#21 Solution | Stacking + Hill Climbing

First, I would like to start by sincerely thanking Kaggle for hosting this episode of the playground series. In this post, I will briefly outline my approach.

Base Models

The table below summarizes the various models that were built using the data:

Model	**Number of Models**
Catboost	20+
LGBM	50+
LGBM-Goss	10+
XGBoost	50+
Logistic	10+
Multinomial NB	10+
GradientBoostedTreesLearner	5+

All the base model were trained over the same 10-fold cross-validation strategy, StratifiedKFold(n_splits=10, shuffle=True, random_state=42).

Stackers

Most of the top-scoring public notebooks used LogisticRegression as the final_estimator in the StackingClassifier. However, in many of my experiments, I did not achieve good alignment between the local cross-validation score and the public leaderboard score. As a result, I decided to try a different model as the final_estimator. I found that using LinearDiscriminantAnalysis as the final_estimator in the StackingClassifier generated a consistent alignment between the local cross-validation score and the public leaderboard score. I also experimented with GaussianNB as the stacker and observed similar behavior as with LogisticRegression. Notably, I trained over 50 different stackers using LinearDiscriminantAnalysis as the final_estimator.

Hill Climbing

The table below displays the public leaderboard scores of my top four stackers:

Stacker	**Public LB score**
1	0.38239
2	0.38233
3	0.38224
4	0.38211

Then, I saved the out-of-fold predictions from the above stackers and used them to estimate the weights for the ensemble using hill climbing. The public leaderboard score of the hill climbing ensemble was 0.38247 and the private score was 0.38385.

What worked

- Treating all features as categorical as pointed out in a couple of discussions.
- Adding multiple copies of the original dataset for training purposes.

What didn't work

- Feature engineering