In [56]:

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import graphviz

# Assignment 2
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB

# Assignment 3
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster import KMeans
from sklearn.preprocessing import normalize
from scipy import stats
```

In [57]:

```
data = pd.read_csv('../../Desktop/CS3481- Fundamentals of Data Science/vertebr
al_column_data/column_3C.dat', sep=' ', header=None)
data.columns = ['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lor
dosis_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric','degree_
spondylolisthesis numeric','class']
features = ['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordosi
s_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric','degree_spon
dylolisthesis numeric']
classes = ['disk hernia (DH)', 'spondylolisthesis (SL)', 'normal (NO)']
```

In [58]:

```
X=data.iloc[:,0:6].values
Y=data.iloc[:,6].values
#print (len(data))
#print (data.shape)
```

In [59]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0,random_sta
te=10,shuffle=True)
```

In [60]:

```
X = X_train
```

In [ ]:

```

```

In [14]:

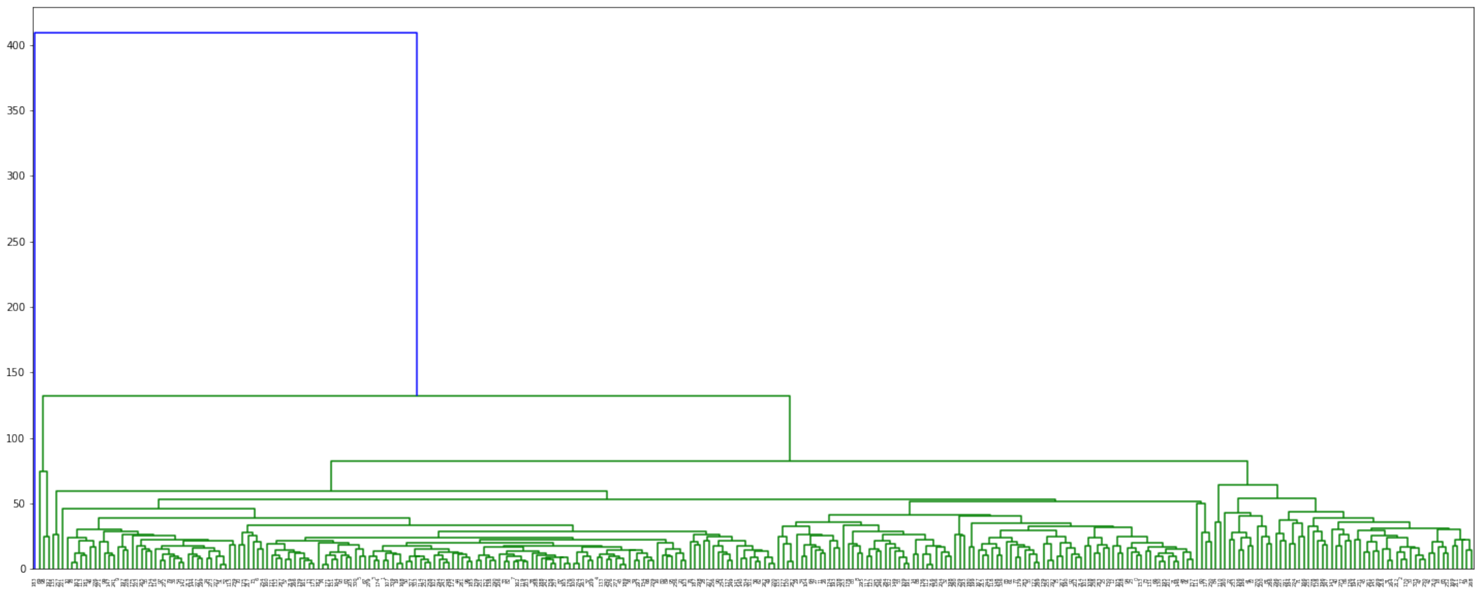
```
# Before performing any clustering algorithm, we first test and see how is our data is and if we need any
# pre-processing. As hierarchical clustering is performed first, we use group average version to build the
# clustering model as it's the intetimediate of single link and complete linkt to use it to test how is our raw data
```

In [15]:

```
Z = linkage(X, 'average')
```

In [16]:

```
plt.figure(figsize=(25, 10))
dendrogram(Z)
plt.show()
```



In [17]:

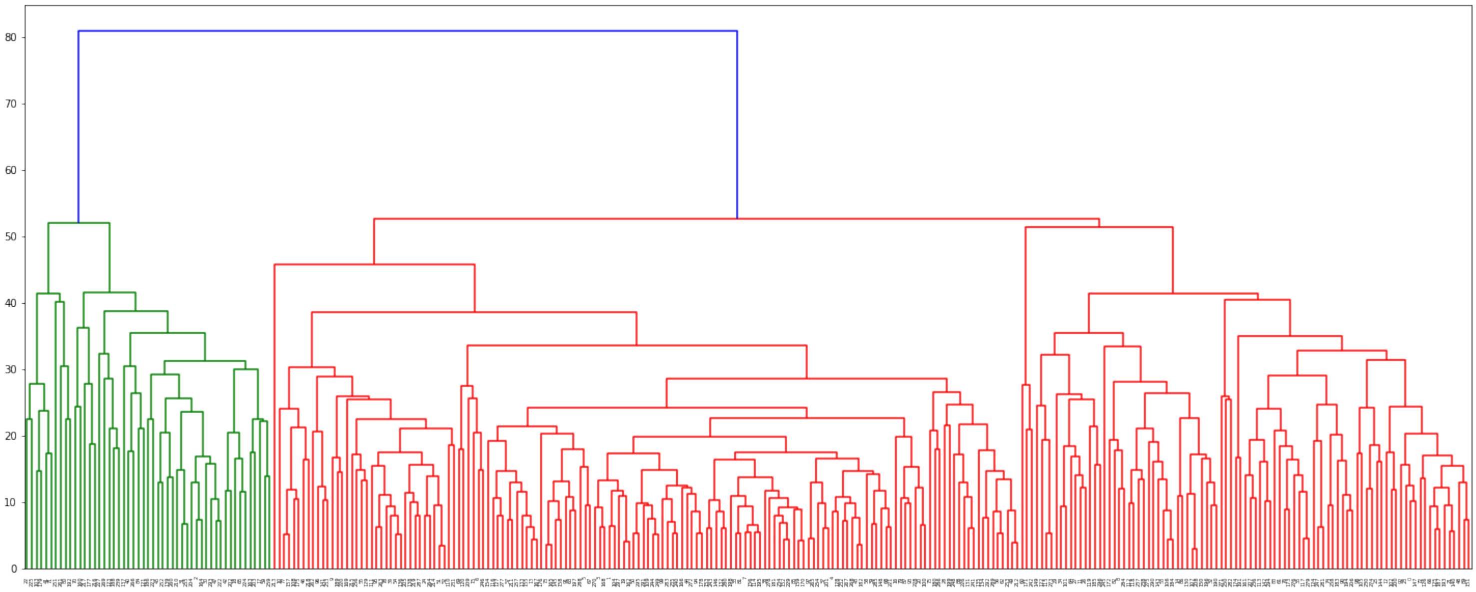
```
# As shown above, the distances between different clusters at lower level of t
he tree is less, clusters are not
# abled to distinguihed from each other, and the outliers are affecting the re
sult. The hierarchical clustering
# dendrogram resulted does not look good as well. As a first step in clusterin
g, the rule of thumb is to
# normalize/standaradize data as clustering uses distance meatures such as
# Euclidean distance. Thus, attributes with larger ranges will automatically h
ave more importance than other
# attributes which is false as we know, because all attributes must be equally
important. Thus normalizing the range
# of all attributes to scale using min/max or standardize using z-score is imp
ortant to bring all attributes to
# similar properties. These two are main normalization techniques. Normalizati
on will improve the clustering result.
# However, it is not clear, which is better, min-max normalization technique o
r z-score standardization technique.
# The former one scales the values between 0 and 1, and the latter one makes s
ure the values have mean 0, and
# standard deviation of 1. Both are normalization techniques in general, just
that standardization is also used in
# some cases as a terminology.
# However, before we test, which two of the normalization technique to use, we
first try to eliminate the outliers in
# our data. This is done by assuming normal distribution, and using z-score to
standardize all the values. We can
# assume normal distribution because from intuition we know the most important
data are usually clustered around the
# mean and the outliers are are the values at the far left or right away from
the mean. So we eliminate values which
# are 3 standard deviations away to include around 99.7% of the data as we kno
w from the 68–95–99.7 rule. Any more
# from this value, I am afraid too much outliers will be included, and any les
s must delete important information as
# well. I have tested 2, and 4, and my hypothesis was correct, so I sticked to
3.
```

In [18]:

```
# Eliminating outliers
```

In [19]:

```
df = pd.DataFrame(data=X)
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
X = df.values
Z = linkage(X, 'average')
plt.figure(figsize=(25, 10))
dendrogram(Z)
plt.show()
```

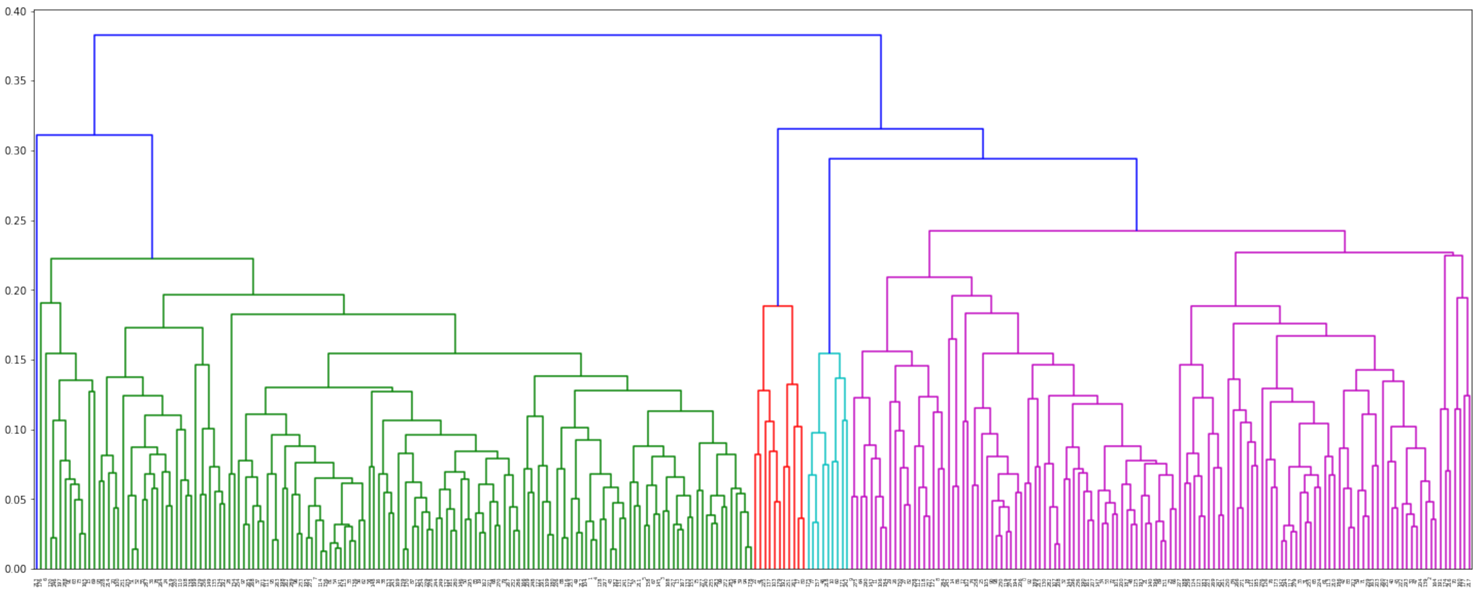


In [20]:

```
# As seen above, we can see that, eliminating outliers have improved the resul
t greatly compared to earlier result.
# However, we can furthur improve by reducing the imporsmotance of large scale
d attributes and normalize the data.
```

In [21]:

```
# First method of normalization - Scale to value between 0-1 --> min-max norma
lization
Xa = normalize(X)
Z = linkage(Xa, 'average')
plt.figure(figsize=(25, 10))
dendrogram(Z)
plt.show()
```



In [22]:

```
# As seen above, normalization has improved the clustering result. The model c
an distingush between different sets
# of data more easily and thus create more distinguishable clusters. Thus, it
can extract more information from the
# data.
```

In [23]:

```
# Second method of normalization - Standardize to mean 0, variance 1 --> z-sco
re normalization
```

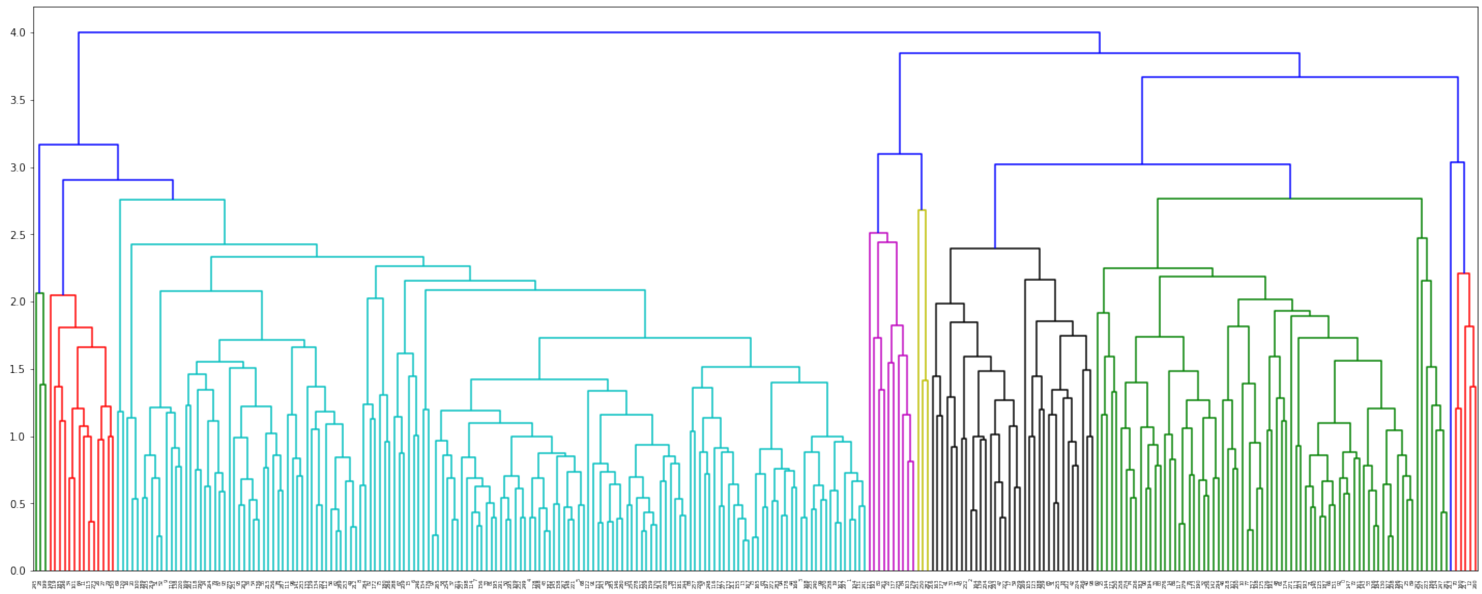
In [ ]:

```

```

In [24]:

```
Xb = stats.zscore(X)
Z = linkage(Xb, 'average')
plt.figure(figsize=(25, 10))
dendrogram(Z)
plt.show()
```



In [25]:

```
# Compared to the previous normalization technique, this one is able to extrac
t more information, distinguish more
# clusters from each other. So we will use  this normalization technique and p
roceed further.
```

In [9]:

```
X = X_train
df = pd.DataFrame(data=X)
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
X = df.values
X = stats.zscore(X)
```

In [ ]:

```

```

In [ ]:

```
# Question 1
```

In [27]:

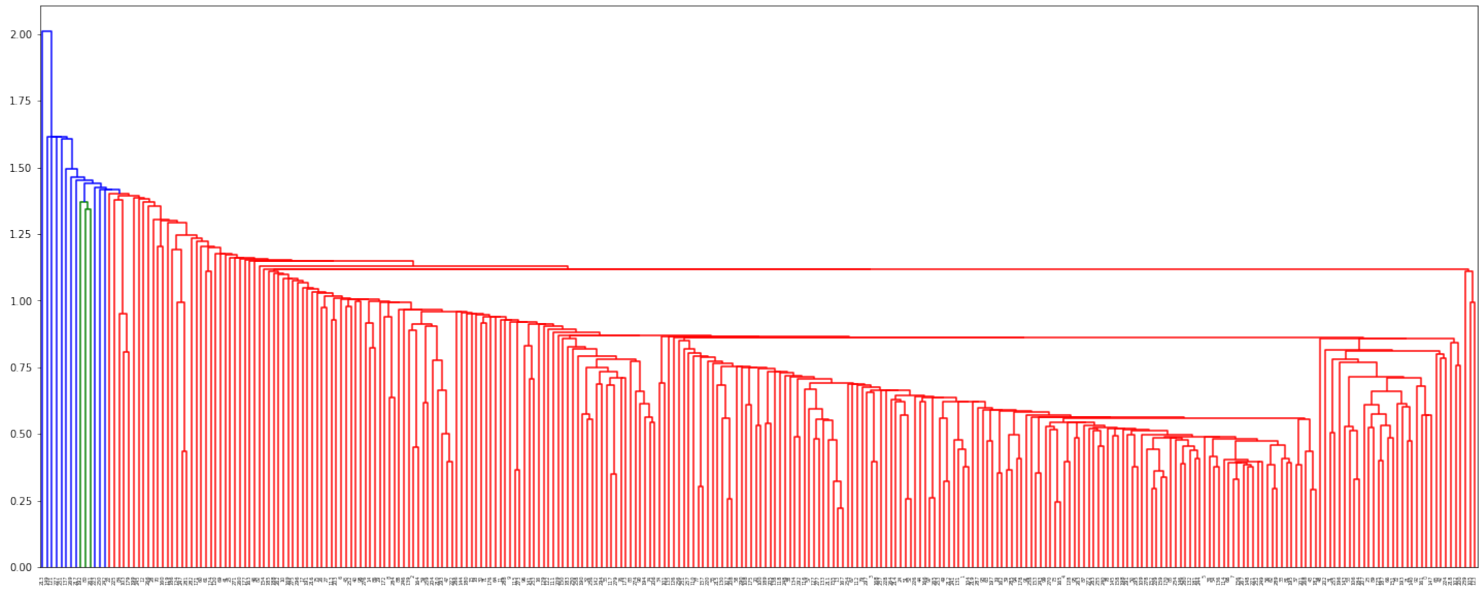
```
# Single link Hierarchical clustering
```

In [28]:

```
Z1 = linkage(X, 'single')
```

In [29]:

```
plt.figure(figsize=(25, 10))
dendrogram(Z1)
plt.show()
```



In [31]:

```
kclusters1 = fcluster(Z1, 3, criterion='maxclust')
kclusters1
```

Out[31]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

In [ ]:

```

```

In [32]:

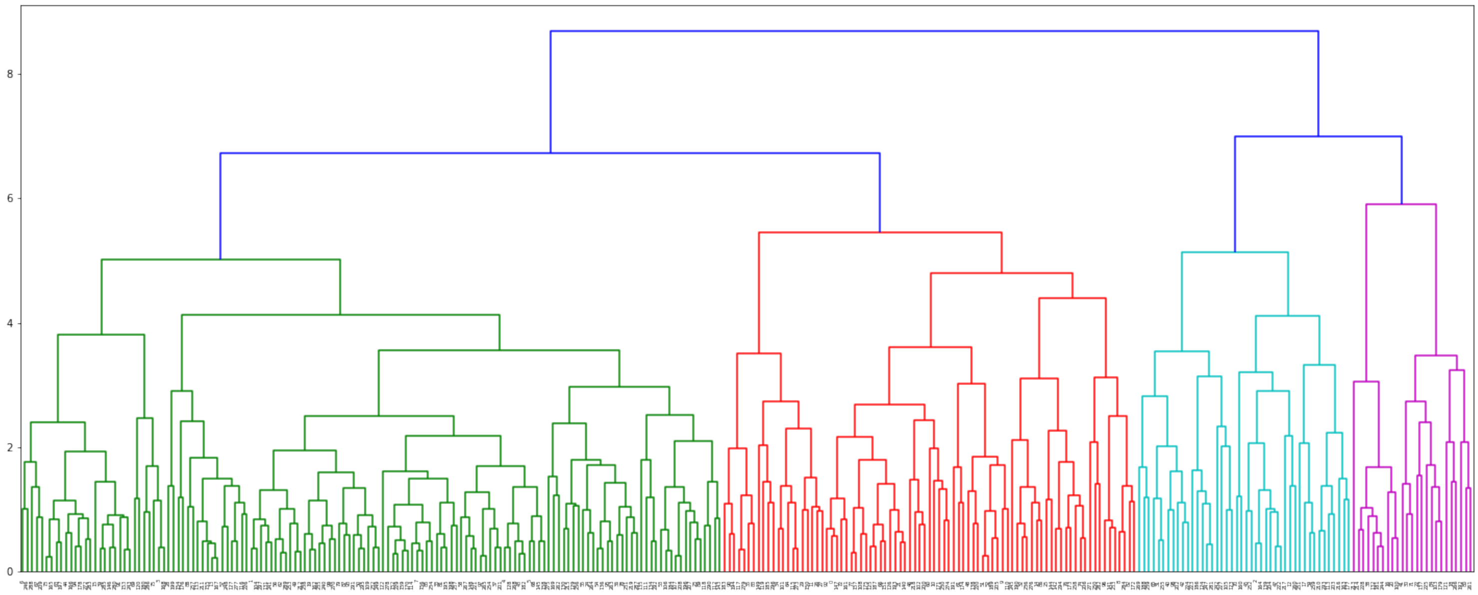
```
# Complete link Hierarchical clustering
```

In [33]:

```
Z2 = linkage(X, 'complete')
```

In [34]:

```
plt.figure(figsize=(25, 10))
dendrogram(Z2)
plt.show()
```



In [35]:

```
kclusters2 = fcluster(Z2, 3, criterion='maxclust')
kclusters2
```

Out[35]:

```
array([1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 3, 2, 3, 1,
3, 1,
       3, 1, 1, 1, 1, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 1, 3, 1, 2, 3,
2, 1,
       1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
1, 2,
       1, 1, 1, 1, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 1, 3, 1, 2, 1, 1,
1, 1,
       1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1, 2, 2, 1, 1, 1, 1, 1,
1, 1,
       3, 1, 1, 1, 1, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
       1, 2, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 3, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 3, 3, 1, 2, 2,
1, 1,
       1, 1, 2, 2, 2, 3, 1, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
1, 1,
       2, 1, 3, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 3,
1, 1,
       1, 1, 3, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1], dtype=int32)
```

In [ ]:

```

```

In [36]:

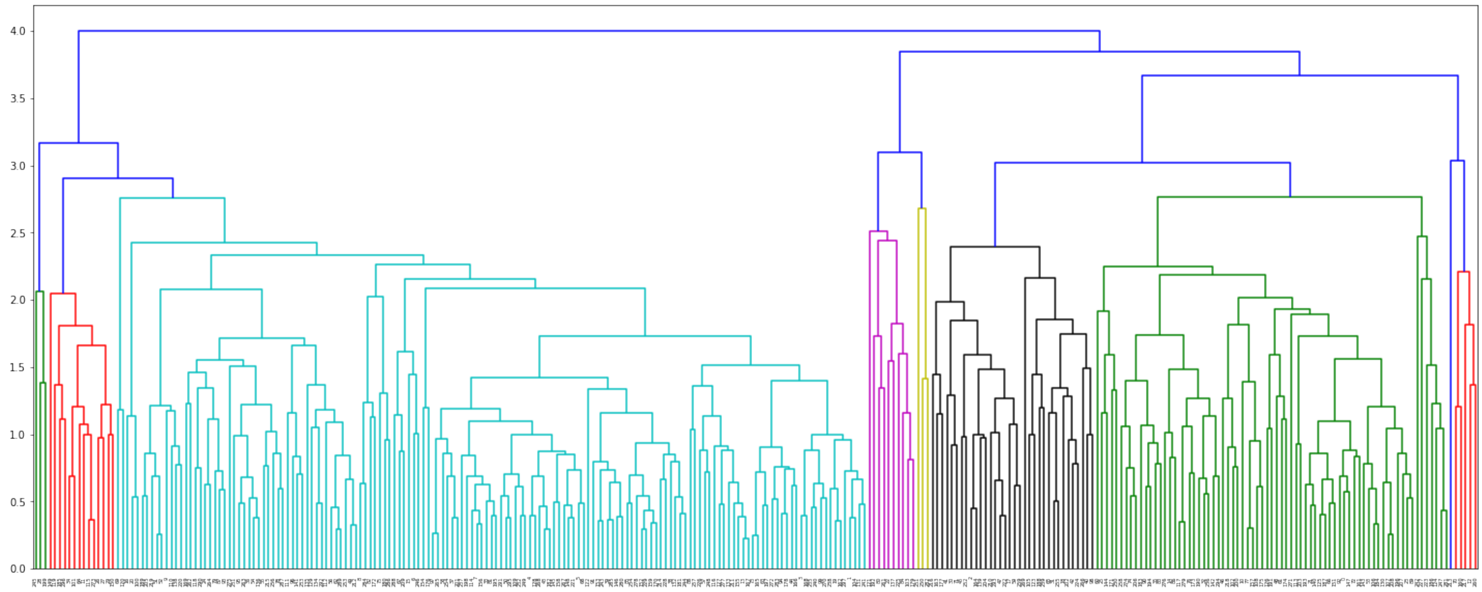
```
# Group average Hierarchical clustering
```

In [38]:

```
Z3 = linkage(X, 'average')
```

In [39]:

```
plt.figure(figsize=(25, 10))
dendrogram(Z3)
plt.show()
```



In [40]:

```
kclusters3 = fcluster(Z3, 3, criterion='maxclust')
kclusters3
```

Out[40]:

```
array([3, 1, 3, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 1, 3, 1, 1, 3, 3, 1,
1, 3,
       2, 3, 1, 3, 1, 1, 1, 1, 3, 3, 3, 1, 1, 3, 1, 1, 1, 1, 3, 3,
3, 1,
       1, 3, 3, 3, 3, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 3, 3, 3, 1, 1,
1, 3,
       3, 1, 1, 1, 3, 3, 3, 1, 3, 1, 3, 3, 1, 1, 2, 1, 1, 3, 2, 1,
3, 1,
       1, 3, 3, 1, 3, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 2, 1, 3, 3, 3,
3, 1,
       1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 2, 1, 3, 3, 3, 3, 1, 1, 1,
3, 1,
       1, 1, 1, 1, 1, 2, 1, 3, 3, 1, 3, 3, 3, 1, 1, 3, 1, 1, 1, 3,
1, 1,
       1, 1, 1, 3, 1, 1, 3, 3, 1, 3, 3, 1, 1, 1, 1, 1, 1, 3, 1, 3,
3, 3,
       1, 3, 1, 2, 1, 1, 1, 3, 3, 1, 3, 3, 3, 1, 3, 3, 2, 3, 3, 1,
3, 1,
       1, 1, 3, 1, 3, 3, 3, 1, 3, 3, 3, 1, 3, 1, 1, 3, 1, 1, 3, 3,
3, 1,
       1, 1, 3, 3, 3, 2, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 3, 1, 1, 3,
1, 1,
       3, 1, 1, 1, 1, 3, 1, 1, 2, 2, 3, 1, 1, 3, 1, 1, 3, 3, 3, 2,
1, 1,
       1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 3, 1, 3, 1, 1, 3, 1, 3, 2, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1], dtype=int32)
```

In [ ]:

```

```

In [41]:

```
# Comparisons between the 3 different versions of hierarchical clustering
```

In [43]:

```
# As seen from the dendrogram and also from the partitional clustering of the hierarchical clustering with 3 clusters
# by cutting the dendrogram at a certain level which basically results in partitional clustering, single link
# is not doing as good as complete and average versions of the model. Complete and average versions can extract more
# information from data and distinguish clusters from each other more readily than single link version. In dendrogram,
# it can be seen form the different colors present to differentiate the clusters. There are more colors in complete
# and average versions. And in partitional clustering, with 3 cluster partitions, single link version
# classifies most of the data into cluster "1", and there is little or no cluster "2" and "3" at all. In complete
# version, there is some more variety and has more cluster "2" and "3", although cluster "1" is still the most among
# the 3 clusters. And in group version, there's much more of "3" or "2" comapared to previous 2 versions. So average
# version of Hierarchical clustering has more uniform number of cluster "1", "2" and "3". Thus it can distinguish
# different clusters more uniformly. In conclusion, the intermediate version of single link and complete link,
# which is the average version is a more appropriate method of performing Hierarchical clusterin in this case.
# Also, complete link version is ranked second best, and single link version is the worse among all 3.
```

In [ ]:

```

```

In [44]:

```
# Question 2
```

In [ ]:

```
# I will choose complete link and average versions of the Hierarchical cluster
ing to identify possible patterns and
# investigate the type of information we can get from the results obtained. Si
ngle link version is not going to be
# studied as it gives very poor representational data and information.
# Between complete and average versions, it can be seen that in general the gr
oup version connected between clusters at
# lowever level of the dendrogram with much less distance values and this can
intuitively mean that group version
# could recognize smaller colonies more easily and lable them as distinguishab
le clusters. This is why, there are
# more colors in the dendrogram and more clusters. At higher level of the dend
rogram , it can be seen that complete
# version has distance values of different clusters much larger, than that of
group, this is because complete version
# defines the minimum distance as the largest distance of any two point of two
clusters.
# For complete version, it can be seen that the clustering could recognize 4 m
ain clusters each of reasonable size.
# However the cluster size gradually decreases as seen from the dendogram amon
g its clusters. This gives the
# information that, there are 4 main clusters of different gradually decreasin
g sizes.
# For average version, it recognizes around 9 clusters, and it's seen that it
can distinguish more specific clusters.
# It can be thought that, the 4 main clusters in complete version have parts w
here another clusters can be formed
# which can be seen from the average version. This shows that there's some var
iety in the data itself in the 4 main
# clusters in complete version.
# For complete version, the difference of the distance values of the merges be
tween the higher and lower levels of
# dendrogram is large. This signifies, smaller similar clusters are grouped fi
rst, and the larger clusters which
# are futhur from each other are grouped last. This is what we want in cluster
ing. However this relationship is not
# as apparent in group version, so this could be an indicator that complete ve
rsion is better.
# If the data is interepreted at 2 cluster level of dendrogram, the average ve
rsion has 2 clusters of somewhat more
# equal size of clusters. However, it's quite unequal for the complete version
. But when the two clusters are grouped
# into one, the distance value for average version is much less than that of c
omplete version. This implies,
# grouping the data into 2 equal sizes does not show the proper cluster inform
ation because clusters are supposed to
# show that differences in data in the form of clusters. The higher the distan
ce value, the higher the difference
# between the clusters. Thus, the larger distance values usually, signifies a
correct clustering of data as we want
# to distinguish between different types of data in terms of clusters as much
as possible.
```

In [ ]:

```

```

In [106]:

```
# Question 3
```

In [ ]:

```
# The comparision done in this question is done by comparing the number of dat
a points in the clusters. For example,
# in 3 clusters, cluster A, B and C are just named to differentiate 3 differen
t clusters and compared to another
# method. But the cluster A in the new method does not refer to the same type
of class label of cluster A from another
# method. The names are just to differentiate the 3 different clusters. And th
e amount of data points in the clusters
# is used to compare and see how similar are the proportions. This can easily
and roughly tell us how good the
# clustering method is performing or similar to another clustering method by c
omparing the proportions of data points
# in the clusters from one model/clustering method to that of another
# 3 clusters will be used only as the class labels in the original dataset has
3 labels
# The performance of partitional clustering derived from hierarchical clusteri
ng will be compared to that of K-means
# and the proportions of original class labels will be used to check which is
performing better. Both, the complete
# and average versions of hierarchical clustering will be used to test.
```

In [107]:

```
# For complete link version
```

In [177]:

```
kclusters2 = fcluster(Z2, 3, criterion='maxclust')
kk = list(kclusters2)
print("From clustering solutions:")
print("Cluster A: ", kk.count(1))
print("Cluster B: ", kk.count(2))
print("Cluster C: ", kk.count(3))
```

```
From clustering solutions:
Cluster A:  229
Cluster B:  44
Cluster C:  25
```

In [109]:

```
km2 = KMeans(n_clusters=3)
km2.fit(X)
kk = list(km2.labels_)
print("From K-means:")
print("Cluster A: ", kk.count(0))
print("Cluster B: ", kk.count(1))
print("Cluster C: ", kk.count(2))
```

```
Cluster A:  108
Cluster B:  56
Cluster C:  134
```

In [144]:

```
kk = list(Y_train)
print("From original set of class labels:")
print("Cluster A: ", kk.count('DH'))
print("Cluster B: ", kk.count('SL'))
print("Cluster C: ", kk.count('NO'))
```

```
From original set of class labels:
Cluster A:  60
Cluster B:  150
Cluster C:  100
```

In [ ]:

```
# (a)
# Comapring for 3 clusters, the partitional clusterings derived from hierarchi
cal clusterings and k-means, it can be
# observed that k-means distributes the data points between the 3 clusters mor
e evenly. In the clustering solution
# the cluster A has abnormally more data points.
# (b)
# By comparing the 3 partitional clustering solution and K-means to the origin
al class labels of the dataset, we can
# see how well the clustering algorithms clustered the data properly with corr
ect proportions.
# Judging from the labels from the original dataset, one cluster has 60 data p
oints, one has 150, and the last one
# has 100. I didn't eliminate the outliers because it is small compared to the
original dataset size of only 12 points
# I just need to compare the proportions to have the conclusion I want to get.
Thus, two clusters need to have large
# number of data points and the difference in the number of data points betwee
n each clusters is around 50 roughly.
# Comparing these facts with the two clustering models, K-means is performing
relatively better.
```

In [ ]:

```
```

In [147]:

```
# For average version
```

In [153]:

```
kclusters3 = fcluster(Z3, 3, criterion='maxclust')
kk = list(kclusters3)
print("From clustering solutions:")
print("Cluster A: ", kk.count(1))
print("Cluster B: ", kk.count(2))
print("Cluster C: ", kk.count(3))
```

```
From clustering solutions:
Cluster A:  172
Cluster B:  13
Cluster C:  113
```

In [172]:

```
km2 = KMeans(n_clusters=3)
km2.fit(X)
kk = list(km2.labels_)
print("From K-means:")
print("Cluster A: ", kk.count(0))
print("Cluster B: ", kk.count(1))
print("Cluster C: ", kk.count(2))
```

```
From K-means:
Cluster A:  56
Cluster B:  134
Cluster C:  108
```

In [178]:

```
kk = list(Y_train)
print("From original set of class labels:")
print("Cluster A: ", kk.count('DH'))
print("Cluster B: ", kk.count('SL'))
print("Cluster C: ", kk.count('NO'))
```

```
From original set of class labels:
Cluster A:  60
Cluster B:  150
Cluster C:  100
```

In [180]:

```
# (a)
# The derived clustering solution, is performing more similar to that of k-mea
ns in this case, as not only one cluster
# abnormally has high number of data points, but two clusters have, which is s
imilar to that of K-means. But still,
# in the clustering solution the size of the largest cluster is much larger th
an the smallest cluster. The
# smallest cluster has abnotmally small number of data points. K-means is dist
ributing the data points more evenly.
# (b)
# Comparing the two models to proportions of class labels from the original da
taset, we can see that K-means is
# giving clusters which have proportions much similar to that of original data
set class labels. The smallest cluster
# in K-means has 56 data points in K-means very close to original small cluste
r value of 60 and the largest cluster
# is 134 to the original value of 150, which is much more close to that of wha
t clustering solutions is giving which
# is 172. Similarly, for the medium cluster, the number of data points is more
closer to the original set, for K-means
```

In [183]:

```
# In conclusion, K-means is performing better than the partitional clusterin s
olutions derived from the hierarchical
# clustering method in both complete version and average version.
```

In [ ]:

```
```

In [184]:

```
# Question 4
```

In [83]:

```
# In this part, different subsets of attributes from the dataset are selected
and hierarchical clustering is performed
# on the new subset and then compared to the original hierarchical clustering
structure. I will create subsets,
# each will have 4 attributes, by eliminating 2 of the attributes. An arbitrar
y simple subset selection I used is
# by eliminating the first 2 attributes first, then the next two, and finally,
the last two. This way, each
# attribute is eliminated once only and is considered in the subset twice so i
t's fair. Many other combinations can
# also form but this is good enough to generalize. The hierarchical clustering
version I will stick to is the complete
# link version because it generalizes the data clustering well, its performanc
e isn't poor nor the clsuterings are
# so detailed.
```

In [85]:

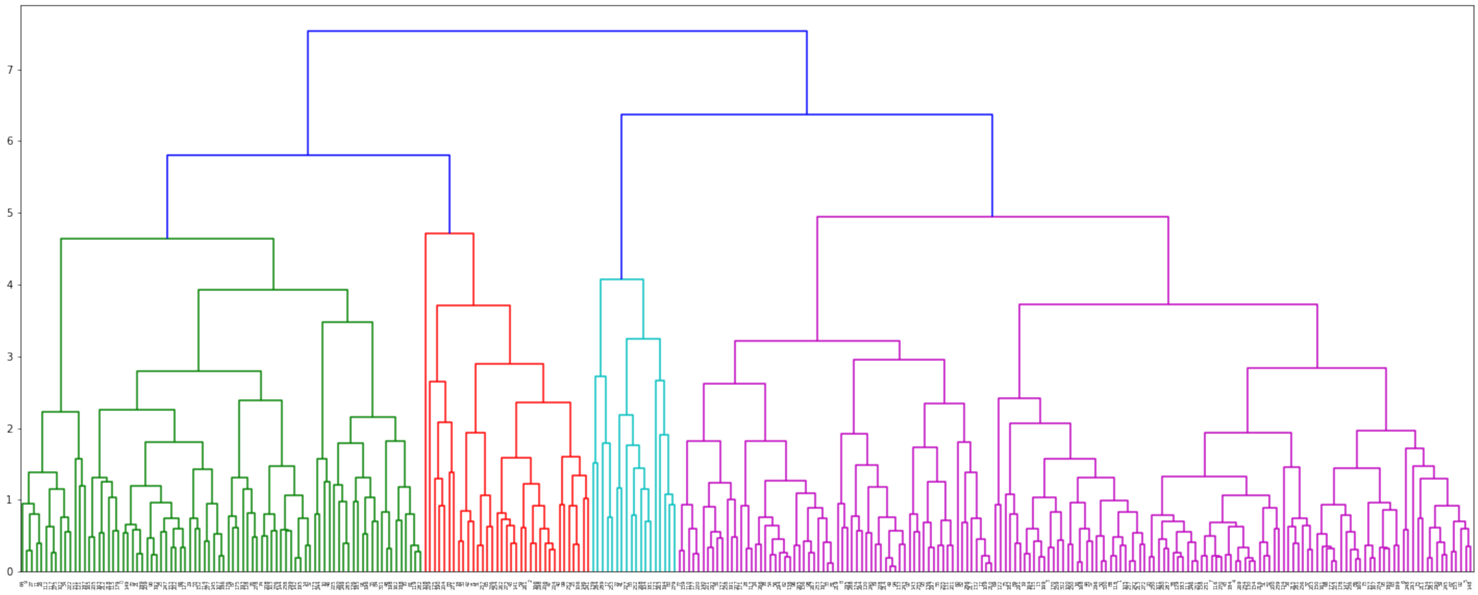
```
# Round 1 - Eliminate 0,1
```

In [81]:

```
X = X_train
df = pd.DataFrame(data=X)
df.columns = features
del df[features[0]]
del df[features[1]]
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
XX = df.values
XX = stats.zscore(XX)
```

In [84]:

```
ZZ1 = linkage(XX, 'complete')
plt.figure(figsize=(25, 10))
dendrogram(ZZ1)
plt.show()
```



In [ ]:

```
# Comparing the result from the subset dataset to the original clustering, it can be seen that again, 4 main clusters
# are found but are of different proportions than that of original one. The disatance values of the clusters at
# higher level of the dendrogram is smaller in the new result this shows that, the original clusterings could
# cluster data more properly for data which are apart.
```

In [ ]:

```

```

In [90]:

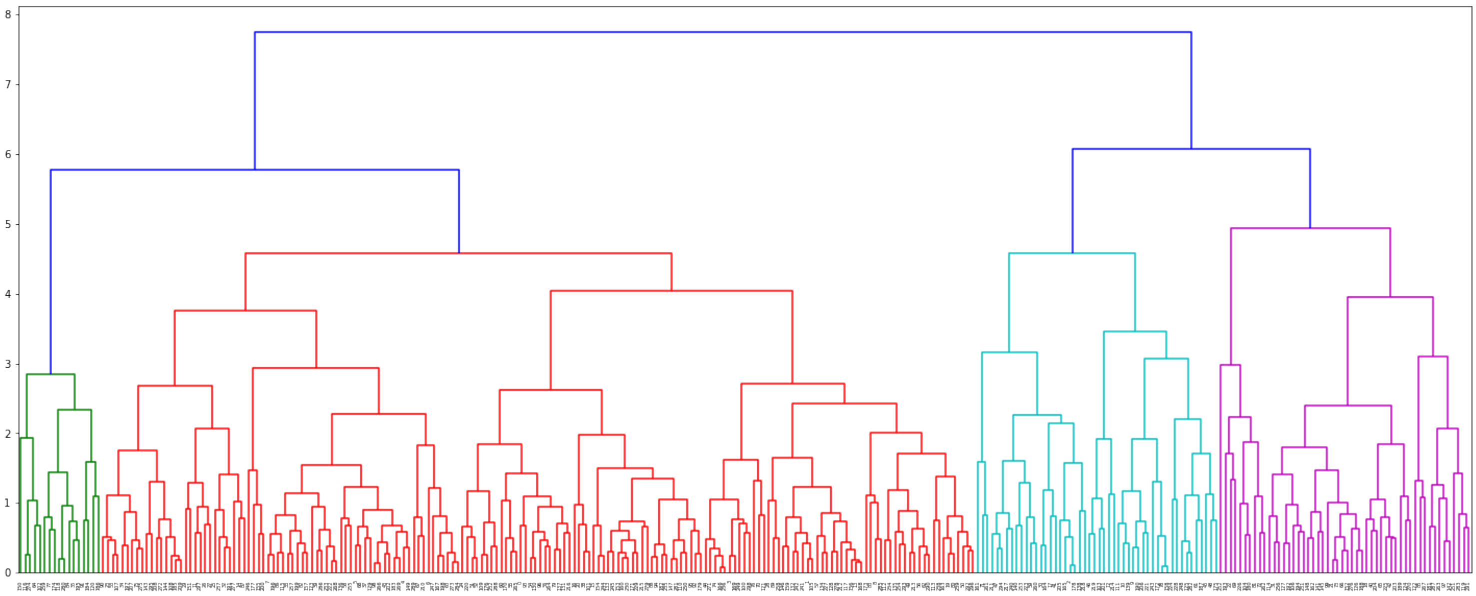
```
# Round 2 - Eliminate 2,3
```

In [91]:

```
X = X_train
df = pd.DataFrame(data=X)
df.columns = features
del df[features[2]]
del df[features[3]]
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
XX = df.values
XX = stats.zscore(XX)
```

In [92]:

```
ZZ2 = linkage(XX, 'complete')
plt.figure(figsize=(25, 10))
dendrogram(ZZ2)
plt.show()
```



In [ ]:

```
# For the new result, it has shown that it can still show 4 main clusters, but one of them is shrunk considerably,
# and its data points joined the neighbouring cluster. On the other half of the data, the clusters remain almost the
# same. Thus, this shows that, there's a clear decline in cluster recognizition affecting a very specific group
# of data points.
```

In [ ]:

```

```

In [93]:

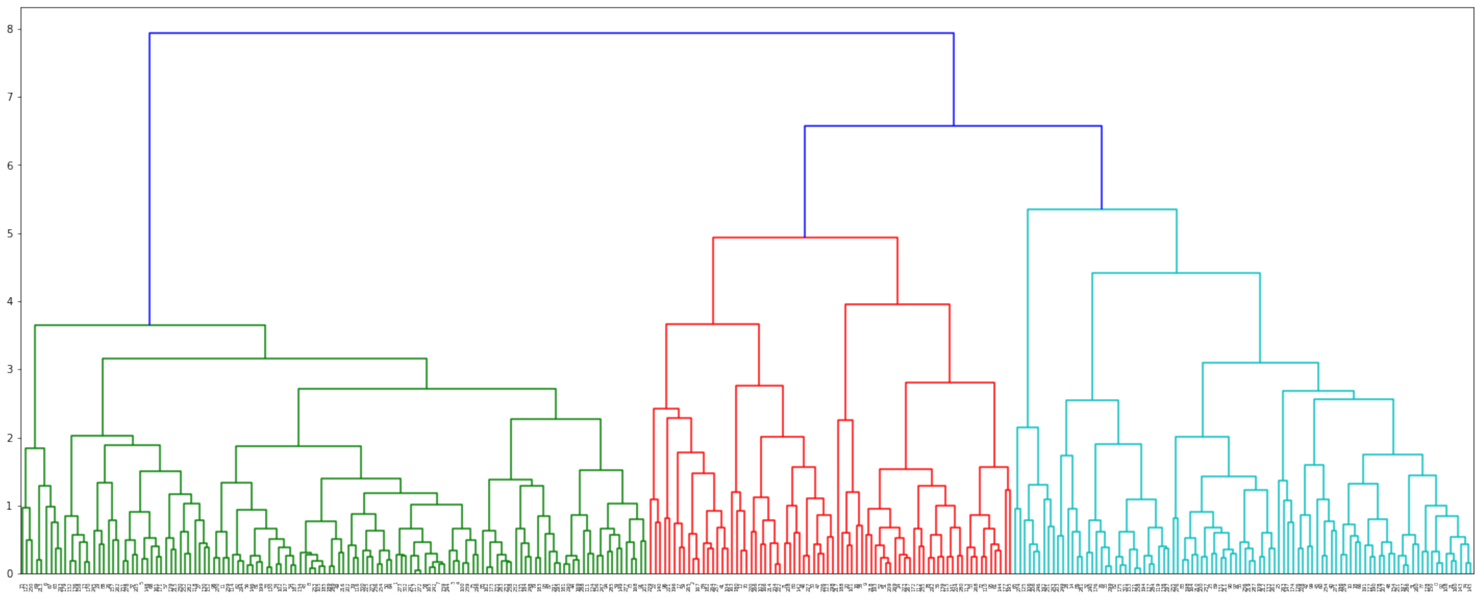
```
# Round 3 - Eliminate 4,5
```

In [94]:

```
X = X_train
df = pd.DataFrame(data=X)
df.columns = features
del df[features[4]]
del df[features[5]]
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
XX = df.values
XX = stats.zscore(XX)
```

In [95]:

```
ZZ3 = linkage(XX, 'complete')
plt.figure(figsize=(25, 10))
dendrogram(ZZ3)
plt.show()
```



In [96]:

```
# The result of this new clustering is quite pleasing to see as there are clea
r 3 distinct clusters formed. And all
# of them are in relatively good size indicating, it's not a small cluster of
outliers, but group of data points
# belonging to a dinstint group. Also, we know that the original dataset, ther
e are 3 labels, so this new result
# can show/predict that more relatively. I think it's because, as decrease in
number of attributes, resulted in
# less complication of clustering the data, so the clustering could be general
ized more easily to represent the
# whole data. The new clustering recognizes 3 main clusters represented by the
3 colors which is 1 less than the
# original clustering.
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```

```

In [ ]:

```

```