第二回コンペ（Home credit）説明文

Kitakami

1. 全体的な方針

基本的な方針は以下の通りです

とりあえずそのままの予測してみる→予測値の重要度を確認→重要度を参考に特徴量を追加・削除

初期の方はこの流れで予測値を改善していました。モデルは速度を重視して lightGBM を選択しました。xgboost や catboost、NN なども試しましたが精度上がらず、パラメータの探索にも時間がかかったので lightGBM のみを最後まで使用しました。途中から CV が伸び悩んで（CV : 0.760 程度）からはパラメータの探索に注力していきました。結果的に上位に入れたのはこのパラメータの探索が大きかったと思います（CV : 0.763 程度）。

2. データ加工

・欠損値

lightGBM ということで、補完でそれほど変化なかったので NaN のままで学習させました。CODE_GENDER の XNA は train だけだったので行ごと削除、DAYS_EMPLOYED の 365243 は nan 埋め、DAYS_LAST_PHONE_CHANGE の 0 は nan として扱うことにしています。

・特徴量エンジニアリング

数値データ

AMT_系、DAYS_系、EXT_SOURCE 系の重要度が高い傾向にあったのでこれらを中心に特徴量を作りました。途中からネタ切れになったので上位の notebook からよさそうな特徴量を拝借させてもらいました。ChatGPT に無差別大量に特徴量を作ってもらうという戦略も試しましたが、あまり意味のある新しい特徴量は作れず、特徴量が多くなりすぎて把握できなくなったので止めました。ですが、ChatGPT で特徴量を作るのはドメイン理解がない時には有効的なのかなとも思いました（先に notebook を参考にしていたので今回の場合、意味はなかったですが）。あと ChatGPT だとカラム名も考えてくれるのでそこも便利だと思います。特徴量追加によってできた inf のデータは 0 埋めしています。

カテゴリ変数

label・one-hot・target エンコーディングそれぞれ試した結果、one-hot が一番良好だったので one-hot を採用（上位の notebook だと label が良いと書かれていましたが今回の条件だと one-hot が良かったです）。

・特徴量の削除

特徴量の削除はスコアにはあまり影響しませんでした（計算コストは軽減されるので意味が全くないわけではない）。ですので、初期の方は削除を行っていましたが途中から削除をやめています。

3. パラメータの探索

特徴量でスコアが動かなくなってからはパラメータの探索に注力しました。最初は手動で max_depth 値をチューニングし（これが大きくスコアに影響したと思われる）、他のパラメータを optuna でチューニングしました。探索の際は時間がかかるので 1fold のみでやっていました。

4. 失敗と反省

一回目のコンペは忙しくちゃんと参加できず、二回目が実質初めての参加になりました。そういったこともあって、何個か失敗を犯しました。失敗から学べることの方が多いと思うので今回一番苦戦したファイル管理について記しておこうと思います。

・ファイル管理

提出したファイルがどのファイルだったのかわからなくなることがよくありました。基本的には LB 上でデータの最初の方が見られますが、うっかり上書きしたりなどどうしようもなくなる時もあると思います。他にも、どのデータ・モデル・パラメータで予測したのかなど、行方不明になることが多かったです（結果的にパラメータが行方不明になったため公開したコードでは提出したデータを完全に再現できません（TA の方に通達済み））。python はインタプリタ方式ということもあり上書きミスもしやすいです。こんなところで失敗はしたくないのでファイル管理はしっかりとしないといけないなと思い知らされました。

・簡易的な対処方法：ファイル名にタイムスタンプを押す

datetime モジュールを使ってファイル名にタイムスタンプを押すのが一番いいのかなと思いました。さらにスプレッドシートや Notion などで管理するのがよさそうですね。

（なにかいい方法があればぜひ DM 等で教えてください！！）

サンプルコード（いつのデータなのかをファイル名に）

```
from datetime import datetime
# 現在の日時を取得
now = datetime.now()
# 日時を文字列に変換（例: 20230711_123456）
timestamp_str = now.strftime("%Y%m%d_%H%M%S")
# ファイル名を作成
filename = f'predictions_{timestamp_str}.csv'
# 予測データを CSV ファイルに保存
predictions.to_csv(filename, index=False)
```

モデル保存の関数（使用したデータファイル_モデルの種類_保存日時）

```
import joblib
def save_model(model, data_filename, model_type):
    # 現在の日時を取得
    now = datetime.now()
    # 日時を文字列に変換（例: 20230711_123456）
    timestamp_str = now.strftime("%Y%m%d_%H%M%S")
    # ファイル名を作成
    filename = f'{data_filename}_{model_type}_{timestamp_str}.pkl'
    # モデルをファイルに保存
    joblib.dump(model, filename)
    print(f'Model saved as {filename}')
```