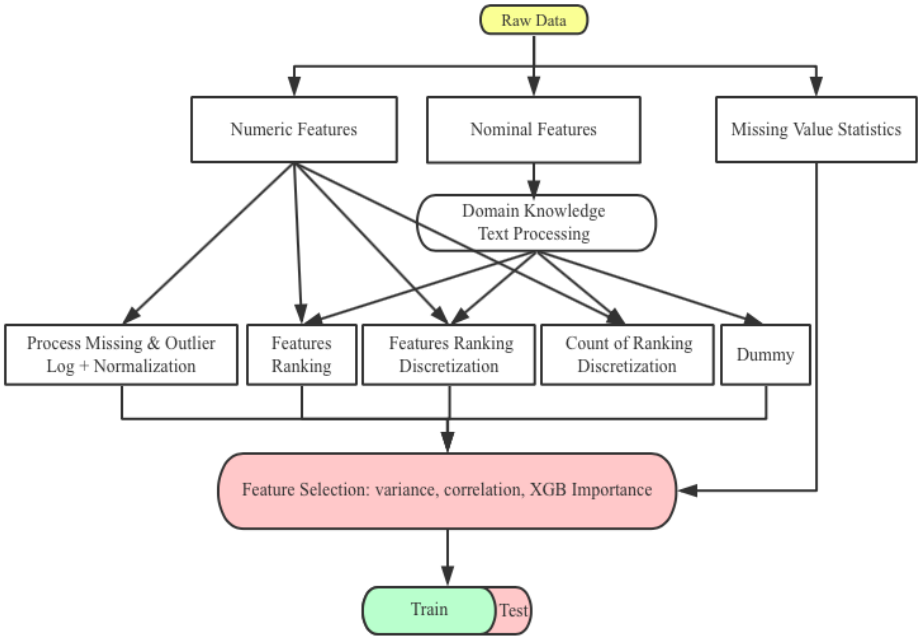
This is my strategy for the Transfer Learning Challenge of Credit Risk Prediction of Qianhai[1]. This document introduces the order to execute the given files and a brief explanation of the objective of each file is provided. To note, modeling is conducted in Python 3.

The chart provides an overview of data preparation,which is the task of the first four files.



1.Qianhai_solution.py

Statistical features.

2.Configure.py

Configure sets up the development environment for the following files.

3.Preprocessing.py

In the raw data, numerical variables are log transformed and normalized, then dummies of nominal variables are generated.

4. Statistics_Features.py

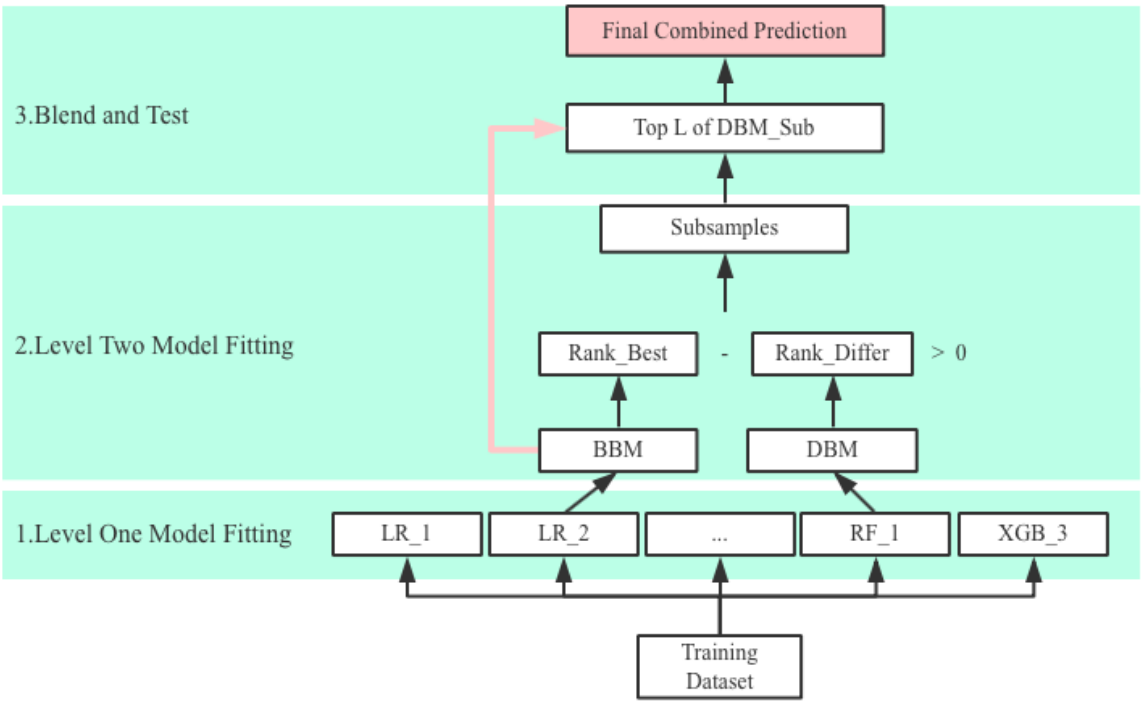
Statistics_Features is a streamlined version for Qianhai_solution.py, it could be skipped if Qianhai_solution is successfully executed. In this way, the processed data is made of two

[1]https://www.kesci.com/apps/home/#!/lab/dataset/58e4663a9ed26b1e09bfbaaf/document

parts: statistical features from Qianhai_solution.py and processed numeric and nominal features from Preprocessing.py.

5.Feature_Selection

Ahead of modeling, processed data is selected in this section. Features that have low variance or high correlation with other features are excluded.



The following files are in charge of modeling. As can be seen from the diagram above, modeling consists of three steps: firstly, a couple of base models from LR to XGB are used to fit the training data. Afterwards in step two, samples that are suspicious to be misclassified by the single best base model are fitted by a linear model. Eventually in the last step the results from BBM and DBM_Sub are blended. Outputs of this model will be compared with the results of a normal stacking model, the one has the highest AUC in hold-out will be used as final prediction to submit.

5.run.py - level_one_wrapper

Thanks to the prepared data via the first five steps, method of level_one_wrapper, which is Level One Model Fitting in the chart above, is to fit the data via a bunch of algorithms.

7.BBM_DBM.py

BBM_DBM reads the results of level_one_wrapper and tells which algorithm is the single best base model (BBM) based on the average AUC via 5-fold CV. Additionally it indicates the base model differs from BBM to the largest scale (DBM) according to the PCC values among all base models. Samples have high positive values in the difference between Ranking_BBM and Ranking_DBM (Ranking_BBM-Ranking_DBM) are highly supposed to be misclassified by BBM (normally XGBoost) but more accurately evaluated by DBM (in general a linear model).

8.load_train_data.py

load_train_data plots the ranking difference of instances between BBM and DBM so as to determine the subsamples for level two model fitting.

9.run.py - level_two_wrapper

On the selected samples from the last step, the model DBM_Sub is fitted, which is linear model in Level Two Model Fitting. This model gives more accurate evaluations on the chosen data than BBM.

10.run.py - level_one_predict

The models fitted in level_one_wrapper are used to fit the hold-out test.

11.run.py - level_two_predict

The outputs of level_one_predict are used as inputs to fit a normal stacking model.

12.local_predit_verrify_tune_final.py

Tune optimal values of three parameters: L, intervales, the lowest ranking difference for Strategy One and Strategy Two of parameter tuning are verified respectively, based on the observed data for the unseen data.

13.local_predit_verrify.py

The optimal values of parameters from the last step are tested in the hold-out data set in this step. Comparing BBM_Holdout, Stacking_Holdout and the model depicted above, the model of the three has the highest hold-out AUC will be used as final prediction to submit.