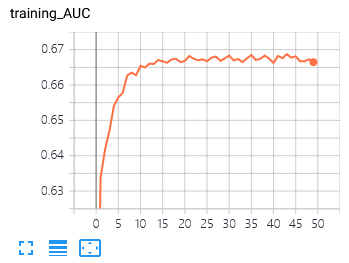
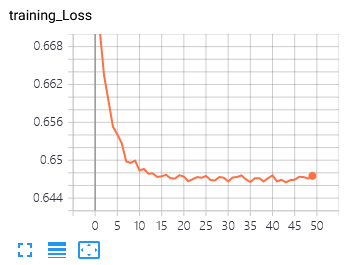
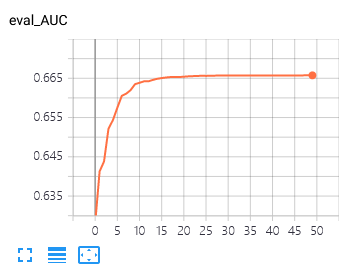
**model overview**

For a classification problem to predict the probability of individuals, the first came into one’s head is the logistic regression model. It is a generalized linear model, so it is less likely to become overfitting than a nonlinear model. And it has some perfect properties that is convenient for us to make regression diagnosis. However, the input features are not that complete, which means some features have many missing values, and for some other features, the proportion of having zero values is more than 90%. Due to the imperfect condition of our data, we also consider the gradient boosting decision tree. Specifically, we use the lightGBM algorithm.

**logistic regression**

For the logistic regression model, we use PyTorch to realize it. First, let all the input features pass through a fully connected layer and the output of this layer is one-dimensional, where is the parameters to be estimated. Then let the output of the first layer pass through an active function, which is the sigmoid function here. Then, the output is the probability of each individual. As for some technical details, we use ADAM as the optimizer and reduce the learning rate dynamically by monitoring the evaluation loss.

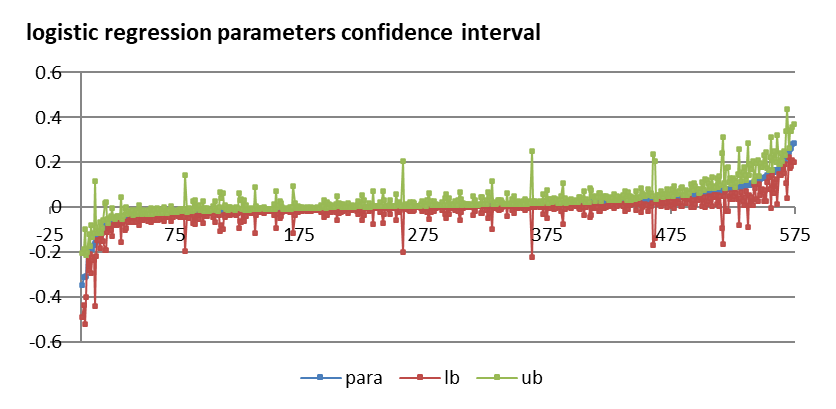
Before training, we split the dataset to make 20% of data the evaluation dataset. Since the training data is significantly unbalanced, we spent a little trick on choosing the epoch train data. Now let’s denote the total number of vaccinated samples as and the number of unvaccined sample as . In every epoch of training, we first used all the vaccinated samples, whose sample size was , as the training data. Then, we also chose from the unvaccined samples randomly. In this case, we could make sure that the training data for each epoch was strictly balanced and the total sample size was . If we train for enough epoches, we should utilize all the information from the original dataset.

**logistic regression model diagnosis**

We used tensorboard to visualize and monitor the training process. On evaluation dataset, the logistic regression model converges smoothly and the model performance on both training and evaluating dataset are consistent, that means the over-fitting situation doesn’t happen. The final AUC score is about 0.665. The next step is to calculate the standard deviation of the estimated parameters, which is given by

where is the diagonal matrix generated by . The rule of feature selection is that, if falls into such an interval , then is insignificant. After this step, we have about 300 variables which are significant. What is worth mentioning is that, though some features have a large , they are not significant due to a relatively larger standard deviation. By the way, there is no need to do Lasso logistic regression to enforce some to , since our sample size is greatly larger than the number of variables.



The chart above is the model parameters sorted by values, the green points are the upper bounds for each parameter and the red points are the lower bounds. If the x axis crosses the green or the red line, the corresponding parameter would be considered insignificant. As is shown in the middle part in this chart, most parameters, whose values are near 0, are not significant. What’s more, in the head and tail of this graph do exist some parameters, which have large parameter values but relatively larger standard deviation.

**bosting tree**

Boosting tree is a set of nonlinear decision tree models. To make a brief introduction, after generating a single tree , we will generate a new tree to fit the regression residuals. So the predicting model is given by

We use lightGBM here because it is fast and precise as well. To avoid over-fitting, we choose the maximum depth of a tree ti be 6, maximum number of leaves in a single tree to be 60. This cannot be too small since we have about 800 input features and should be less than . What’s more, we set the minimum data in every leaf to 1000, and the number of trees to 300.

Similarly, we split the dataset to make 20% of data the evaluation dataset. As mentioned before, our data is unbalanced, and we cannot use the batch-training method in lightGBM, since trees are not additive. So, using the prior notation and , we choose randomly 4 subsamples from unvaccinated samples and the size of each subsample is . Then we concatenate the vaccinated sample with these subsamples respectively to get 4 training datasets. On each dataset, we train the lightGBM model with the same hyperparameters. For evaluation and holdout data, we use these 4 fully trained lightGBM models to predict the probability of each individual and average them as the final predicted probability. We considered the average performance of each parameter on the 4 datasets, that is, how many times were they chosen as a tree node.

