Home Credit – Credit Risk Model Stability

By Andrew Scott

Problem Statement

Statistics and Machine Learning have long played a key role in determining credit default likelihood. Traditional approaches have been algorithms such as Linear Discriminant Analysis, Logistic Regression, and Decision Trees. Today, one of the most popular and well performing algorithms are Gradient-Boosting Machines. Due to very high visibility and growth of the consumer loan industry, this topic has been studied in detail and models have been highly optimized already. Any small increase in performance can be considered an accomplishment. With the rebirth of Neural Networks and recent steps in the space, there is potential for these deep learning methods to become more popular in credit risk modeling and offer a route for improvement.

According to Home Credit, a consumer credit provider who focuses on responsible lending to people with little or no credit history, a scoring model which stays accurate over time is important. If behaviors and factors change then a model may lose its utility and begin to suggest bad loans or even prevent applicants without banking history to borrow money.

The Kaggle host, Home Credit, argues that if data science could help better predict loan repayment capabilities then consumer lending can become more inclusive and accessible. Our goal here is to explore different machine learning models that can stay effective over time. This will be measured by a Gini Stability metric calculated over a series of weeks.

$$\text{gini}=2*\text{AUC}-1$$

A linear regression, $a{\cdot}x{+}b$, is fit through the weekly Gini scores, and a falling_rate is calculated as $\min(0,a)$

. This is used to penalize models that drop off in predictive ability.

Finally, the variability of the predictions are calculated by taking the standard deviation of the residuals from the above linear regression, applying a penalty to model variability.

The final metric is calculated as

$$\text{stability metric}=mean(gini)+88.0\cdot min(0,a)-0.5\cdot std(\text{residuals})$$

Daniel Herman, Tomas Jelinek, Walter Reade, Maggie Demkin, Addison Howard. (2024). Home Credit - Credit Risk Model Stability. Kaggle. https://kaggle.com/competitions/home-credit-credit-risk-model-stability

Data Preparation

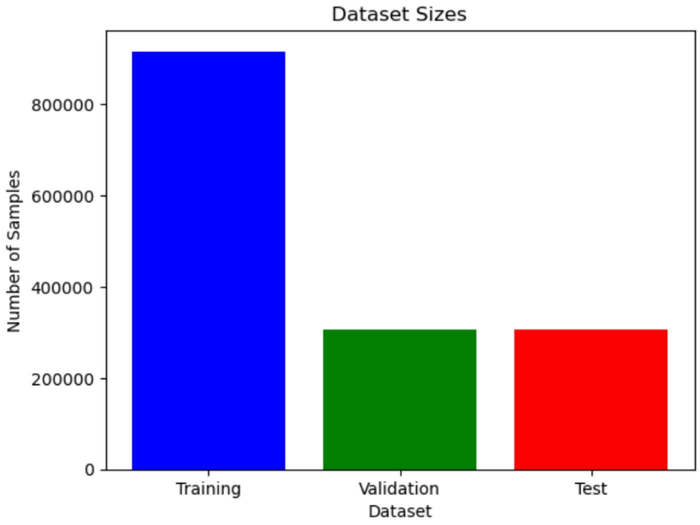
There are several files provided of which some are internal data sources and others external. There is a “Base” table which provides a unique observation, the outcome, and the associated time period. There are supporting tables which include internal static information, Credit Beareau, demographic, and tax data.

The data are nested in nature and thus some of the files must be aggregated per case_id (or per observation). For example, Payments Overdue is aggregated by Max, to represent the feature of that case for the maximum payment which is overdue. The different cleaned datasets are then joined on case_id and we end up with a DataFrame containing 46 features per case.

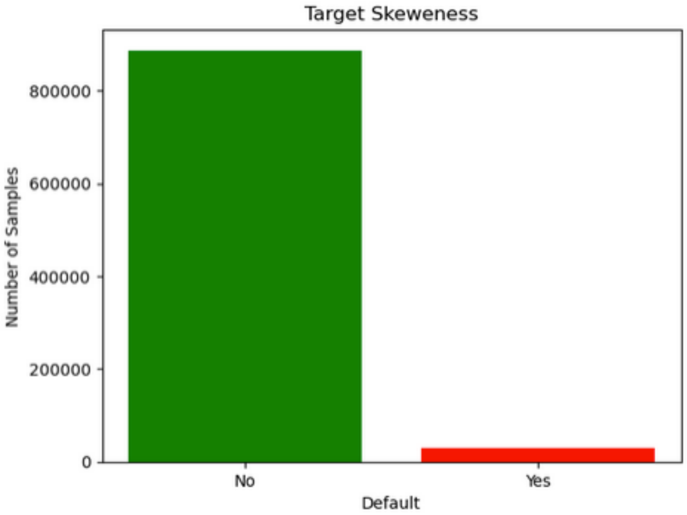
Columns are either numeric or of string or object type, we ensure that the string and object columns are converted to categorical. The first model, Light GBM, can accept the categorical input. However, past this we use SciKit Learn and Keras Tensorflow so we then need to pre-process some more. For numerical features, we substitute null values with zero and then scale between 0 and 1. For the categorical variables, we one hot encode using SciKit Learn’s encoding functionality.

Exploratory Analysis

The data provided is of high volume with over 1 million observations. This leaves us with over 800k training samples which is a good number for training machine learning models and even neural networks.



Consequently, when dealing with a problem like Credit Default we encounter the problem of unbalanced outcome. The fact of the matter is that the target outcome we are trying to predict, default or failure to repay loans, occurs a fraction of the time compared to the alternative. That said, we must look at the data to see what that fraction might be in this case. As you can see below, the Default = Yes outcome is much smaller than not. 28,872 observations are Default, or 3.15%.



We will explain how we account for this imbalance in our models to follow.

Methodology and Results

Light GBM

Light GBM is a lightweight Gradient-Boosting Machine algorithm which is built on tree-based learning. According to the documentation, it yields benefits such as Faster training speed, low memory usage, and better accuracy than traditional tree methods (*LIGHTGBM documentation*). The host provided a basic Light GBM model to act as a baseline for the competition. When reviewing other submissions and code submitted to the competition, it also appears that most other participants are utilizing this algorithm.

Gradient Boosting Machines appear to be an extremely popular technique in credit scoring Data Science. In this study, we expanded upon the base model by adjusting the below parameters with the goal of increasing performance.

- Max Depth: 5 ->8
- Number of Leaves: 20 -> 40
- Learning Rate: 0.05 -> 0.025
- Number of Estimators: 1,000 -> 2,000

This hyper-tuning improved the baseline algorithm by about one-percent. The baseline model already performs considerably well and so it is challenging to beat a well-suited model. This model yields a 75.5% AUC and a 46.7% Stability Score (reminder: the Stability Score is the aforementioned custom Gini Coefficient adaptation).

Logistic Regression

Logistic Regression is another popular algorithm applied to credit scoring problems. It offers simplicity and transparency unlike other methods like ensemble models or neural networks. However, they may fall short on dealing with imbalanced datasets.

Despite being a popular classifier, Logistic Regression did not work well with this feature set. The AUC was 50% and Stability score effectively 0%. Perhaps there is too many features and the model is too complex to work well with Logistic Regression.

Decision Tree

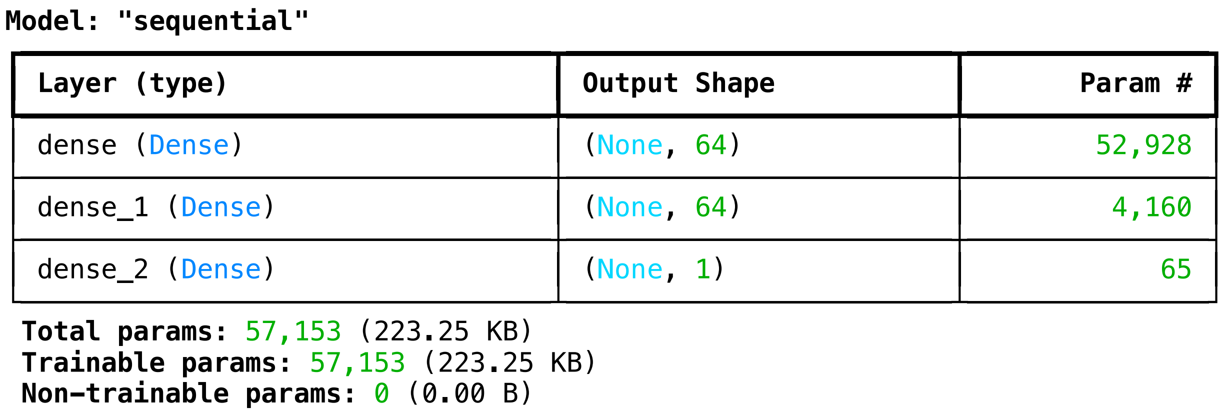
The next model attempted was a Decision Tree. Since Decision Trees are a basis of Gradient Boosting Machines, we figured it would be worth to try a simple Decision Tree model which would be parsimonious to a degree. Eight levels were passed as a hyper-parameter as maximum depth for the tree.

The performance of this model was almost identical to Logistic Regression where the AUC was 50% and Stability score was 0%. Perhaps the lack of voting through ensemble is causing a lack of performance on these more rudimentary methods.

Neural Networks

Neural Networks have been leveraged recently in studies and sometimes in industry for Credit Scoring, but are more common in other business applications. There is far less prior work in the subject of NNs in Credit Risk scoring. This is likely because the bagging models work well and a majority of focus remains in that field. It could also be due to the fact that regulations prevent the application of deep learning when making credit decisions. However, it is clear that deep learning does in fact work well in this case.

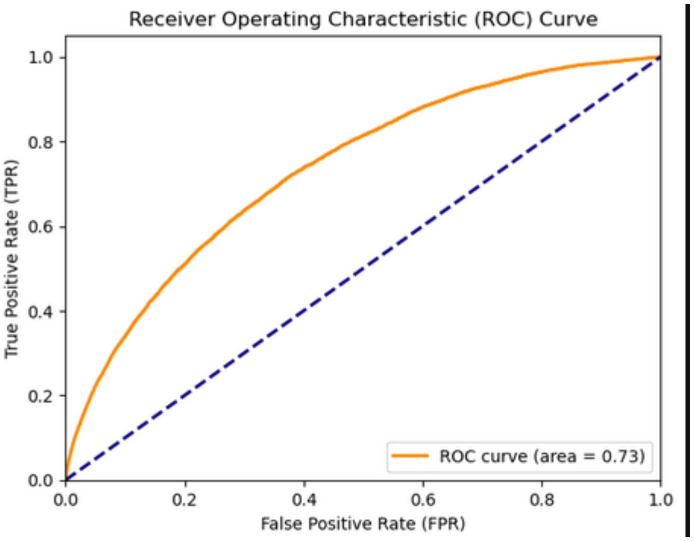
A: Simple Multi-Layer Perceptron



```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                   │ (None, 64)             │        52,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 64)             │         4,160 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 57,153 (223.25 KB)
 Trainable params: 57,153 (223.25 KB)
 Non-trainable params: 0 (0.00 B)
```

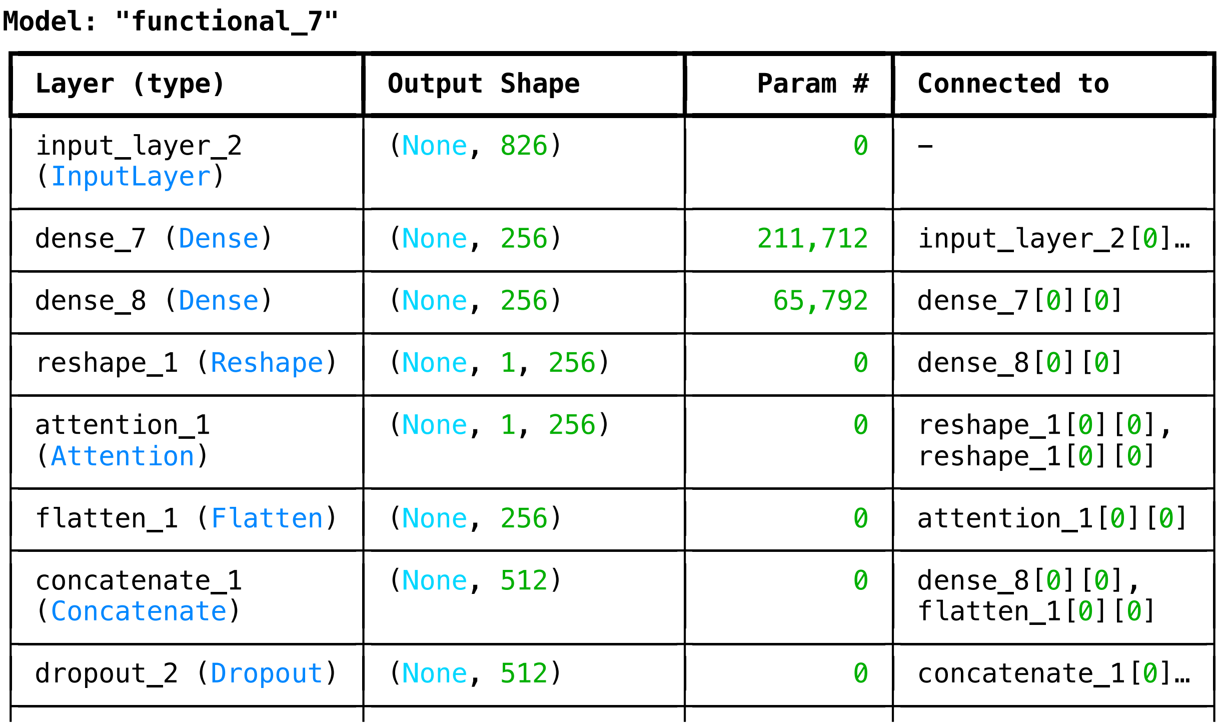
Hyper-parameter Summary
Optimizer: Adam

Loss: Binary CrossEntropy
Epochs: 10
Batch Size: 2048
With Validation

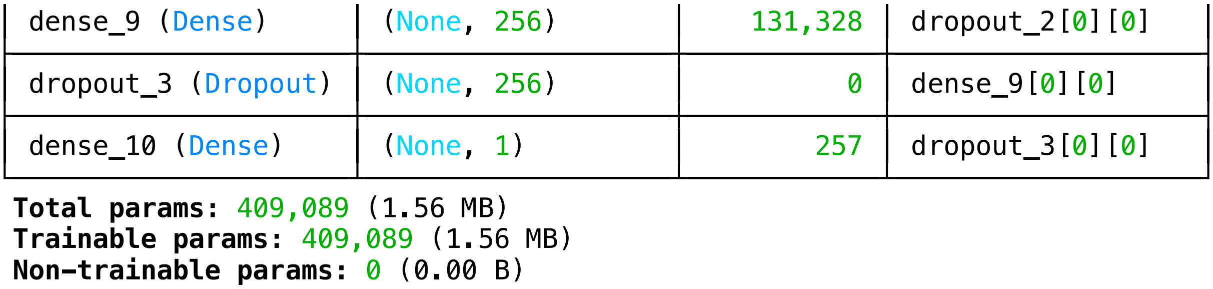


This model is a Multi-Layer Perceptron implemented in Keras Tensorflow. This model performed very well compared to the other models tried and evenly well with the best model so far, Light GBM. Accuracy was 96%, AUC was 74% and Stability score 43%.

B: Multi-Layer Perceptron with Attention, Dropout & Class Weights

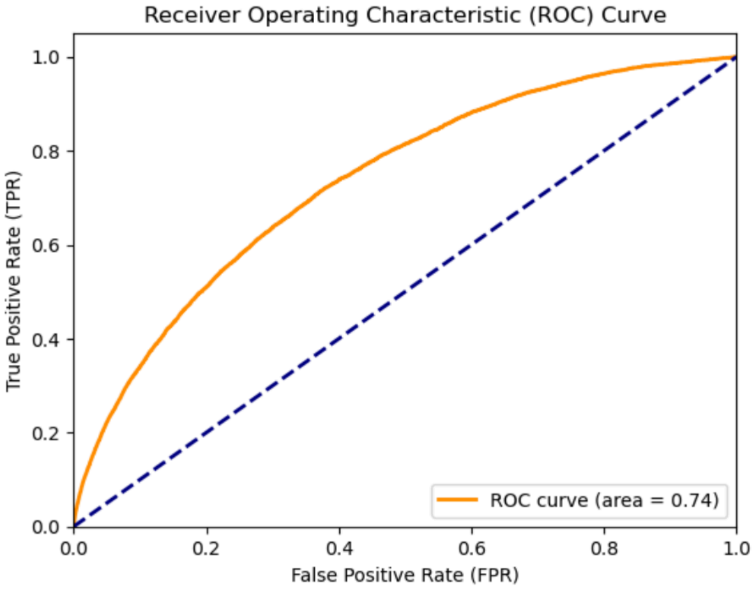


```
Model: "functional_7"
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ input_layer_2       │ (None, 826)       │          0 │ -                 │
│ (InputLayer)        │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_7 (Dense)     │ (None, 256)       │    211,712 │ input_layer_2[0]… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_8 (Dense)     │ (None, 256)       │     65,792 │ dense_7[0][0]     │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ reshape_1 (Reshape) │ (None, 1, 256)    │          0 │ dense_8[0][0]     │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ attention_1         │ (None, 1, 256)    │          0 │ reshape_1[0][0],  │
│ (Attention)         │                   │            │ reshape_1[0][0]   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ flatten_1 (Flatten) │ (None, 256)       │          0 │ attention_1[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ concatenate_1       │ (None, 512)       │          0 │ dense_8[0][0],    │
│ (Concatenate)       │                   │            │ flatten_1[0][0]   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dropout_2 (Dropout) │ (None, 512)       │          0 │ concatenate_1[0]… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
```

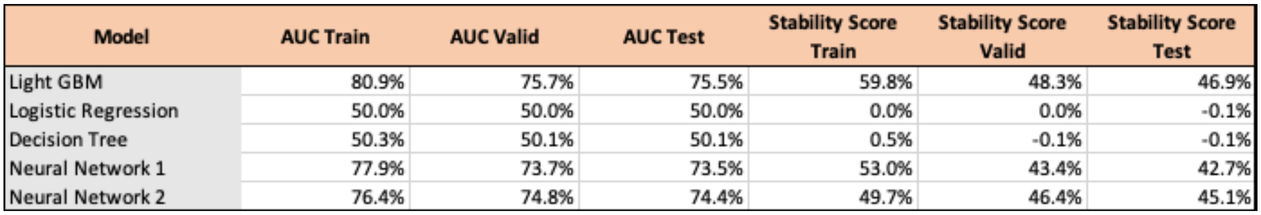


```
│ dense_9 (Dense)       │ (None, 256)       │     131,328 │ dropout_2[0][0]       │
├───────────────────────┼───────────────────┼─────────────┼───────────────────────┤
│ dropout_3 (Dropout)   │ (None, 256)       │           0 │ dense_9[0][0]         │
├───────────────────────┼───────────────────┼─────────────┼───────────────────────┤
│ dense_10 (Dense)      │ (None, 1)         │         257 │ dropout_3[0][0]       │
└───────────────────────┴───────────────────┴─────────────┴───────────────────────┘
 Total params: 409,089 (1.56 MB)
 Trainable params: 409,089 (1.56 MB)
 Non-trainable params: 0 (0.00 B)
```

Hyper-parameter Summary
Optimizer: Adam
Loss: Binary CrossEntropy
Epochs: 30
Batch Size: 2048
With Validation
With Class Weights



In this second Multi-Layer Perceptron, we implemented a few additional tools. First, we introduced **Attention**, with the goal of paying self-attention to the encoded input which may help the model determine which features are of most importance. Secondly, we introduced dropout to help generalize the model. Lastly, we added two more dense layers for a total of 3 dense hidden layers. Epochs were also increased to 30. This model is substantially more complex.



Model	AUC Train	AUC Valid	AUC Test	Stability Score Train	Stability Score Valid	Stability Score Test
Light GBM	80.9%	75.7%	75.5%	59.8%	48.3%	46.9%
Logistic Regression	50.0%	50.0%	50.0%	0.0%	0.0%	-0.1%
Decision Tree	50.3%	50.1%	50.1%	0.5%	-0.1%	-0.1%
Neural Network 1	77.9%	73.7%	73.5%	53.0%	43.4%	42.7%
Neural Network 2	76.4%	74.8%	74.4%	49.7%	46.4%	45.1%

Insights

Both the hyper-tuned Light GBM and the Multi-layer Perceptrons performed well on this dataset. For the competition, we submitted the Light GBM model because it did perform slightly better on the stability metric. However, both these models can be considered “Black Box” models. In the banking industry, there is significant regulation and control. A degree of transparency is required to be compliant with these regulations to ensure fair credit lending. One thing to consider when investigating the potential of Deep Learning in this environment is the ability to explain the model which is difficult.

A takeaway from this exercise would be to focus on explaining predictions using tools like SHAP and LIME and bundle that with a performant neural network.

Additionally, we would want to explore other options for dealing with imbalanced classification which may include techniques such as oversampling or SMOTE.

In conclusion, Deep Learning offers new possibilities in the Credit Scoring industry and should be studied and implemented more.

References

Daniel Herman, Tomas Jelinek, Walter Reade, Maggie Demkin, Addison Howard. (2024). Home Credit - Credit Risk Model Stability. Kaggle. https://kaggle.com/competitions/home-credit-credit-risk-model-stability

Microsoft. (n.d.). *LIGHTGBM documentation*. Welcome to LightGBM’s documentation! - LightGBM 4.0.0 documentation. https://lightgbm.readthedocs.io/en/stable/