Coursera Peer-Graded Assignment:Unsupervised Machine Learning

by-Aarohan Verma

Background

User knowledge is where a person who uses knowledge generated through research to make informed decisions about policies, programs and/or best practices. This dataset was part of the Ph.D. Thesis of Dr. Hamdi Tolga Kahraman from Turkey back in 2009. It is an unlabelled dataset containing 5 features as explained in the column descriptions. The data contains real information about the student's knowledge status about the subject of Electrical DC Machines. The insipiration is to find clusters within data, explore their properties and make meaningful conclusions and actions / recommendations that might add value to the process from where the data was obtained.

Data Set Characteristics: Multivariate

Number of Instances: 403

Area: Education

Attribute Characteristics: Real

Number of Attributes: 5

Associated Tasks: Clustering/Classification

Attribute Information:

STG (The degree of study time for goal object materails),

SCG (The degree of repetition number of user for goal object materails)

STR (The degree of study time of user for related objects with goal object)

LPR (The exam performance of user for related objects with goal object)

PEG (The exam performance of user for goal objects)

UNS (The knowledge level of user)

Class Distribution

Very Low: 50

Low:129

Middle: 122

High 130

Total:403

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

In [2]:

In [2]:
```
df_train=pd.read_excel('Data_User_Modeling_Dataset.xls')
```

In [3]:
```
df_train.head()
```

Out[3]:

	STG	SCG	STR	LPR	PEG	UNS
0	0.00	0.00	0.00	0.00	0.00	Very Low
1	0.08	0.08	0.10	0.24	0.90	High
2	0.06	0.06	0.05	0.25	0.33	Low
3	0.10	0.10	0.15	0.65	0.30	Middle
4	0.08	0.08	0.08	0.98	0.24	Low

Exploratory Data Analysis

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

Out[4]:

	STG	SCG	STR	LPR	PEG
count	403.000000	403.000000	403.000000	403.000000	403.000000
mean	0.353141	0.355940	0.457655	0.431342	0.456360
std	0.212018	0.215531	0.246684	0.257545	0.266775
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.200000	0.200000	0.265000	0.250000	0.250000
50%	0.300000	0.300000	0.440000	0.330000	0.400000
75%	0.480000	0.510000	0.680000	0.650000	0.660000
max	0.990000	0.900000	0.950000	0.990000	0.990000

Clearly, values for all the columns lie in range 0-1, hence there is no need of scaling

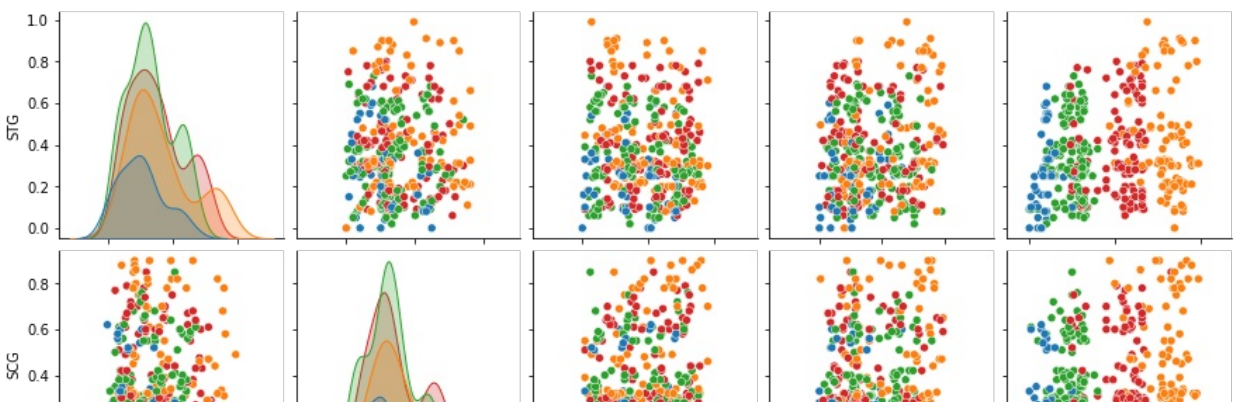
In [5]:
```
df_train.info()
```

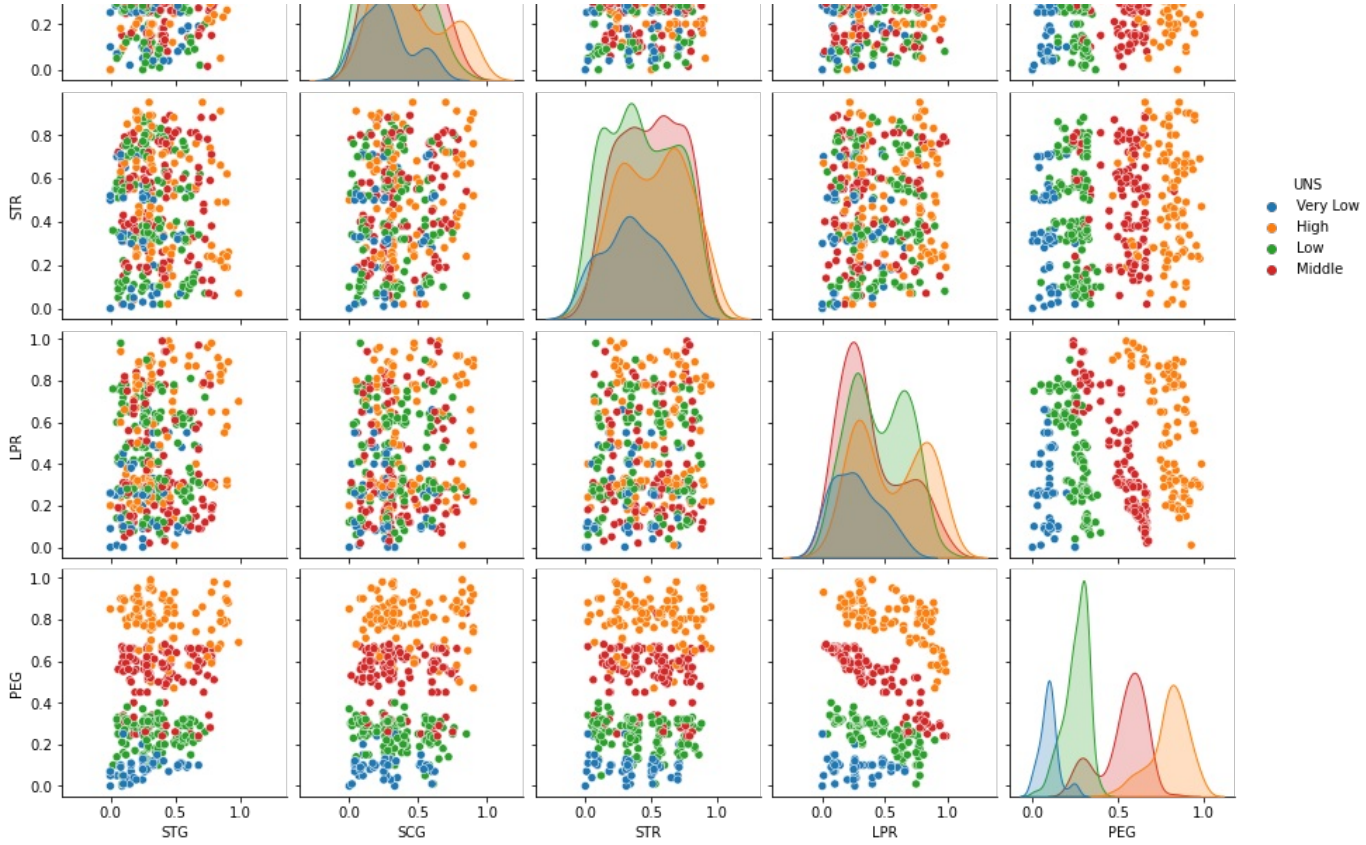
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 403 entries, 0 to 402
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   STG     403 non-null    float64
 1   SCG     403 non-null    float64
 2   STR     403 non-null    float64
 3   LPR     403 non-null    float64
 4   PEG     403 non-null    float64
 5   UNS     403 non-null    object 
dtypes: float64(5), object(1)
memory usage: 19.0+ KB
```

Clearly, there are no categorical attributes, hence no need of encoding

In [6]:
```
sns.pairplot(df_train,hue='UNS')
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x225fca580a0>
```





Dataset has high degree of entropy

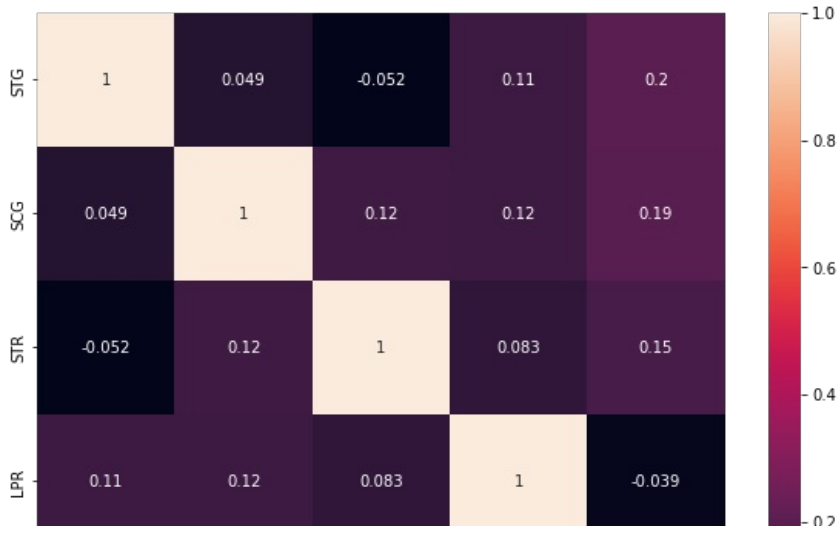
Notice: The plot above shows the distribution in each variable / column. We can focus our attention on the PEG box. As we know that PEG is exam performance of user for goal objects. We can see that the higher result exam of a user, this user tends to be classified into the class of "high knowledge" and vice versa.

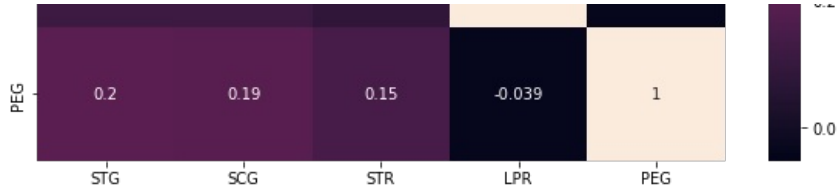
In [7]:
```
df_train.isnull().sum()
```

Out[7]:
```
STG    0
SCG    0
STR    0
LPR    0
PEG    0
UNS    0
dtype: int64
```

In [8]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df_train.iloc[:,:-1].corr(),annot=True)
```

Out[8]:
```
<AxesSubplot:>
```



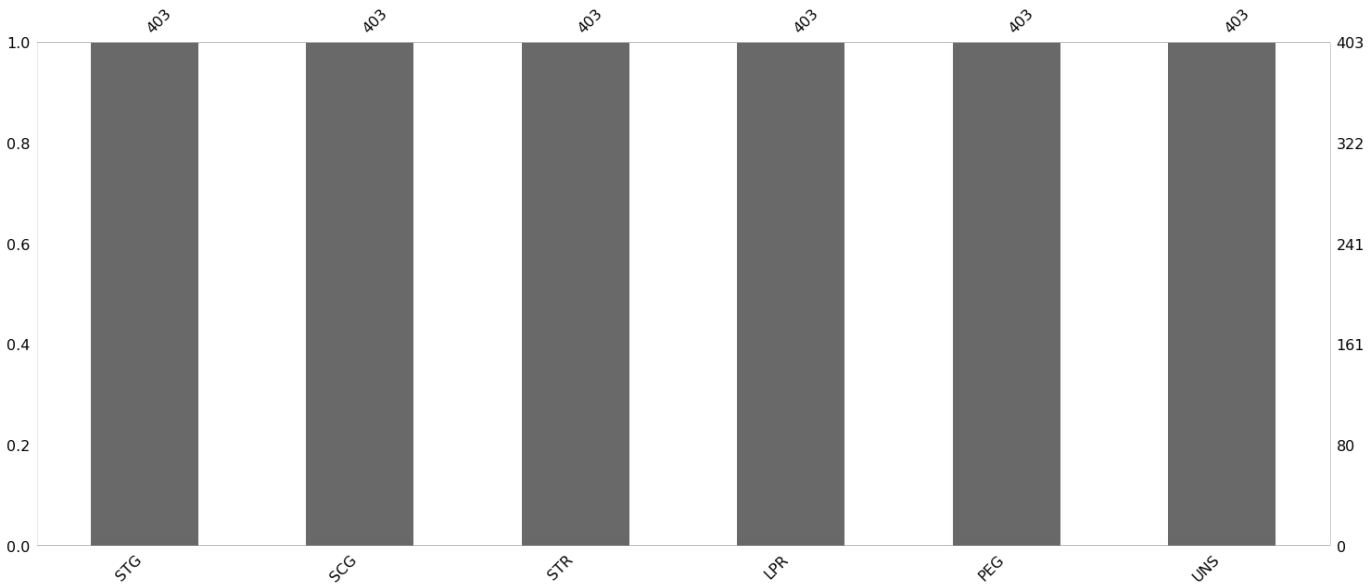


In [9]:
```
df_train.shape
```

Out[9]:
```
(403, 6)
```

In [10]:
```
msno.bar(df_train)
```

Out[10]:
```
<AxesSubplot:>
```



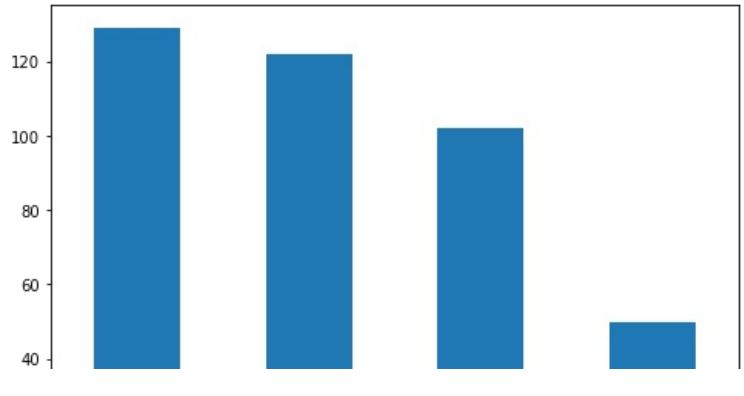
Clearly, there are no missing values to impute

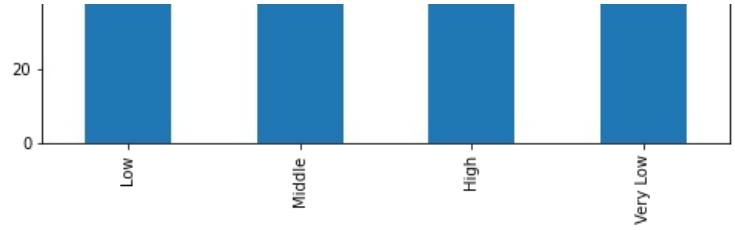
Distribution of Classes

In [11]:
```
df_train['UNS'].value_counts()
```

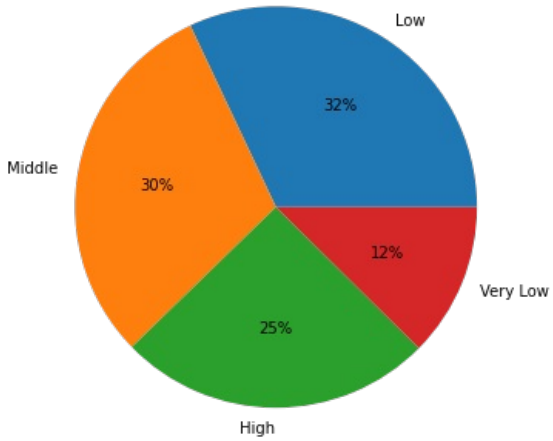
Out[11]:
```
Low         129
Middle      122
High        102
Very Low     50
Name: UNS, dtype: int64
```

In [12]:
```
plt.figure(figsize=(8,6))
df_train['UNS'].value_counts().plot(kind='bar')
plt.show();
```





In [13]:
```
plt.figure(figsize=(8,6))
plt.pie(df_train['UNS'].value_counts(),labels=df_train['UNS'].value_counts().index,autopct='%.0f%%')
plt.show()
```



Linear Discriminant Analysis

Un-comment Cell 23 to perform Cluster analysis on LDA transformed dataset

In [14]:
```
df_train.head()
```

Out[14]:

	STG	SCG	STR	LPR	PEG	UNS
0	0.00	0.00	0.00	0.00	0.00	Very Low
1	0.08	0.08	0.10	0.24	0.90	High
2	0.06	0.06	0.05	0.25	0.33	Low
3	0.10	0.10	0.15	0.65	0.30	Middle
4	0.08	0.08	0.08	0.98	0.24	Low

In [15]:
```
df_train['UNS_Numerical']=df_train['UNS'].replace({'Very Low':0,'Low':1,'Middle':2,'High':3})
data=df_train[['STG','SCG','STR','LPR','PEG']]
target=df_train['UNS_Numerical']
```

In [16]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [17]:
```
lda=LinearDiscriminantAnalysis(n_components=3)
data_r2=lda.fit(data,target).transform(data)
```

In [18]:
```
lda.explained_variance_ratio_
```

Out[18]:
```
array([9.95724278e-01, 3.60372480e-03, 6.71996749e-04])
```

In [19]:
```
data_r2=pd.DataFrame(data_r2,columns=['C1','C2','C3'])
```

In [20]:
```
df_train['C1']=data_r2['C1']
```

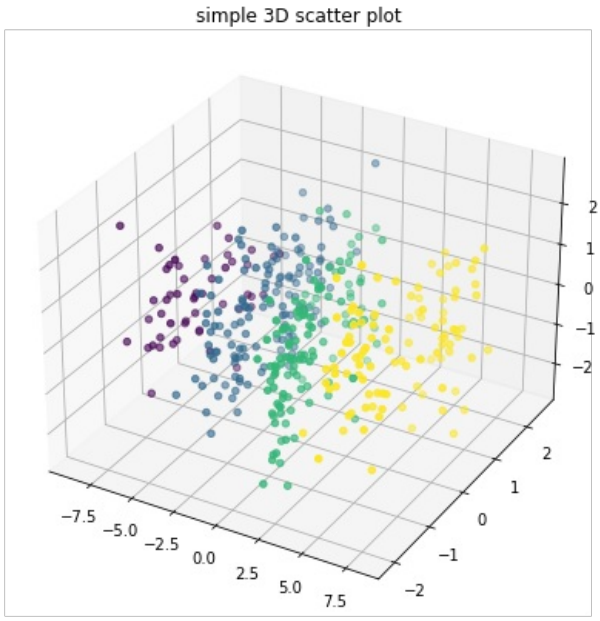
```
df_train['C2']=data_r2['C2']
df_train['C3']=data_r2['C3']
df_train.head()
```

Out[20]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112

In [21]:

```
from mpl_toolkits import mplot3d

fig = plt.figure(figsize = (10, 7))
ax = plt.axes(projection ="3d")
ax.scatter3D(df_train['C1'], df_train['C2'], df_train['C3'], c = df_train['UNS_Numerical'])
plt.title("simple 3D scatter plot")

plt.show()
```

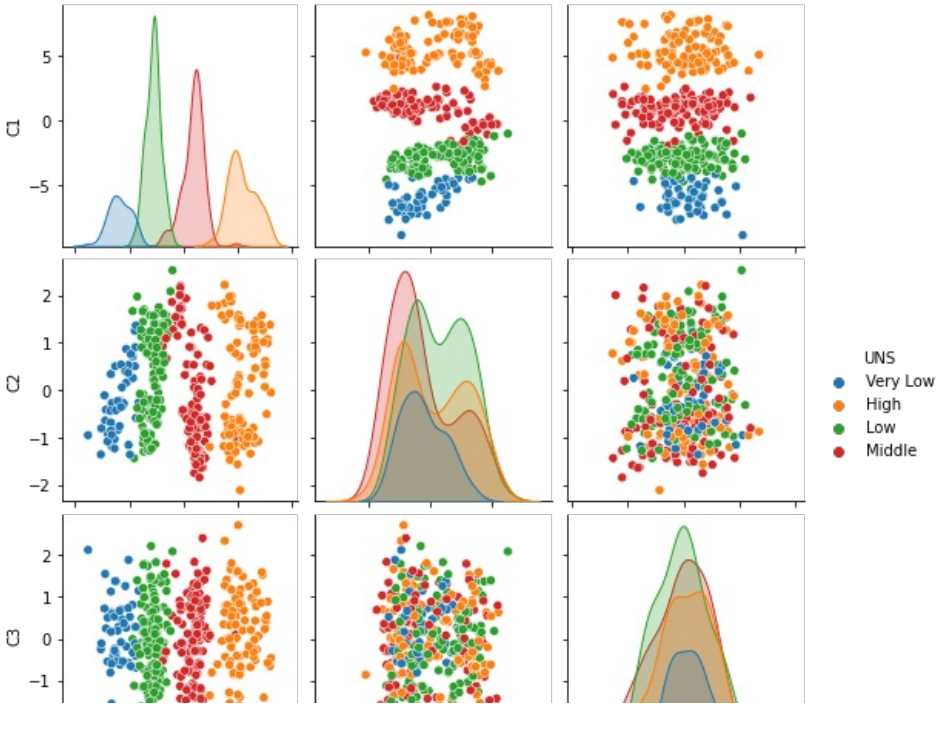


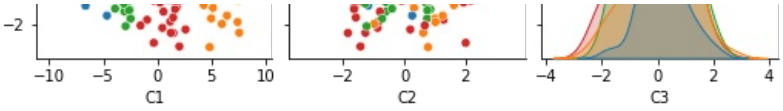
In [22]:

```
sns.pairplot(df_train[['C1','C2','C3','UNS']],hue='UNS')
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x22580e9f730>
```





In [23]:
```
#data=df_train[['C1','C2','C3']]
```

Cluster Analysis

Note: It is prior known from the dataset that no. of clusters(classes) are 4

In [24]:
```
from sklearn import metrics
```

In [25]:
```
models_silhouette_score=list()
models_calinski_harabasz_score=list()
models_davies_bouldin_score=list()
```

Clustering Metrics under Consideration:

1. Silhouette Score:

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

2. Calinski Harabasz Index:

If the ground truth labels are not known, the Calinski-Harabasz index (sklearn.metrics.calinski_harabasz_score) - also known as the Variance Ratio Criterion - can be used to evaluate the model, where a higher Calinski-Harabasz score relates to a model with better defined clusters.

The index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared)

3. Davies Bouldin Index:

If the ground truth labels are not known, the Davies-Bouldin index (sklearn.metrics.davies_bouldin_score) can be used to evaluate the model, where a lower Davies-Bouldin index relates to a model with better separation between the clusters.

This index signifies the average ‘similarity’ between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves.

Zero is the lowest possible score. Values closer to zero indicate a better partition.

Function to generate report for clustering model

In [26]:
```
def model_analysis(model,label):

    #Print Metric Scores for Model
    print('Silhouette Score:',metrics.silhouette_score(data, model.labels_, metric='euclidean'))
    print('Calinski Harabasz Score:',metrics.calinski_harabasz_score(data, model.labels_))
    print('Davies Bouldin Score:',metrics.davies_bouldin_score(data, model.labels_))

    print('')

    #Collect Metric Scores for Comparison
    models_silhouette_score.append(metrics.silhouette_score(data, model.labels_, metric='euclidean'))
    models_calinski_harabasz_score.append(metrics.calinski_harabasz_score(data, model.labels_))
    models_davies_bouldin_score.append(metrics.davies_bouldin_score(data, model.labels_))

    #Display Pairplot as per clustering performed by model
    sns.pairplot(df_train[['STG','SCG','STR','LPR','PEG',label]],hue=label,palette='tab10')

    fig, ax = plt.subplots(2,2,figsize=(14,12))
```

```
    #Display Scatterplot for 'STG' vs 'PEG'
    ax[0,0].scatter(df_train['STG'],df_train['PEG'],c=df_train[label],cmap='plasma')
    ax[0,0].grid()
    ax[0,0].set_xlabel('STG')
    ax[0,0].set_ylabel('PEG')

    #Display Scatterplot for 'SCG' vs 'PEG'
    ax[0,1].scatter(df_train['SCG'],df_train['PEG'],c=df_train[label],cmap='plasma')
    ax[0,1].grid()
    ax[0,1].set_xlabel('SCG')
    ax[0,1].set_ylabel('PEG')

    #Display Scatterplot for 'STR' vs 'PEG'
    ax[1,0].scatter(df_train['STR'],df_train['PEG'],c=df_train[label],cmap='plasma')
    ax[1,0].grid()
    ax[1,0].set_xlabel('STR')
    ax[1,0].set_ylabel('PEG')

    #Display Scatterplot for 'LPR' vs 'PEG'
    ax[1,1].scatter(df_train['LPR'],df_train['PEG'],c=df_train[label],cmap='plasma')
    ax[1,1].grid()
    ax[1,1].set_xlabel('LPR')
    ax[1,1].set_ylabel('PEG')

    plt.show()
```

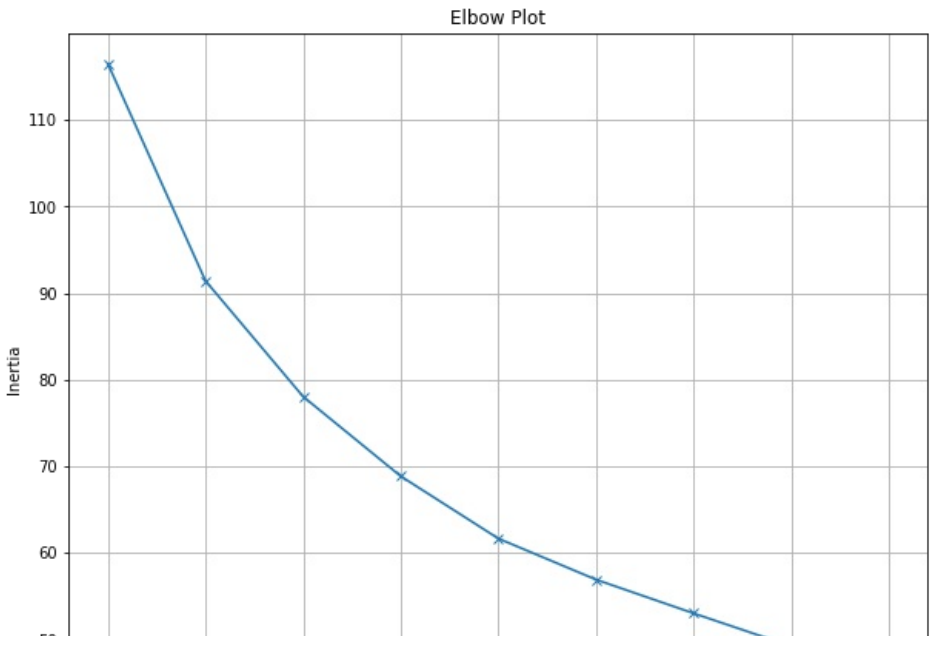
K-Means Clustering

In [27]:
```
from sklearn.cluster import KMeans
```

Elbow Plot

In [28]:
```
kmeans_inertia=list()

#Collecting Inertia for various no. of clusters
for i in range(1,10):

    km=KMeans(n_clusters=i,init='k-means++',n_init=10,random_state=1)
    km.fit(data)
    kmeans_inertia.append(km.inertia_)

kmeans_inertia=pd.DataFrame({'Clusters':list(range(1,10)),'Inertia':kmeans_inertia,})
```

In [29]:
```
#Generating Elbow Plot to determine ideal no. of clusters
plt.figure(figsize=(10,8))

plt.title('Elbow Plot')
plt.plot(kmeans_inertia['Clusters'],kmeans_inertia['Inertia'],marker='x')
plt.xlabel('Clusters')
plt.ylabel('Inertia')
plt.grid()

plt.show()
```



Elbow Point can be observed for around 4 cluster which is consistent with data

In [30]:
```
kmeans_inertia
```

Out[30]:

	Clusters	Inertia
0	1	116.481996
1	2	91.411999
2	3	78.044187
3	4	68.848534
4	5	61.649824
5	6	56.908940
6	7	53.014307
7	8	49.208816
8	9	46.148784

Fitting K-Means Model for 4 clusters

In [31]:
```
km=KMeans(n_clusters=4,init='k-means++',n_init=10,random_state=1)
km.fit(data)
```

Out[31]:
```
KMeans(n_clusters=4, random_state=1)
```

In [32]:
```
df_train['KMeans_Labels']=km.labels_
```

In [33]:
```
df_train.head()
```

Out[33]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0

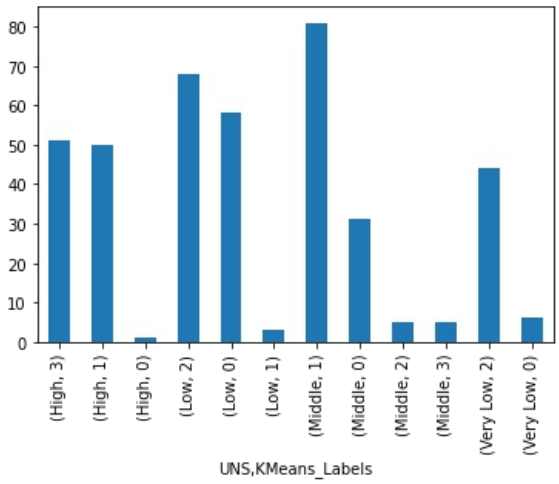
Clustering Analysis

In [34]:
```
grp=df_train.groupby(['UNS'])['KMeans_Labels'].value_counts()
print(grp)
```

```
UNS       KMeans_Labels
High      3                51
          1                50
          0                 1
Low       2                68
          0                58
          1                 3
Middle    1                81
          0                31
          2                 5
          3                 5
Very Low  2                44
          0                 6
Name: KMeans_Labels, dtype: int64
```

In [35]:
```
grp.plot(kind='bar')
```

Out[35]:
```
<AxesSubplot:xlabel='UNS,KMeans_Labels'>
```



'High UNS' : Mixture of Cluster 3, 1 and 0

'Medium UNS' : Mixture of Cluster 1, 0, 2 and 3

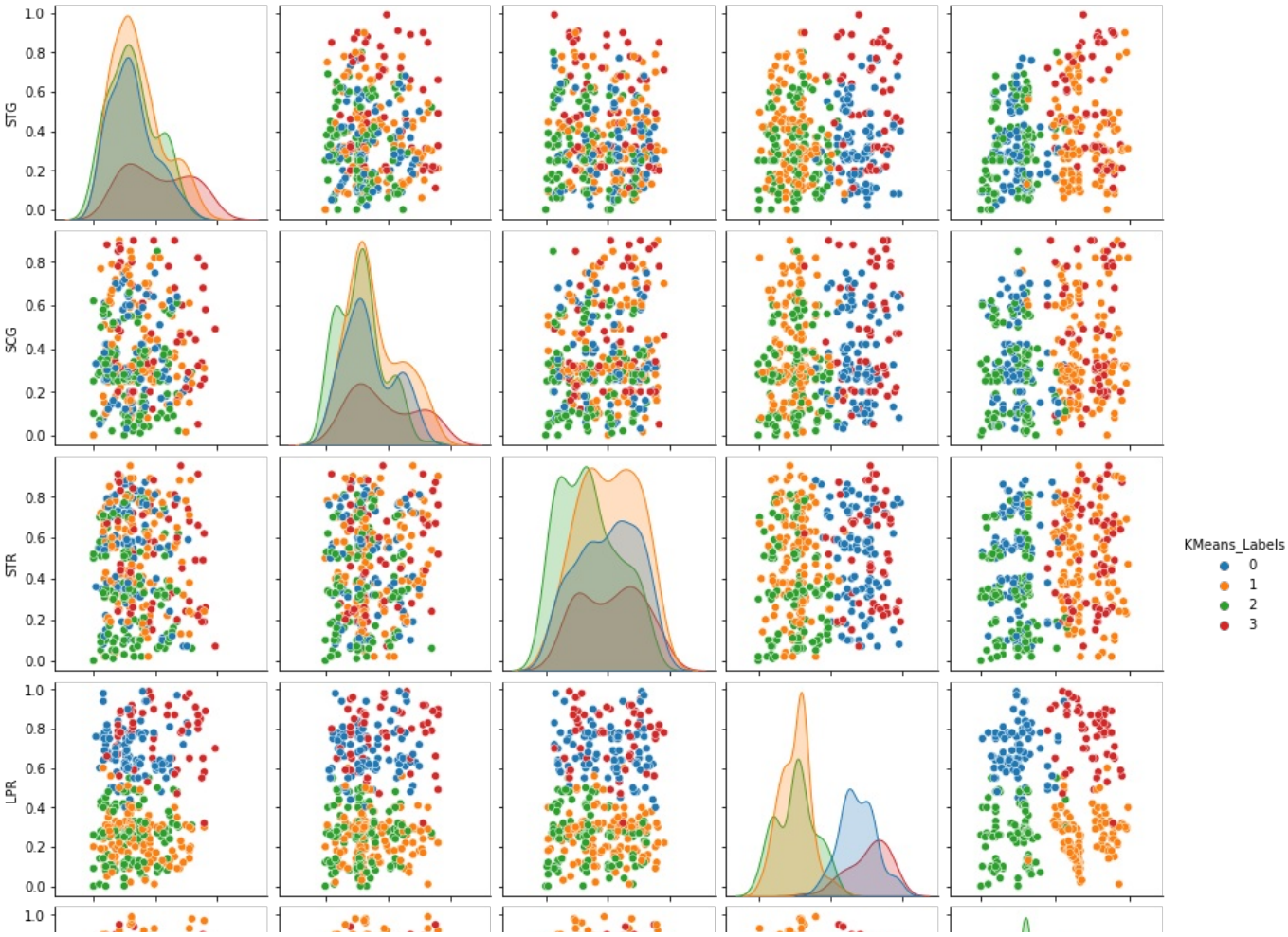
'Low UNS' : Mixture of Cluster 2, 0 and 1

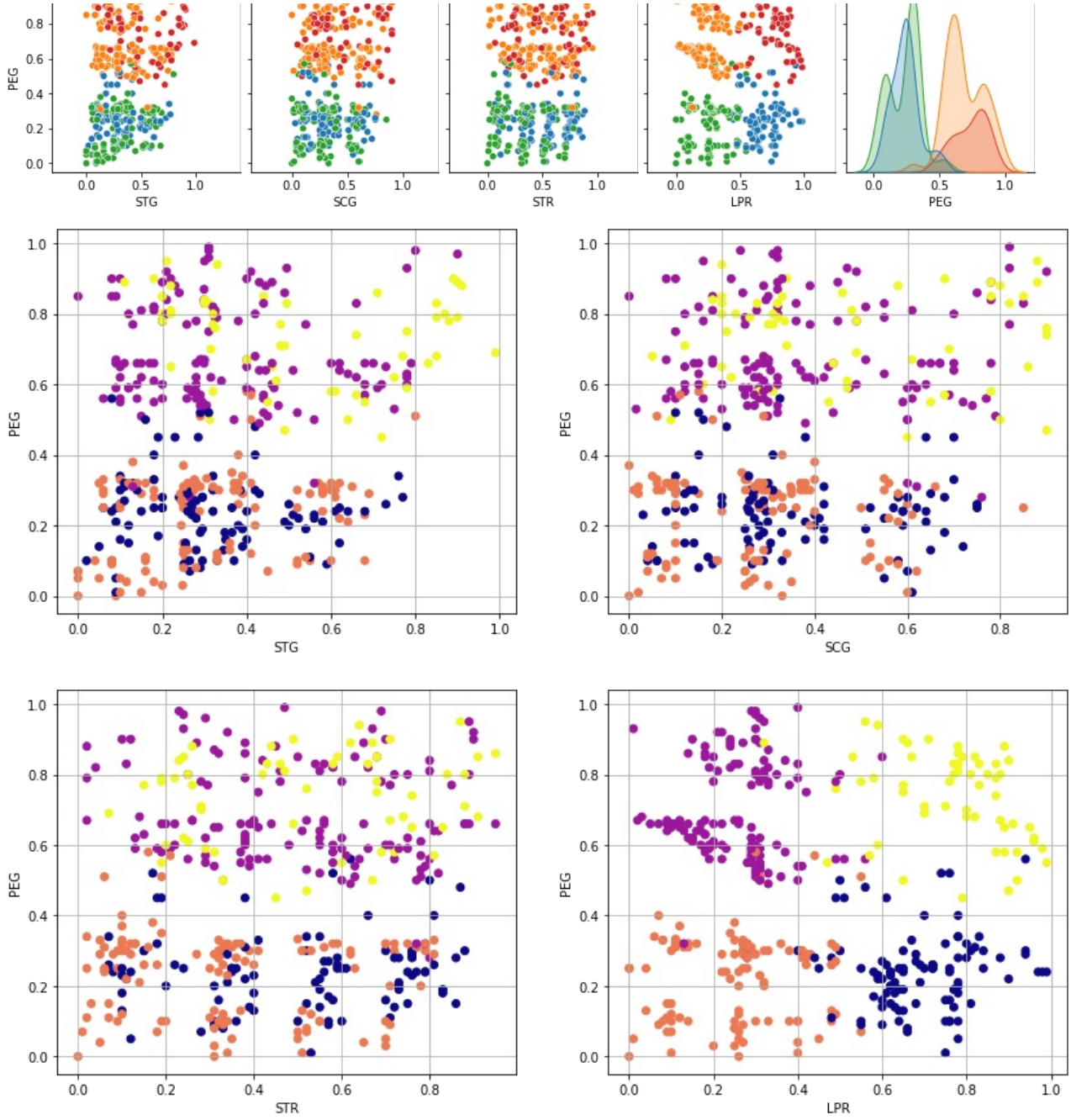
'Very Low UNS' : Mixture of Cluster 2 and 0

Model Report

In [36]:
```
model_analysis(km,'KMeans_Labels')
```

```
Silhouette Score: 0.194848968838817
Calinski Harabasz Score: 92.01721682147956
Davies Bouldin Score: 1.6132302777590333
```





Inferences:

Cluster 1 or 3 can be identified as 'High UNS'

Cluster 1 can be identified as 'Medium UNS'

Cluster 0 or 2 can be identified as 'Low UNS'

Cluster 0 or 2 can be identified as 'Very Low UNS'

Hierarchial Clustering

In [37]:
```
from sklearn.cluster import AgglomerativeClustering
```

Fitting Hierarchial-Clustering Model for 4 clusters

In [38]:
```
ag=AgglomerativeClustering(n_clusters=4,linkage='ward',compute_full_tree=True)
ag.fit(data)
```

Out[38]:
```
AgglomerativeClustering(compute_full_tree=True, n_clusters=4)
```

In [39]:
```
df_train['HC_Labels']=ag.labels_
```

In [40]:
```
df_train.head()
```

Out[40]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2	0
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1

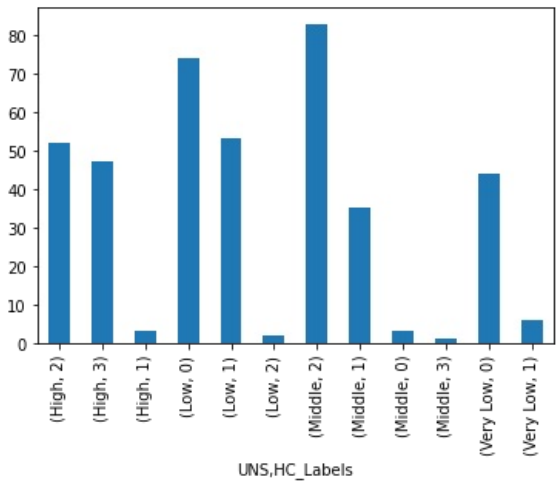
Clustering Analysis

In [41]:
```
grp=df_train.groupby(['UNS'])['HC_Labels'].value_counts()
print(grp)
```

```
UNS       HC_Labels
High      2            52
          3            47
          1             3
Low       0            74
          1            53
          2             2
Middle    2            83
          1            35
          0             3
          3             1
Very Low  0            44
          1             6
Name: HC_Labels, dtype: int64
```

In [42]:
```
grp.plot(kind='bar')
```

Out[42]:
```
<AxesSubplot:xlabel='UNS,HC_Labels'>
```



'High UNS' : Mixture of Cluster 2 and 3

'Medium UNS' : Mixture of Cluster 2, 1, 0 and 3

'Low UNS' : Mixture of Cluster 0, 1 and 2

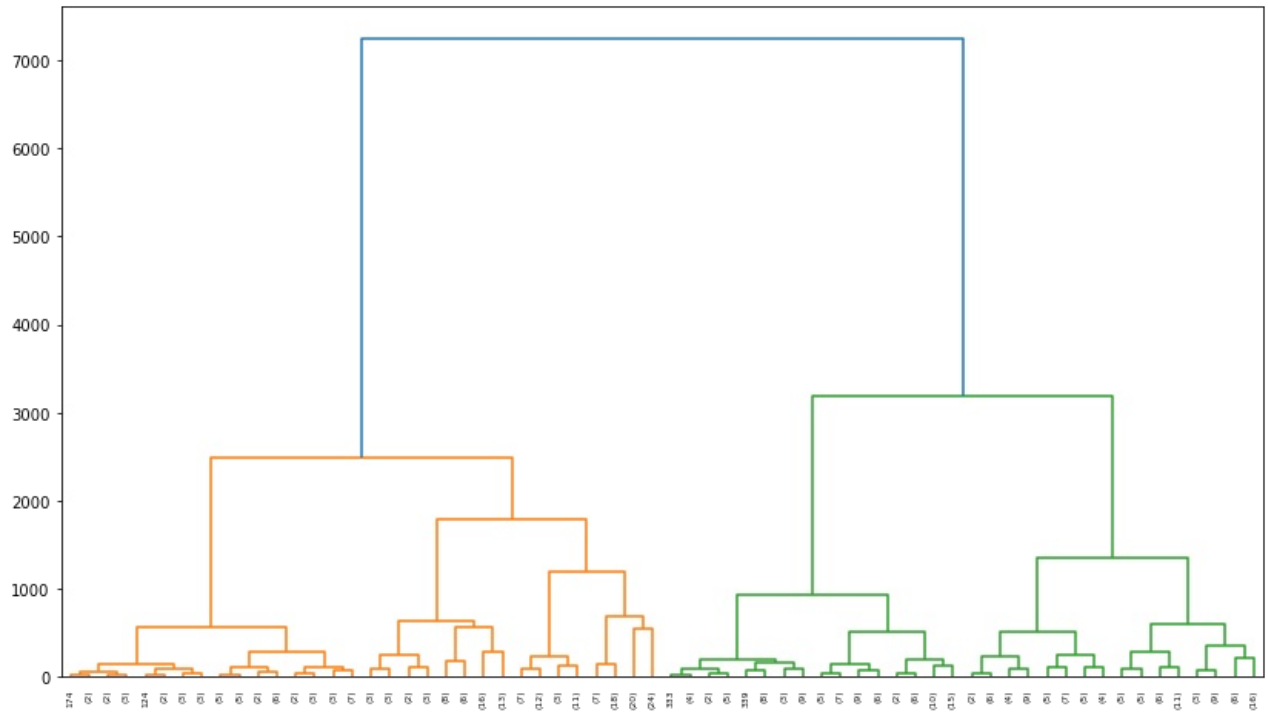
'Very Low UNS' : Mixture of Cluster 0 and 1

In [43]:
```
from scipy.cluster import hierarchy
```

Dendrogram Tree of Hierarchy

Dendrogram Tree of Hierarchy

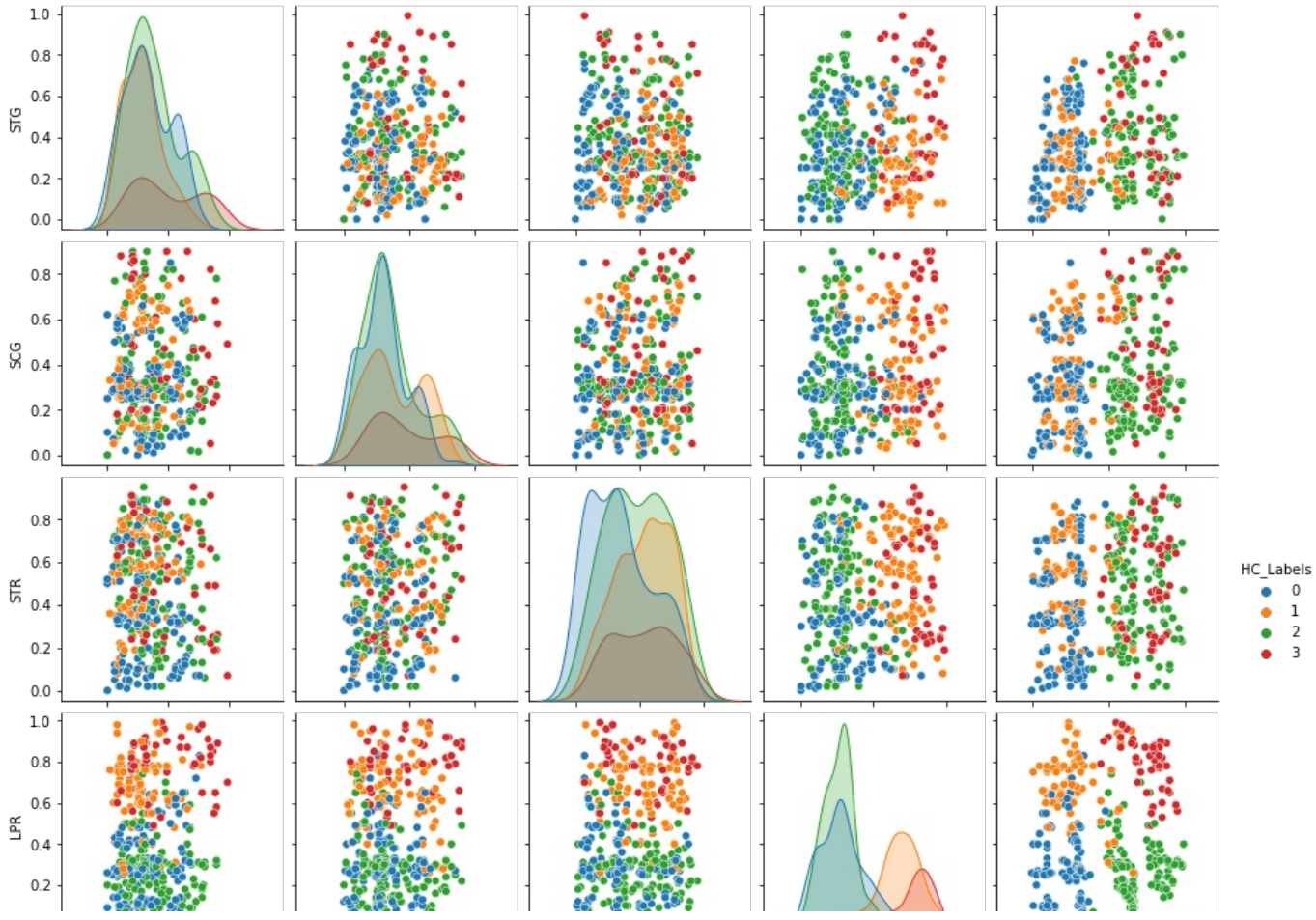
In [44]:
```
z=hierarchy.linkage(ag.children_,method='ward')
fig,ax=plt.subplots(figsize=(14,8))
den=hierarchy.dendrogram(z,orientation='top',show_leaf_counts=True,p=5,truncate_mode='level',ax=ax)
```

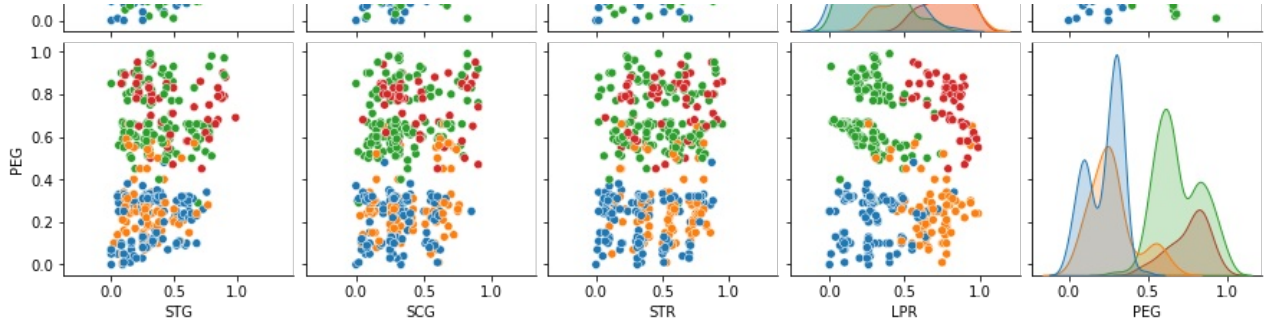


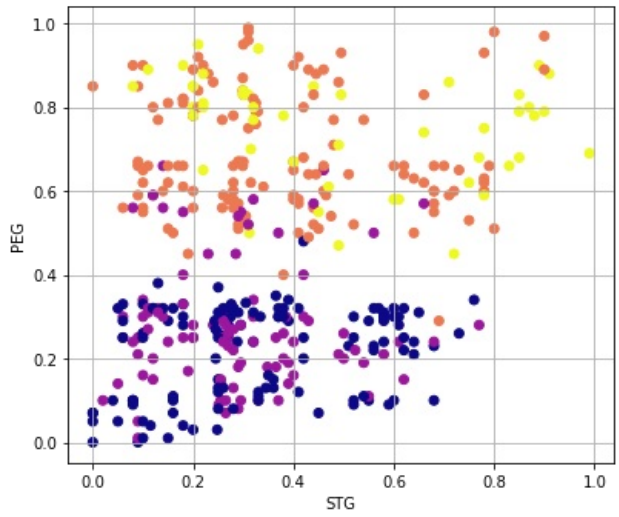
Model Report

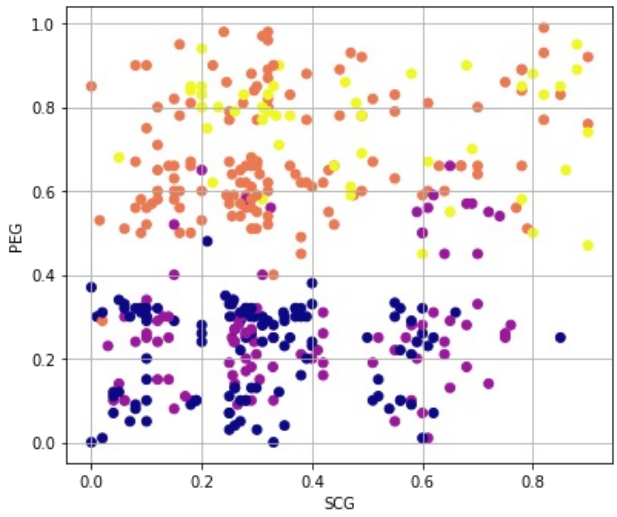
In [45]:
```
model_analysis(ag,'HC_Labels')
```

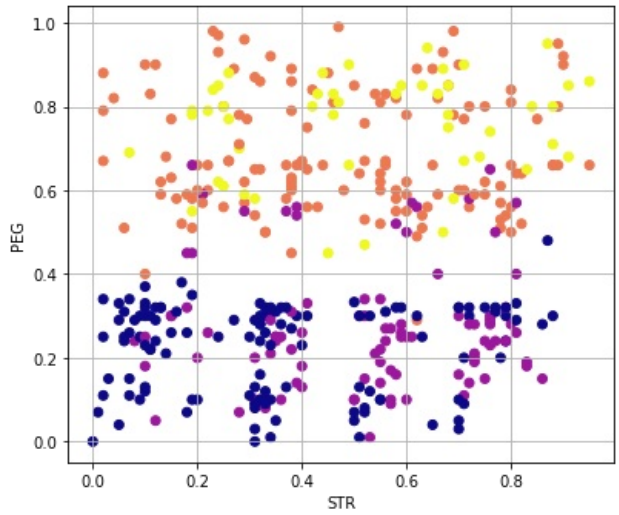
```
Silhouette Score: 0.17045086535201248
Calinski Harabasz Score: 79.25267998834383
Davies Bouldin Score: 1.7346757844071719
```

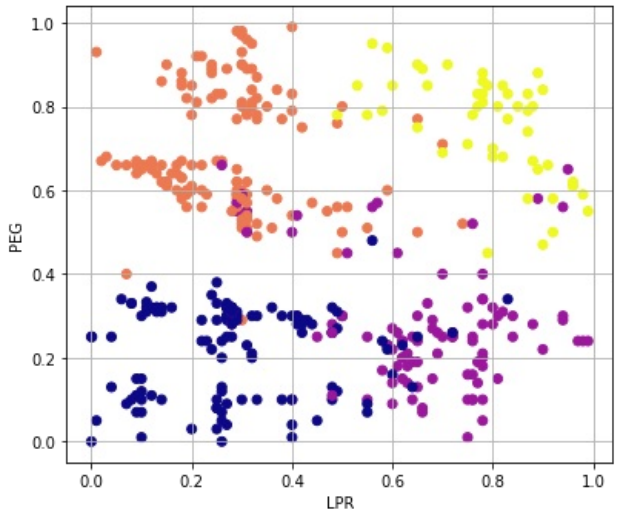












Inferences:

Cluster 2 or 3 can be identified as 'High UNS'

Cluster 2 can be identified as 'Medium UNS'

Cluster 0 or 1 can be identified as 'Low UNS'

Cluster 0 or 1 can be identified as 'Very Low UNS'

Birch Clustering

In [46]:
```
from sklearn.cluster import Birch
```

Fitting Birch Model for 4 Cluster

In [47]:
```
brch=Birch(n_clusters=4,threshold=0.5)
brch.fit(data)
```

Out[47]:
```
Birch(n_clusters=4)
```

In [48]:
```
df_train['Birch_Labels']=brch.labels_
```

In [49]:
```
df_train.head()
```

Out[49]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels	Birch_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2	0	1
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2	0
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0	1
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1	1

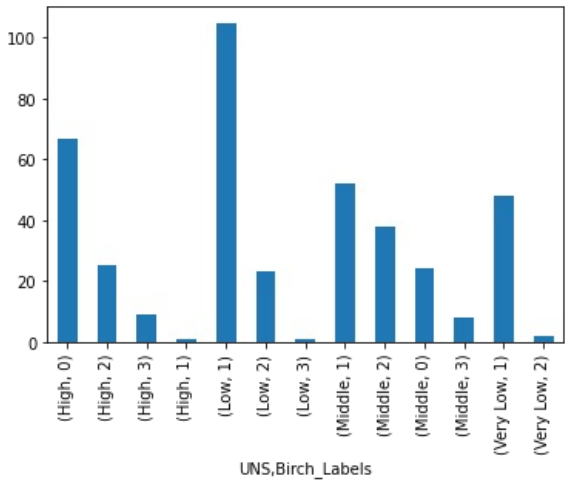
Cluster Analysis

In [50]:
```
grp=df_train.groupby(['UNS'])['Birch_Labels'].value_counts()
print(grp)
```

```
UNS       Birch_Labels
High      0               67
          2               25
          3                9
          1                1
Low       1              105
          2               23
          3                1
Middle    1               52
          2               38
          0               24
          3                8
Very Low  1               48
          2                2
Name: Birch_Labels, dtype: int64
```

In [51]:
```
grp.plot(kind='bar')
```

Out[51]:
```
<AxesSubplot:xlabel='UNS,Birch_Labels'>
```



'High UNS' : Mixture of Cluster 0, 2, 3 and 1

'Medium UNS' : Mixture of Cluster 1, 2, 0 and 3

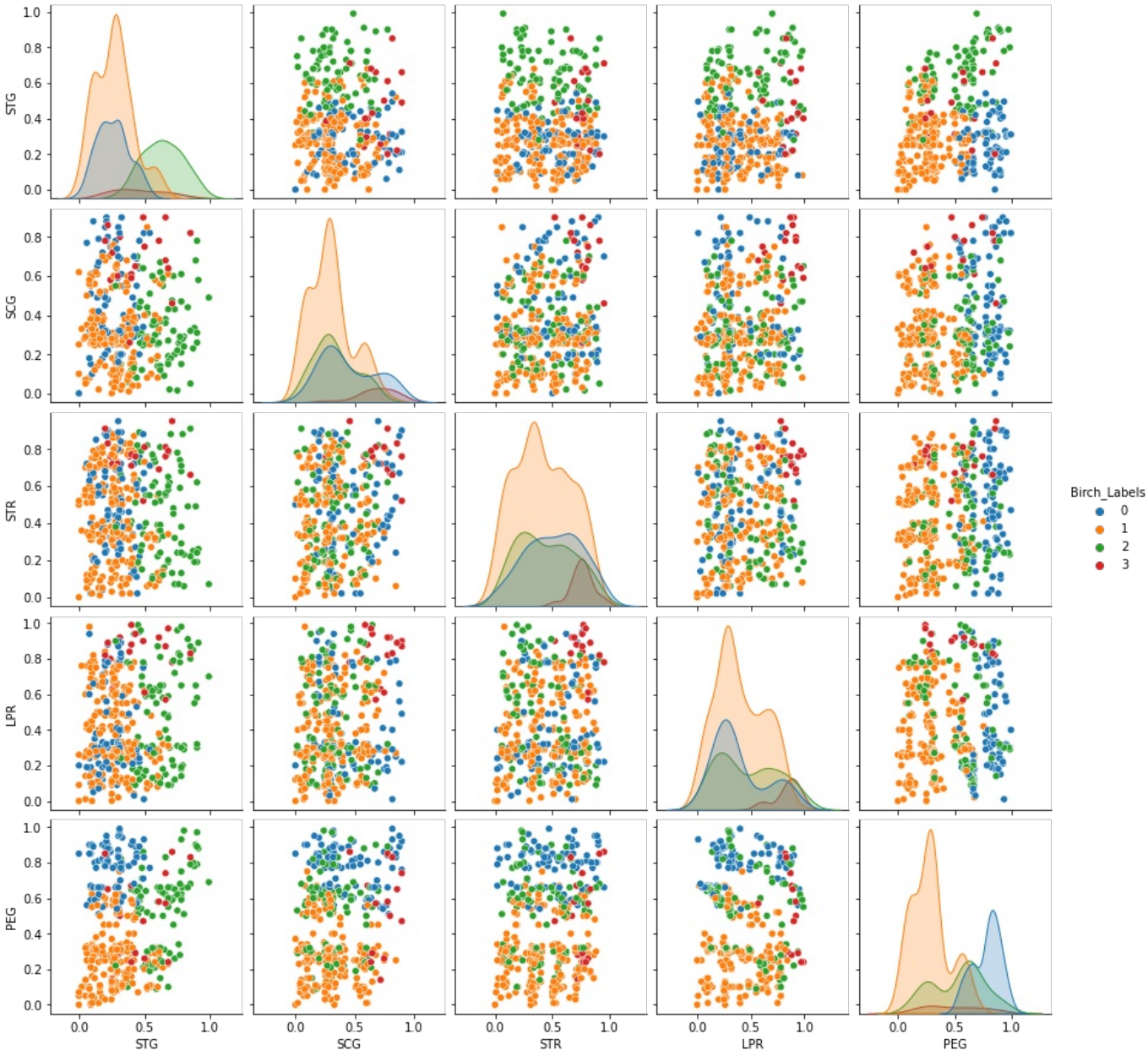
'Low UNS' : Mixture of Cluster 1, 2 and 3

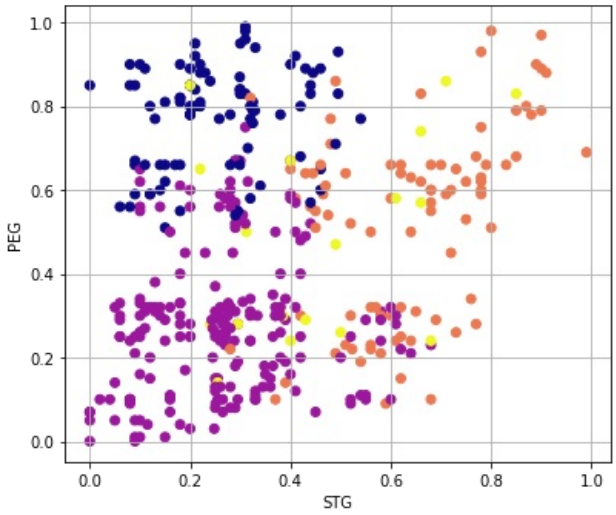
'Very Low UNS' : Mixture of Cluster 1 and 2

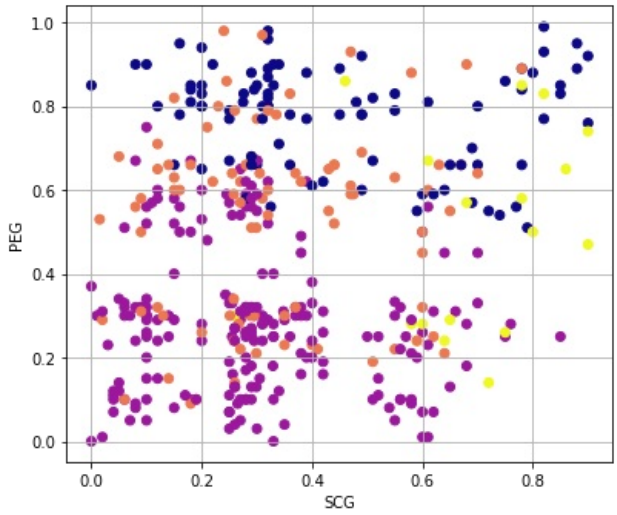
Model Report

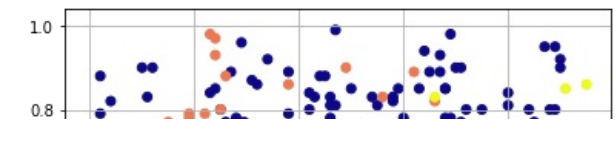
In [52]:
```
model_analysis(brch,'Birch_Labels')
```

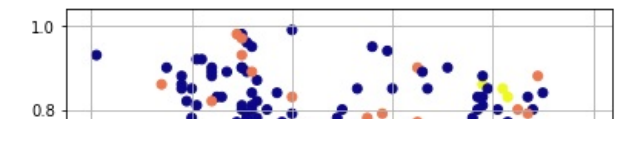
```
Silhouette Score: 0.14177768237174443
Calinski Harabasz Score: 56.72472320536432
Davies Bouldin Score: 1.81160100485888
```

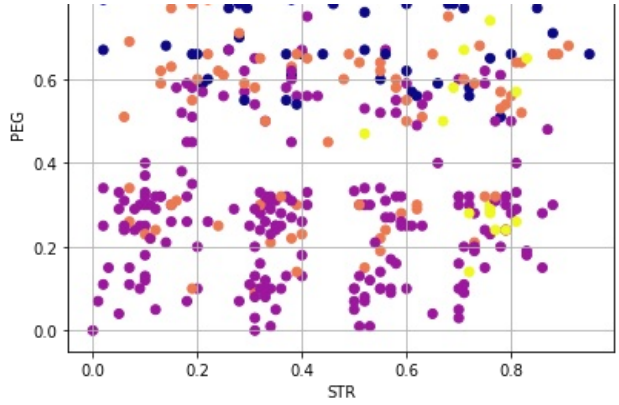


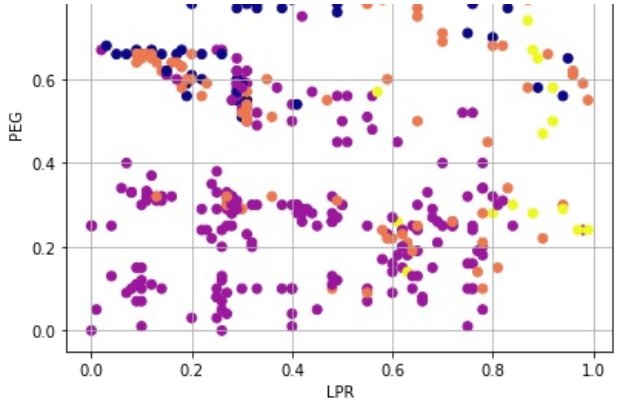












Inferences:

Cluster 0 or 2 can be identified as 'High UNS'

Cluster 0, 1 or 2 can be identified as 'Medium UNS'

Cluster 1 or 2 can be identified as 'Low UNS'

Cluster 1 or 2 can be identified as 'Very Low UNS'

Clusters are less distinctive

Spectral Clustering

In [53]:
```
from sklearn.cluster import SpectralClustering
```

Fitting Spectral Model for 4 Cluster

In [54]:
```
sc=SpectralClustering(n_clusters=4,assign_labels='discretize')
sc.fit(data)
```

Out[54]:
```
SpectralClustering(assign_labels='discretize', n_clusters=4)
```

In [55]:
```
df_train['SC_Labels']=sc.labels_
```

In [56]:
```
df_train.head()
```

Out[56]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels	Birch_Labels	SC_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2	0	1	1
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2	0	3
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0	1	1
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1	1	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1	1	0

Cluster Analysis

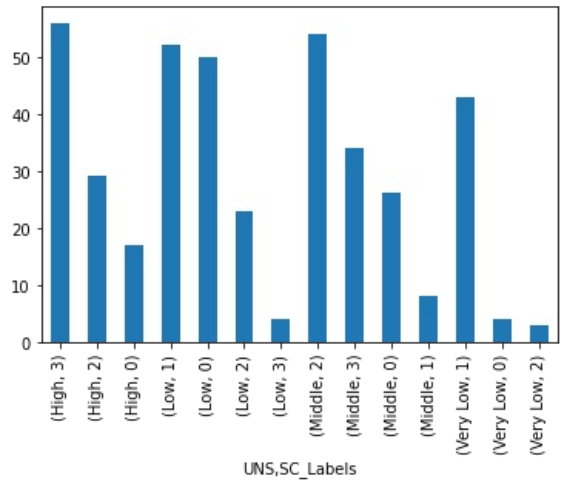
In [57]:
```
grp=df_train.groupby(['UNS'])['SC_Labels'].value_counts()
print(grp)
```

```
UNS       SC_Labels
High      3            56
          2            29
          0            17
Low       1            52
          0            50
          2            23
```

```
          3             4
Middle    2            54
          3            34
          0            26
          1             8
Very Low  1            43
          0             4
          2             3
Name: SC_Labels, dtype: int64
```

In [58]:
```
grp.plot(kind='bar')
```

Out[58]:
```
<AxesSubplot:xlabel='UNS,SC_Labels'>
```



'High UNS' : Mixture of Cluster 3, 0 and 2

'Medium UNS' : Mixture of Cluster 3, 2, 0 and 1

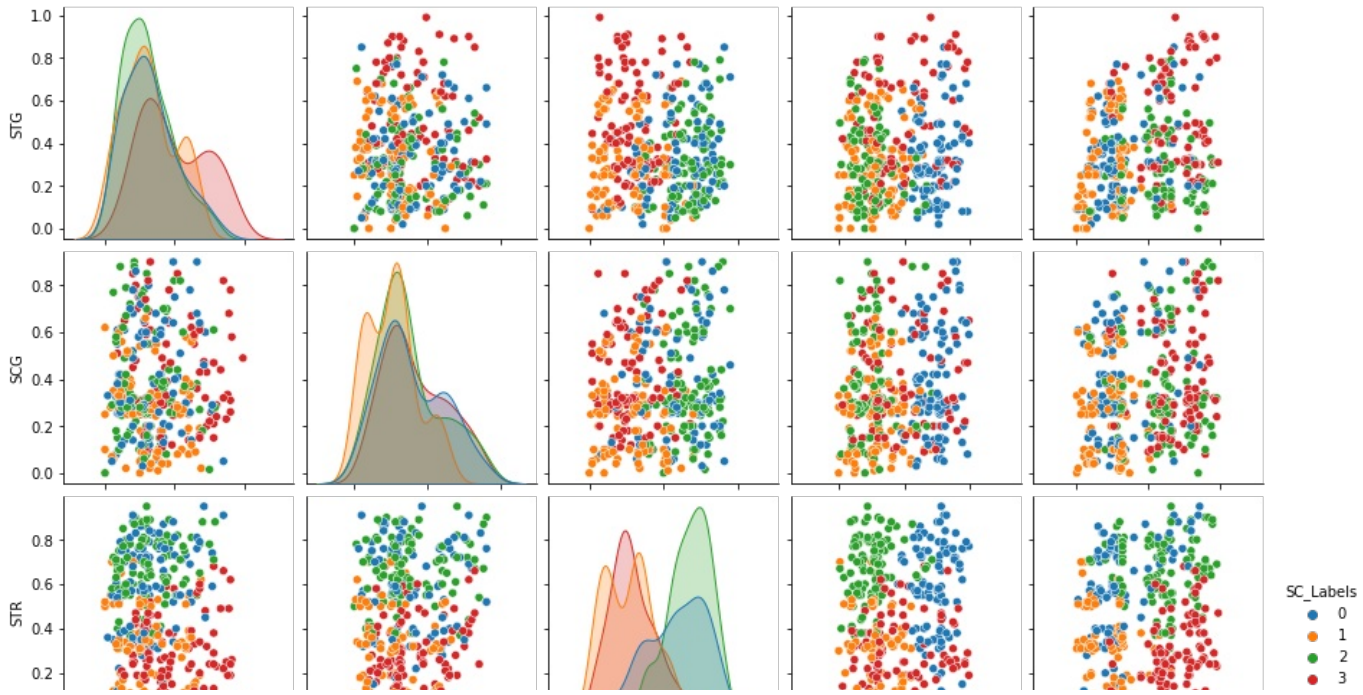
'Low UNS' : Mixture of Cluster 1, 2 and 0

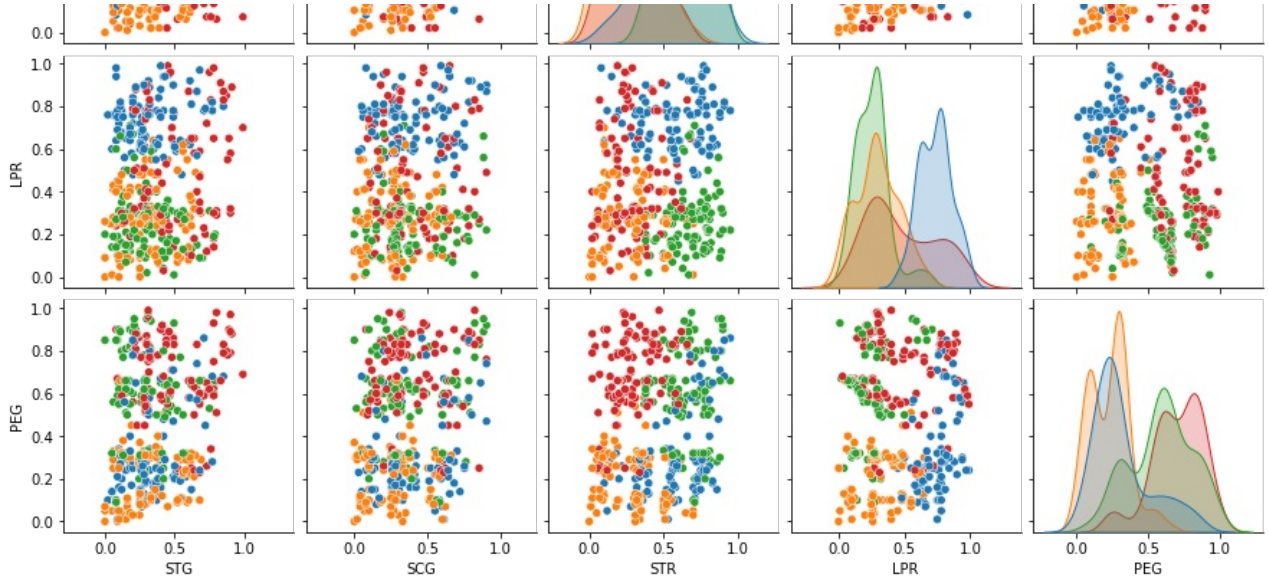
'Very Low UNS' : Mixture of Cluster 1 and 2

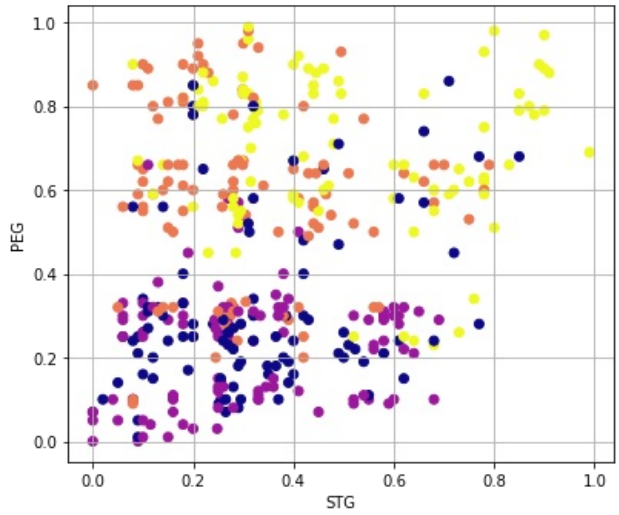
Model Report

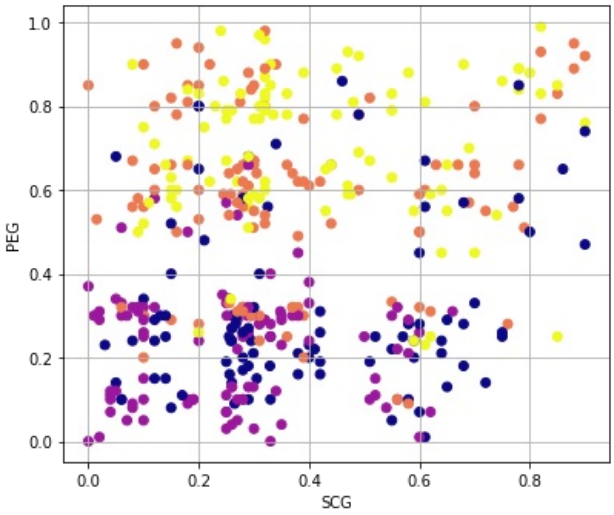
In [59]:
```
model_analysis(sc,'SC_Labels')
```

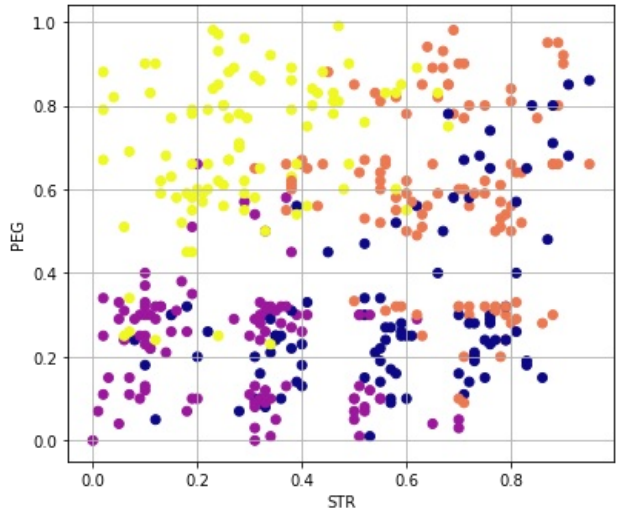
```
Silhouette Score: 0.18306504607047686
Calinski Harabasz Score: 85.47262240656778
Davies Bouldin Score: 1.5988109066825642
```

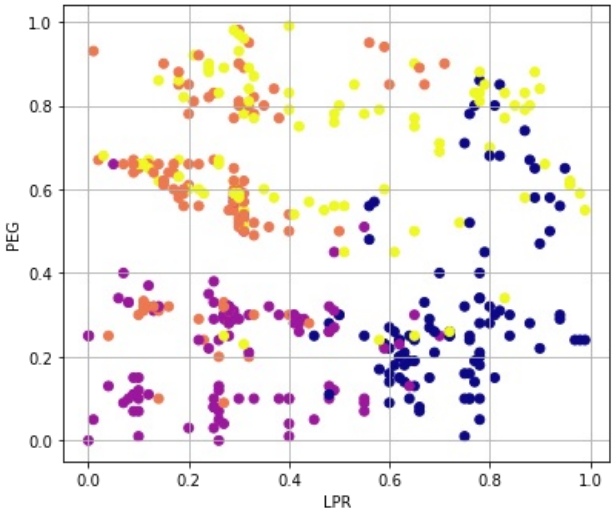












Inferences:

Cluster 0 or 3 can be identified as 'High UNS'

Cluster 2 or 3 can be identified as 'Medium UNS'

Cluster 1 or 2 can be identified as 'Low UNS'

Cluster 1 or 2 can be identified as 'Very Low UNS'

Clusters are less distinctive

OPTICS Clustering

OPTICS Clustering

In [60]:
```
from sklearn.cluster import OPTICS
```

Fitting OPTICS Model

In [61]:
```
opt=OPTICS()
opt.fit(data)
```

Out[61]:
```
OPTICS()
```

In [62]:
```
df_train['OPTICS_Labels']=opt.labels_
```

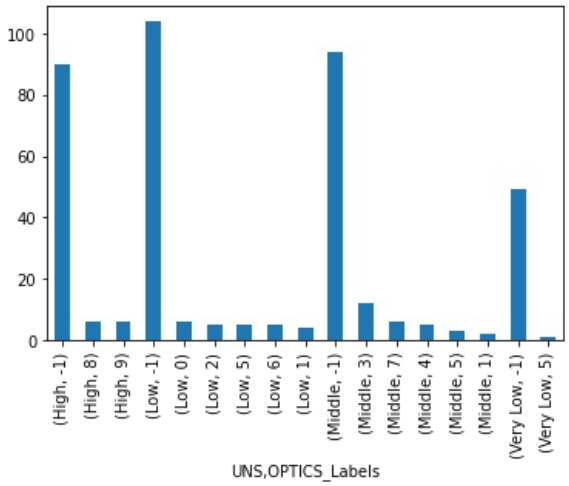
Cluster Analysis

In [63]:
```
grp=df_train.groupby(['UNS'])['OPTICS_Labels'].value_counts()
print(grp)
```

```
UNS       OPTICS_Labels
High      -1                90
           8                 6
           9                 6
Low       -1               104
           0                 6
           2                 5
           5                 5
           6                 5
           1                 4
Middle    -1                94
           3                12
           7                 6
           4                 5
           5                 3
           1                 2
Very Low  -1                49
           5                 1
Name: OPTICS_Labels, dtype: int64
```

In [64]:
```
grp.plot(kind='bar')
```

Out[64]:
```
<AxesSubplot:xlabel='UNS,OPTICS_Labels'>
```



Too many clusters

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

Out[65]:

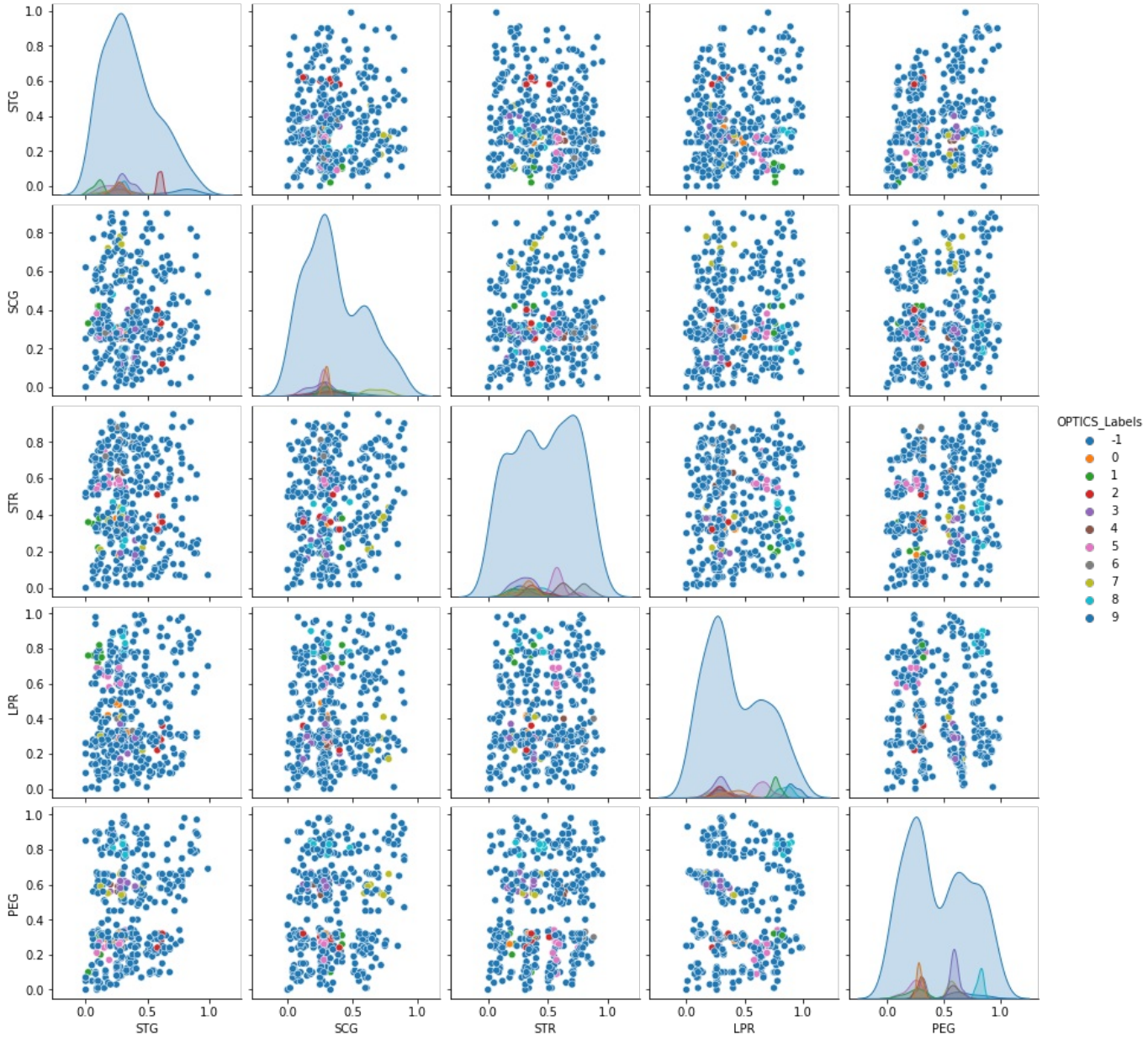
	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels	Birch_Labels	SC_Labels	
0	0.00	0.00	0.00	0.00	0.00	Very	0	-8.815143	-0.949209	2.123050	2	0	1	1	

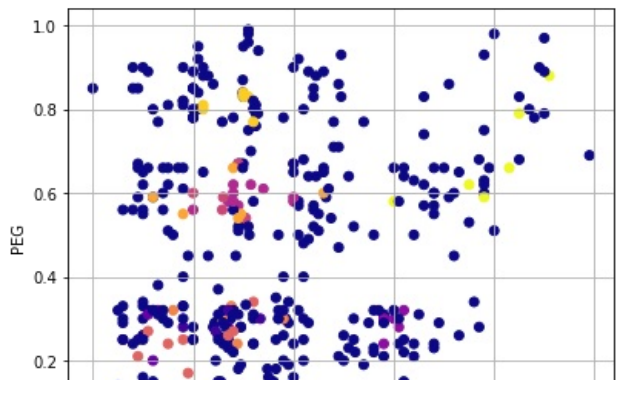
						Low								
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2	0	3
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0	1	1
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1	1	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1	1	0

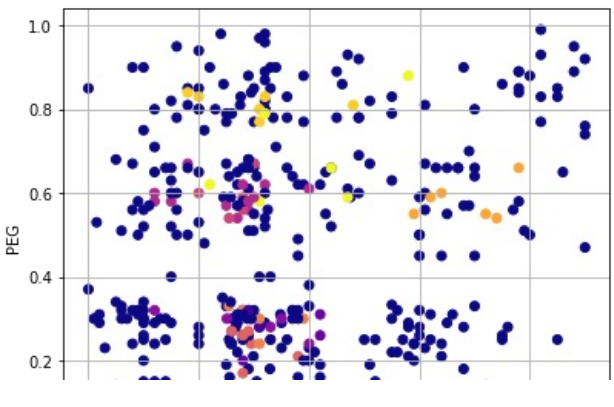
Model Report

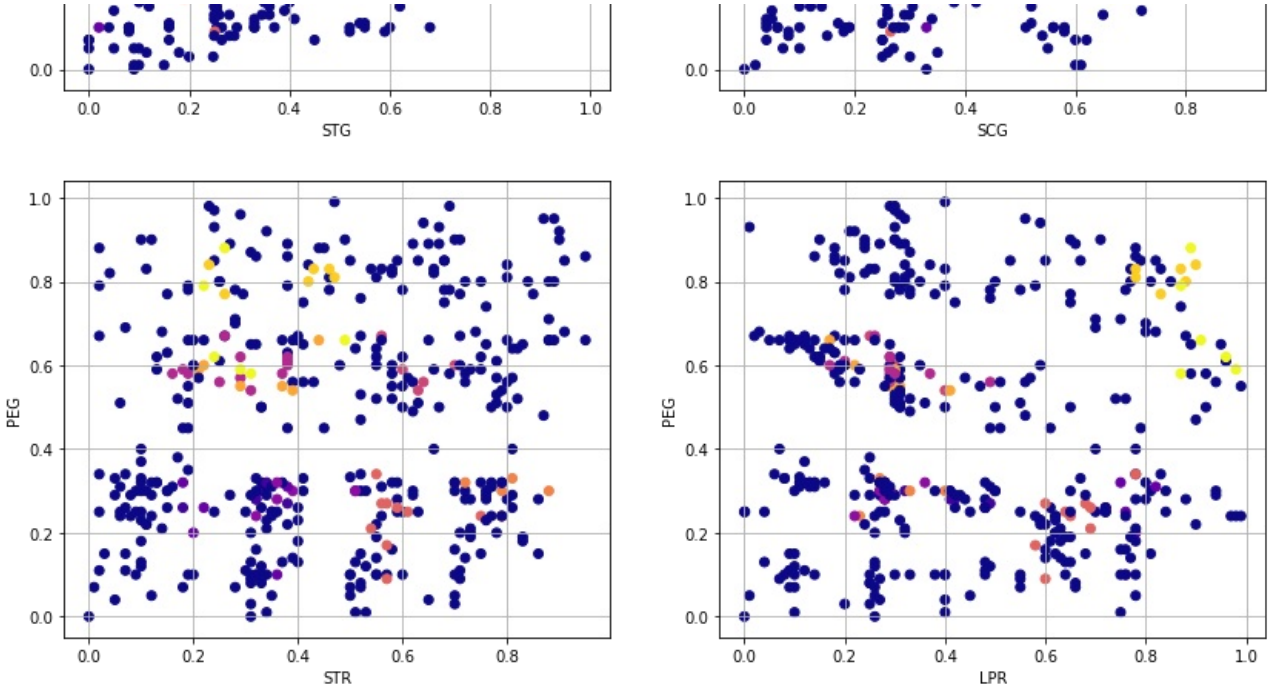
In [66]:
```
model_analysis(opt,'OPTICS_Labels')
```

```
Silhouette Score: -0.3169575625553002
Calinski Harabasz Score: 4.498975554625626
Davies Bouldin Score: 1.8331493777788863
```









Inference: Clusters are not distinctive at all due to high degree of entropy

MeanShift Clustering

In [67]:
```
from sklearn.cluster import MeanShift
```

Fitting MeanShift Model

In [68]:
```
ms=MeanShift(bandwidth=0.4)
ms.fit(data)
```

Out[68]:
```
MeanShift(bandwidth=0.4)
```

In [69]:
```
df_train['MeanShift_Labels']=ms.labels_
```

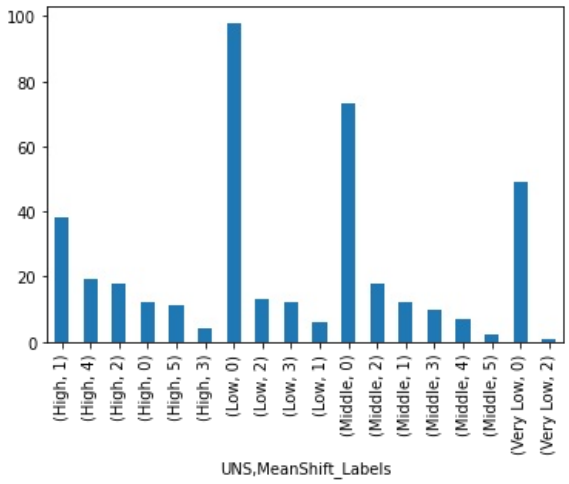
Cluster Analysis

In [70]:
```
grp=df_train.groupby(['UNS'])['MeanShift_Labels'].value_counts()
print(grp)
```

```
UNS       MeanShift_Labels
High      1                   38
          4                   19
          2                   18
          0                   12
          5                   11
          3                    4
Low       0                   98
          2                   13
          3                   12
          1                    6
Middle    0                   73
          2                   18
          1                   12
          3                   10
          4                    7
          5                    2
Very Low  0                   49
          2                    1
Name: MeanShift_Labels, dtype: int64
```

In [71]:
```
grp.plot(kind='bar')
```

Out[71]:
```
<AxesSubplot:xlabel='UNS,MeanShift_Labels'>
```



Too many clusters

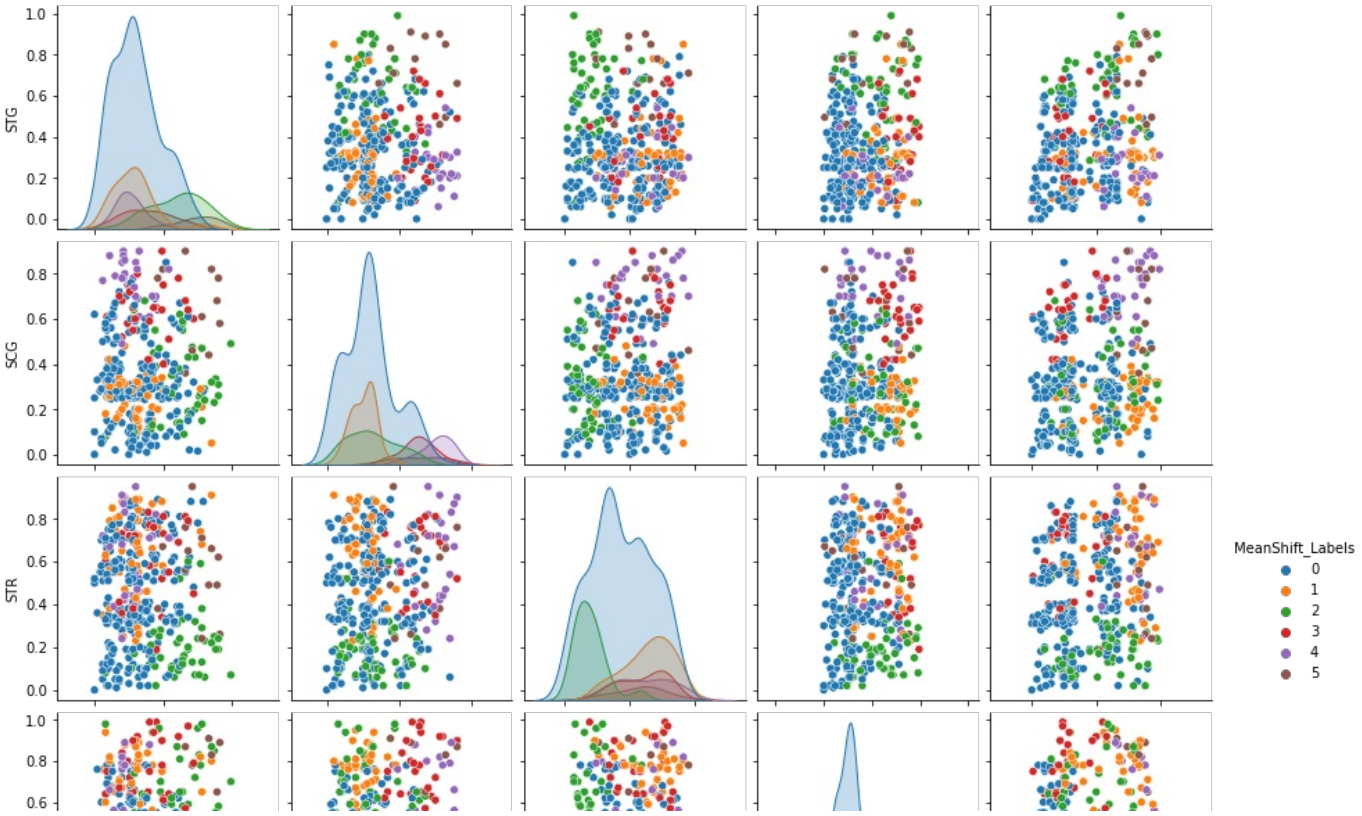
In [72]:
```
df_train.head()
```

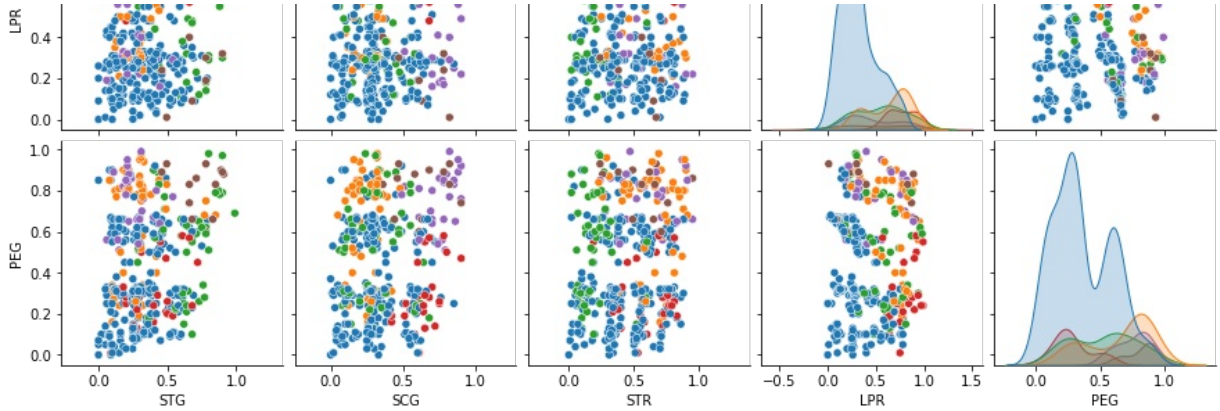
Out[72]:

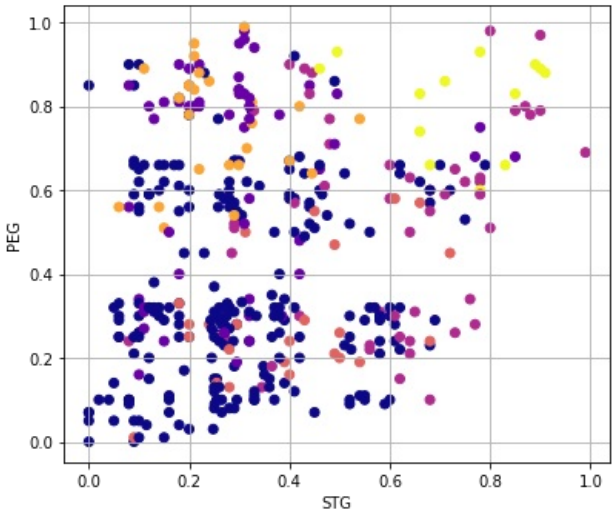
	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels	Birch_Labels	SC_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2	0	1	1
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2	0	3
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0	1	1
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1	1	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1	1	0

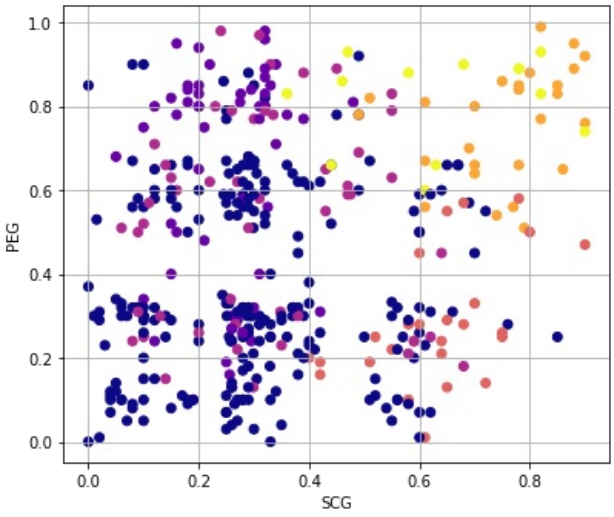
In [73]:
```
model_analysis(ms,'MeanShift_Labels')
```

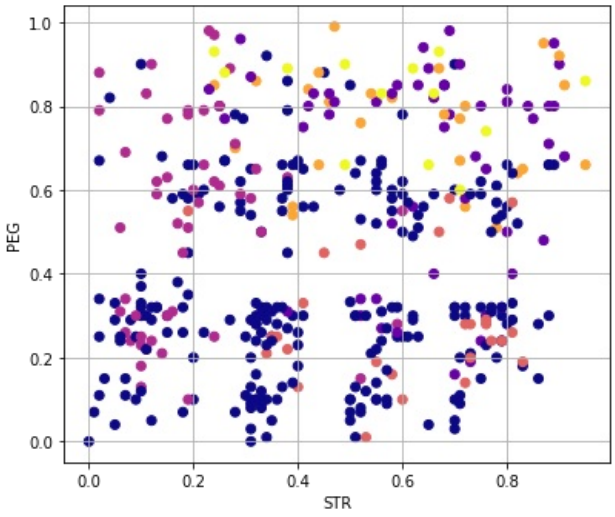
```
Silhouette Score: 0.10422530867269249
Calinski Harabasz Score: 41.31294634730933
Davies Bouldin Score: 1.6671423155305067
```

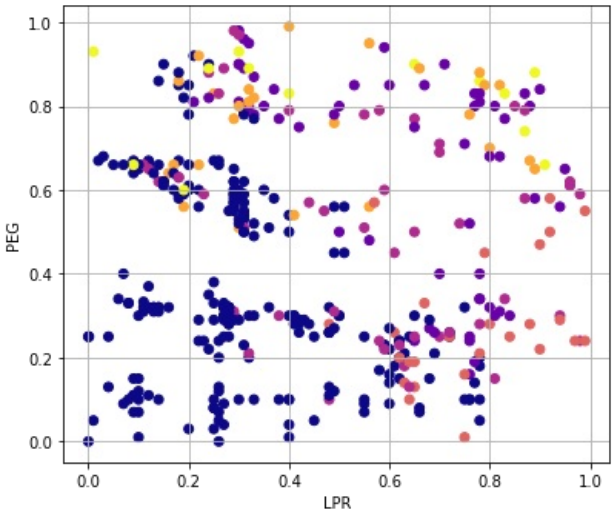












Inference: Clusters are not distinctive at all due to high degree of entropy

Conclusion

In [74]:
```
pd.DataFrame([models_silhouette_score,models_calinski_harabasz_score,models_davies_bouldin_score],columns=['K-Mea
```

Out[74]:

	K-Means	Hierarchial	Birch	Spectral	OPTICS	MeanShift
Silhouette Score	0.194849	0.170451	0.141778	0.183065	-0.316958	0.104225
Calinski_Harabasz Score	92.017217	79.252680	56.724723	85.472622	4.498976	41.312946
Davies_Bouldin Score	1.613230	1.734676	1.811601	1.598811	1.833149	1.667142

1. Silhouette Score:

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

2. Calinski Harabasz Index:

A higher Calinski-Harabasz score relates to a model with better defined clusters.

3. Davies Bouldin Index:

Zero is the lowest possible score. Values closer to zero indicate a better partition.

Clearly K-Means Model is offering best degree of clustering amongst all models in our case

Cluster 1 or 3 can be identified as 'High UNS'

Cluster 1 can be identified as 'Medium UNS'

Cluster 0 or 2 can be identified as 'Low UNS'

Cluster 0 or 2 can be identified as 'Very Low UNS'

There is distinct separation between High/Medium and Low/Very Low UNS Clusters

In [75]:
```
df_train
```

Out[75]:

	STG	SCG	STR	LPR	PEG	UNS	UNS_Numerical	C1	C2	C3	KMeans_Labels	HC_Labels	Birch_Labels	SC_Labels
0	0.00	0.00	0.00	0.00	0.00	Very Low	0	-8.815143	-0.949209	2.123050	2	0	1	1
1	0.08	0.08	0.10	0.24	0.90	High	3	5.065758	-0.874352	2.713808	1	2	0	3
2	0.06	0.06	0.05	0.25	0.33	Low	1	-2.985485	-0.314041	2.215907	2	0	1	1
3	0.10	0.10	0.15	0.65	0.30	Middle	2	-1.565771	1.178651	1.791894	0	1	1	1
4	0.08	0.08	0.08	0.98	0.24	Low	1	-1.008529	2.534450	2.086112	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
398	0.90	0.78	0.62	0.32	0.89	High	3	5.970680	-0.980799	-1.992274	3	2	2	3
399	0.85	0.82	0.66	0.83	0.83	High	3	7.421386	1.017474	-1.944556	3	3	3	3
400	0.56	0.60	0.77	0.13	0.32	Low	1	-2.939158	-1.234232	-2.045379	1	0	2	2
401	0.66	0.68	0.81	0.57	0.57	Middle	2	2.603471	0.193764	-2.123721	3	1	3	0
402	0.68	0.64	0.79	0.97	0.24	Middle	2	-0.332272	2.007104	-2.443699	0	1	3	0

403 rows × 16 columns

In [76]:
```
model_analysis(km,'KMeans_Labels')
```

```
Silhouette Score: 0.194848968838817
Calinski Harabasz Score: 92.01721682147956
Davies Bouldin Score: 1.6132302777590333
```