【戦略概要】
まず、今回も前回と同様で基本的な戦略としては、Kaggle で公開されている上位の Code・Discussion を全て読み込み取り入れられるデータを全て使用するという戦略を取りました。ただし今回第一回 Competition の Titanic とは異なり、配布されたデータが実際の Kaggle の Competition では複数であったものの、1 つのデータのみの配布であったので Kernel にあるコードの完全コピーは出来ず、Kernel を前回よりもきちんと理解した上で使用する必要があったのが少し大変でした。
【実際に行ったこと】
基本的に Kaggle 等のコンペティションにおいて必要なことは、特徴量エンジニアリング、モデルの作成、ハイパーパラメータの調整、アンサンブル学習の 4 つの手法です。

① 【モデルの選定】今回、上位複数の Kernel を見た結果、基本的には LightGBM が最も高い精度を出していたため、モデルの作成は LightGBM に絞り、特徴量エンジニアリングから進めていきました。
② 【特徴量エンジニアリング①】今回のデータ予測では債務不履行者の予測をするため、「お金をきちんと返しそうな人を表す特徴量」と「お金を返さなそうな人を不真面目な人を表す特徴量」の作成をまずは自己流で行いました。その後、LightGBM では学習の時間はかかるが特徴量を沢山増やした所で予測の精度に悪影響がないというような記事を見つけたため、上位の Kernel で使用されている特徴量を片っ端から突っ込むという手段を取りました。
③ 【特徴量エンジニアリング②】片っ端から突っ込んだ結果特徴量が増えてしまったので、特徴量を減らすために、Null Importance や XAI での重要特徴量の選定を行い、特徴量を削って学習させるということを試しましたが精度は向上せず Kernel から引用し、片っ端から増やした特徴量はそのままの状態となりました。
④ 【特徴量エンジニアリング③】LightGBM では特徴量同士の四則演算の関係を自身で特徴量として入力をしないと情報として入力をしてくれないという記事を読み、全ての特徴量の四則演算結果を新しい特徴量として追加するということを行いましたが、毎回のモデルの学習に時間がかかりすぎる（1 日以上かかる）かつ、feature_fraction を 0.1 といった低い値に設定して特徴量の増加に対して悪影響が出ないようにしたにも関わらず普通に精度が下がったので四則演算を闇雲に突っ込むのはやめました
⑤ 【特徴量エンジニアリング④】カテゴリ変数については、例えば学歴のようなものは 1, 2, 3, 4 と順位付けをしてラベルエンコーディングするという手法も取りましたが、今回アンサンブル用に作成した方ほうではいくつかこの方法を実施しましたが、もう片方ではラベルエンコーディングをするよりワンホットエンコーディングをした方が良い結果が出たので基本ワンホットエンコーディングを使用しました。また、ターゲットエンコーディングという手法も試みましたが精度は上がりませんでした。
⑥ 【特徴量エンジニアリング⑤】欠損値等については、EDA を行った結果明らかに特異な値に対しては変更を加えましたが、LightGBM では欠損値を気にせずモデルの学習ができますし、

何より平均や中央値、重要特徴量でクラスタリングした近傍値での平均・中央値等複数の方法で欠損値を埋めて学習を行った結果精度が下がるという現象が起きたため、欠損値処理については簡易的に留めました。

ここまでの特徴量エンジニアリングを経て、ハイパーパラメータを調整しないで大体 LB での精度の最大値が 0.7666 程度でした。

⑦ 【ハイパーパラメータ調整】ハイパーパラメータの調整は Optuna を用いて行い、結果として 0.76743 まで精度が向上しました。

⑧ 【アンサンブル学習】いくら特徴量エンジニアリングをしても全く精度が 0.76743 から伸びずにいた時に、もう 1 つモデルを作成し、かつ今使用しているモデルの Seed を変更してアンサンブル学習をした結果精度が一気に 0.768 以上まで向上しました。あとは、各モデルの重みづけをすると精度が向上したので特に理由もなく、LB で思考できる回数まで適当に重みづけを行い最も精度が高かったデータを提出しました。