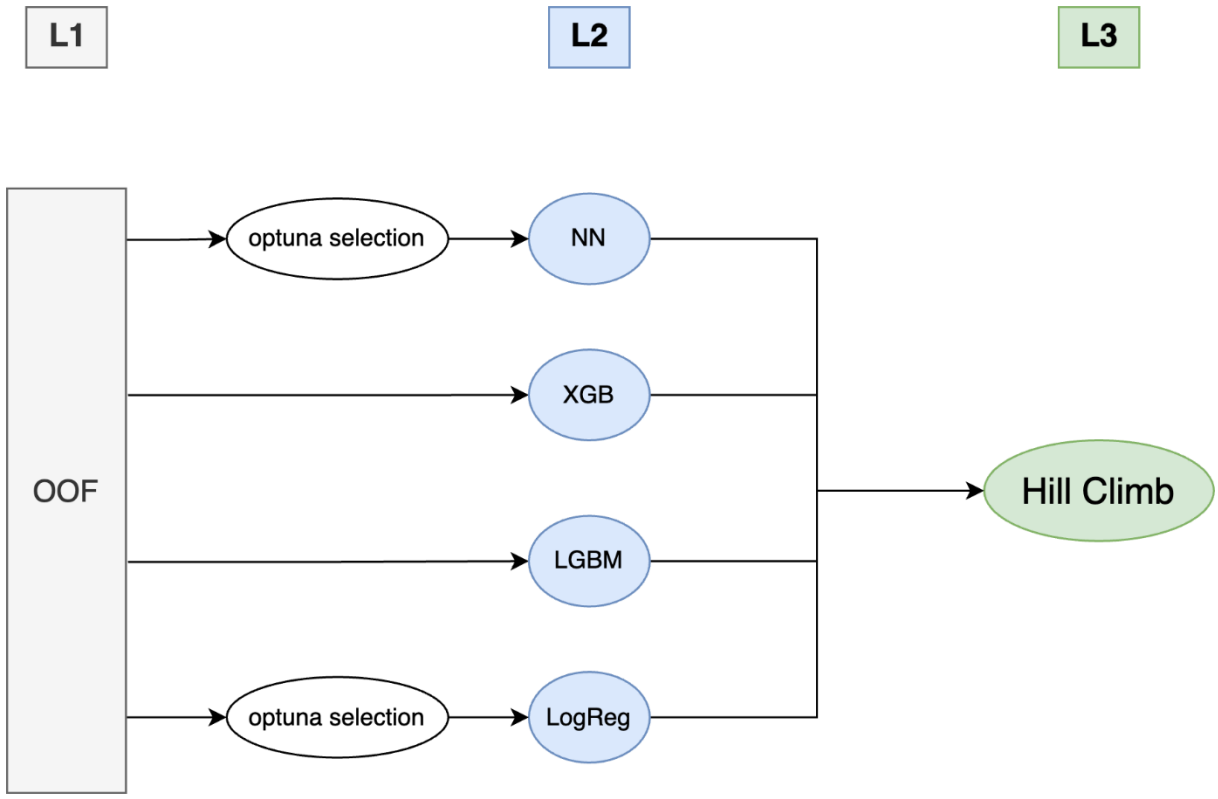
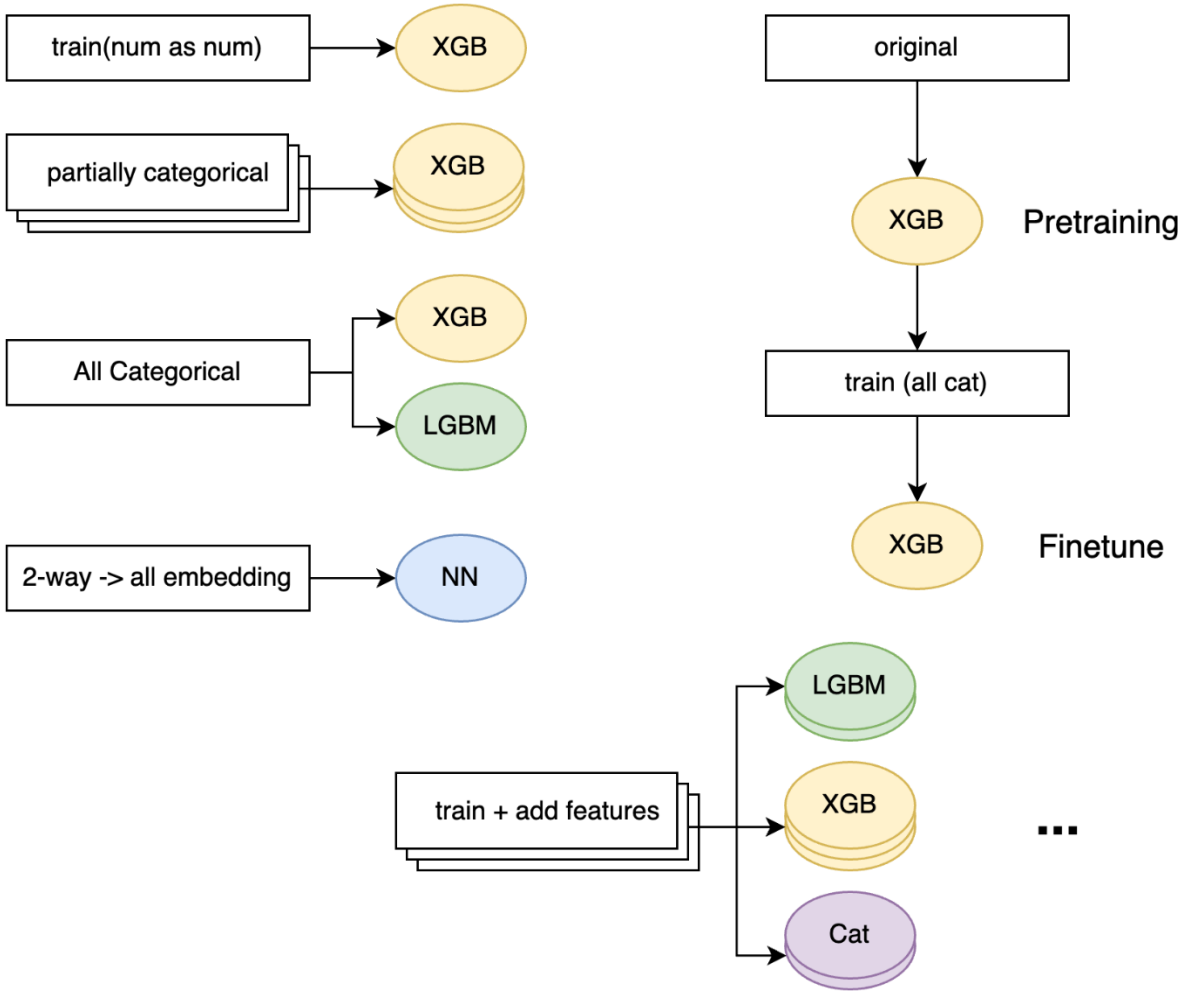
2nd Place Solution - L3 Ensemble of 100+ OOFs

First of all, I want to express my gratitude to the organizers for hosting another incredibly fun Playground competition this month.
I'd also like to thank everyone who shared discussions, code, and various insights. It was a very enjoyable and educational experience.

My solution is a 3-layer ensemble:
100 OOF predictions (from a 5-fold SKF) → L2 models (XGB, LGBM, LogReg, NN) → L3 Hill Climb.
All experiments and submissions were done exclusively using Kaggle Notebooks.



L1 - The Model Zoo



- Multiple **XGBoost** models trained with different combinations of categorical features. Some were trained on the original dataset only, while others were trained on train + original (with a weight of 4.0 for the original data).
- An **XGBoost** model pre-trained on the original data and then fine-tuned on the train data.
- **XGBoost** models trained with additional engineered features, such as binned features, clustering features, and Target Encoding.
- An **XGBoost** model that included latent features generated by a Supervised AutoEncoder.
- **XGBoost** models using auc and merror as their eval_metric.
- **ExtraTrees**, **RandomForest**, and **NN** models trained using the predictions of XGBoost as features.

- Several **LightGBM** models, mirroring the XGBoost experiments.
- **Logistic Regression** and three types of **Neural Networks** (TabTransformer, simple NN).

Here is a summary of the CV scores for the main L1 models:

Model	**CV (mAP@3)**
All Cat XGB	0.37768
All Cat LGBM	0.37483
CB over XGB	0.37707
Finetune XGB	0.37753
NN	0.36067
XGB + FE	0.345 - 0.365

Most other models had scores between 0.34 and 0.369. While they didn't perform as well individually, they contributed to the diversity of the ensemble. Notably, models with added features like Target Encoding or latent features from the Supervised AutoEncoder scored around 0.35, but significantly improved the final ensemble performance.

L2 & L3 - Stacking and Hill Climbing

I used 100 of the OOF predictions to train L2 stacking models: **XGBoost**, **LightGBM**, **Logistic Regression**, and a **Neural Network**.

- For **Logistic Regression** and the **NN**, I found that using too many OOF features hurt both CV and LB performance. I used **Optuna** to select the best subset of OOFs for these models. I wanted to use forward selection, but opted for Optuna due to computational constraints.
- For **XGBoost** and **LightGBM**, I used all 100 OOF features without any parameter tuning.

Finally, I blended the predictions of these L2 models using a **L3 Hill Climb** algorithm, optimizing directly for mAP@3, to create my final submission.

Here are the CV scores for the L2 and L3 models:

Model	CV (mAP@3)	CV (logloss)
L2_NN	0.38321	1.87040
L2_LGBM	0.38343	1.87014
L2_XGB	0.38348	1.86997
L2_LogReg	0.38362	1.87074

Model	CV (mAP@3)	CV (logloss)
L3_Hill Climb	**0.38418**	**1.86958**

Discussion & Thoughts

As many of us discovered, **XGBoost** performed exceptionally well in this competition. It was difficult to achieve a similar score with CatBoost. I haven't yet reached a satisfying conclusion as to why there was such a significant performance gap.

One interesting observation was the optimal max_depth.

- When treating **all columns as categorical**, the optimal max_depth was around **7-8**.
- When treating numerical columns as numbers, the optimal max_depth was much higher, around **16-18**.

Here's my hypothesis:
By treating numerical columns as categorical, the model can learn non-linearities and interactions more efficiently with fewer splits, while also being less sensitive to outliers. This combination likely allowed for a more robust and generalizable model to be built with a shallower max_depth (7-8). The very high max_depth of 16 when using raw numerical features might have been a sign of the model trying to force-fit the data's complex relationships through a series of linear splits, potentially leading to overfitting. Categorization seems to have provided a "hint" that helped the model interpret these complexities more appropriately.

A question for the community: I'm still uncertain about using Hill Climb as an L2 model. Since it's a purely linear combination, I suspect that cross-validation might not be strictly necessary, though it feels like the safest approach. Is it effective as an L2 model, or is it better reserved for a final L3 blend? I would be very grateful to hear your thoughts, experiences, or theoretical explanations on this topic.
(As an aside, in my experiments, the selected OOF combinations were nearly identical across folds, which suggests CV might not have been essential).

Key Takeaways from This Competition

- **Ensembling is Key:** When using Hill Climb, it's more important to have **diverse models** (even with varying scores) than a collection of similar high-scoring models.
- **Stacking with Tree Models:** XGBoost and LightGBM are relatively robust to noise when used as stackers, making OOF feature selection less critical.
- **Stacking with Linear/NN Models:** Logistic Regression and NNs are more sensitive to noise, and their performance often improves with careful OOF feature selection.
- **Weighting Original Data:** Giving a significantly larger weight to the original dataset can sometimes lead to a performance boost.
- **Optimization Target:** While directly optimizing mAP@3 is difficult (optimizing mlogloss is usually a good proxy), Hill Climbing seemed more stable when optimizing for mAP@3 directly rather than for logloss.
- **Feature Engineering:** Treating all features as categorical (or binned) can be a very effective strategy, but be aware that it can change the optimal hyperparameters, like max_depth.
- **The Final Week:** It's best to ignore the Public Leaderboard in the final week to avoid overfitting and second-guessing your CV strategy.