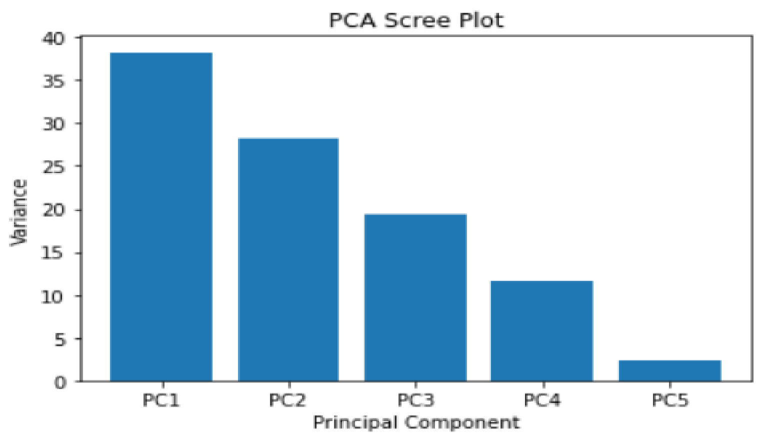
Question 2

[CM2]Representation Learning

1. **Use a scree-plot to look at the cumulative variance represented by the PCA eigenvectors,**

Scree Plot:

```
pca = PCA(n_components = 5)
pca_features = pca.fit_transform(Extracted_fetures)
percent_variance = np.round(pca.explained_variance_ratio_*100, decimals=2)
columns = ['PC1', 'PC2', 'PC3', 'PC4','PC5']
plt.bar(x= range(1,6), height=percent_variance, tick_label=columns)
plt.xlabel('Principal Component')
plt.ylabel('Variance')
plt.title('PCA Scree Plot')
plt.show()
```



```
original_data.nunique()
```

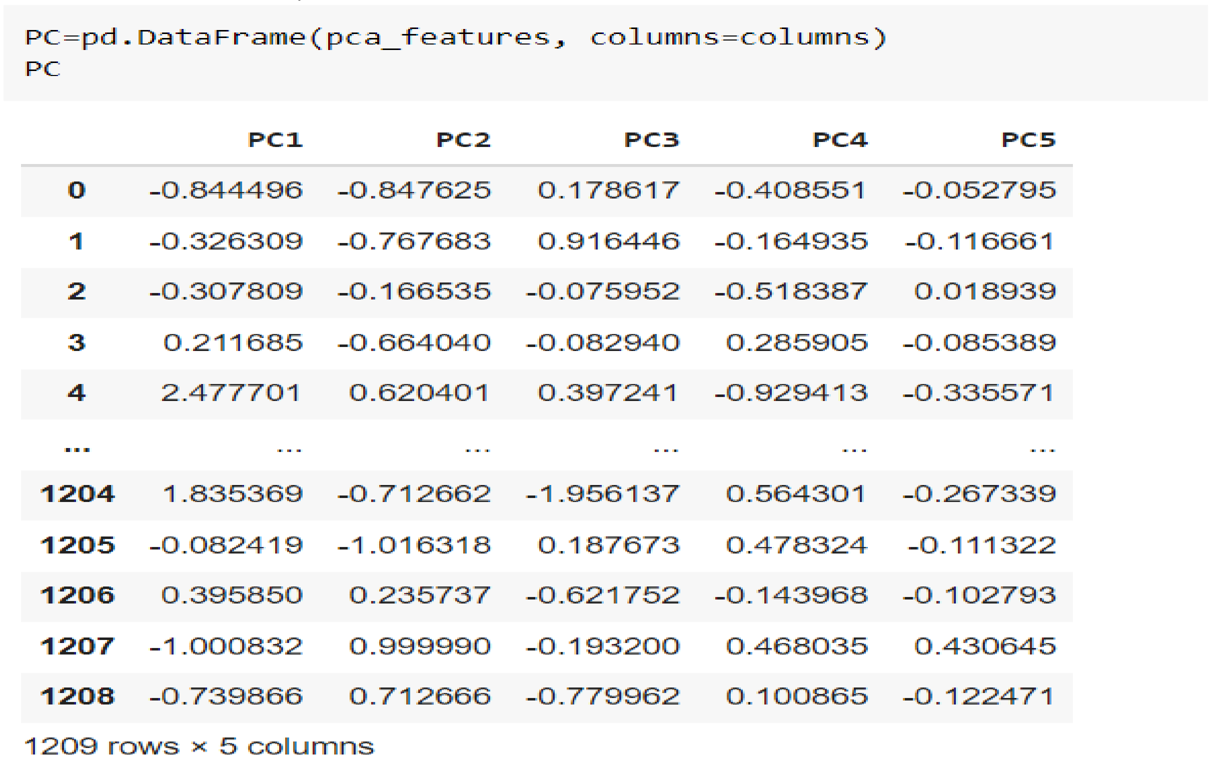
```
Day                                  30
State ID                             41
State                                41
Lat                                  41
Long_                                41
Active                             1162
Incident_Rate                      1159
Total_Test_Results                 1127
Case_Fatality_Ratio                1159
Testing_Rate                       1127
Resident Population 2020 Census      41
Population Density 2020 Census       41
Density Rank 2020 Census             41
SexRatio                             11
Confirmed                             2
Deaths                                2
Recovered                             2
outliers                              1
dtype: int64
```

Q]
Give advice on the best number of reduced features

Answer]

From the total number of unique observations over each columns in our data, we can observe that 5 columns from Active to Testing Rate show variation and hence these are selected as the best number of reduced features to represent our data for PCA.

Therefore, Number of Components for PCA are chosen to be 5.

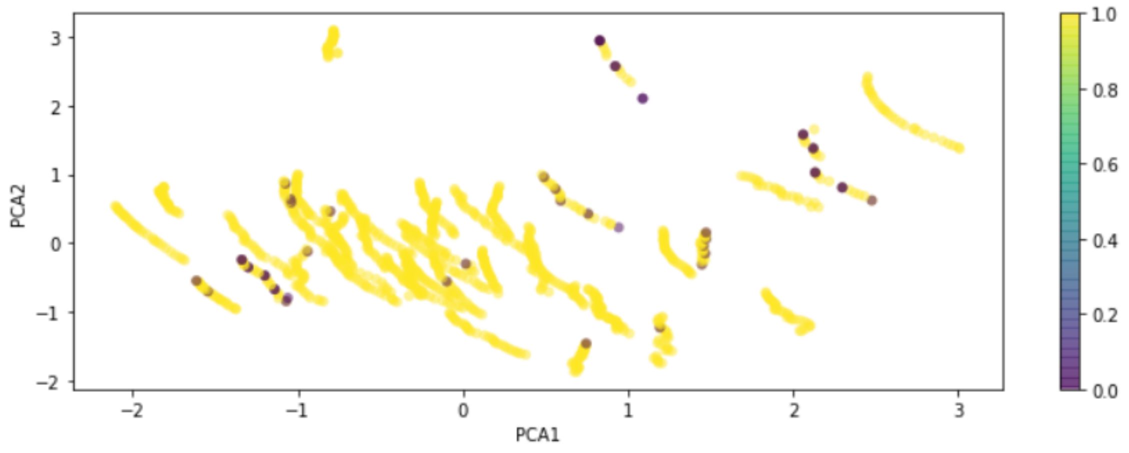


```
PC=pd.DataFrame(pca_features, columns=columns)
PC
```

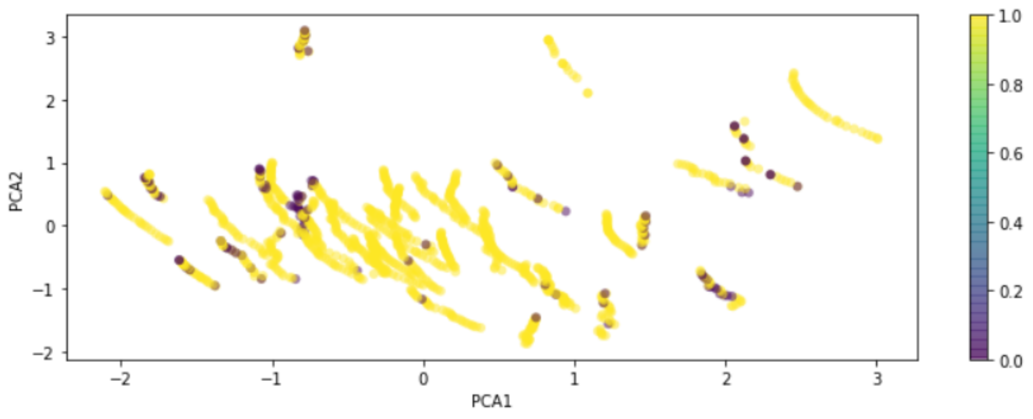
	PC1	PC2	PC3	PC4	PC5
0	-0.844496	-0.847625	0.178617	-0.408551	-0.052795
1	-0.326309	-0.767683	0.916446	-0.164935	-0.116661
2	-0.307809	-0.166535	-0.075952	-0.518387	0.018939
3	0.211685	-0.664040	-0.082940	0.285905	-0.085389
4	2.477701	0.620401	0.397241	-0.929413	-0.335571
...	...	...	...	...	...
1204	1.835369	-0.712662	-1.956137	0.564301	-0.267339
1205	-0.082419	-1.016318	0.187673	0.478324	-0.111322
1206	0.395850	0.235737	-0.621752	-0.143968	-0.102793
1207	-1.000832	0.999990	-0.193200	0.468035	0.430645
1208	-0.739866	0.712666	-0.779962	0.100865	-0.122471

1209 rows × 5 columns

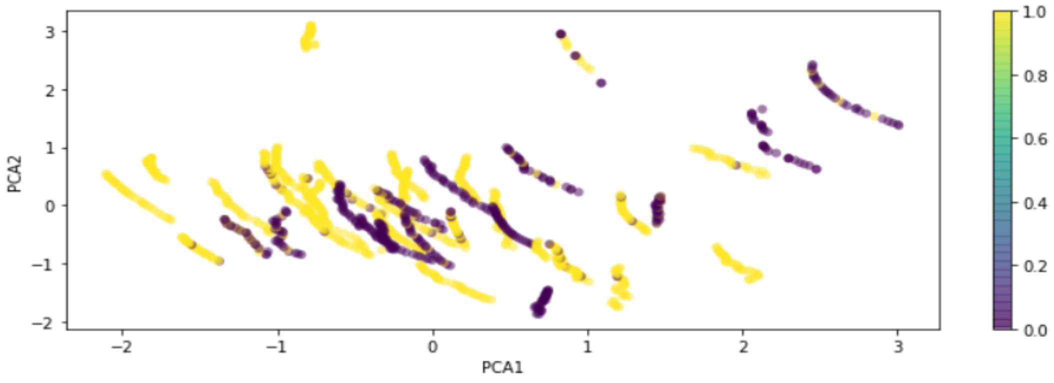
```
# plot by Confirmed
y = original_data.iloc[:, 14:17]
pca = PCA(n_components = 2)
pca_features = pca.fit_transform(Extracted_fetures)

figure(figsize=(12,4))
plt.scatter(pca_features[:, 0], pca_features[:, 1],
            c=y["Confirmed"], edgecolor='none', alpha=0.5,)
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.colorbar();
```



```
[249] # plot by deaths
      y = original_data.iloc[:, 14:17]

      pca = PCA(n_components = 2)
      pca_features = pca.fit_transform(Extracted_fetures)
      figure(figsize=(12,4))
      plt.scatter(pca_features[:, 0], pca_features[:, 1],
                  c=y["Deaths"], edgecolor='none', alpha=0.5,)
      plt.xlabel('PCA1')
      plt.ylabel('PCA2')
      plt.colorbar();
```



```
[253] # plot by Recovered
      y = original_data.iloc[:, 14:17]

      pca = PCA(n_components = 2)
      pca_features = pca.fit_transform(Extracted_fetures)
      figure(figsize=(12,4))
      plt.scatter(pca_features[:, 0], pca_features[:, 1],
                  c=y["Recovered"], edgecolor='none', alpha=0.5,)
      plt.xlabel('PCA1')
      plt.ylabel('PCA2')
      plt.colorbar();
```

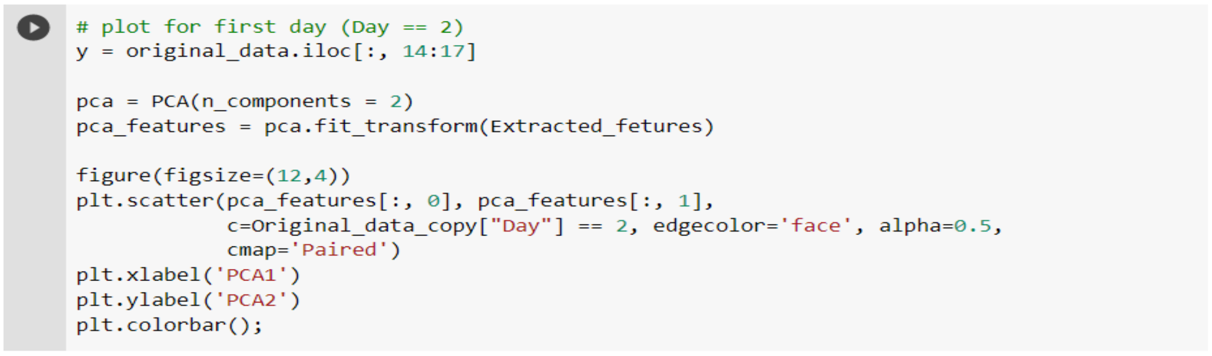


By comparing the PCA plot of Confirmed, Deaths and Recovered we can come to the conclusion that:

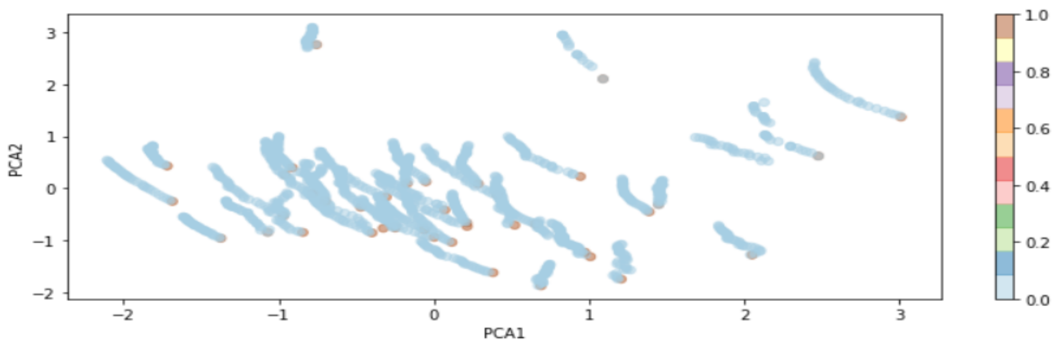
Labels	
Confirmed	We can observe more true cases than false cases here
Deaths	We can observe more true cases than false cases here
Recovered	We can observe more false cases than true cases here

2. Plot the data points, or a useful subset of them (select out one day per state? or use colour by State?) on the first two PCA vectors

Plot by Day:

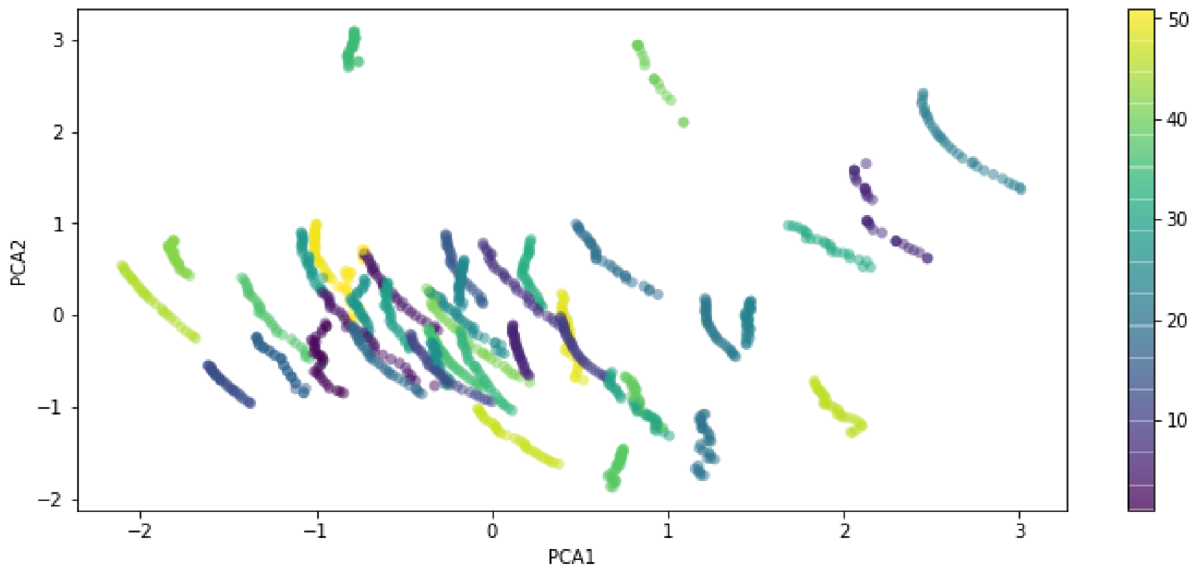


```
# plot for first day (Day == 2)
y = original_data.iloc[:, 14:17]

pca = PCA(n_components = 2)
pca_features = pca.fit_transform(Extracted_fetures)

figure(figsize=(12,4))
plt.scatter(pca_features[:, 0], pca_features[:, 1],
            c=Original_data_copy["Day"] == 2, edgecolor='face', alpha=0.5,
            cmap='Paired')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.colorbar();
```



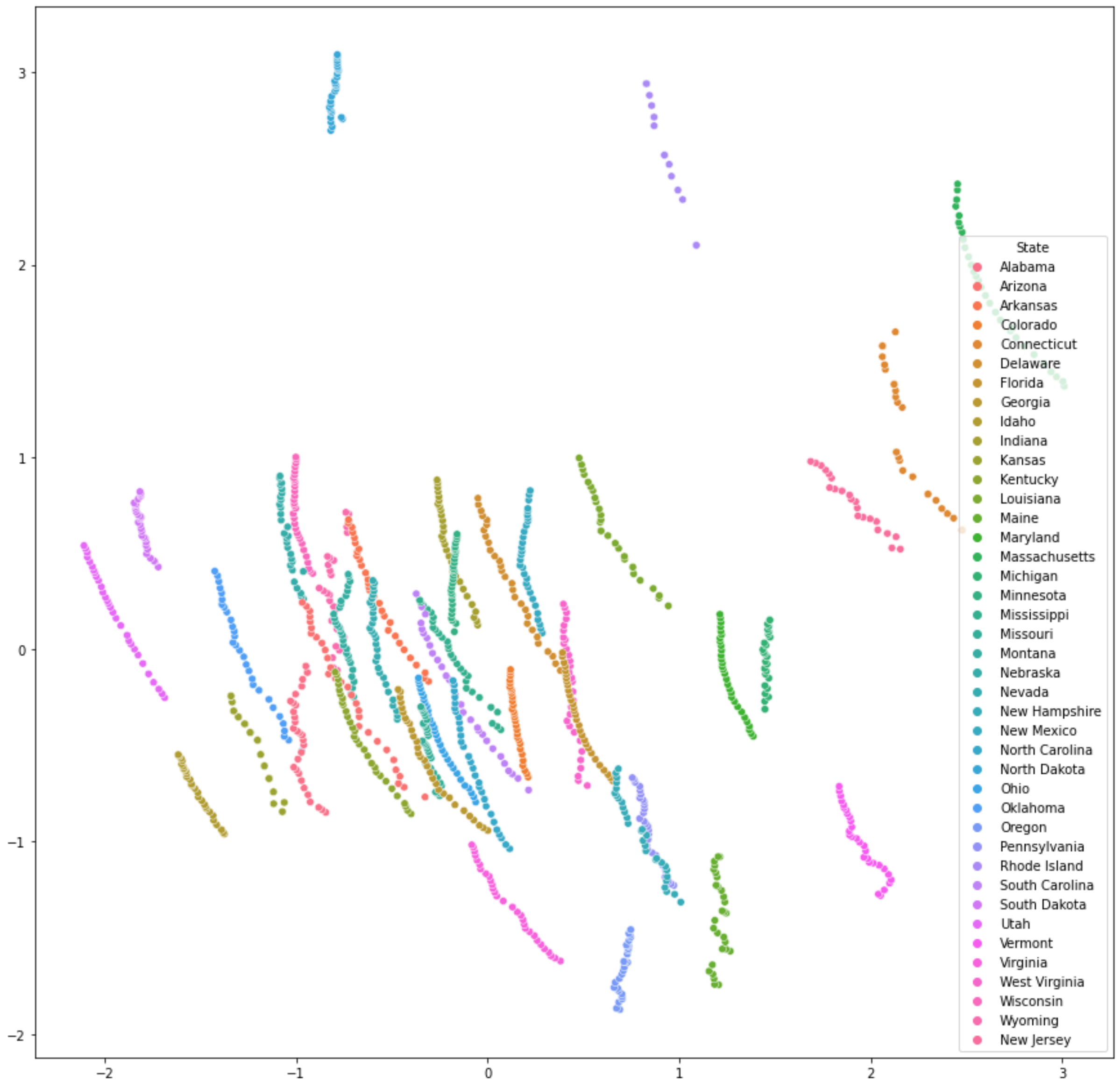
Here we can observe that for the day = 1, number of true cases are more than false cases.

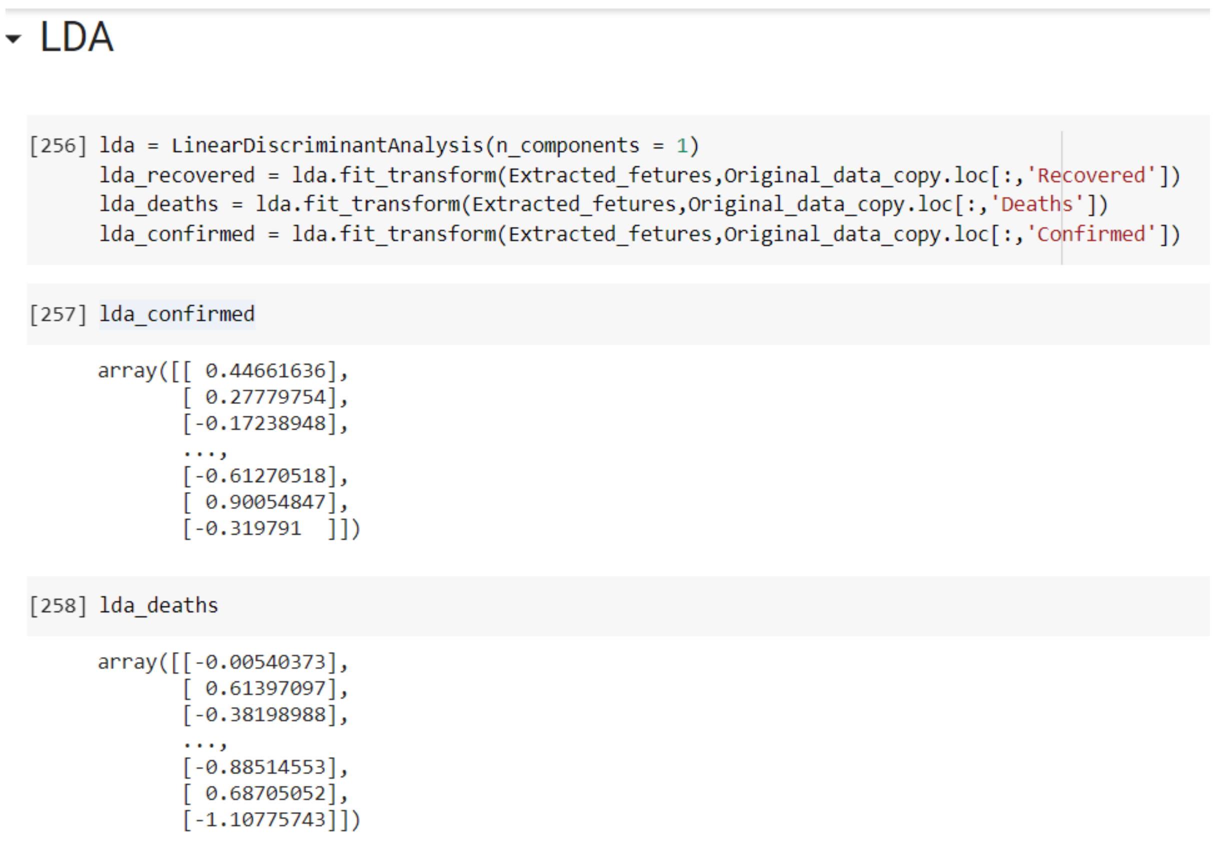
Plot by State ID:



Plot by State:

```
[ ]  # plot by State
     y = original_data.iloc[:, 14:17]

     pca = PCA(n_components = 2)
     pca_features = pca.fit_transform(Extracted_fetures)

     figure(figsize=(15,15))
     sns.scatterplot(pca_features[:, 0], pca_features[:, 1],hue=Original_data_copy["State"])
```





LDA

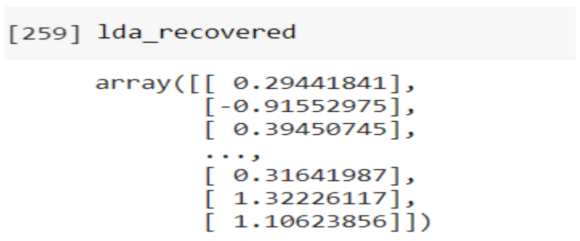
```
[256] lda = LinearDiscriminantAnalysis(n_components = 1)
      lda_recovered = lda.fit_transform(Extracted_fetures,Original_data_copy.loc[:,'Recovered'])
      lda_deaths = lda.fit_transform(Extracted_fetures,Original_data_copy.loc[:,'Deaths'])
      lda_confirmed = lda.fit_transform(Extracted_fetures,Original_data_copy.loc[:,'Confirmed'])
```

```
[257] lda_confirmed
```

```
      array([[ 0.44661636],
             [ 0.27779754],
             [-0.17238948],
             ...,
             [-0.61270518],
             [ 0.90054847],
             [-0.319791  ]])
```

```
[258] lda_deaths
```

```
      array([[-0.00540373],
             [ 0.61397097],
             [-0.38198988],
             ...,
             [-0.88514553],
             [ 0.68705052],
             [-1.10775743]])
```



```
[259] lda_recovered
```

```
      array([[ 0.29441841],
             [-0.91552975],
             [ 0.39450745],
             ...,
             [ 0.31641987],
             [ 1.32226117],
             [ 1.10623856]])
```

Q]
Does the LDA method provide better results for one label more than the others?

Answer]

Linear Discriminant Analysis (LDA) is a supervised approach for identifying the linear discriminants that reflect the axes that optimise separation between various classes. Each class is given a Gaussian density by the model, which assumes that all classes have the same covariance matrix.

Since the labels in the dataset are unbalanced(quite a few more True cases than there are False cases) Recovered is the most balanced, so it should be easier to train and hence we get better results with LDA here. Deaths is next and Confirmed is the least balanced