'], test_
```

Implementing SVM

```
In [19]: from sklearn import svm
         svma= svm.SVC()
         svma.fit(x_train, y_train)
         svm_prediction= svma.predict(x_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarni
ng: The default value of gamma will change from 'auto' to 'scale' in version 0.
22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
In [20]: from sklearn.metrics import classification_report
         from sklearn import metrics
         print(classification_report(y_test,svm_prediction))
         print("Accuracy:",metrics.accuracy_score(y_test, svm_prediction))
         metrics.confusion_matrix(y_test, svm_prediction)
```

```
              precision    recall  f1-score   support

           1       0.98      0.98      0.98        53
           2       0.98      0.98      0.98        60
           3       0.99      0.99      0.99       141

    accuracy                           0.99       254
   macro avg       0.99      0.99      0.99       254
weighted avg       0.99      0.99      0.99       254

Accuracy: 0.9881889763779528
```

```
Out[20]: array([[ 52,   1,   0],
                [  0,  59,   1],
                [  1,   0, 140]], dtype=int64)
```

K fold Validation

```
In [21]: from sklearn.model_selection import cross_val_score
         clf = svm.SVC(kernel='linear', C=1)
         scores = cross_val_score(clf, data, raw_data['class'], cv=5)
         scores
```

```
Out[21]: array([0.94117647, 0.95294118, 0.94705882, 0.95857988, 0.95209581])
```

In [22]:
```
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.95 (+/- 0.01)
```

Applying PCA

In [23]:
```
# Principal Component Analysis
from numpy import array
from sklearn.decomposition import PCA
# create the PCA instance
pca = PCA(8)
# fit on data
pca.fit(data)
print(pca.components_)
print(pca.explained_variance_)
# transform data
pcadata = pca.transform(data)
print(pcadata)
```

```
[[ 0.27273518  0.28741899  0.30232752  0.27011948  0.09689464  0.19559503
   0.31072282 -0.30890325  0.30755002  0.27825022  0.29676248  0.30688497
   0.26356413 -0.04037111  0.04217391  0.05878818  0.03739285  0.08389378]
 [-0.09402491  0.13257726 -0.04946733 -0.19626984 -0.26136031 -0.11137445
   0.0705384  -0.00837394  0.08256217  0.12324944  0.09259617  0.07752009
   0.21356472  0.48006222 -0.04169996 -0.10051474 -0.50896972 -0.51312981]
 [ 0.07263159  0.19190467 -0.04621247 -0.1126621  -0.06594468  0.20306458
  -0.10502353  0.1056912  -0.09627636  0.21855804 -0.16056044 -0.10705279
   0.18415833 -0.1008689   0.62746259 -0.58199598  0.0454722   0.04921917]
 [ 0.12924861 -0.07540313  0.11541801 -0.24458828 -0.6237589   0.23324969
   0.03143173  0.0453379   0.0567906  -0.00417757 -0.04156598  0.02793803
  -0.11765801 -0.13062663  0.34796787  0.54110943 -0.05623672  0.06510857]
 [ 0.16250793 -0.14182777 -0.0888463   0.13330928  0.08845701 -0.63156104
   0.07899073 -0.0645369   0.07642302 -0.25482915  0.16684589  0.1168687
  -0.00504046  0.16370232  0.5606893   0.10269531  0.18777601 -0.10693811]
 [ 0.23689551 -0.06283367 -0.01550653 -0.15034991 -0.55492977 -0.28721853
   0.1038946  -0.08474375  0.11127167 -0.06923381  0.09890243  0.1427139
  -0.06509349 -0.20013661 -0.3520953  -0.47126749  0.26222663  0.04719469]
 [ 0.2591551  -0.37473332  0.13690977  0.17132755  0.04059489  0.4352907
```

Splitting data

In [24]:
```
x_train, x_test, y_train, y_test= train_test_split(pcadata, raw_data['class'], t
```

Implementing SVM

In [25]:

```
svma= svm.SVC()
svma.fit(x_train, y_train)
svm_prediction= svma.predict(x_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarni
ng: The default value of gamma will change from 'auto' to 'scale' in version 0.
22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [26]:

```
print(classification_report(y_test,svm_prediction))
print("Accuracy:",metrics.accuracy_score(y_test, svm_prediction))
metrics.confusion_matrix(y_test, svm_prediction)
```

```
              precision    recall  f1-score   support

           1       0.92      0.92      0.92        53
           2       0.97      0.98      0.98        60
           3       0.98      0.97      0.98       141

    accuracy                           0.96       254
   macro avg       0.96      0.96      0.96       254
weighted avg       0.96      0.96      0.96       254

Accuracy: 0.9645669291338582
```

Out[26]:

```
array([[ 49,   2,   2],
       [  0,  59,   1],
       [  4,   0, 137]], dtype=int64)
```

K fold validation

In [27]:

```
from sklearn.model_selection import cross_val_score
clf = svm.SVC(kernel='linear', C=1)
scores = cross_val_score(clf, data, raw_data['class'], cv=5)
scores
```

Out[27]:

```
array([0.94117647, 0.95294118, 0.94705882, 0.95857988, 0.95209581])
```

In [28]:

```
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.95 (+/- 0.01)
```

In [30]:

```
#To increase the accuracy, I have replaced outliers with medians and also standar
#In this case, applying PCA is reducing the accuracy. And in this case, PCA is no
#k fold validation for both the cases is same
#Without PCA:
#Accuracy: 98.8 kfold accuracy: 95
#With PCA:
#Accuracy: 96.4 kfold accuracy: 95
```