Tutorial

Linear Regression tutorial 4

Anik Bhowmick
AE20B102
IDDD Data Science

ID5055 Foundations of Machine Learning

September 22, 2023

Problem 1

a) Run linear, non-linear, lasso, ridge regression on the given cosine data. Report your observations.

b) Find the minimum value of num poly features such that the model fits properly.

Solution. a)

0.1 Linear Regression

Here, we fitted the raw data without any feature manipulation. The plot we obtained is the following

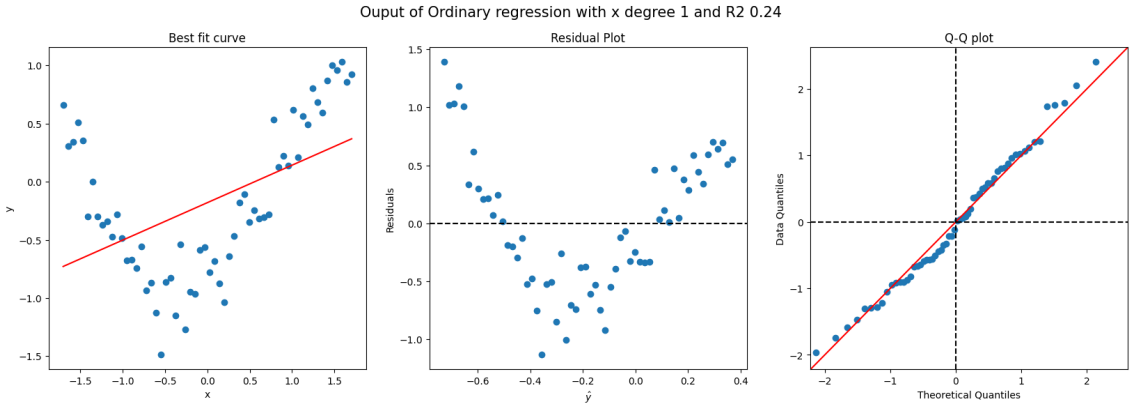


Figure 1: Linear Regression with one feature

- The model performance is extremely poor. It doesn't even fit the data. The Bias is very high.
- So, with one single feature, a linear regression model can't fit a non-linear function.
- Effect of this poor model is again translated into the poor R^2 score.

0.2 Non-linear Regression without regularization

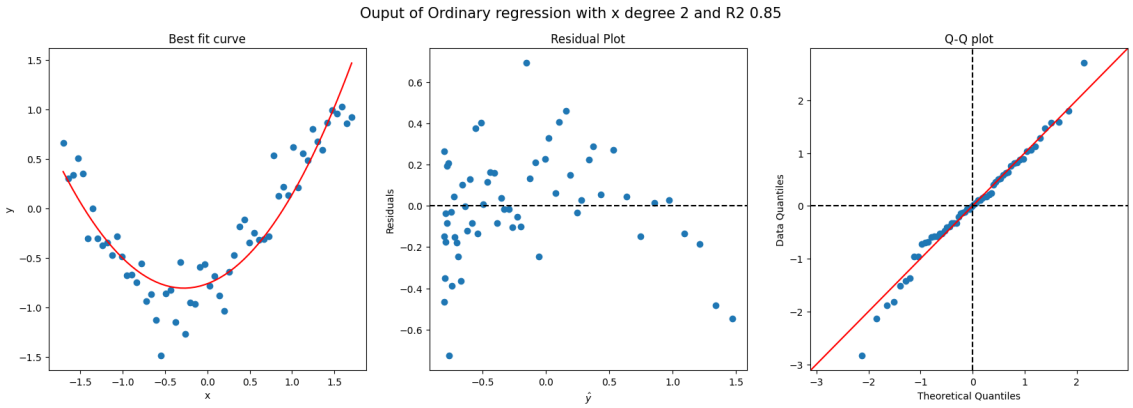


Figure 2: Non Linear Regression with degree of x 2

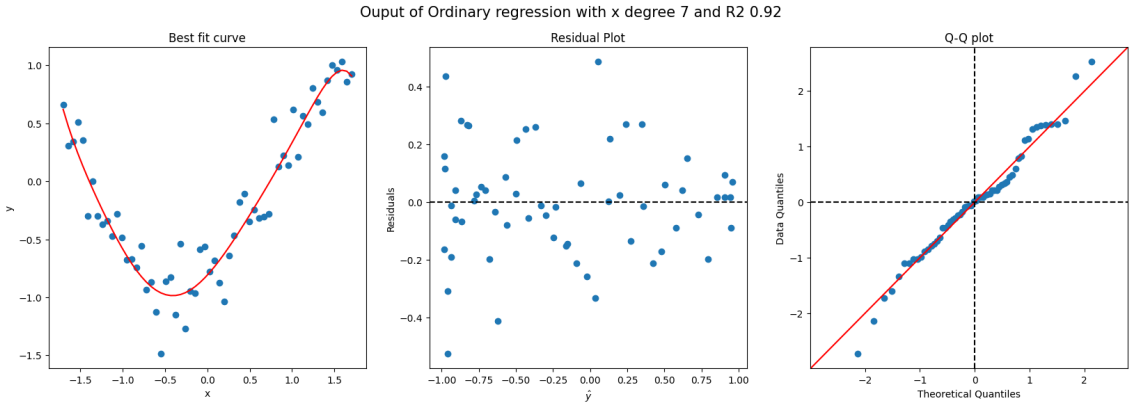


Figure 3: Non Linear Regression with degree of x 7

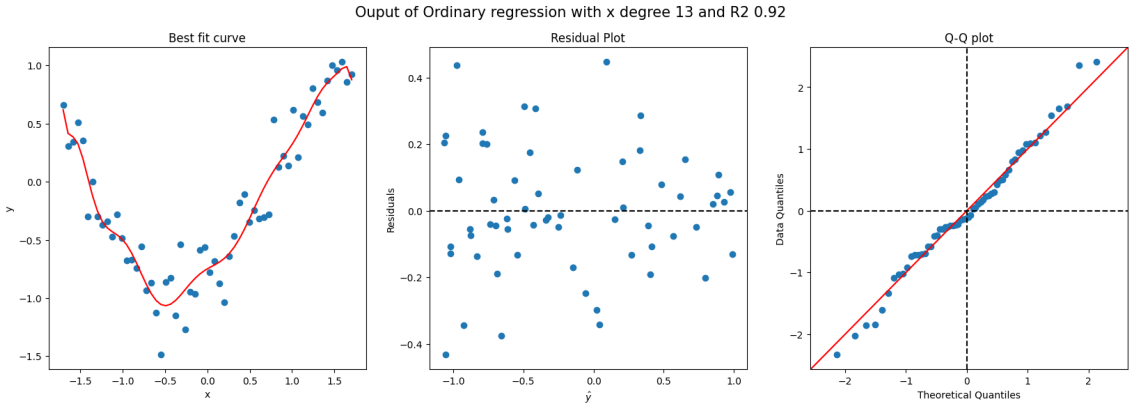


Figure 4: Non Linear Regression with degree of x 13

- With 2 features, the model fitting is not that good. It's still underfitting the data.
- With 7 features, the model fitting is quite better. The R^2 score has increased significantly. The bias is fairly less. Also, looking at the plot, we don't think the model started overfitting data.
- For the 3rd plot, although the R^2 score is the same as the 2nd plot, undoubtedly the model is overfitting. This observation confirms that the $R2$ score alone can't help in deciding whether a model is good or bad.

0.3 Ridge Regression

This part investigates how the model performance goes with various values of regularization constants for Ridge Regression.

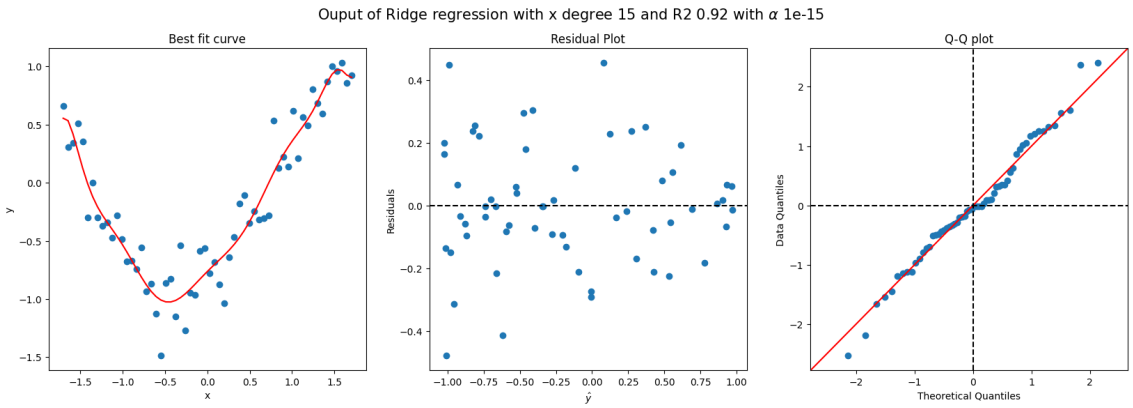


Figure 5: Highest degree of x is 15 and Regularization constant 10^{-15}

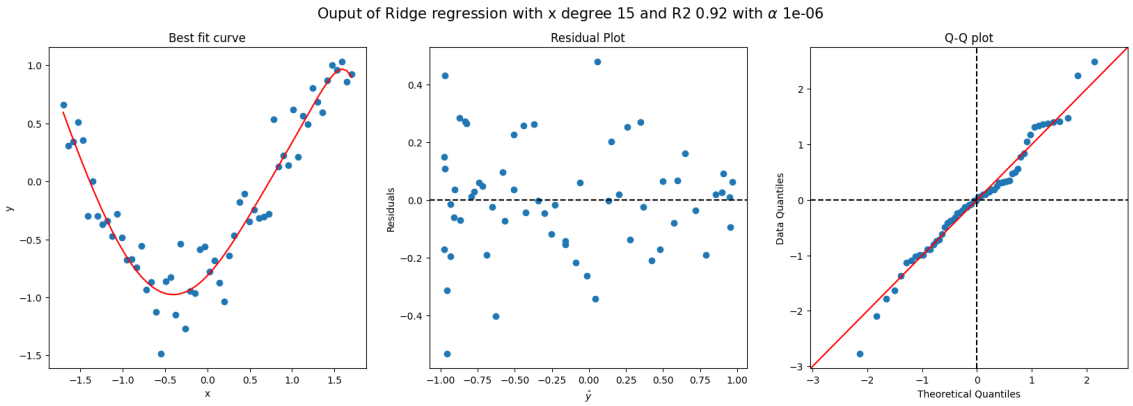


Figure 6: Highest degree of x is 15 and Regularization constant 10^{-6}

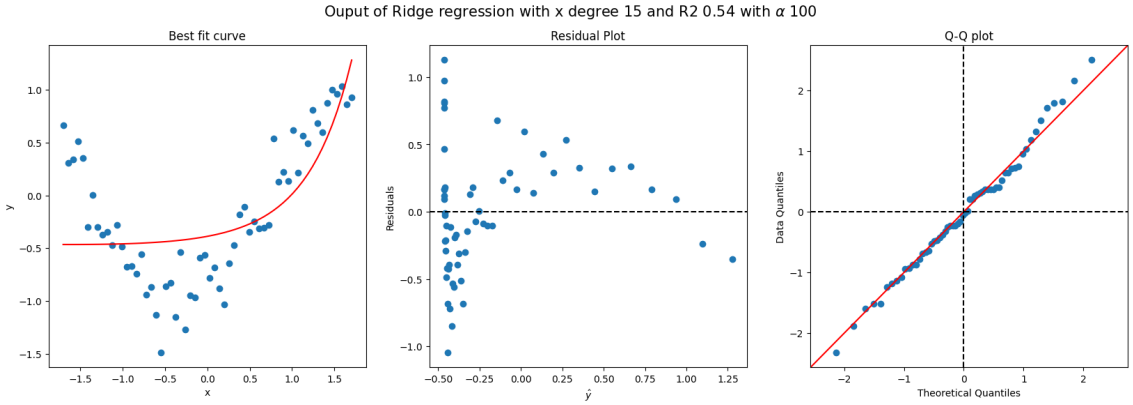


Figure 7: Highest degree of x is 15 and Regularization constant 10^{2}

0.4 Lasso Regression

b)

Importing libraries

In [589…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import scipy.cluster.hierarchy as shc
from sklearn import metrics
from sklearn.cluster import AgglomerativeClustering, DBSCAN, SpectralClustering
from sklearn.preprocessing import StandardScaler
```

Load the data

In [590…

```
Data=np.load("/content/test_data.npy",allow_pickle=True).item()
```

In [591…

```
X,labels=Data ["data"],Data ["labels"]
```

Plot functions

In [592…

```
def plot_clusters(data,true_labels=None,title_tru="True_clusters"):
  for i in range(4):
    for j in range(i+1,4):
      plt.scatter(data[:,i],data[:,j],c=true_labels)
      plt.xlabel(f'Feature {i}')
      plt.ylabel(f'Feature {j}')
      plt.title(title_tru)
      plt.show()

def compare_plot(data,true_labels,cluster_labels,title_cluster):
  for i in range(4):
    for j in range(i+1,4):
      fig,ax=plt.subplots(1,2,figsize=(12,5))
      ax[0].scatter(data[:,i],data[:,j],c=true_labels)
      ax[0].set_title("Original Data with ground labels")
      ax[0].set_xlabel(f'Feature {i}')
      ax[0].set_ylabel(f'Feature {j}')
      ax[1].scatter(data[:,i],data[:,j],c=true_labels)
      ax[1].set_title("Predicted by "+title_cluster)
      ax[1].set_xlabel(f'Feature {i}')
      ax[1].set_ylabel(f'Feature {j}')
      plt.show()
```

Hierarchical Agglomerative Clustering

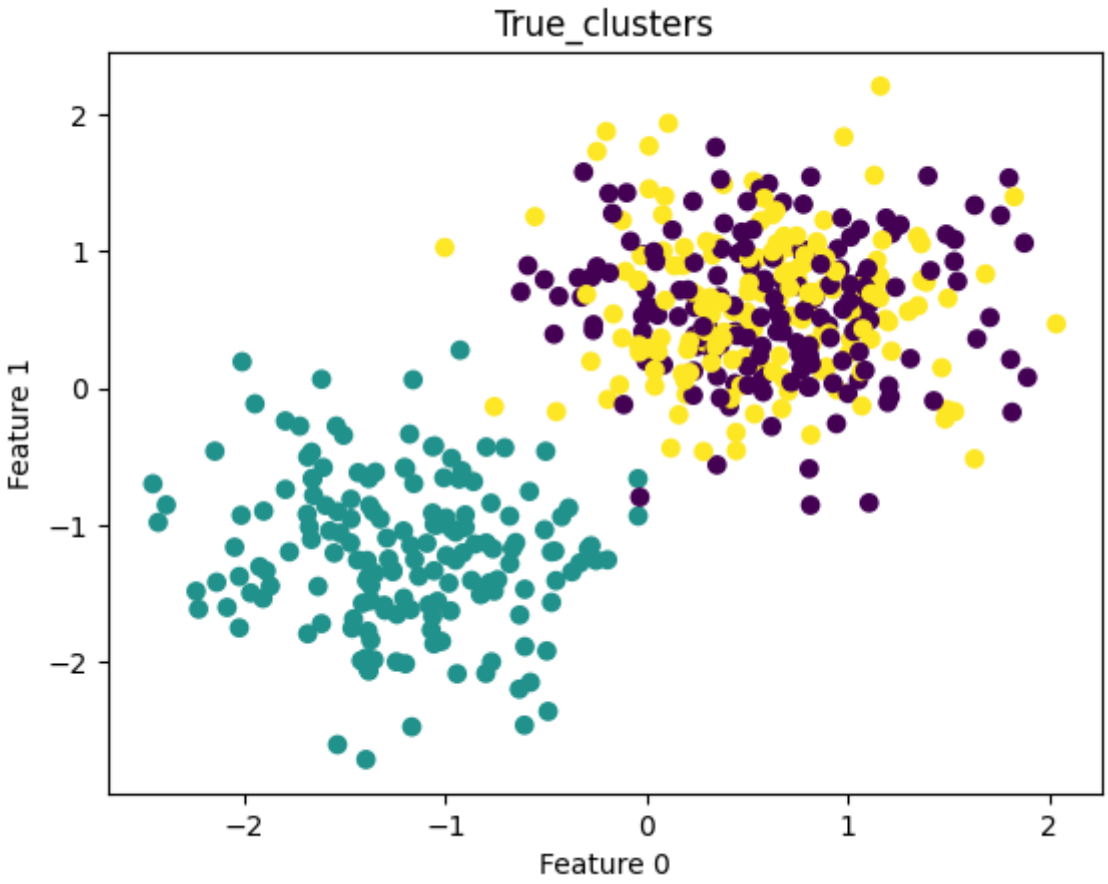
In [593…

```
S=StandardScaler()
scaled_x=S.fit_transform(X)
```

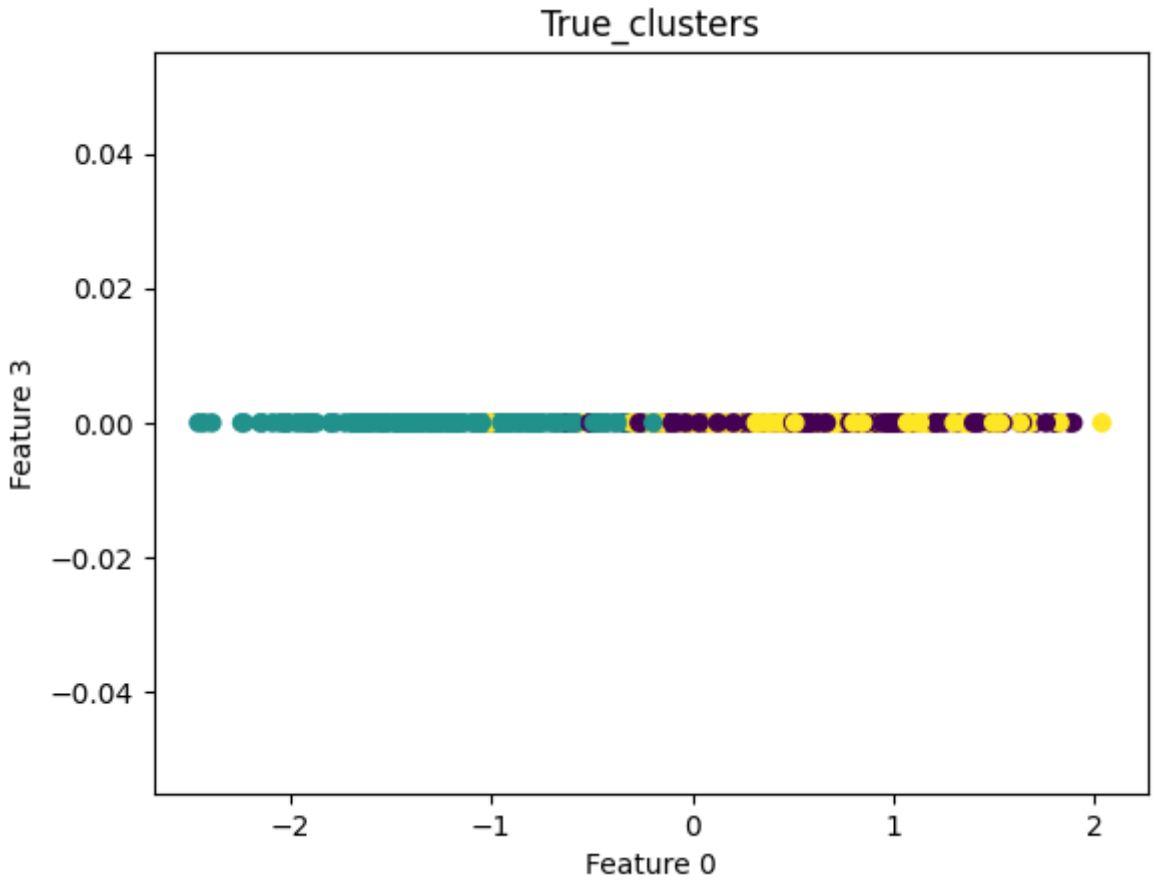
Visualizing the data with ground labels

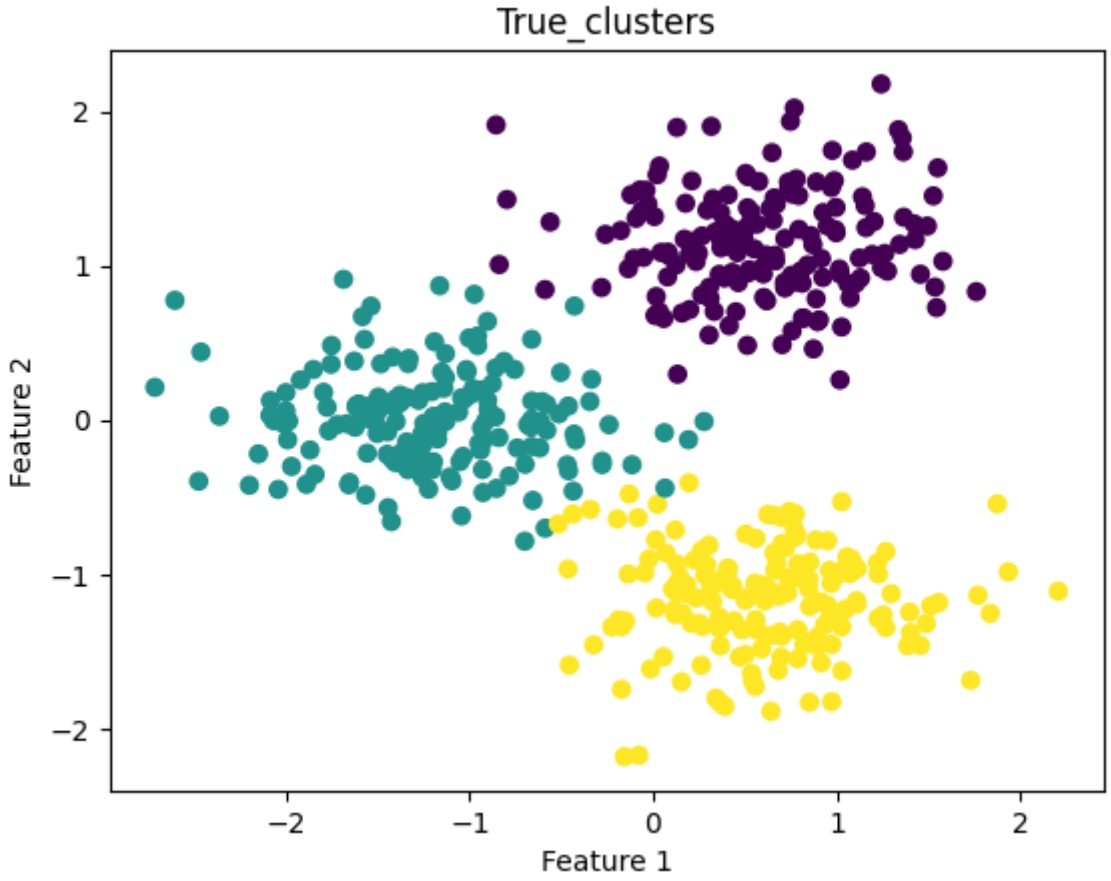
In [594…

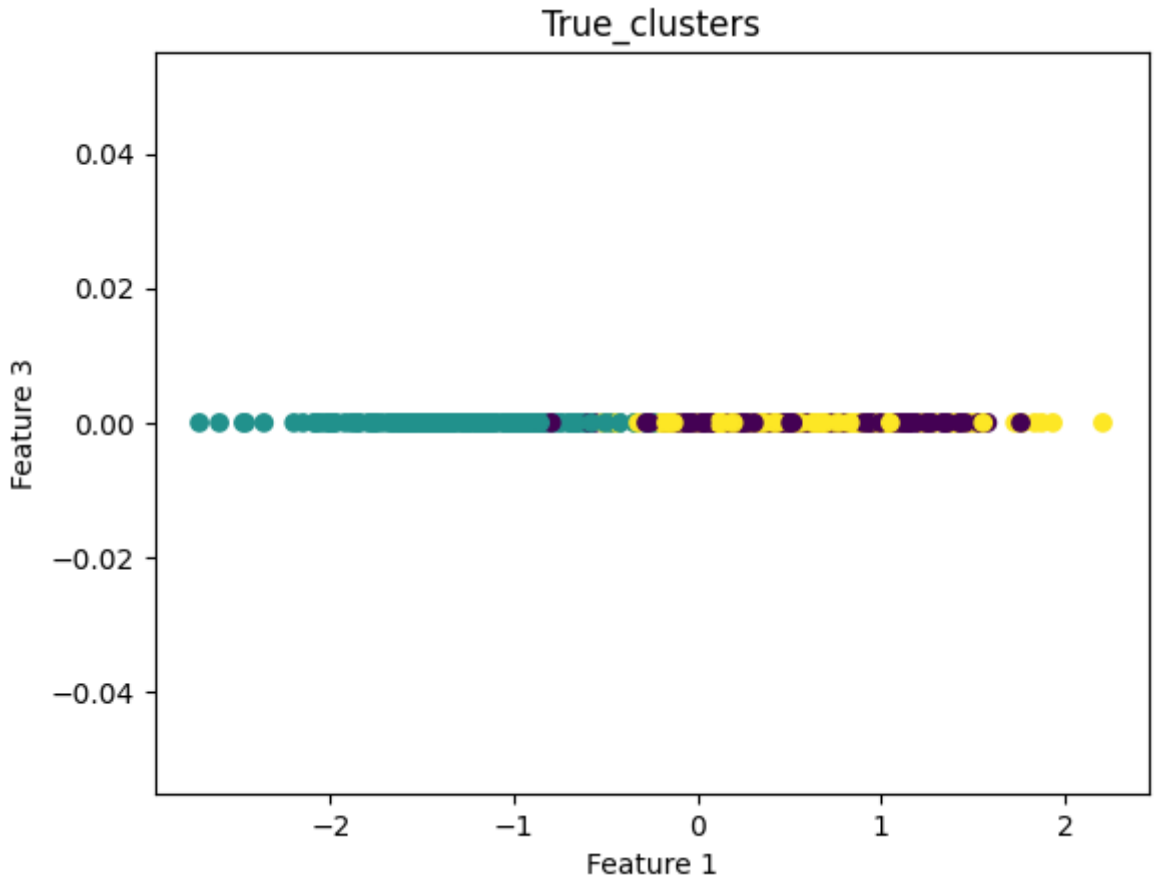
```
plot_clusters(data=scaled_x,true_labels=labels)
```

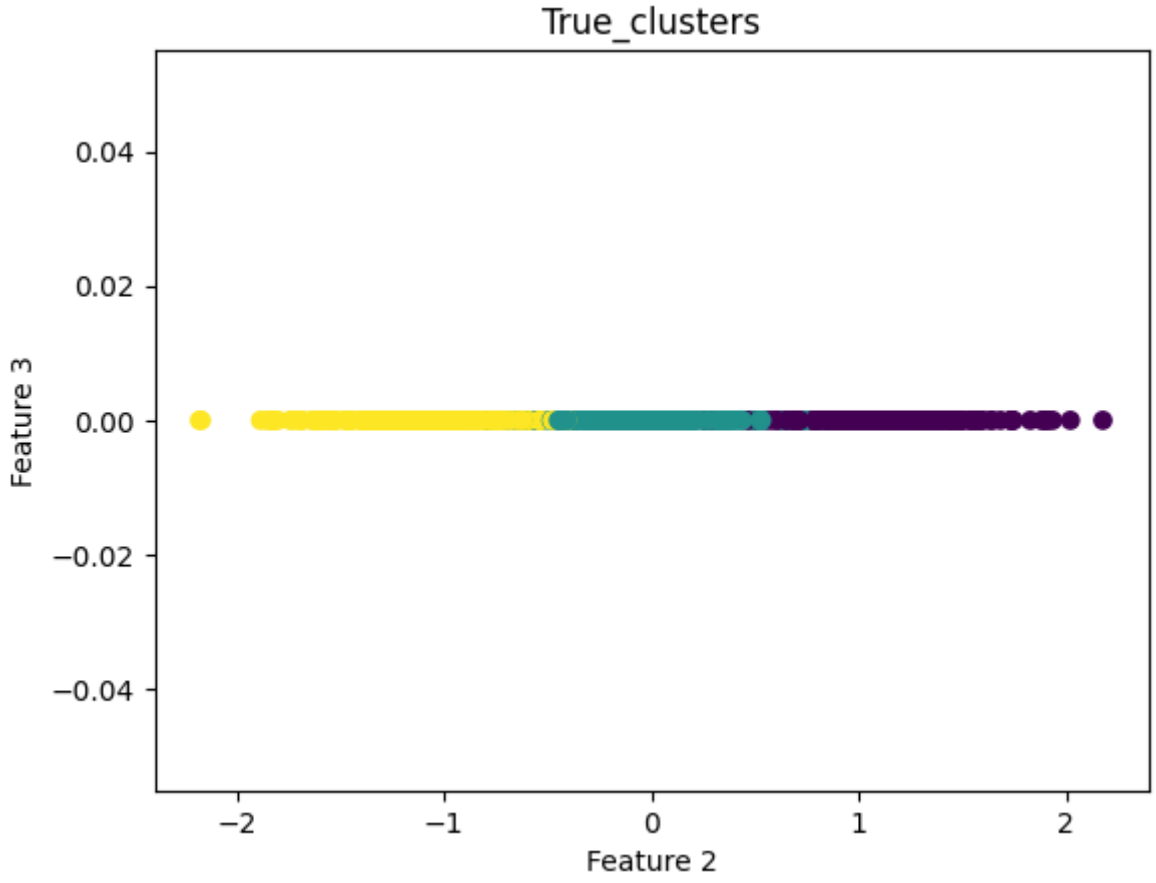

True_clusters
Feature 1
Feature 0
2
1
0
−1
−2
−2
−1
0
1
2


True_clusters
Feature 2
Feature 0
2
1
0
−1
−2
−2
−1
0
1
2


True_clusters
0.04
0.02
0.00
−0.02
−0.04
Feature 3
−2
−1
0
1
2
Feature 0


True_clusters
2
1
0
−1
−2
Feature 2
−2
−1
0
1
2
Feature 1





In [595…

```
def get_best_cluster_number_hc(X,labels,clusters):
  R=0
  M=0
  CR=0
  CM=0
  best_labels_R=np.zeros(X.shape[0])
```

```
  best_labels_M=np.zeros(X.shape[0])
  for i in clusters:
    agg_cluster=AgglomerativeClustering(n_clusters=i,metric='euclidean',linkage='c
    agg_cluster.fit(X)
    pred_labels=agg_cluster.labels_
    rand_score=metrics.adjusted_rand_score(labels,pred_labels)
    mutual_info_score=metrics.adjusted_mutual_info_score(labels,pred_labels)
    print(f'For {i} clusters the rand_score is : {rand_score}')
    print(f'For {i} clusters the mutual_info_score is : {mutual_info_score}')
    if rand_score>R:
      CR=i
      R=rand_score
      best_labels_R=pred_labels
    if mutual_info_score>M:
      CM=i
      M=mutual_info_score
      best_labels_M=pred_labels
  return (R,CR,best_labels_R),(M,CM,best_labels_M)
```

In [596…

```
clus=[1,2,3,4,5,6]#try several cluster and pick the one with highest mutual info s
A,B=get_best_cluster_number_hc(scaled_x,labels=labels,clusters=clus)
```

```
For 1 clusters the rand_score is : 0.0
For 1 clusters the mutual_info_score is : 0.0
For 2 clusters the rand_score is : 0.5711835334476844
For 2 clusters the mutual_info_score is : 0.7334012369626347
For 3 clusters the rand_score is : 1.0
For 3 clusters the mutual_info_score is : 1.0
For 4 clusters the rand_score is : 0.891353849423029
For 4 clusters the mutual_info_score is : 0.9149401169151868
For 5 clusters the rand_score is : 0.7681460488985477
For 5 clusters the mutual_info_score is : 0.8406746240013129
For 6 clusters the rand_score is : 0.6970602539070745
For 6 clusters the mutual_info_score is : 0.7991201552864633
```

Best score and number of cluster by agglomerative clustering

In [597…

```
print(f'best rand index score {A[0]} and number of clusters {A[1]}')
print(f'best mutual information score {B[0]} and number of clusters {B[1]}')
```

```
best rand index score 1.0 and number of clusters 3
best mutual information score 1.0 and number of clusters 3
```

Silhouette Score corresponding to labels with max rand_score

In [598…

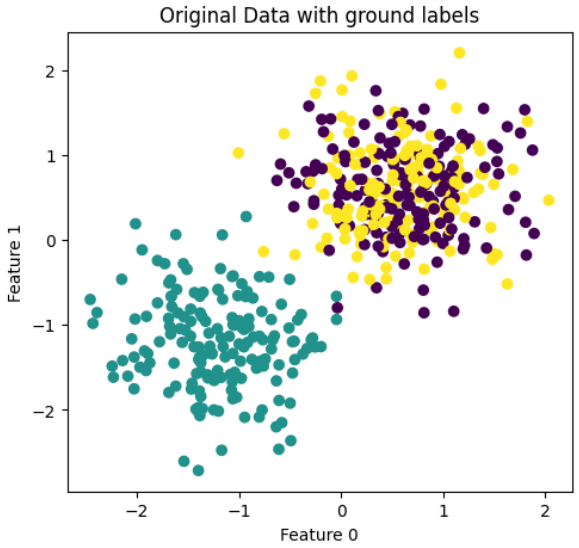
```
s_hc=metrics.silhouette_score(scaled_x,A[2])
print(f'Silhouette score : {s_hc}')
```

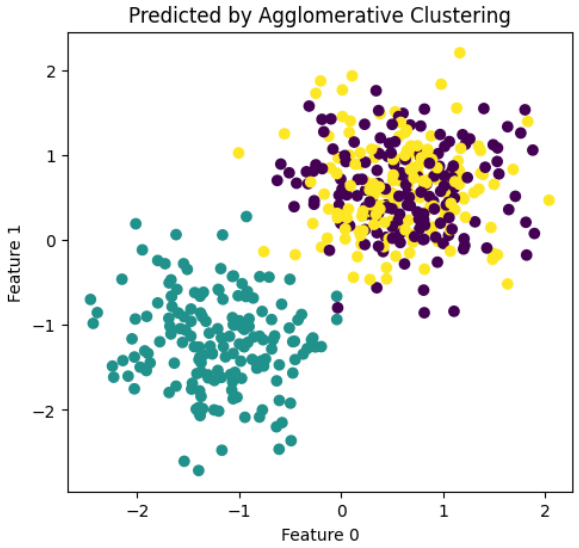
```
Silhouette score : 0.5878535006929564
```

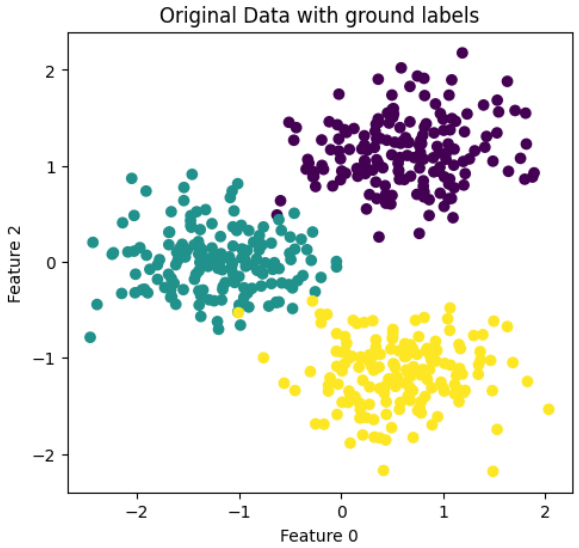
Plot with Predicted Labels

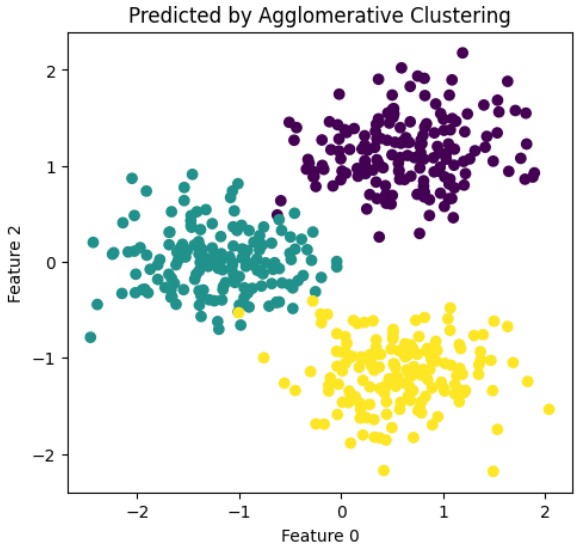
In [599…

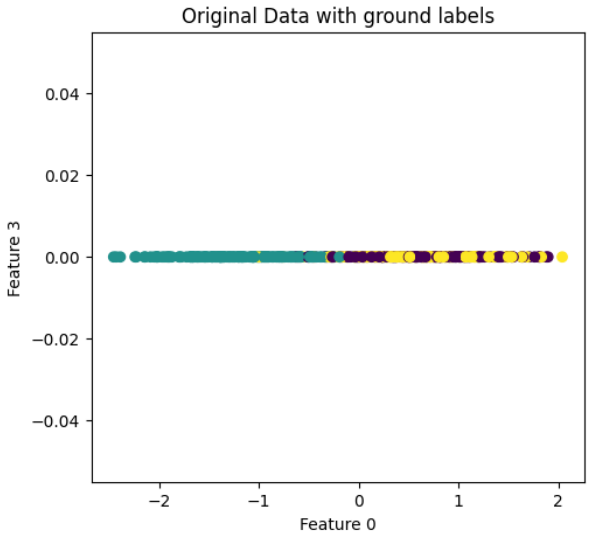
```
compare_plot(scaled_x,labels,cluster_labels=A[2],title_cluster='Agglomerative Clus
```

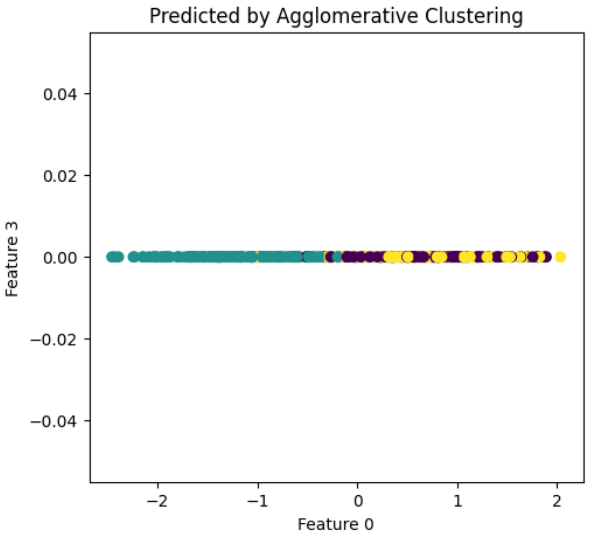

Original Data with ground labels
Feature 1
Feature 0

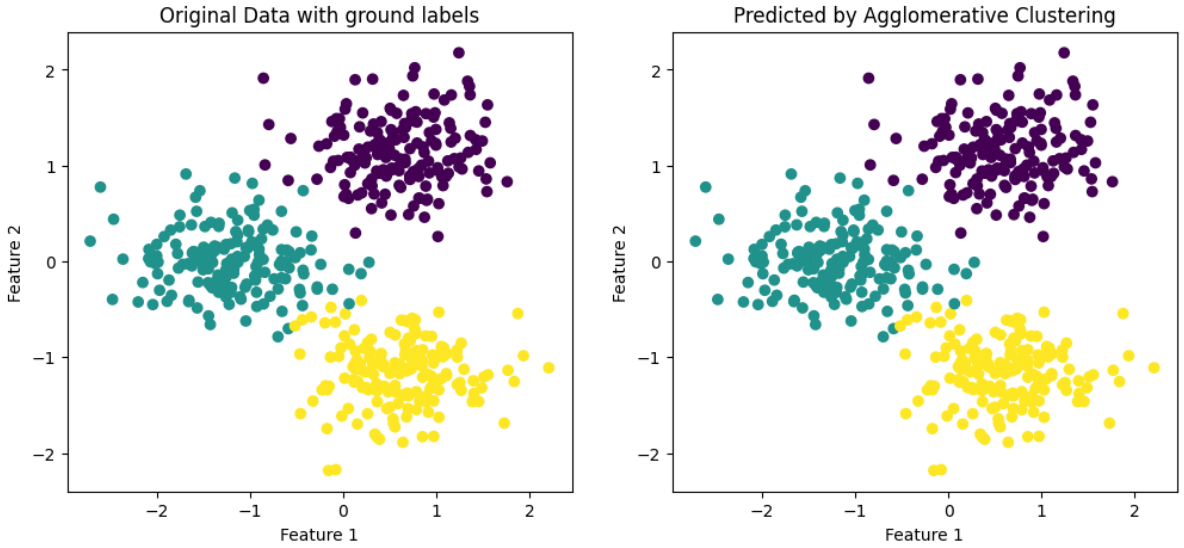

Predicted by Agglomerative Clustering
Feature 1
Feature 0

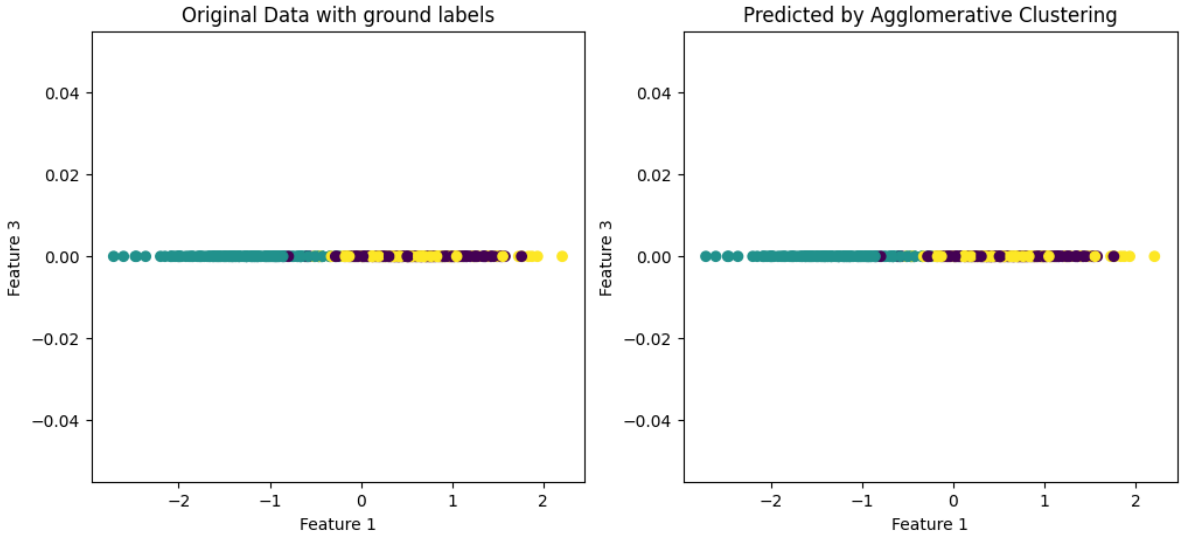

Original Data with ground labels
Feature 2
Feature 0

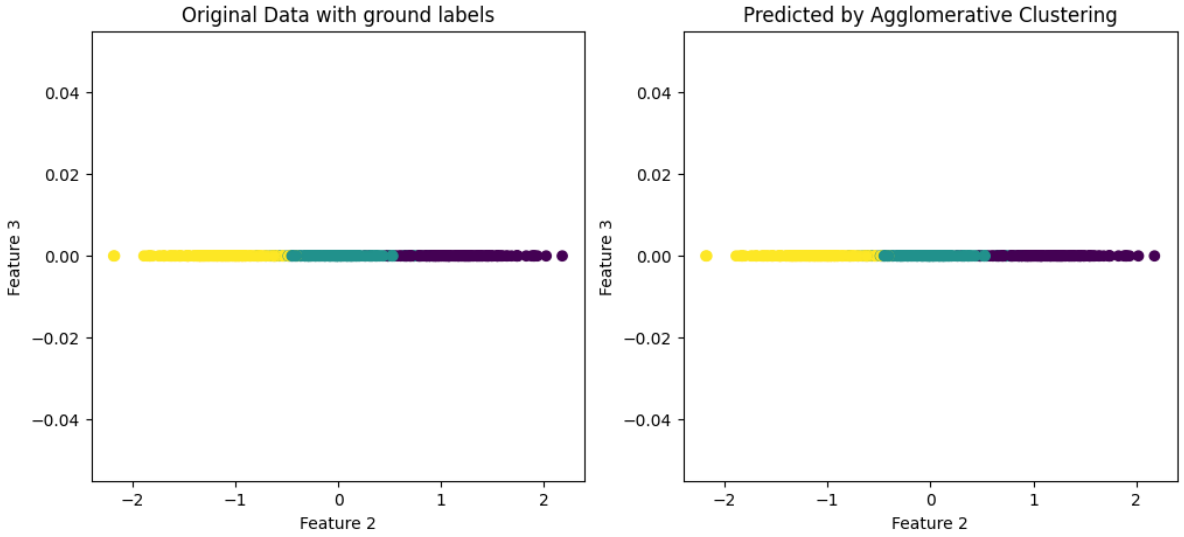

Predicted by Agglomerative Clustering
Feature 2
Feature 0


Original Data with ground labels
Feature 3
Feature 0


Predicted by Agglomerative Clustering
Feature 3
Feature 0

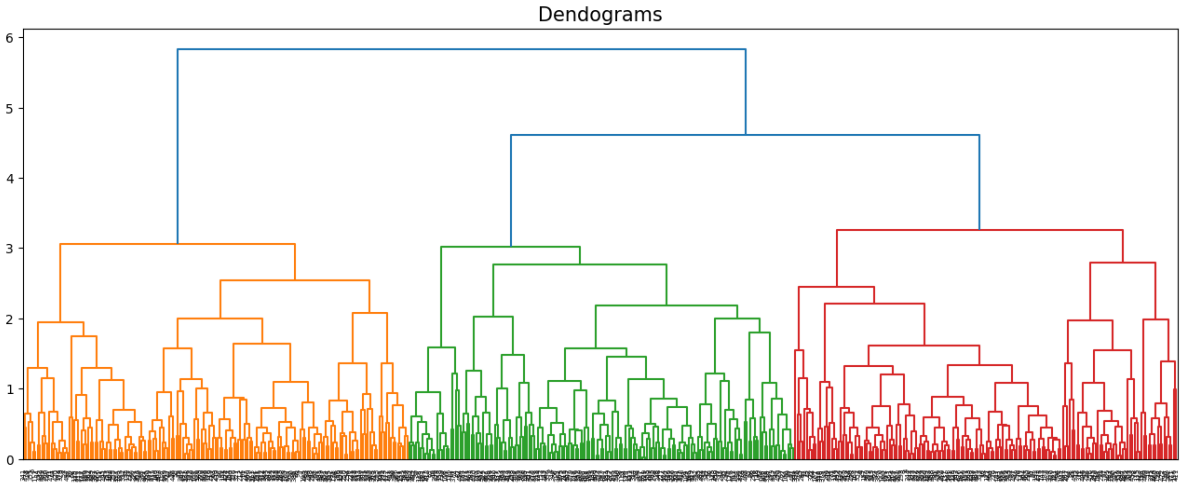






In [600…

```
plt.figure(figsize=(16,6))
plt.title("Dendograms",fontsize=15)
Z=shc.linkage(scaled_x,method="complete")
dend=shc.dendrogram(Z)
```



This also suggests 3 clusters are present in the given data

DBSCAN

In [601…

```
def get_best_cluster_number_db(X,labels):
  R=0
  M=0
  CR=0
  CM=0
  best_labels_R=np.zeros(X.shape[0])
  best_labels_M=np.zeros(X.shape[0])
  e=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]#various epsilon radius
  for i in e:
    db=DBSCAN(eps=i,min_samples=3,metric='euclidean')
    db.fit(X)
    pred_labels=db.labels_
    rand_score=metrics.adjusted_rand_score(labels,pred_labels)
    mutual_info_score=metrics.adjusted_mutual_info_score(labels,pred_labels)
    print(f'For {i} epsilon the rand_score is : {rand_score}')
    print(f'For {i} epsilon the mutual_info_score is : {mutual_info_score}')
    if rand_score>R:
      CR=i
      R=rand_score
      best_labels_R=pred_labels
    if mutual_info_score>M:
      CM=i
      M=mutual_info_score
      best_labels_M=pred_labels
  return (R,CR,best_labels_R),(M,CM,best_labels_M)
```

In [602…

```
C,D=get_best_cluster_number_db(scaled_x,labels=labels)
```

```
For 0.1 epsilon the rand_score is : 0.0001504660760463416
For 0.1 epsilon the mutual_info_score is : 0.026225009678618516
For 0.2 epsilon the rand_score is : 0.04977396174948663
For 0.2 epsilon the mutual_info_score is : 0.29413812425436414
For 0.3 epsilon the rand_score is : 0.6746679696725888
For 0.3 epsilon the mutual_info_score is : 0.6944192672135717
For 0.4 epsilon the rand_score is : 0.855901493945305
For 0.4 epsilon the mutual_info_score is : 0.8276158944361047
For 0.5 epsilon the rand_score is : 0.9577815135999573
For 0.5 epsilon the mutual_info_score is : 0.9335191180627351
For 0.6 epsilon the rand_score is : 0.9759934521503237
For 0.6 epsilon the mutual_info_score is : 0.9585671330154851
For 0.7 epsilon the rand_score is : 0.5661551263318177
For 0.7 epsilon the mutual_info_score is : 0.7149041566628729
For 0.8 epsilon the rand_score is : -7.903772152369691e-06
For 0.8 epsilon the mutual_info_score is : -0.0016442859802544261
For 0.9 epsilon the rand_score is : 0.0
For 0.9 epsilon the mutual_info_score is : 0.0
```

Best score and number of cluster by DBSCAN

In [603…

```
print(f'best rand index score {C[0]} and epsilon radius {C[1]}')
print(f'best mutual information score {D[0]} and epsilon radius {D[1]}')
```

```
best rand index score 0.9759934521503237 and epsilon radius 0.6
best mutual information score 0.9585671330154851 and epsilon radius 0.6
```

Silhouette Score corresponding to labels with max rand_score

In [604…

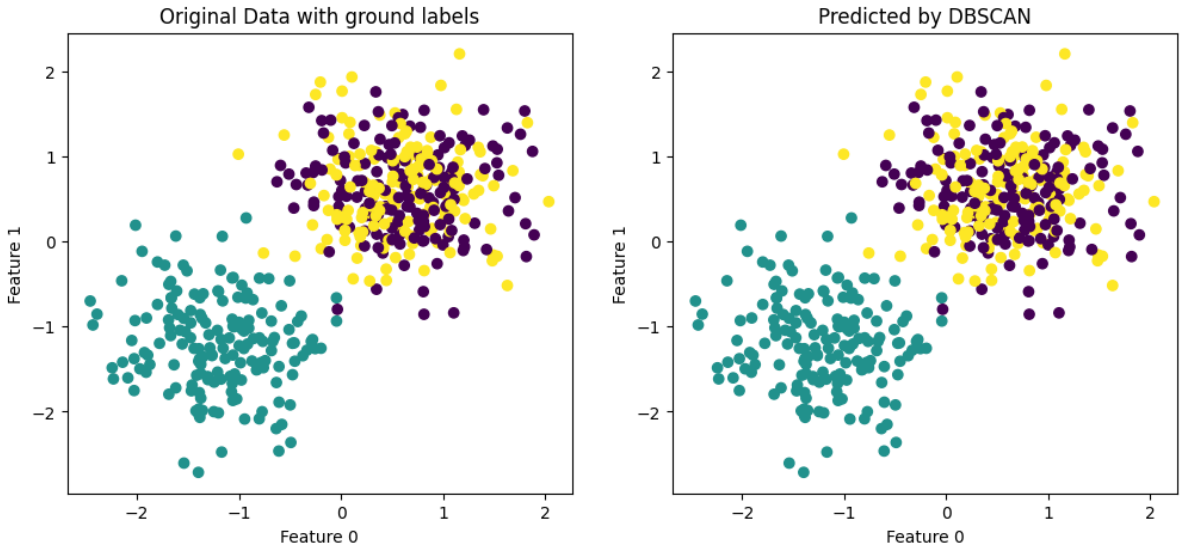
```
s_db=metrics.silhouette_score(scaled_x,C[2])
print(f'Silhouette score : {s_db}')
```

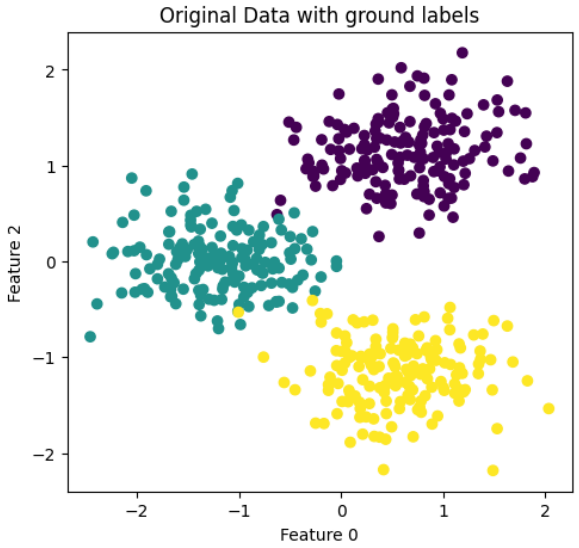
```
Silhouette score : 0.5562688222138219
```

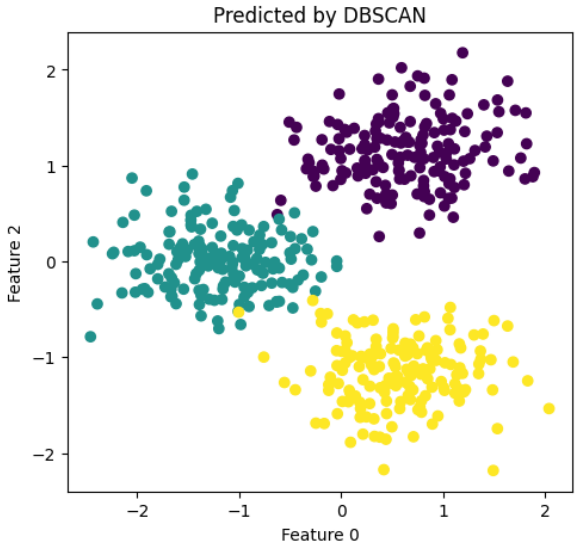
Plot with Predicted Labels

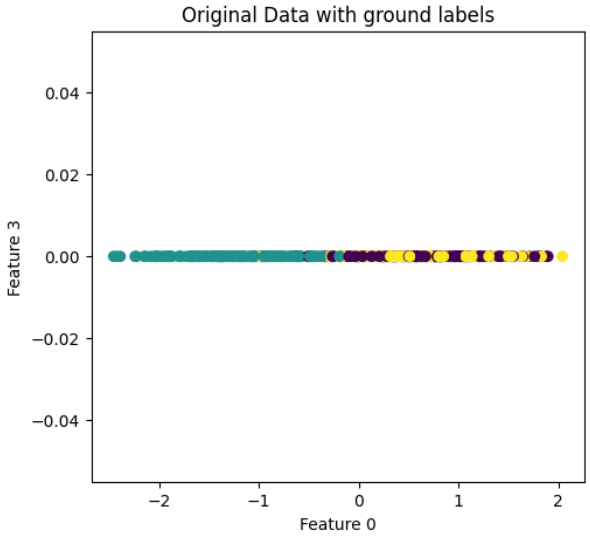
In [605…

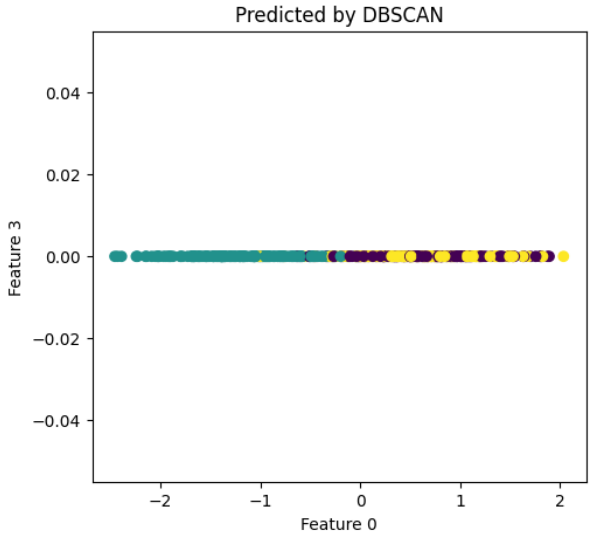
```
compare_plot(scaled_x,labels,cluster_labels=C[2],title_cluster='DBSCAN')
```

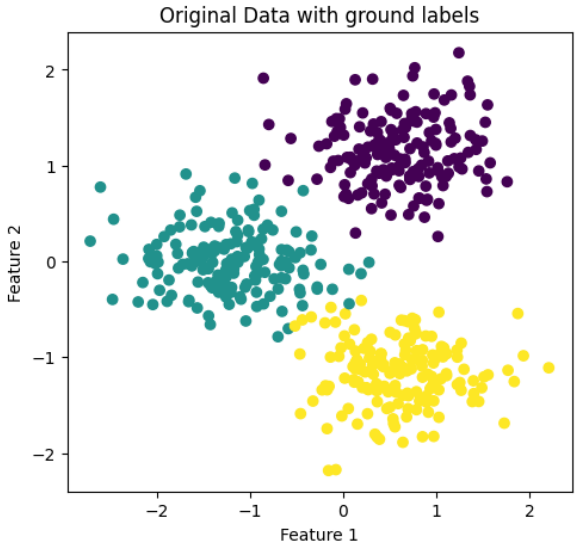


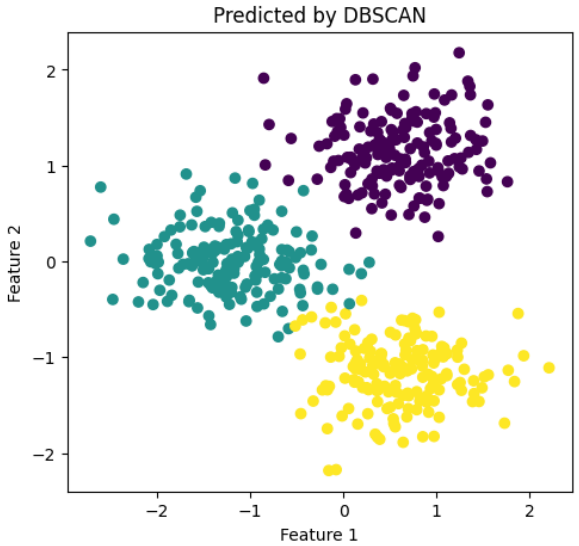

Original Data with ground labels
Feature 2
Feature 0

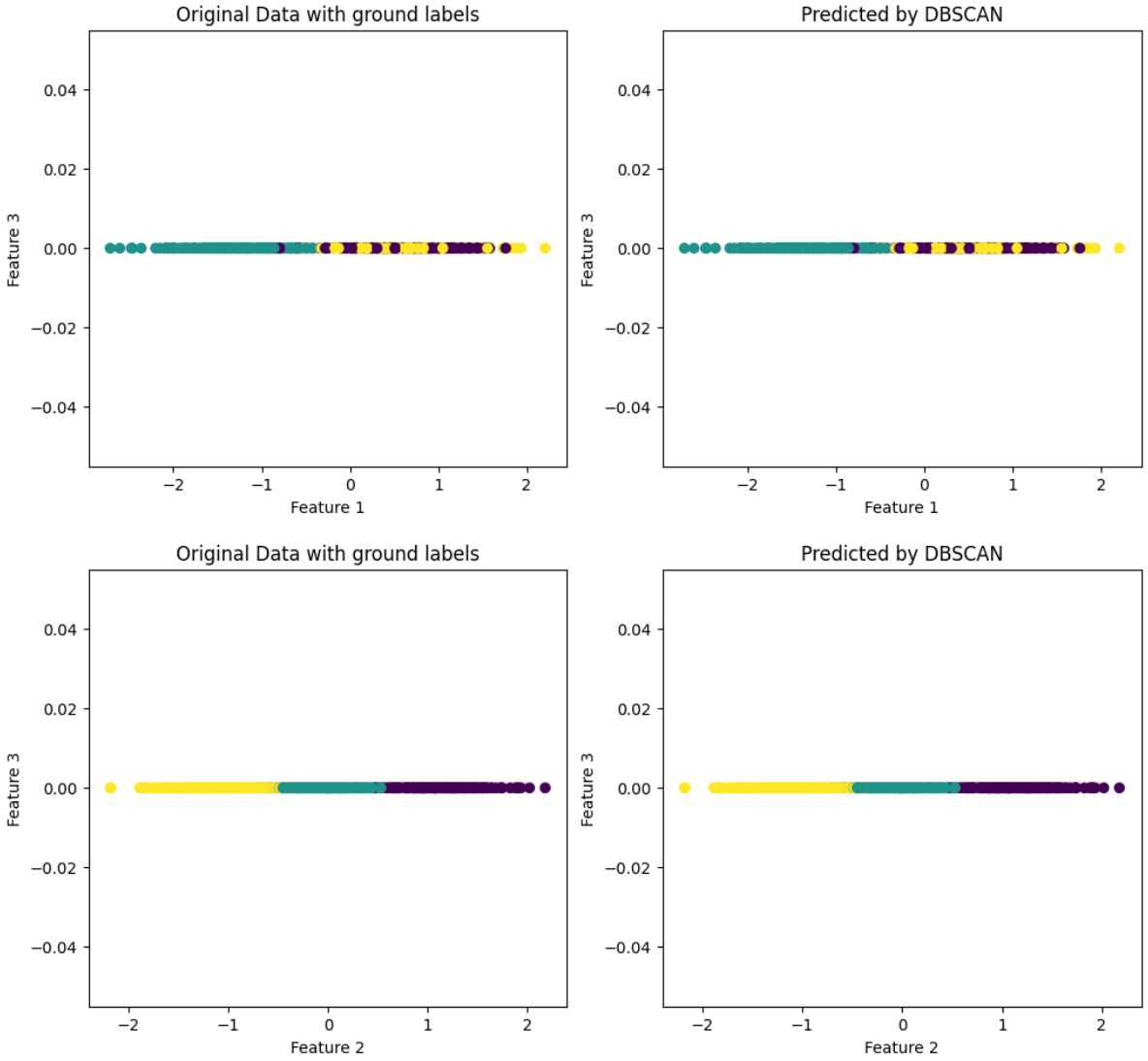

Predicted by DBSCAN
Feature 2
Feature 0


Original Data with ground labels
Feature 3
Feature 0


Predicted by DBSCAN
Feature 3
Feature 0


Original Data with ground labels
Feature 2
Feature 1


Predicted by DBSCAN
Feature 2
Feature 1



Spectral Clustering

In [606…

```
def get_best_cluster_number_sc(X,labels,num_clus):
  R=0
  M=0
  CR=0
  CM=0
  best_labels_R=np.zeros(X.shape[0])
  best_labels_M=np.zeros(X.shape[0])
  for i in num_clus:
    SC=SpectralClustering(n_clusters=i,affinity="nearest_neighbors",assign_labels=
    SC.fit(X)
    pred_labels=SC.labels_
    rand_score=metrics.adjusted_rand_score(labels,pred_labels)
    mutual_info_score=metrics.adjusted_mutual_info_score(labels,pred_labels)
    print(f'For {i} clusters the rand_score is : {rand_score}')
    print(f'For {i} clusters the mutual_info_score is : {mutual_info_score}')
    if rand_score>R:
      CR=i
      R=rand_score
      best_labels_R=pred_labels
      A=SC.affinity_matrix_
    if mutual_info_score>M:
      CM=i
      M=mutual_info_score
```

```
        best_labels_M=pred_labels
    return (R,CR,best_labels_R),(M,CM,best_labels_M),A
```

In [607…

```
E,F,c=get_best_cluster_number_sc(scaled_x,labels,clus)
```

```
For 1 clusters the rand_score is : 0.0
For 1 clusters the mutual_info_score is : 0.0
For 2 clusters the rand_score is : 0.5711835334476844
For 2 clusters the mutual_info_score is : 0.7334012369626347
For 3 clusters the rand_score is : 1.0
For 3 clusters the mutual_info_score is : 1.0
For 4 clusters the rand_score is : 0.8720115739236267
For 4 clusters the mutual_info_score is : 0.9058444342815657
For 5 clusters the rand_score is : 0.7364869235348988
For 5 clusters the mutual_info_score is : 0.8288512885673662
For 6 clusters the rand_score is : 0.5959711069669162
For 6 clusters the mutual_info_score is : 0.7659446747638127
```

In [608…

```
print(f'best rand index score {E[0]} and number of clusters is {E[1]}')
print(f'best mutual information score {F[0]} and number of clusters is {F[1]}')
```

```
best rand index score 1.0 and number of clusters is 3
best mutual information score 1.0 and number of clusters is 3
```

Silhouette Score corresponding to labels with max scores(Rand_Index and Mutual information)

In [609…

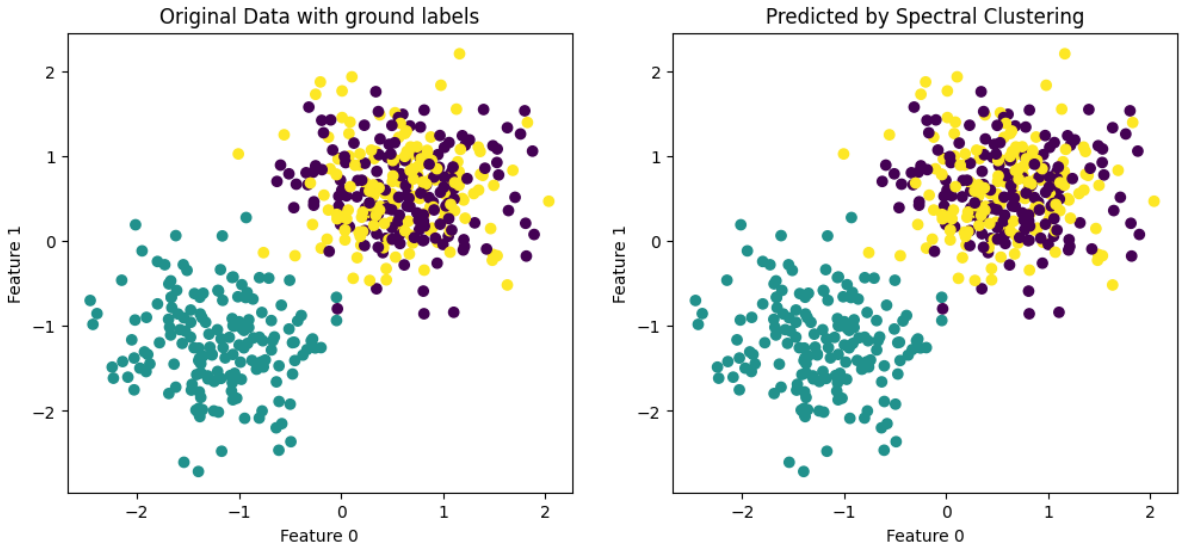
```
s_ss=metrics.silhouette_score(scaled_x,E[2])
print(f'Silhouette score :  {s_ss}')
```

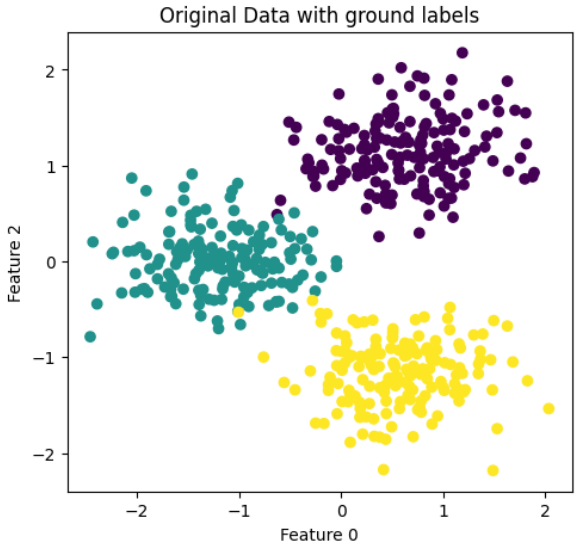
```
Silhouette score :  0.5878535006929564
```

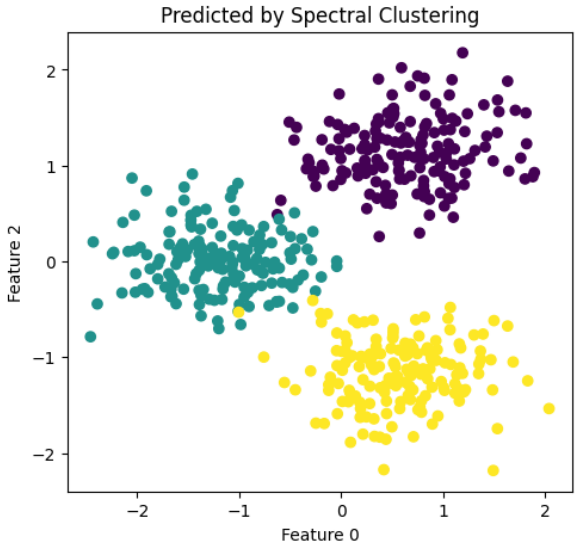
Plot with Predicted Labels

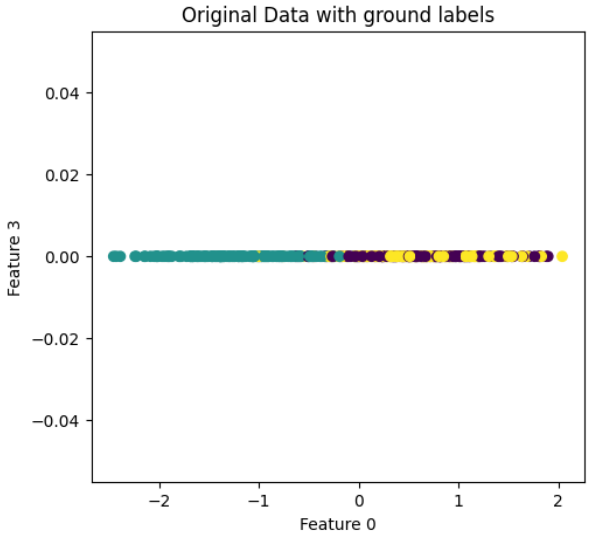
In [610…

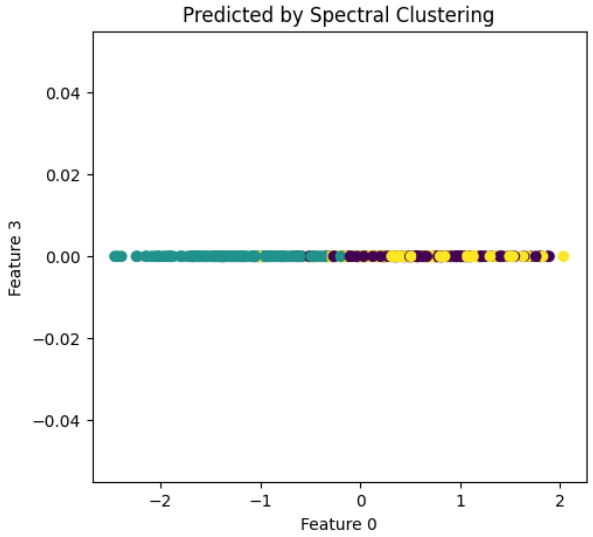
```
compare_plot(scaled_x,labels,cluster_labels=E[2],title_cluster='Spectral Clusterin
```

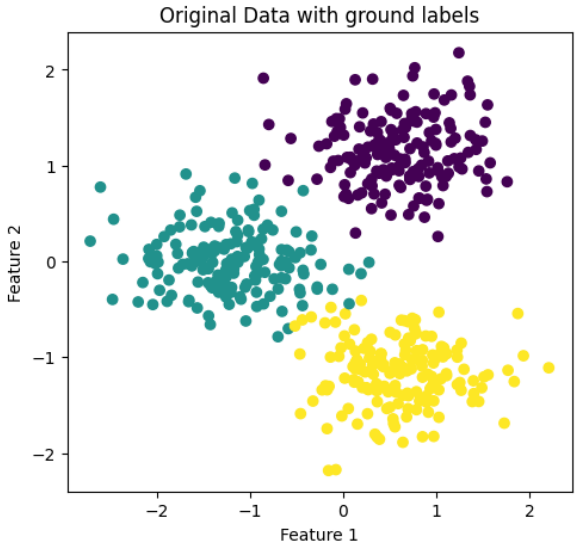


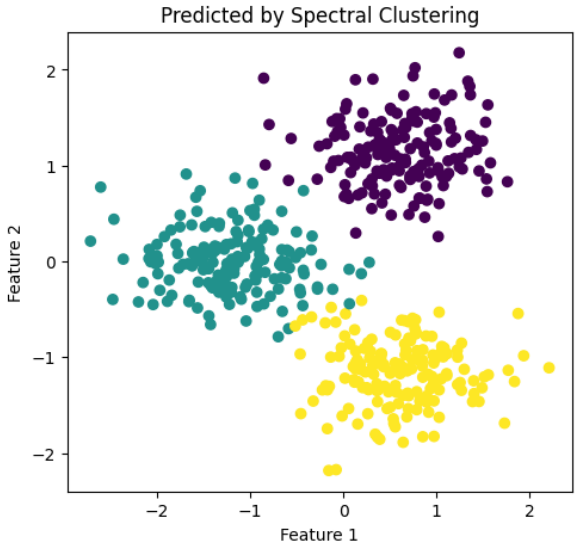

Original Data with ground labels
Feature 2
Feature 0

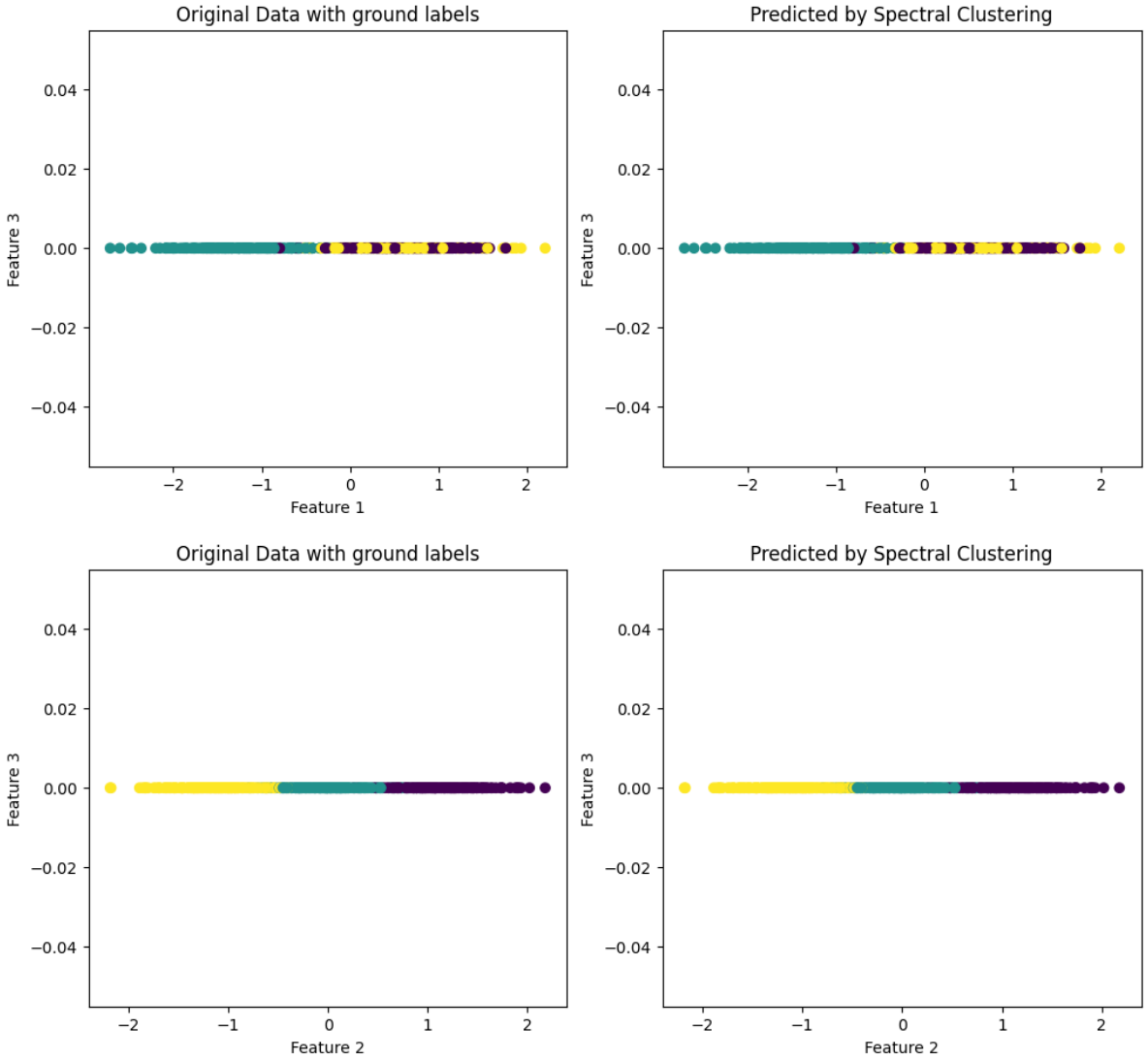

Predicted by Spectral Clustering
Feature 2
Feature 0


Original Data with ground labels
Feature 3
Feature 0


Predicted by Spectral Clustering
Feature 3
Feature 0


Original Data with ground labels
Feature 2
Feature 1


Predicted by Spectral Clustering
Feature 2
Feature 1



Eigen values and vectors visualization in reduced dimension

In [611…

```
c.shape#affinity matrix dimension
```

Out[611]:

```
(500, 500)
```

In [612…

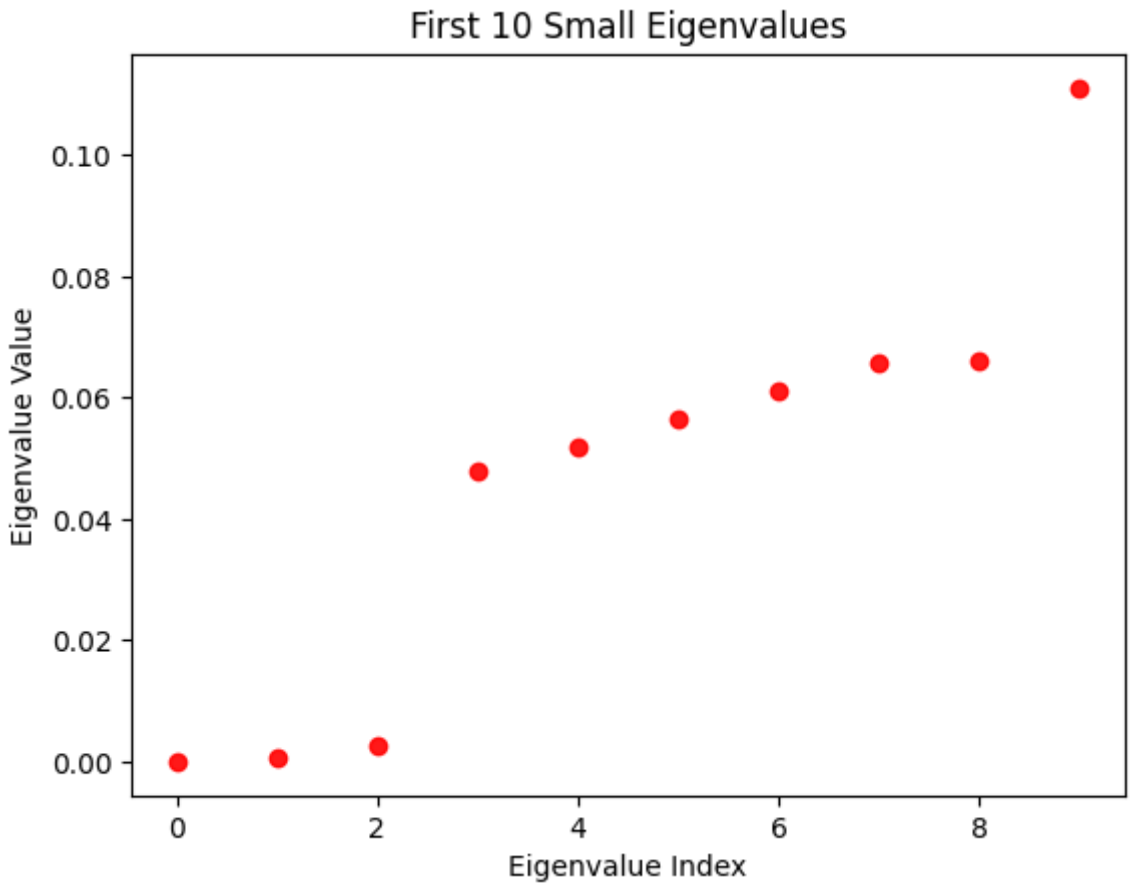
```
def get_norm_laplacian(affinity_matrix):
  degree_matrix = np.diag(np.sum(affinity_matrix.toarray(), axis=1))
  laplacian_matrix = degree_matrix - affinity_matrix.toarray()
  norm_laplace=np.matmul(np.linalg.inv(degree_matrix),laplacian_matrix)
  return norm_laplace
def get_eigens(lap,k=10):
  vals, vecs=np.linalg.eig(lap)
  index=np.argsort(vals)
  least_vals=vals[index[:k]]
  vec=vecs[:,index[:k]]
  return np.real(least_vals),np.real(vec)
def plot_eigvals(vals):
  x = np.arange(len(vals))
  plt.scatter(x, vals,marker='o',alpha=0.9,color='r')
  plt.title("First 10 Small Eigenvalues")
  plt.xlabel("Eigenvalue Index")
  plt.ylabel("Eigenvalue Value")
  plt.show()
```

```
def plot_eigvecs(vector,num_elems=200,k=5):
  fig,ax=plt.subplots(1,k,figsize=(20,5))
  for i in range(k):
    ax[i].plot(np.arange(num_elems),vector[:num_elems,i])
    ax[i].set_title(f'eigen vector {i}')
```

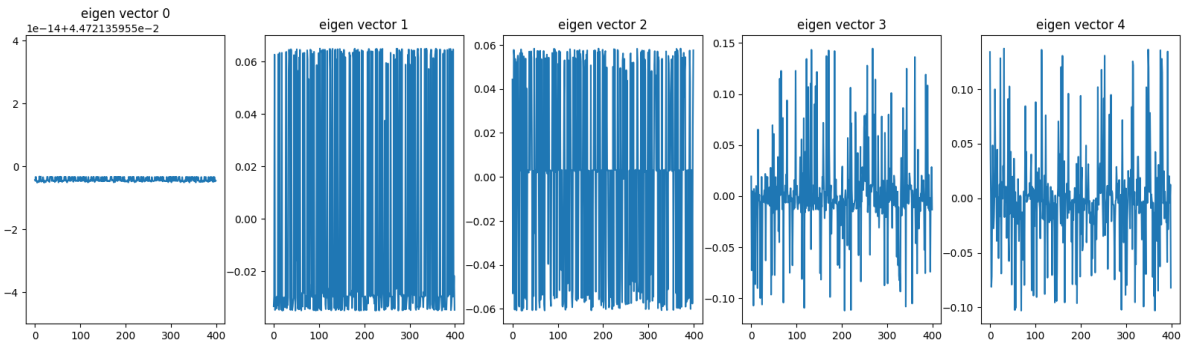
In [613…
```
L=get_norm_laplacian(c)
```

In [614…
```
val,vec=get_eigens(L,k=10)
```

In [615…
```
plot_eigvals(np.sort(val))
```



In [616…
```
plot_eigvecs(vec,num_elems=400,k=5)
```



In [617…
```
Algos=['Agglomerative Clustering','DBSCAN','Spectral Clustering']
Rand_score=[A[0],C[0],E[0]]
Mutual_info_score=[B[0],D[0],F[0]]
sil_score=[s_hc,s_db,s_ss]
T={'Algorithm':Algos,'best Rand Index':Rand_score,'best Mutual Information':Mutual_
Data=pd.DataFrame(T)
Data.set_index('Algorithm',inplace=True)
```

In [618…
```
Data
```

Out[618]:

Algorithm	best Rand Index	best Mutual Information	Silhouette Score
Agglomerative Clustering	1.000000	1.000000	0.587854
DBSCAN	0.975993	0.958567	0.556269
Spectral Clustering	1.000000	1.000000	0.587854

In [619…
```
Data.to_csv('Result.csv')
```