Credit Underwriting Technical Presentation

For more personal insights, please refer to the threads by Olivier, Ryan, and myself, as well as Phil's comment.

Introduction

From my prior experience with credit underwriting, I’ve come to appreciate that this is one of the most complex problems for applying machine learning. Data in this domain tends to be very heterogeneous, collected over different time frames, and coming from many different sources that may change and alter in the midst of the data collection process. Coming up with a proper target variable is also a very tricky process, requiring deep domain knowledge and refined business analysis skills. I want to commend Home Credit and Kaggle for providing such a great dataset, which was leak-free and very amenable to machine learning techniques.

Based on what is known about credit underwriting, and similar machine learning problems, it was clear that two things would be crucial for building a good model for this competition:

1. A good set of smart features.
2. A diverse set of base algorithms.

We were able to utilize four main sources of feature diversity, along with a few minor additional ones.

Data and Feature Engineering

Feature Engineering (Ryan)

Like many people, I started my base models with simple aggregates over `SK_ID_PREV` and `SK_ID_BUREAU` features for each `SK_ID_CURR`. I also created many features based on division and subtraction from the `application_train.csv`. The most notable division was by `EXT_SOURCE_3`, which gave me small boosts to my CV and translated positively to the LB. I also found decent boosts by using a label encoder for categorical variables, applied to all categorical variables in both `application_train.csv` and the LAST application in `previous_application.csv`.

Aside from aggregate features over `SK_ID_PREV` and `SK_ID_BUREAU`, I also used different slices of data to compute aggregates:

- **previous_application.csv**: Aggregates for the last 3, 5, and first 2, 4 applications.
- **installment_payments.csv**: Aggregates for the last 2, 3, 5 payments, and over different periods such as 60, 90, 180, and 365 days, filtered on `DAYS_INSTALMENT`. Aggregates over all installments that were past due.
- **POS_CASH_balance.csv, credit_card_balance.csv**: Similar methodology as `installment_payments.csv`.

Feature Engineering (Olivier)

I computed a yearly interest rate that became one of the highest scoring features in my models. I also attempted to create predictions on a few tables (bureau and previous application) but did not get the boost reported by other teams. With new skilled members joining, bringing features/dataset that gave a real boost to our CV/LB, I focused on stacking.

Feature Engineering (Phil)

The most important features I engineered, in descending order of importance (measured by gain in the LGBM model), were:

- `neighbors_target_mean_500`: Mean TARGET value of the 500 closest neighbors of each row.
- `region_id`: The `REGION_ID_POPULATION` field treated as a categorical feature.
- `debt_credit_ratio_None`: Sum of all credit debt over the sum of all credit.
- `credit_annuity_ratio`: `AMT_CREDIT / AMT_ANNUITY`.
- `prev_PRODUCT_COMBINATION`: `PRODUCT_COMBINATION` value from the most recent previous application.
- `DAYS_CREDIT_mean`: Mean `CREDIT_DAYS` value from the bureau table.
- `credit_goods_price_ratio`: `AMT_CREDIT / AMT_GOODS_PRICE`.
- `last_active_DAYS_CREDIT`: Most recent `DAYS_CREDIT` value from active loans in the bureau.
- `credit_downpayment`: `AMT_GOOD_PRICE - AMT_CREDIT`.
- `AGE_INT`: `int(DAYS_BIRTH / -365)`.
- `installment_payment_ratio_1000_mean_mean`: Mean of `AMT_PAYMENT - AMT_INSTALMENT` for installments where `DAYS_INSTALLMENT > -1000`.
- `annuity_to_max_installment_ratio`: `AMT_ANNUITY / (maximum installment)`.

Feature Engineering (Yang)

I used features from open solutions, modifying the time periods to include more variance. I applied weighted means (using the time period as the weight) to create features related to annuity, credit, and payment, extracting personal credit habits. Additionally, I generated some KPIs constructed by income, payment, and time, which positively impacted the CV.

Feature Selection and Reduction (Bojan)

Restricting the feature set proved useful, as aggregating features resulted in thousands of features, many of which were redundant or noisy. I used frequency-encoded categorical features with numerical features and ran simple forward feature selection using Ridge regression. This reduced the original set of over 1600 features to just 240. Adding more features brought the total to 287, achieving CV scores of 0.7985 and LB scores of 0.802-0.803. Combining features from new team members led to supersets numbering 1800-2000 features.

Base Models

All base models were trained on StratifiedKFold with 5-folds.

Models Used:

- **Olivier**: LightGBM, FastRGF, FFM (with below expectations CV results).
- **Bojan**: XGBoost, LightGBM, CatBoost, Linear Regression. XGB models trained with `gpu_hist` on a GPU, LightGBMs on CPU. CatBoost models helped with metafeature diversity.
- **Ryan and Yang**: Several LightGBM models on engineered datasets and combinations with other datasets. Ryan also tried FFM without success.

Neural Networks (Michael Jahrer)

Neural networks underperformed compared to boosted trees in terms of AUC. Using DAE+NN improved performance slightly. DAE involved denoising autoencoder preprocessing, with 10000-10000-10000 topology for the first models, later changed to fewer, larger hidden layers. Best AUC was achieved with a DAE with one hidden layer of 50k neurons, followed by a 1000-1000 supervised NN. Despite neural net optimizations, LightGBM with a small learning rate outperformed them.

Ensembling

We used three levels of ensembling with a pipeline for CSV generation, sharing meta files across team members, and centralizing concatenation work.

- **Level 1**: NN, XGBoost, LightGBM, Hill Climber linear model.
- **Level 2**: NN, ExtraTree, Hill Climber.
- **Final Prediction**: An equal-weighted blend of the Level 2 predictions.

Insights and Recommendations

- Hyperparameter tuning was minimal. We relied on diverse models for ensembling rather than optimizing individual models.
- Train/test prediction had an AUC of over 0.98, suggesting potential for adversarial validation or pseudolabeling.
- Imputing `EXT_*` features did not help base models, as these features were consistent between train and test sets.
- Our top three base models would have placed in the top 10 individually, suggesting feature engineering and selection are crucial.

In conclusion, feature engineering and selection were the most important steps in this competition, and a single, highly optimized model might outperform complex ensembles.