Lab # 6: Decision Tree Regression and Random Forest

6.1 Objectives

36. Understand the Decision Tree Regression and Random Forest
37. Apply the Decision Tree Regression and Random Forest through python.

6.2 Pre-Lab

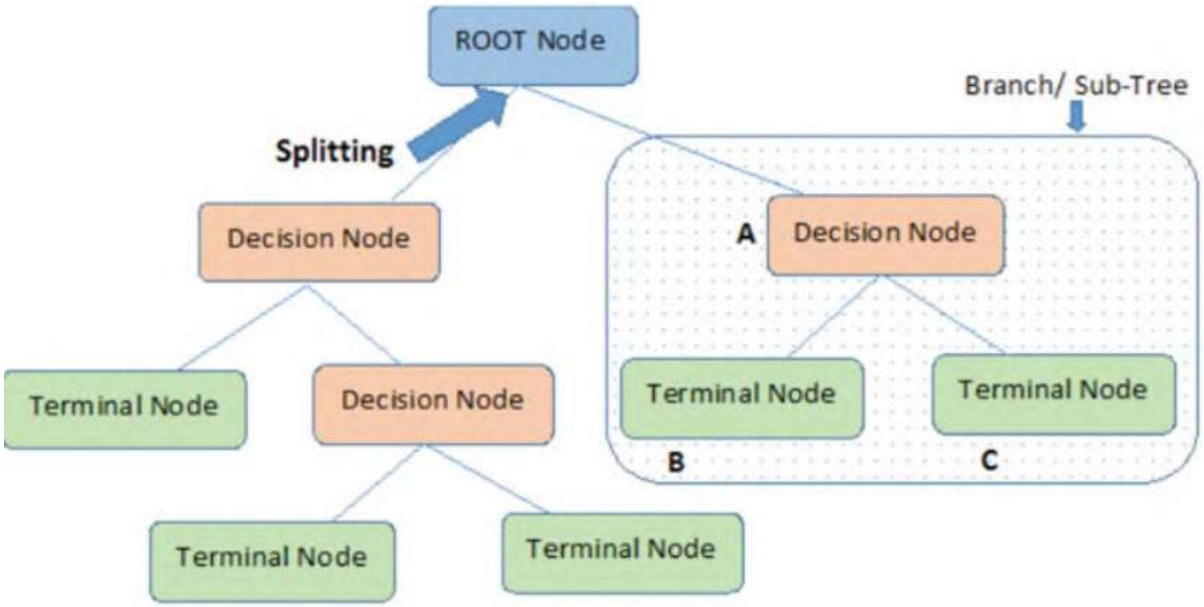
6.2.1 Decision Tree Regression

A decision tree is arriving at an estimate by asking a series of questions to the data,each question narrowing our possible values until the model gets confident enoughto make a single prediction. The order of the question and their content are beingdetermined by the model. In addition, the questions asked are all in a True/Falseform.

This is a little tough to grasp because it is not how humans naturally think, andperhaps the best way to show this difference is to create a real decision tree from. Inthe above problem x1, x2 are two features that allow us to make predictions for thetarget variable y by asking True/False questions.

The decision of making strategic splits heavily affects a tree's accuracy. Thedecision criteria are different for classification and regression trees. Decision treesregression normally use mean squared error (MSE) to decide to split a node intotwo or more sub-nodes. Suppose we are doing a binary tree; the algorithm will firstpick a value and split the data into two subsets. For each subset, it will calculate theMSE separately. The tree chooses the value with results in smallest MSE value.

Let us examine how is Splitting Decided for Decision Trees Regressor in moredetail. The first step to create a tree is to create the first binary decision.



1. We need to pick a variable and the value to split on such that the two groups are as different from each other as possible.
2. For each variable, for each possible value of the possible value of that variable see whether it is better.
3. Take weighted average of two new nodes (mse*num_samples).

To sum up, we now have:

38. A single number that represents how good a split is, which is the weighted average of the mean squared errors of the two groups that create.
39. A way to find the best split, which is to try every variable and to try every possible value of that variable and see which variable and which value gives us a split with the best score.
40.

Training of a decision tree regressor will stop when some stopping condition ismet:

1. When you hit a limit that was requested (for example: `max_depth`).
2. When your leaf nodes only have one thing in them (no further split is possible, MSE for the train will be zero but will overfit for any other set—not a useful model).

6.2.2 Random Forest

What is a Random Forest? And how does it differ from a Decision Tree?The fundamental concept behind random forest is a simple but powerful one—the wisdom of crowds. In data science speak, the reason that the random forestmodel works so well is: A large number of relatively uncorrelated models (trees)operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments withlow correlations (like stocks and bonds) come together to form a portfolio thatis greater than the sum of its parts, uncorrelated models can produce ensemblepredictions that are more accurate than any of the individual predictions. The reasonfor this wonderful effect is that the trees protect each other from their individualerrors (as long as they do not constantly all err in the same direction). While sometrees may be wrong, many other trees will be right, so as a group the trees are ableto move in the correct direction. So the prerequisites for random forest to performwell are:

1. There needs to be some actual signals in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

So how does random forest ensure that the behavior of each individual tree is nottoo correlated with the behavior of any of the other trees in the model? It uses thefollowing two methods:

Bagging (Bootstrap Aggregation)—Decision trees are very sensitive to the datathey are trained on—small changes to the training set can result in significantlydifferent tree structures. Random forest takes advantage of this by allowing eachindividual tree to randomly sample from the dataset with replacement, resultingin different trees. This process is known as bagging.

Feature Randomness—In a normal decision tree, when it is time to split a node,we consider everypossible feature and pick the one that produces the mostseparation between the observations in the left node vs. those in the right node.In contrast, each tree in a random forest can pick only from a random subsetof features. This forces even more variation among the trees in the model andultimatelyresults in lower correlation across trees and more diversification.

As Random Forest is actually a collection of Decision Trees, this makes thealgorithm slower and less effective for real-time predictions. In general, RandomForest can be fast to train, but quite slow tocreate predictions once they are trained.

This is due to the fact that it has to run predictions on each individual tree and thenaverage their predictions to create the final prediction.Each individual tree in the random forest splits out a class prediction and the classwith the most votes become our model's prediction. Decision Trees do suffer fromoverfitting while Random Forest can prevent overfitting resulting in better prediction most of the time.

6.3 In-Lab

In-Lab Task 1

Write the following decision tree implementation in python.

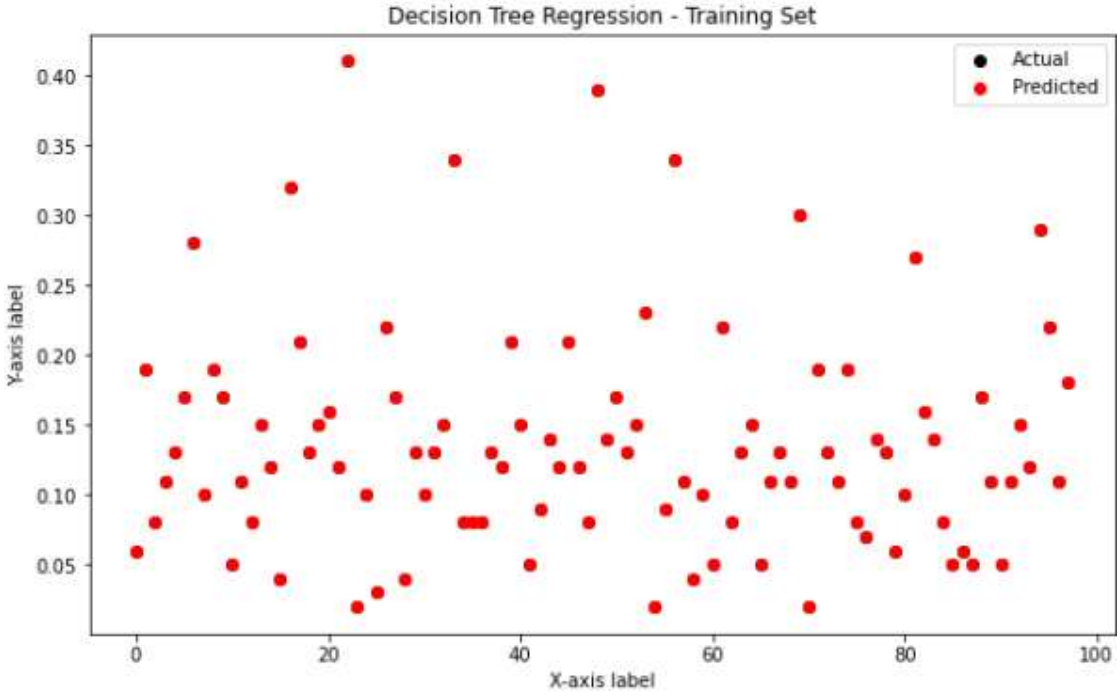
```
# ================================================================================
# Model 2 decision tree
# ================================================================================

model2 = tree.DecisionTreeRegressor()
model2.fit(X_train, y_train)
print("Decision Tree")
print("==============================")
y_pred_train2 = model2.predict(X_train)
RMSE_train2 = mean_squared_error(y_train,y_pred_train2)
print("Decision Tree Train set: RMSE {}".format(RMSE_train2))

y_pred_test2 = model2.predict(X_test)
RMSE_test2 = mean_squared_error(y_test,y_pred_test2)
print("Decision Tree Test set: RMSE {}".format(RMSE_test2))
print("==============================")
```

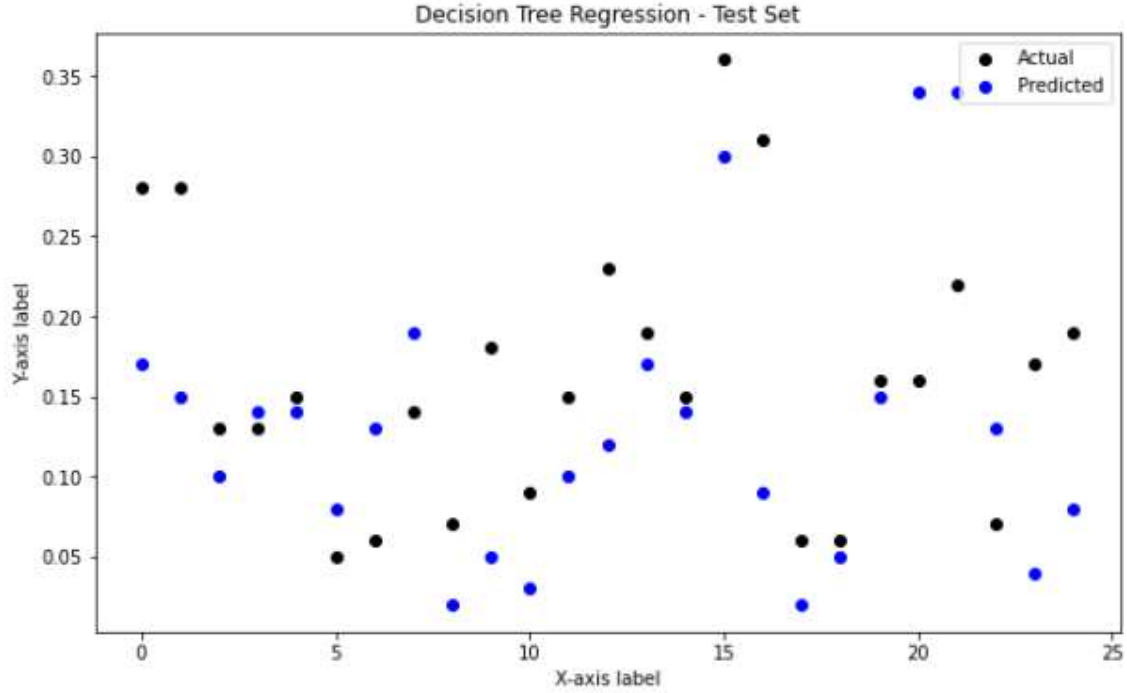
In-Lab Task 2

Display the results of Decision Tree Regression for the Training set.



In-Lab Task 3

Display the results of Decision Tree Regression for the Test set.



In-Lab Task 4

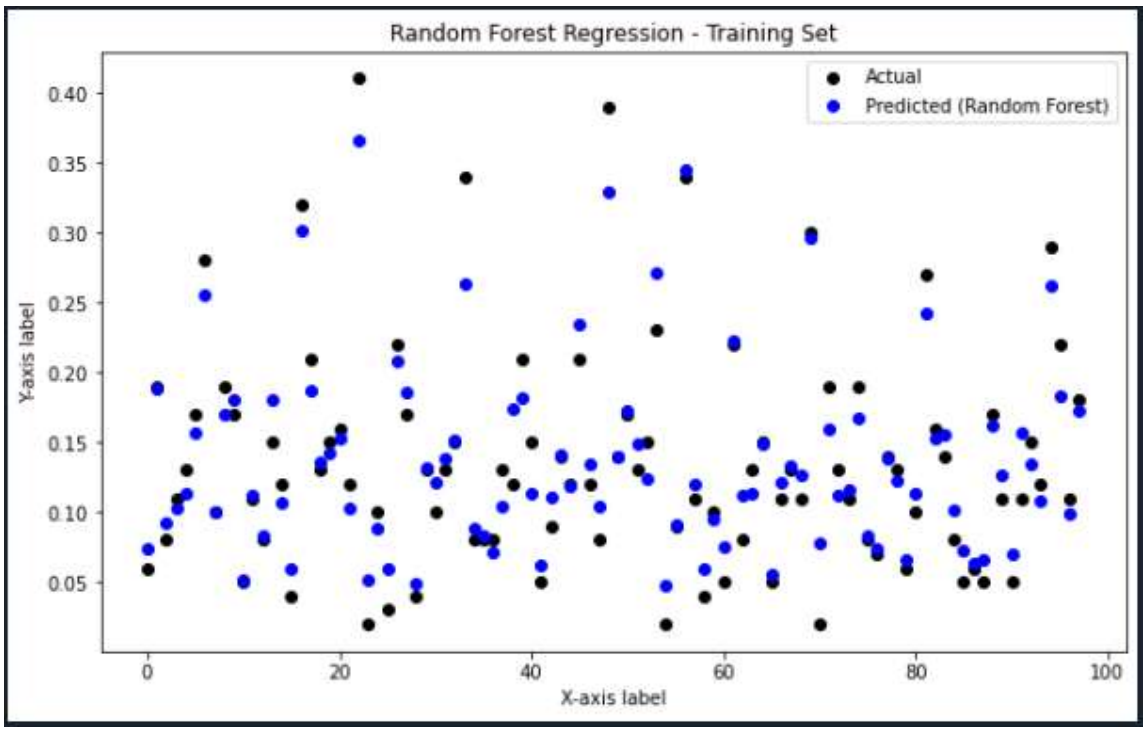
Implement the Random Forest using Python. Understand and discuss the results.

```
# ================================================================================
# Model 3 Random Forest
# ================================================================================

model3 = RandomForestRegressor()
model3.fit(X_train, y_train)
print("Random Forest Regressor")
print("================================")
y_pred_train3 = model3.predict(X_train)
RMSE_train3 = mean_squared_error(y_train,y_pred_train3)
print("Random Forest Regressor TrainSet: RMSE {}".format(RMSE_train3))
print("================================")
y_pred_test3 = model3.predict(X_test)
RMSE_test3 = mean_squared_error(y_test,y_pred_test3)
print("Random Forest Regressor TestSet: RMSE {}".format(RMSE_test3))
print("================================")
```

In-Lab Task 5

Display the result of the model and discuss the results



6.4 Post Lab

Post-Lab Task

Compare the performance of the three models. Which model performs better on the test set, and why do you think it outperforms the others?

Discuss on the strengths and limitations of each model. Discuss any insights gained from the visualizations.

Suggest potential strategies to improve the models' performance. Could feature engineering or hyperparameter tuning enhance the predictions?