Practice: Creating, Scoring, and Assessing Tree-Based Models Using the Python API

1. Use the sampling action set and the srs action to sample 70% of the **bank** data set. Use the partind=TRUE argument to save the partition indicator to the table. Using CASTable in Python, refresh the CAS table object reference, and then use the mean function from the SWAT package to find the proportion of the training cases.

2. Load the decisionTree action set and then train a decision tree using the dtreeTrain action. Save the model to score the validation data later.

3. Train a random forest with 1000 trees using the forestTrain action and be sure to save the model.

4. Train a gradient boosting tree with the gbtreeTrain action and save the model.

5. From the decisionTree action set, use the dtreeScore, forestScore, and gbtreeScore actions to score the validation data using the previously saved models.

6. Load the percentile action set and use the assess action to assess each model. Recall that the input for the assess action has the name with a prefix P_ followed by the target name and modeling level (**P_b_tgt1**). Save the assess results for each model.

7. Download the ROC results for each model by first creating an object reference using CASTable and then bring results to the client with to_frame. For each table, add the model name into the table as a new variable.

8. Combine the local data frame results by row binding them. Then print the model and confusion matrix results for each model at a cutoff of 0.50.

9. Print the misclassification rates for each model by using the **_ACC_** variable in the data frames (1-_ACC_).

10. Plot the ROC curves for each model in one graphic.