1.イントロダクション

今回のコンペは、特徴量生成も重要でしたがハイパーパラメーターの設定による差が大きく出るコンペでだったように感じます。十分な時間の確保が難しかったため、特徴量生成にはあまり時間をかけずにハイパーパラメーターチューニングを主に行いました。

2.特徴量生成

特徴量エンジニアリングは主に Kaggle のサイトのものをそのまま引用した形になっています。めぼしいオリジナルな特徴量としては、object 形式のカラムに対して Target エンコーディングや相関係数の高い外部データの値を用いたことです。この時点ではリーダーボードのスコアが 0.76 ほどとやや低い状態でした。

3.特徴量選択

特徴量選択では lightGBM の importance を参考に行いました。削除をメインに行っており、学習量の少ないオリジナルな特徴量に対して削除を行いました。この時、「optuna.integration.lightgbm」という便利なパッケージがあったのでこれを用いています。自動でハイパーパラメーターチューニングを行ったうえでの学習結果を出してくれるため特徴量選択の場面において非常に役に立ちました。

4.ハイパーパラメーターチューニング

今回最も時間をかけた部分です（勝手に見つけてくれますが)。ハイパーパラメーターチューニングは optuna を使って行っており、公開したコードのような形で行っています。目的としてはとにかく過学習を起こしたくないということで過学習を起こしにくいような値の中で最適なハイパーパラメーターを探すようにしました。

また、チューニングの際に今回最もスコアを上げる要因になったと考えられるものにも出会いました。通常通りに train と valid に分けてハイパーパラメーターを探索したのち、そのハイパーパラメーターで train データすべてと valid で学習させるとよい結果が得られました。おそらく今回のデータセットがたまたまうまくいっただけだと思いますが、コードの記述ミスから良い結果が生まれて驚きました。

5.スタッキング

スタッキングのもととなったのは、オリジナルの特徴量がメインのコード、引用の特徴量がメインのコード、それらをミックスさせた特徴量のコードの 3 つに対して lightGBM と XGboost を使った予測の計 6 つで行っています。これらの組み合わせを変えてスタッキングを行いました。この時参考にしたのがパブリックスコアです。リーダーボードを見ていると、同じスコアの人がいることから非常に少ないデータ数に対するスコアを出されていると思われましたが、ほかに参考にする要素がなかったので仕方なくパブリックスコアを参考にしました。結果として引用の特徴量がメインのコードを使うことはなかったのですが、CV スコアからもやや低い結果が得られていたのでパブリックスコアが乖離していることはなさそうということで提出した組み合わせを用いました。