GCI 2023 Summer 第二回コンペ Home Credit Default Risk 解説

Mayumi_Nakano

モデル: XGBClassifier　Public LB: 0.76751　Private LB: 0.76108　CV 値（AUC スコア）: 0.764109

以降、train データと test データを結合して処理しています。

【欠損値や異常値に対する処理】

今回は、全体的に欠損値が多すぎるので、補完は行いませんでした。そのため、欠損値があっても学習が可能な勾配ブースティング木である XGBoost を用いました。

“AMT_INCOME_TOTAL”: 20,000,000 以上のものは異常値ということで、欠損値（NA）としました。

“DAYS_LAST_PHONE_CHANGE”: 0 であったものは、欠損値（NA）としました。

【特徴量エンジニアリング】

新たに生成した特徴量に関しては、量が多すぎたので割愛させていただきます。基本的には本家 kaggle の Home Credit コンペで上位入賞を果たしていたノートブックやディスカッションを参考にして作成しました。

一方で、特徴量エンジニアリングは多く作れば作るほどいいというわけでもありません。生成した特徴量の多くは引用したものでしたが、その後に自身で特徴量の重要度を示すコードを作成し、使用する特徴量を選別する作業を行いました。選別する基準としては、基本的に特徴量の重要度が 0 であるものに関しては学習で使用しないようにしました。ただし、時に特徴量エンジニアリングによって生成した特徴量 A は重要度が高いにもかかわらず、その特徴量 A を作るのに使用した別の特徴量 B の重要度は極端に低い場合があります。こうした場合では、特徴量 A と特徴量 B は相関を持っている場合が多いため、特徴量 B の重要度がいくら低くても削除しないほうが良いと思います。

【Optuna パラメータチューニング】

今回のコンペで一番差が付くとしたらこのパラメータチューニングなのではないかと思います。僕のおすすめは Optuna という機械学習のパラメータを自動で効率的に探索してくれるライブラリです。これを使うことで、ランダムサーチよりも正確に、全探索よりも高速にパラメータ探索を行ってくれます。

とは言っても、時間はかなりかかるものです。僕は M1 Macbook Air のローカル環境で一度パラメータ探索を行ったのですが、普通に 1 時間くらいかかってしまい、また CPU 温度も 100 度を超えて悲惨でした…

そこで使用したのが、Google Colab の GPU を用いる手法です。使い方は簡単で、やることはたったの 2 つです。一つは Google Colab を使う際に、「ランタイム」→「ランタイムのタイプを変更」→「ハードウェアアクセラレータ」→「GPU」とすることです。もう一つは、Optuna を使用する場合は目的関数のパラメータに「”tree_method”: “gpu_hist”」を加えることです。これによって、ローカル環境や Google Colab の CPU とは比較にならないスピードで計算を行ってくれます。僕の場合はローカル環境 1 時間→Google Colab GPU 5 分くらいでした（笑）

また、Google Colab ではオンラインで計算を行うので、自身の PC に負荷がかからない（温度が上がらない）のも嬉しいポイントです。Google Colab GPU での計算、ぜひやってみてください。