Machine Learning Engineer Nanodegree
Capstone Proposal

Abhishek Bihani

June 15th, 2020

Home Credit Default Risk (Kaggle[1] Competition)

Domain Background

An important fraction of the population finds it difficult to get their home loans approved due to insufficient or absent credit history. This prevents them to buy their own dream homes and at times even forces them to rely on other sources of money which may be unreliable and have exorbitant interest rates. Conversely, it is a major challenge for banks and other finance lending agencies to decide for which candidates to approve housing loans. The credit history is not always a sufficient tool for decisions, since it is possible that those borrowers with a long credit history can still default on the loan and some people with a good chance of loan repayment may simply not have a sufficiently long credit history.

A number of recent researchers[2,3,4] have applied machine learning to predict the loan default risk. This is important since a machine learning-based classification tool to predict the loan default risk which uses more features than just the traditional credit history can be of great help for both, potential borrowers, and the lending institutions.

At a personal level, this project will help me gain an insight into which factors are the most important indicators for a bank when making a loan decision in case I decide to apply for a housing loan in the future.

Problem Statement

The problem and associated data has been provided by Home Call Credit Group[1], and the problem can be described as, *"A binary classification problem where the inputs are various features describing the financial and behavioral history of the loan applicants, in order to predict whether the loan will be repaid or defaulted."*

Datasets and Inputs

The dataset files are provided on the Kaggle website in the form of multiple CSV files and are free to download. The dataset files are described as per Figure 1.

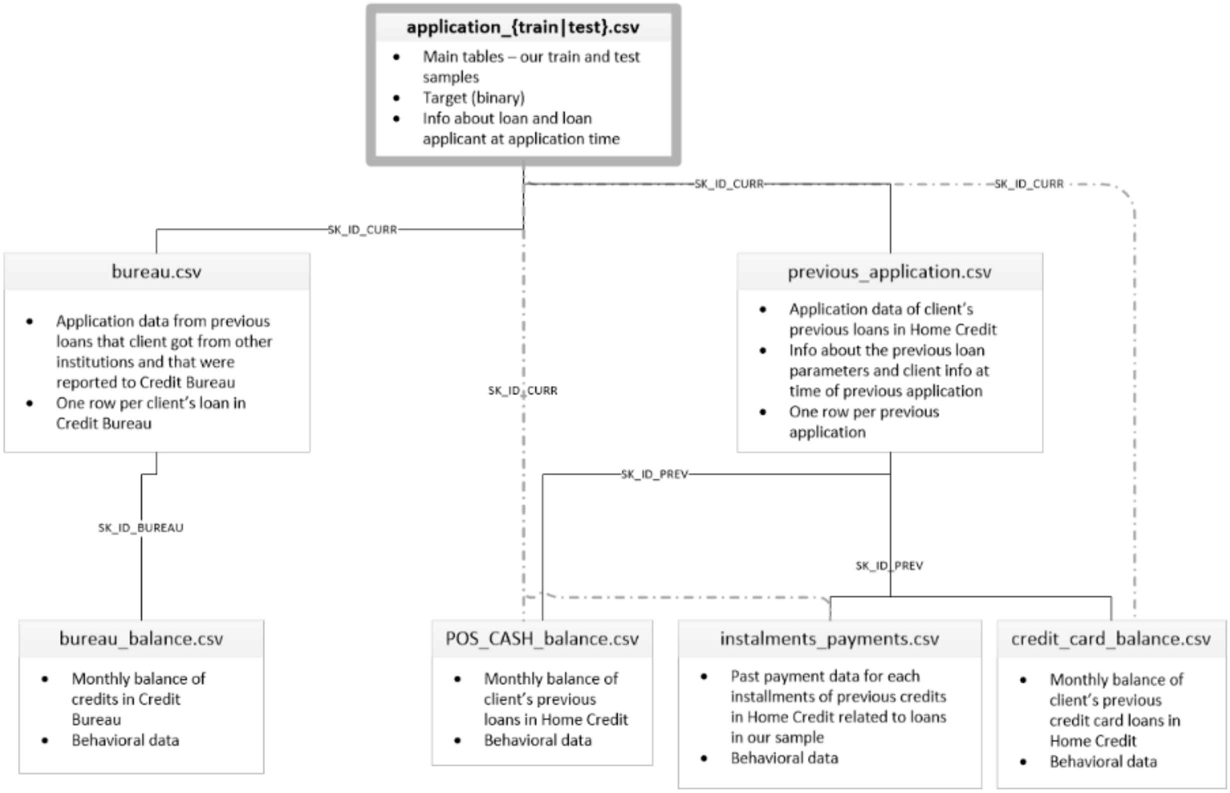
application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_PREV
SK_ID_BUREAU
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Figure 1- Description and connectivity of the Home Credit Default Risk dataset[1]

As seen in Figure 1, the file *application_{train|test}.csv* contains the main table containing the training dataset (307511 samples) and test dataset (48744 samples), with each row representing one loan identified by the feature SK_ID_CURR. The training set contains the variable TARGET with binary values (0: the loan was repaid or 1: the loan was not repaid).

While there are many input files available, due to time and computational constraints, I will be mostly using the main application training and test data files, which contain 121 possible input features and the target feature. The large number of input features and training samples will allow me to identify the important factors and for constructing a credit default risk classification model.

Solution Statement

After the exploratory data analysis and feature engineering, I will use the reduced number of features for training the classification model. Considering that the training dataset may be imbalanced, I will compare different classification methods like logistic regression, random forest, and gradient boosting for selecting the best model for this problem. I will also attempt hyperparameter tuning with K-fold cross-validation and methods like grid search, random search, or Bayesian optimization to improve the model results.

Benchmark Model

Since this is a Kaggle competition, the benchmark model for comparison will be the highest Leaderboard score (AUC score: 0.817)[1]. However, since I am not planning to use all the input features, this may not be possible. Instead, I will train a random forest classifier and use the score obtained on the test dataset as a benchmark to improve on. Nevertheless, my personal goal will be to reach the top 25% of the Kaggle Leaderboard.

Evaluation Metrics

A receiver operating characteristic (ROC) curve summarizes the performance of a binary classification model on the positive class where the x-axis shows False Positive Rate and the y-axis shows the True Positive Rate. A ROC curve does not have bias towards the majority or minority class, making them favorable when using imbalanced data with equal importance for the both classes[5]. The area under the ROC curve can be calculated to find a score between 0 and 1 (perfect model) for a classifier for all threshold values, called the ROC area under curve or AUC. Since the AUC is also used as the evaluation metric in the Kaggle competition, it will be the primary evaluation metric, but the precision and recall will also be calculated.

Project Design

The project will be implemented using different libraries in Python 3 in a Jupyter Notebook[6]. As with most machine learning projects, I expect the majority of the time will be used in the initial portion of data exploration and feature engineering, since it can massively affect the final results. The process of model comparison is expected to take the next-longest duration, followed by time needed for hyperparameter tuning of the final selected method. However, hyperparameter tuning (eg. grid search method) will need to be done with checks and balances to prevent excessively long computations.

After exploratory data analysis to check if there are any anomalous, missing, or duplicate values, I will focus on feature engineering. Moreover, if there are any categorical variables, I will use methods like one-hot encoding to convert them to numerical values. Since 121 input features will be very difficult for any method to handle, I will apply feature selection (eg. rank correlation coefficients between predictor and response features) or feature reduction (eg. principal component analysis) methods to decrease the number of input features. New features may also be constructed from the existing features by applying financial knowledge. While evaluating the different classifiers, I will try different libraries like XGBoost[7] and LightGBM[8] to improve the results and the training time. The model with the best results will be finally selected to use as a credit default risk classification model.

References

1. Home Credit Default Risk Competition (2018). Kaggle. https://www.kaggle.com/c/home-credit-default-risk/overview
2. Bagherpour, A. (2017). Predicting mortgage loan default with machine learning methods. University of California/Riverside.
3. Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. Journal of Banking & Finance, 34(11), 2767-2787.
4. Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018). Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. Electronic Commerce Research and Applications, 31, 24-39.
5. He, H., & Ma, Y. (Eds.). (2013). Imbalanced learning: foundations, algorithms, and applications. John Wiley & Sons
6. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., ... & Ivanov, P. (2016, May). Jupyter Notebooks-a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87-90).
7. Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785-794).
8. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Advances in neural information processing systems (pp. 3146-3154).