Home Credit Default Risk Analysis

Tianyu Du

Oct. 2018

Last update November 3, 2018

Contents

1 Light Gradient Boosting Machine **2**
1.1 Data Preprocessing 2
1.1.1 Dropping Invalid Data 2
1.1.2 Encoding Categorical Features 2
1.1.3 Splitting Training, Testing and Validation Sets 3
1.1.4 Scaling Data 3

1 Light Gradient Boosting Machine

The Task

In this competition, we are going to preform a *binary classification problem*. In our basic model, only features from `application_train.csv` are used.

1.1 Data Preprocessing

Notation 1.1. Each row of raw dataset represents one single *observation*(sample) and each column represents one *feature* or the *target*.

1.1.1 Dropping Invalid Data

Basic `SK_ID_CURR` and `TARGET` are excluded from training features.

Dropping Invalid Features Many of the columns in the raw dataset involves *invalid observations* (marked as `nan`).

One can specify a `DROP_THERSHOLD` parameter ranges from 0% to 100% (default at 10%). Features that are available for too few observations are considered as *invalid features*. Invalid features will be excluded.

With the default setting, 10%, features containing more than 10% invalid observations will be considered as invalid features. And we have 63 features left with the default 10% threshold.

Dropping Specific Features One can specify a `DROP_COLUMNS` parameter to manually drop some features from the raw dataset. In our baseline model, no feature besides `SK_ID_CURR` and `TARGET` are dropped through this manner.

Dropping Invalid Observations After feature selection, observations still with `nan` feature(s) will be dropped, so that we won't have any `nan` element in our feature and target dataset for model training.

1.1.2 Encoding Categorical Features

After feature selection, many of them are *categorical features* in string format. And they are encoded so that they can be fed into our model.

Integer Encoder The package `LabelEncoder` from `sklearn` package is used for categorical feature encoding.
Features are encoded are encoded

1.1.3 Splitting Training, Testing and Validation Sets

1.1.4 Scaling Data

Generally, two methods of scaling, normalization, and standardization, are considered to handle features with significantly different ranges.

After integer-encoding categorical features, all features in the training set are in floating format. We can, therefore, scale them so that all features have similar distributions.

Specifically, since the testing set is unobservable for our model and validation set is used to simulate this property, we are going to fit our feature-scalers on the training set only.

Then, in the model validation phase and evaluate phase, feature-scalers are applied on validation and testing sets.

Normalization. For each feature j, we fit a feature scaler from the training set. Let $\max(x_j)$ and $\min(x_j)$ denotes the largest and smallest observed values of feature j in training set.

For each observation x_{ij}, the normalized proxy z_{ij} is generated following equation (1)

$$z_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{1}$$

Note that after normalization, ranges of all features are normalized and bounded to $[0, 1]$.

Standardization. For each feature j, firstly, the training set standard deviation and mean are calculated and denoted as σ_{x_j} and $\overline{x}_j$.

Then, for each observation x_{ij}, the standardized proxy is generated from the equation (2)

$$z_{ij} = \frac{x_{ij} - \sigma_{x_j}}{\sigma_{x_j}} \tag{2}$$

After standardization, every proxy feature has the same mean of 0 and variance of 1.

Remark 1.1. In our model, standardization methods are used as default.