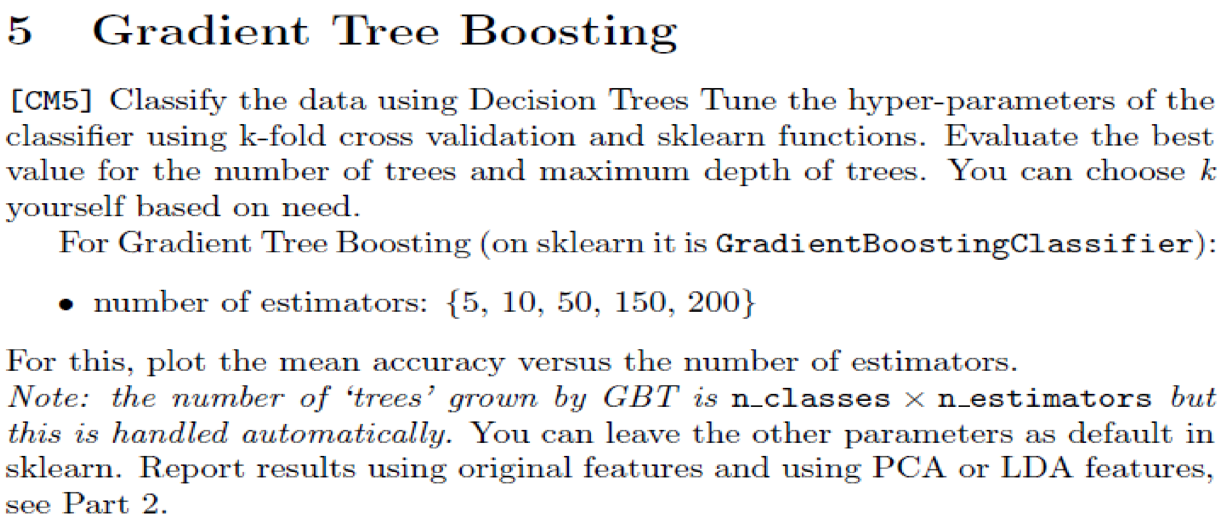
Question 5

[CM5]Gradient Tree Boosting



5 Gradient Tree Boosting

[CM5] Classify the data using Decision Trees Tune the hyper-parameters of the classifier using k-fold cross validation and sklearn functions. Evaluate the best value for the number of trees and maximum depth of trees. You can choose k yourself based on need.

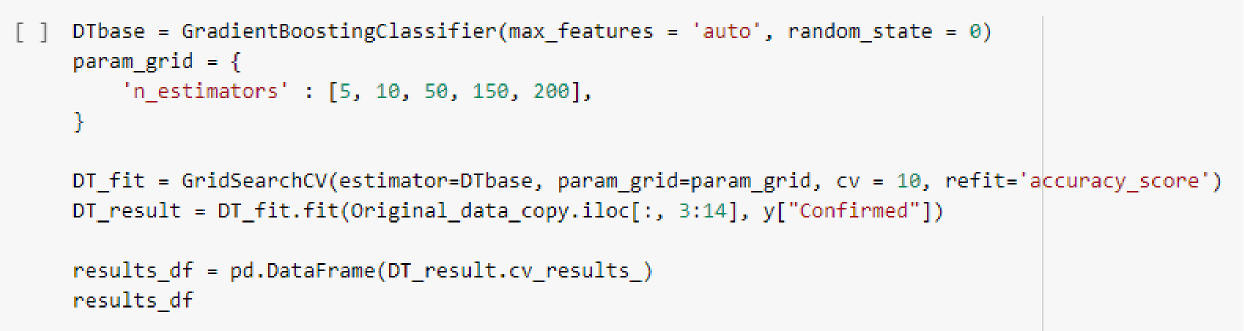
For Gradient Tree Boosting (on sklearn it is `GradientBoostingClassifier`):

- number of estimators: {5, 10, 50, 150, 200}

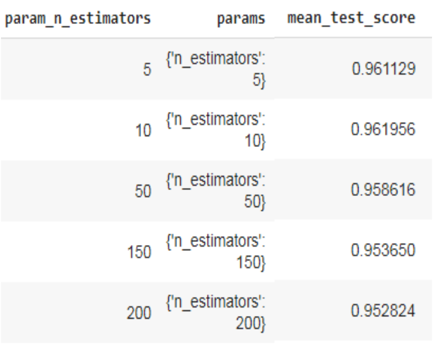
For this, plot the mean accuracy versus the number of estimators.
Note: the number of 'trees' grown by GBT is `n_classes` × `n_estimators` *but this is handled automatically.* You can leave the other parameters as default in sklearn. Report results using original features and using PCA or LDA features, see Part 2.

1. Hyper parameter tuning is performed using 10-fold cross validation on each label to evaluate the best value for number of estimators

Original Features:



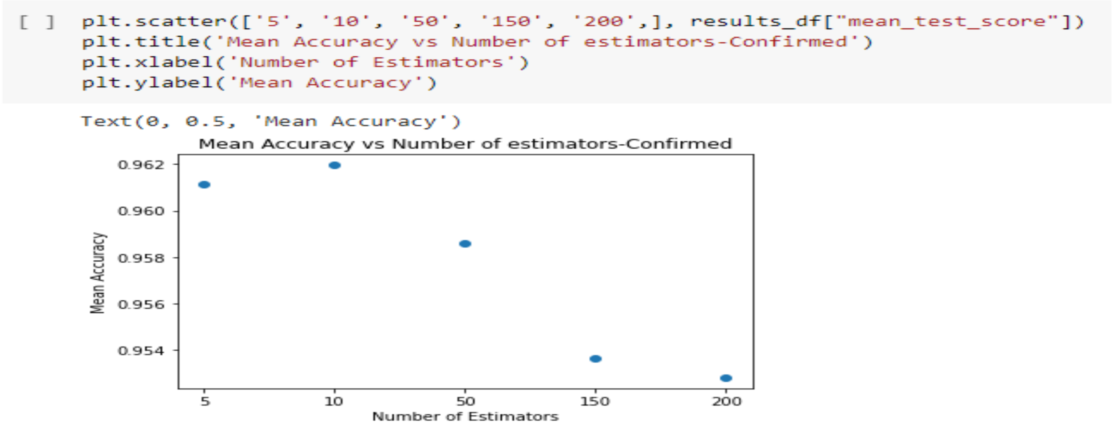
```
[ ] DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
    param_grid = {
        'n_estimators' : [5, 10, 50, 150, 200],
    }

    DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
    DT_result = DT_fit.fit(Original_data_copy.iloc[:, 3:14], y["Confirmed"])

    results_df = pd.DataFrame(DT_result.cv_results_)
    results_df
```



param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.961129
10	{'n_estimators': 10}	0.961956
50	{'n_estimators': 50}	0.958616
150	{'n_estimators': 150}	0.953650
200	{'n_estimators': 200}	0.952824

Plot of Mean Accuracy vs Number of Estimators :

Label-Confirmed

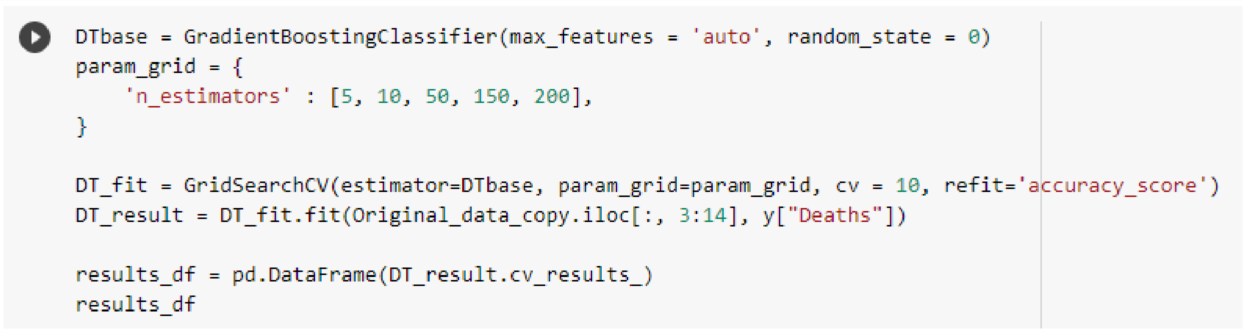


```
[ ]  plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
     plt.title('Mean Accuracy vs Number of estimators-Confirmed')
     plt.xlabel('Number of Estimators')
     plt.ylabel('Mean Accuracy')
```

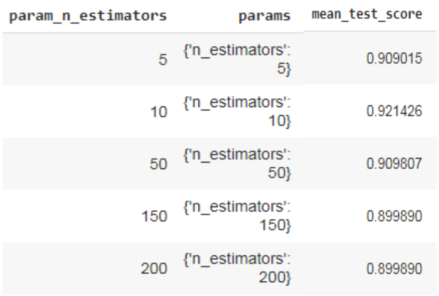
```
Text(0, 0.5, 'Mean Accuracy')
```

Best value for number of estimators=10 with accuracy=96.2%

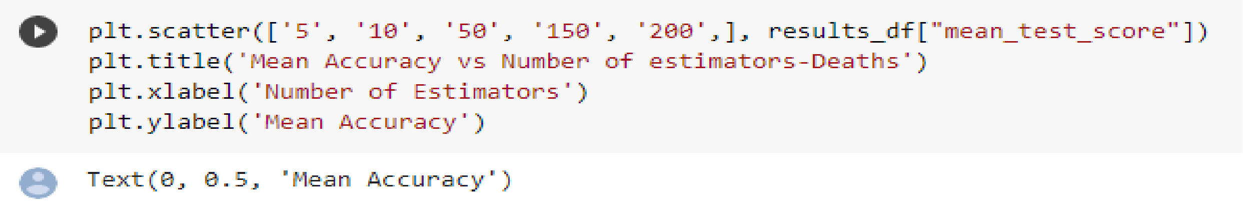
Label-Deaths



```
DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
param_grid = {
    'n_estimators' : [5, 10, 50, 150, 200],
}

DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
DT_result = DT_fit.fit(Original_data_copy.iloc[:, 3:14], y["Deaths"])

results_df = pd.DataFrame(DT_result.cv_results_)
results_df
```

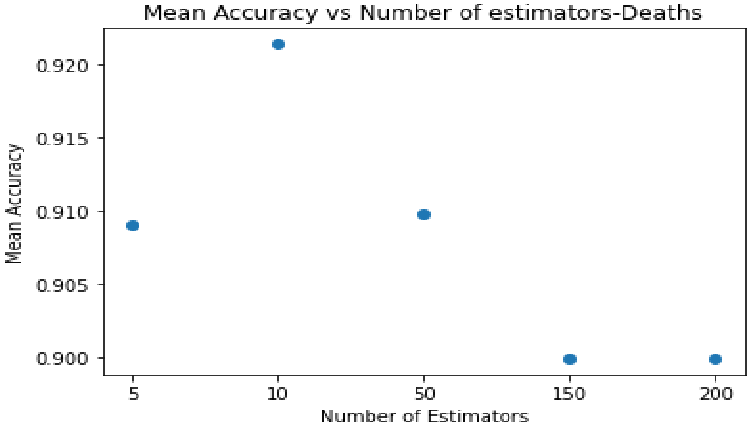


param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.909015
10	{'n_estimators': 10}	0.921426
50	{'n_estimators': 50}	0.909807
150	{'n_estimators': 150}	0.899890
200	{'n_estimators': 200}	0.899890



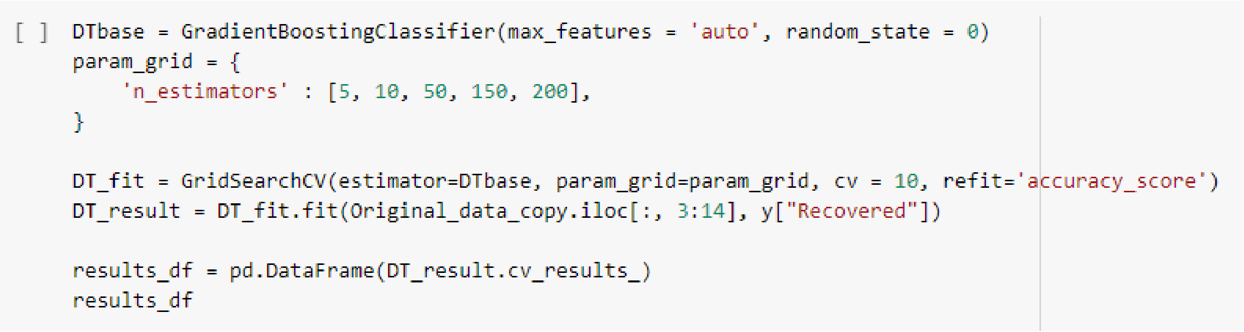
```
plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
plt.title('Mean Accuracy vs Number of estimators-Deaths')
plt.xlabel('Number of Estimators')
plt.ylabel('Mean Accuracy')
```

```
Text(0, 0.5, 'Mean Accuracy')
```

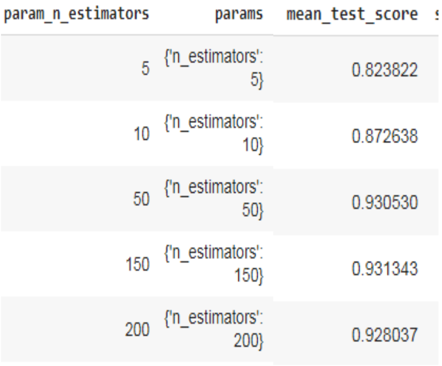


Best value for number of estimators=10 with accuracy=92.14%

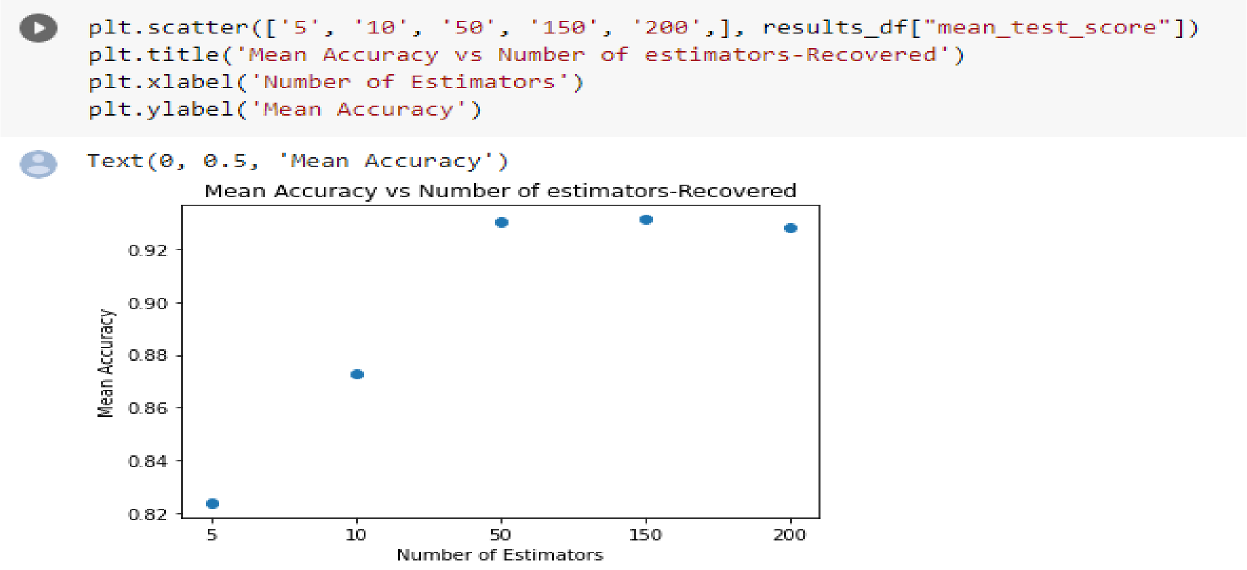
Label -Recovered



```
[ ] DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
    param_grid = {
        'n_estimators' : [5, 10, 50, 150, 200],
    }

    DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
    DT_result = DT_fit.fit(Original_data_copy.iloc[:, 3:14], y["Recovered"])

    results_df = pd.DataFrame(DT_result.cv_results_)
    results_df
```



param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.823822
10	{'n_estimators': 10}	0.872638
50	{'n_estimators': 50}	0.930530
150	{'n_estimators': 150}	0.931343
200	{'n_estimators': 200}	0.928037



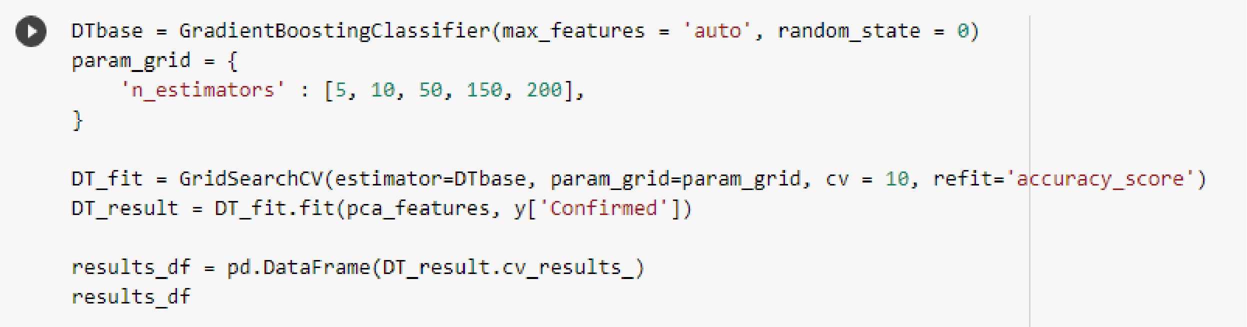
```
plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
plt.title('Mean Accuracy vs Number of estimators-Recovered')
plt.xlabel('Number of Estimators')
plt.ylabel('Mean Accuracy')
```

```
Text(0, 0.5, 'Mean Accuracy')
```

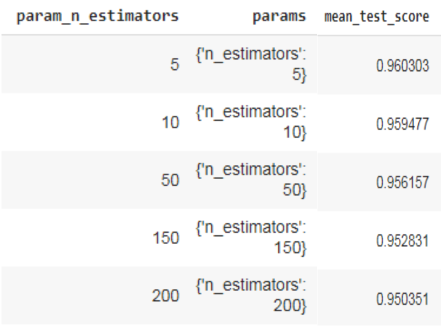
Best value for number of estimators=150 with accuracy=93.13%

PCA:

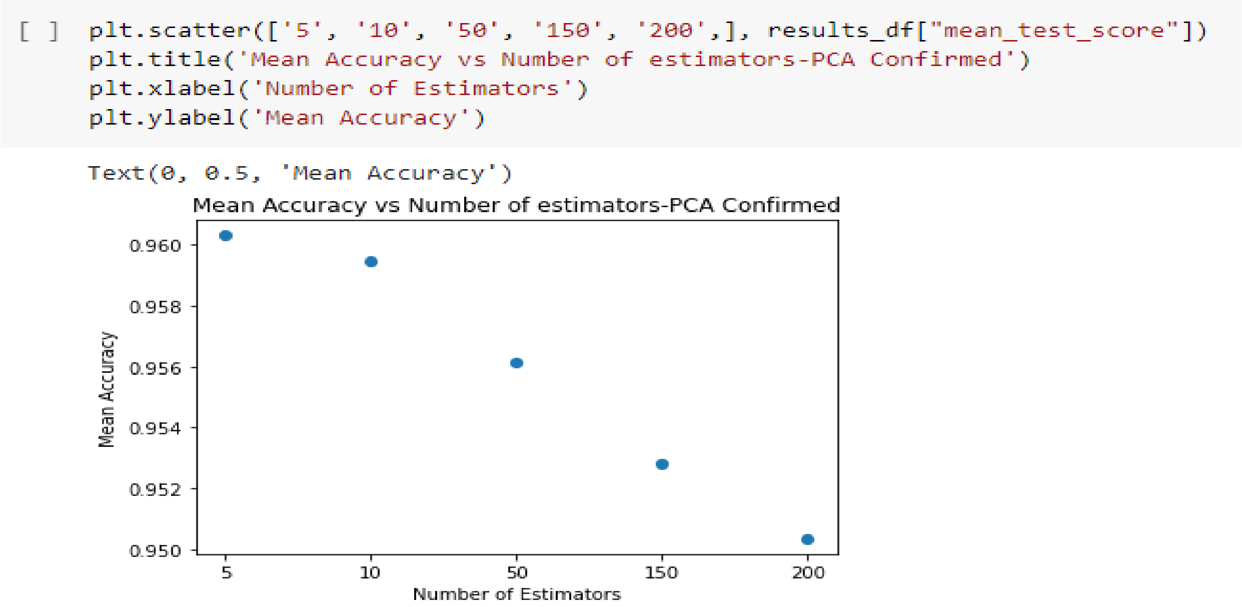
Label-Confirmed



```
DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
param_grid = {
    'n_estimators' : [5, 10, 50, 150, 200],
}

DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
DT_result = DT_fit.fit(pca_features, y['Confirmed'])

results_df = pd.DataFrame(DT_result.cv_results_)
results_df
```



param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.960303
10	{'n_estimators': 10}	0.959477
50	{'n_estimators': 50}	0.956157
150	{'n_estimators': 150}	0.952831
200	{'n_estimators': 200}	0.950351

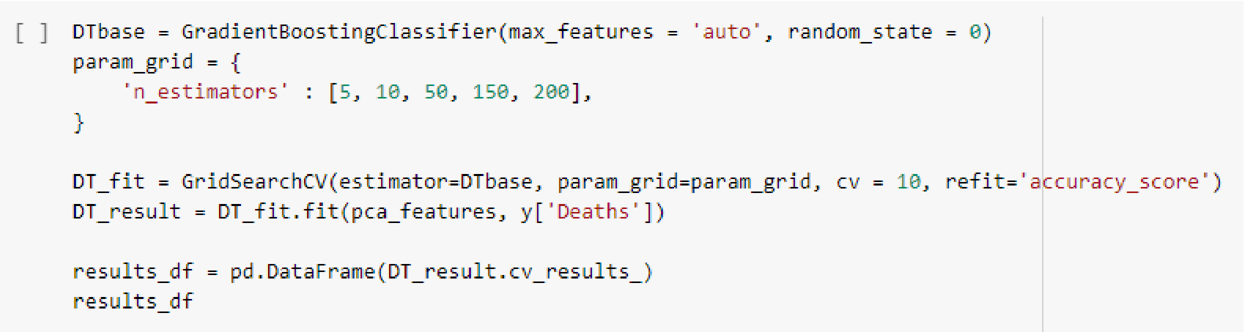


```
[ ]  plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
     plt.title('Mean Accuracy vs Number of estimators-PCA Confirmed')
     plt.xlabel('Number of Estimators')
     plt.ylabel('Mean Accuracy')
```

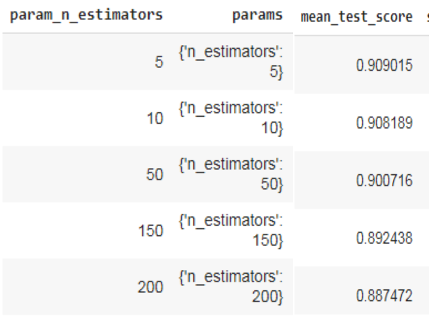
```
Text(0, 0.5, 'Mean Accuracy')
```

Best value for number of estimators=5 with accuracy=96.03%

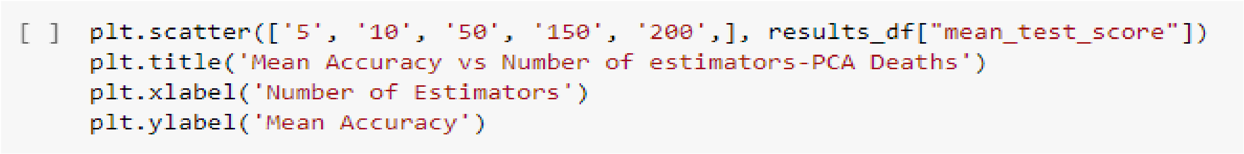
Label-Deaths



```
[ ]  DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
     param_grid = {
         'n_estimators' : [5, 10, 50, 150, 200],
     }

     DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
     DT_result = DT_fit.fit(pca_features, y['Deaths'])

     results_df = pd.DataFrame(DT_result.cv_results_)
     results_df
```

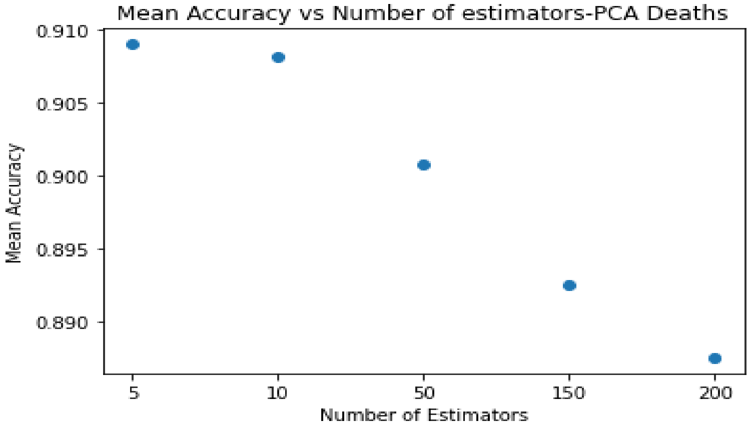


param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.909015
10	{'n_estimators': 10}	0.908189
50	{'n_estimators': 50}	0.900716
150	{'n_estimators': 150}	0.892438
200	{'n_estimators': 200}	0.887472



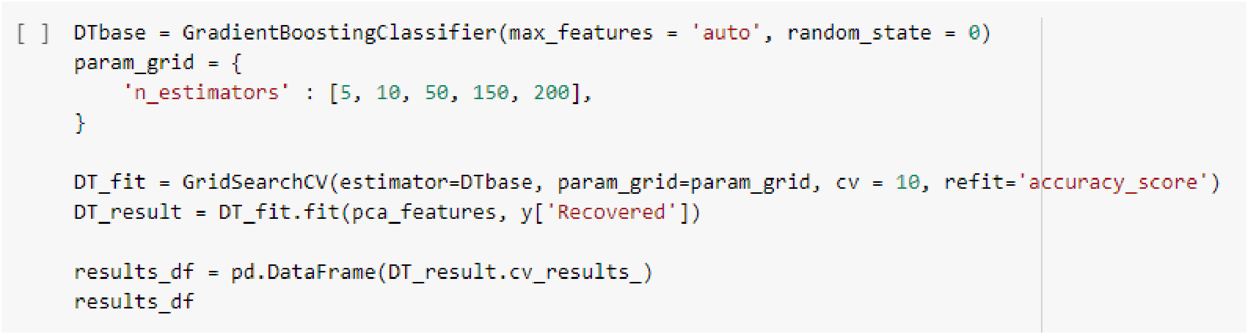
```
[ ]  plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
     plt.title('Mean Accuracy vs Number of estimators-PCA Deaths')
     plt.xlabel('Number of Estimators')
     plt.ylabel('Mean Accuracy')
```

```
Text(0, 0.5, 'Mean Accuracy')
```

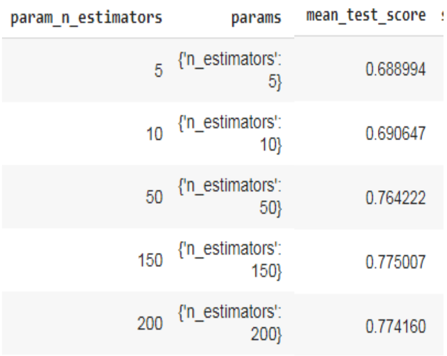


Best value for number of estimators=5 with accuracy=90.90%

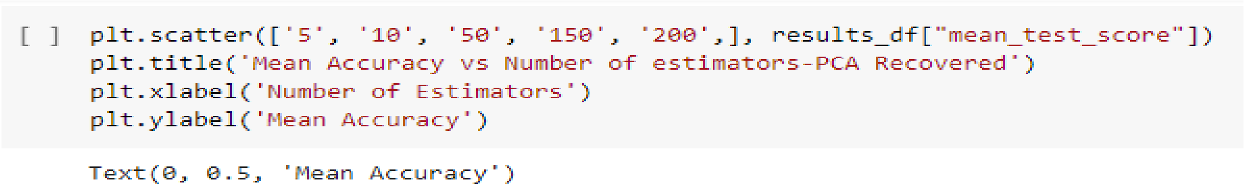
Label-Recovered



```
[ ]  DTbase = GradientBoostingClassifier(max_features = 'auto', random_state = 0)
     param_grid = {
         'n_estimators' : [5, 10, 50, 150, 200],
     }

     DT_fit = GridSearchCV(estimator=DTbase, param_grid=param_grid, cv = 10, refit='accuracy_score')
     DT_result = DT_fit.fit(pca_features, y['Recovered'])

     results_df = pd.DataFrame(DT_result.cv_results_)
     results_df
```

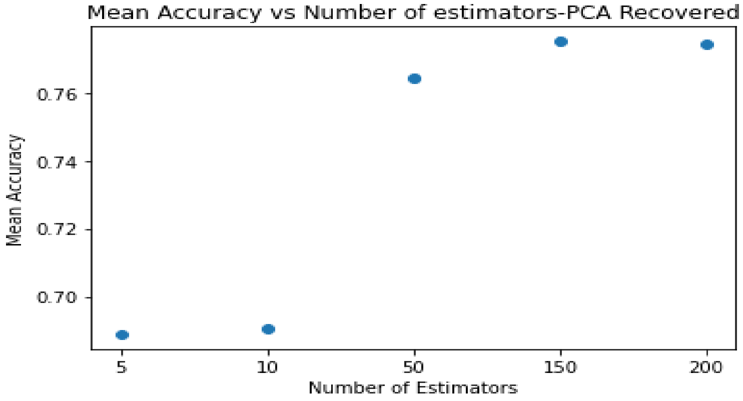


param_n_estimators	params	mean_test_score
5	{'n_estimators': 5}	0.688994
10	{'n_estimators': 10}	0.690647
50	{'n_estimators': 50}	0.764222
150	{'n_estimators': 150}	0.775007
200	{'n_estimators': 200}	0.774160



```
[ ]  plt.scatter(['5', '10', '50', '150', '200',], results_df["mean_test_score"])
     plt.title('Mean Accuracy vs Number of estimators-PCA Recovered')
     plt.xlabel('Number of Estimators')
     plt.ylabel('Mean Accuracy')
```

```
Text(0, 0.5, 'Mean Accuracy')
```



Best value for number of estimators=150 with accuracy=77.50%

Observations:

Original Features:

Label	Number of Estimators	Accuracy
Confirmed	10	96.2%
Deaths	10	92.14%
Recovered	150	93.13%

PCA:

Label	Number of Estimators	Accuracy
Confirmed	5	96.03%
Deaths	5	90.90%
Recovered	150	77.50%