Question 3

[CM3]Decision Tree Classifier

1. Original Feature Space is used for this classification
2. Hyper parameter tuning is performed using 10-fold cross validation on each label to evaluate the best value for Max Depth

```
[262] DTbase = DecisionTreeClassifier(max_features = 'auto', random_state = 0)

      param_grid = {
          'max_depth':[3, 5, 10, None],
      }

      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)

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

Results of Hyper parameter tuning,

param_max_depth	mean_test_score
3	0.742741
5	0.789890
10	0.827066
None	0.784876

Cross-validation is used to estimate a model's predicted level of fit to a data set that is unrelated to the data used to train the model. Any quantitative measure of fit that is appropriate for the data and model can be estimated using it. Here we had used cross validation as 10.

Below we had created a table where we get some different accuracies on different cross validation:

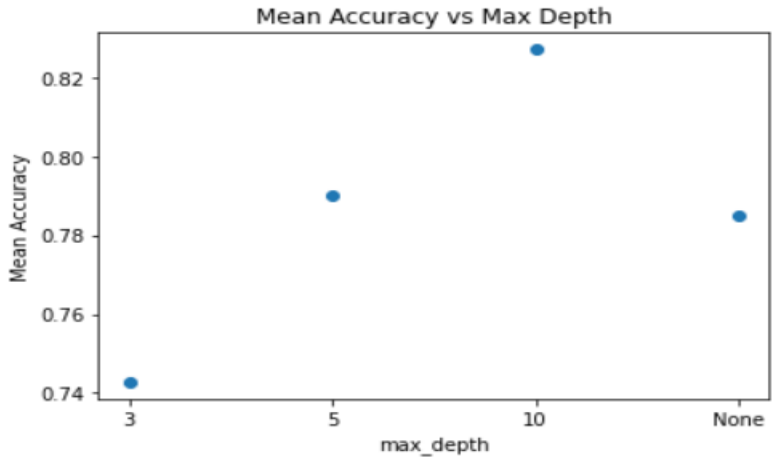
Value of CV	Max_Depth	Accuracy (%)
10	10	82.70
15	10	81.73
20	10	81.01

As the value of CV increase the accuracy decreases.

Plot of Mean Accuracy vs Maximum Depth:

```
plt.scatter(['3', '5', '10', 'None'], results_df["mean_test_score"])
plt.title('Mean Accuracy vs Max Depth')
plt.xlabel('max_depth')
plt.ylabel('Mean Accuracy')
```

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



Therefore, Max Depth value of 10 gives the best accuracy of 82.70%.

When increasing the maximum depth after the value 15, the accuracy slightly changes which is same as the value of accuracy for None.

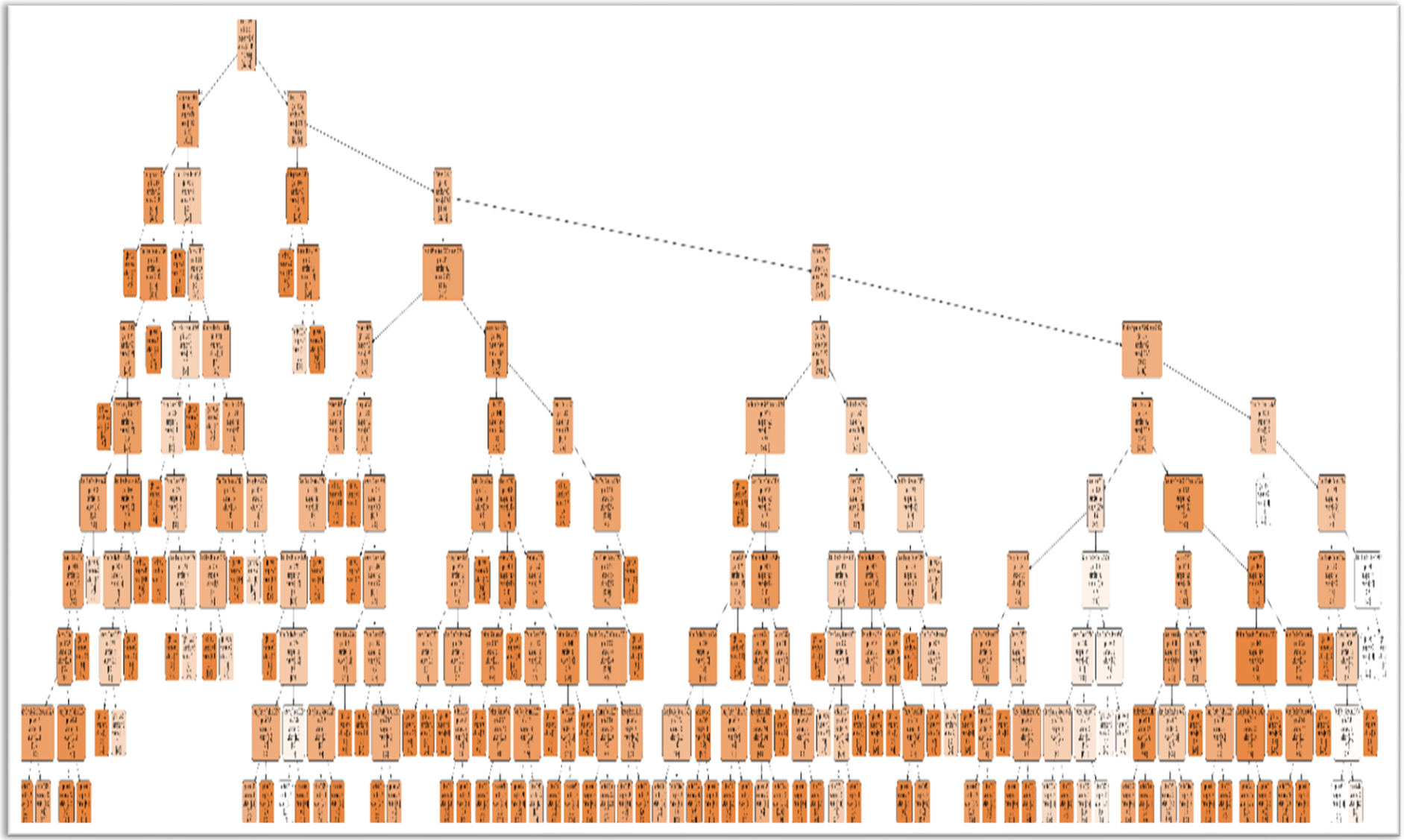
Q] Also, examine the final resulting splitting rules used for the trees. Do they indicate any interesting patterns that explain the data?

```
[ ]  clf = DT_result.best_estimator_
```

```
[ ]  pip install graphviz
```

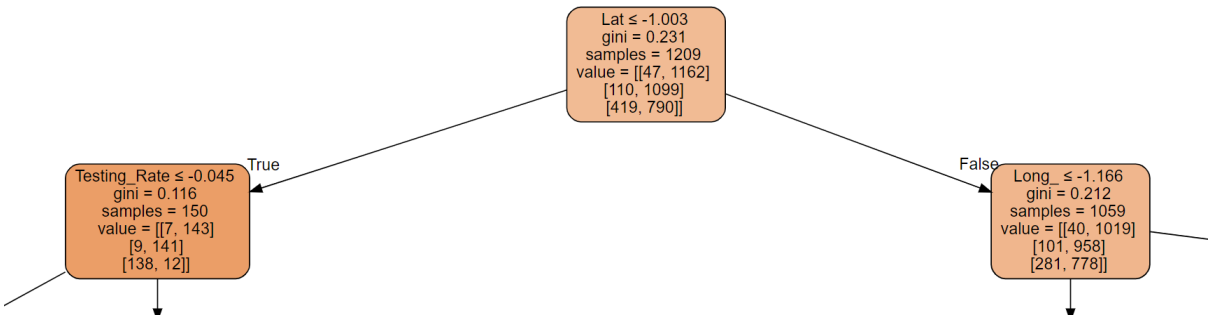
```
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)
```

```
import graphviz

dot_data = tree.export_graphviz(clf, out_file="tree_dot.dot",
                      feature_names=Original_data_copy.iloc[:, 3:14].columns,
                      class_names=y["Confirmed"].unique(),
                      filled=True, rounded=True,
                      special_characters=True)

import pydot
from IPython.display import Image

(graph,) = pydot.graph_from_dot_file("tree_dot.dot")
graph.write_png('dt.png')
Image(filename='dt.png')
```

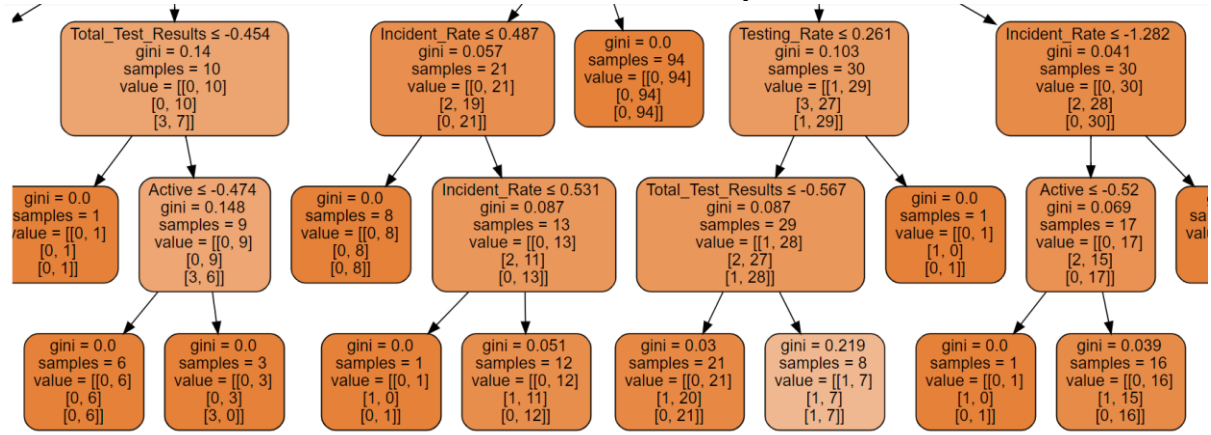


For better clarity:

Initial Split



Portion of Final Split



We can observe from the splitting pattern that, Gini Impurity is preferred to Information Gain because it does not contain logarithms which are computationally intensive.

For each split, the Gini Impurity of each child node is individually calculated. The split with the lowest value of Gini Impurity is selected.

We can observe that the root node was first split at Lat<=-1.003 and gini impurity value of 0.231.Next split is at Testing_Rate <= -0.045 on the left and Long <= -1.166 on the right which is further split into Testing_Rate and Active.

The splitting is unbalanced with more splitting and branching on the right side of root node than the left

This splitting continues similarly until each of the resulting child nodes end up with 2 leaf nodes or 1 leaf node.

In the final splitting, the pattern that can be observed is that the final child node splits are either on Active, Case_Fatality_Ratio, Incident_Rate, Testing_Rate and Total_Testing_Result on every branch. An interesting observation is that these 5 features were the subset of features selected to perform PCA.