Rashomon Code Flow

Simon Dovan Nguyen

April 18, 2025

All code can be found on Github. The links below lead to the function that performs each action.

1 One Iteration Function

1. Set Up:
   (a) Set seed
   (b) Load Dataset
   (c) Train Test Candidate Split Data
      - `df_Test`: 20%
      - `df_Candidate`: 80%
      - Initial `df_Train`: Remaining
   (d) (Batch): Calculate distance/diversity metric

$$d_n^x = \min_m ||x_n - x_m|| \tag{1}$$

   This is the distance from an unlabeled observation n to its nearest labeled neighbor.
2. Learning Procedure
   `For i = 0 to len(df_Train):`
   (a) Prediction Method:
      i. Train TreeFarms model on `df_Train`
         - Results in K number of trees
      ii. Predict label for `df_Test` for all K tree in TreeFarms
      iii. Calculate F1 score (`micro` average) from each tree in TreeFarms model for the `df_Test`
         - Note that the F1 score is over all K trees, including the duplicate trees, even if we are only using the set of unique trees for the selection process (Line 51).
         - Note that my TestErrorFunction still stores both F1 scores: (1) using the duplicate trees and (2) using only the unique trees. (Line 31-55).
   (b) Selection Query
      i. (UNREAL) Restrict ourselves to only the unique trees (`Lines 53 - 56`).
      ii. Predict the labels of `df_Candidate` for each (unique) tree in TreeFarms (`Lines 40 - 51`).
      iii. Calculate recommendation metric (Vote Entropy) for each observation in the candidate set based off of the (unique/duplicate) trees (`Lines 74 - 83`):

$$\text{VoteEntropy}(y, x) = -\sum_{y \in \{0,1\}} \frac{\text{vote}_{\mathcal{C}}(y, x)}{|\mathcal{C}|} \log \frac{\text{vote}_{\mathcal{C}}(y, x)}{|\mathcal{C}|} \tag{2}$$

      where

$$\text{vote}_{\mathcal{C}} = \sum_{c \in \mathcal{C}} \mathbb{I}\{c(x) = y\} \tag{3}$$

      is the number of "votes" that label y receives for x amongst the models in the Rashomon set of trees $\mathcal{C}$.
      iv. Weight VoteEntropy (2) with Diversity Metric 1 with Diversity weight w. (Line 86):

$$\texttt{UncertaintyMetric}(y, x; w) = (1 - w) \cdot \texttt{VoteEntropy} + w \cdot \texttt{Diversity} \tag{4}$$

      v. Recommend the top k candidate observations with the highest vote entropy (`Lines 77 - 81`):

$$\arg\max_x \texttt{UncertaintyMetric}$$

   (c) Update (Lines 67-69)
      i. Add that observation `df_Training`
      ii. Remove that observation from `df_Candidate`
   (d) Repeat Steps (a) - (c)