Home Credit Default Risk

Hemanth Dandu
Institute of Artificial Intelligence
University of Georgia
Athens, GA - 30606
hemanth.dandu@uga.edu

Sumer Singh
Institute of Artificial Intelligence
University of Georgia
Athens, GA - 30606
sumer.singh@uga.edu

Sushanth Kathirvelu
Department of Computer Science
University of Georgia
Athens, GA - 30606
sushanth.kathirvelu@uga.edu

Abstract

This project features two end-to-end approaches taken to solve the Kaggle challenge 'Home Credit Default Risk'. The first approach is using features created manually and the second approach is using an automated feature creation tool. The results of both these methods are compared and it is found that automated feature engineering can create superior features, in a shorter amount of time.

1 Introduction

Due to insufficient or non-existent credit history, many people struggle to get loans. These people are susceptible to exploitation from insincere lenders. Home Credit is striving to help these people at risk. To increase the effectiveness of their efforts, they turn to Kaggle. (1)

The goal of this challenge is to help these at-risk people get a safe loan, by using a variety of alternate data points. The data consists of various information about the applicants, which includes their monthly balances in their credit bureaus, point of sale information, telecommunication records, etc.

2 Data Description

The data is provided in the form of a relational database. There is one main table and six sub tables.

2.1 Main Table

The main table is split into two files, one for training - application_train.csv and one for testing - application_test.csv . Each row corresponds to one applicant. There are 308k applicants for training and 48k applicants for testing. The main table consists of 120 features.

2.2 Bureau Table

This table (bureau.csv) consists of every applicant's previous credits that were reported to Credit Bureau. Bureau Table consists of 1.72 million samples and 15 features. It is connected to the main table using the key 'SK_ID_CURR'

2.3 Bureau Balance Table

This table (bureau_balance.csv) consists of the monthly credit balances of credits reported to Credit Bureau. Bureau Balance Table consists of 27.3 million samples and 2 features. It is connected to Bureau Table with the key 'SK_ID_BUREAU'

2.4 Previous Application Table

This table (previous_application.csv) consists of all clients who have previously applied for Home Credit loans. Previous Application Table consists of 1.67 million samples and 35 features. It is connected to the main table using the key 'SK_ID_CURR'

2.5 Pos Cash Table

This table (POS_CASH_balance.csv) consists of the monthly POS (Point Of Sale) and Cash Loan balances the applicant had with Home Credit. Pos Cash Table consists of 10 million samples and 6 features. It is connected to the main table using the key 'SK_ID_CURR' and connected to Previous Application Table using the key 'SK_ID_PREV'.

2.6 Credit Card Table

This table (credit_card_balance.csv) consists of the monthly balance of previous credit cards that the applicant has with Home Credit. Credit Card Table consists of 3.84 million samples and 21 features. It is connected to the main table using the key 'SK_ID_CURR' and connected to Previous Application Table using the key 'SK_ID_PREV'.

2.7 Installments Table

This table (installments_payments_.csv) consists of the repayment history of previously disbursed credits that the applicant has with Home Credit. Installments Table consists of 13.6 million samples and 6 features. It is connected to the main table using the key 'SK_ID_CURR' and connected to Previous Application Table using the key 'SK_ID_PREV'.

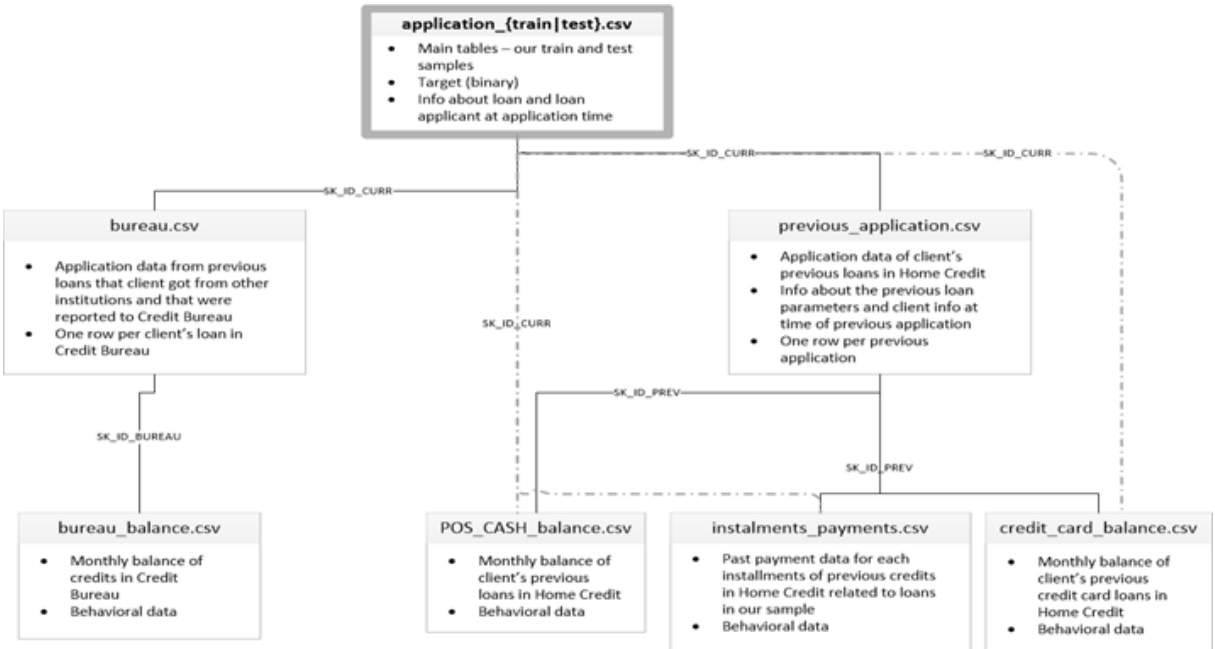

Figure 1: Data Model

3 Exploratory Data Analysis

To understand the data better, we explored the datasets by creating graphs from the train and test sets.

As a part of data exploration, we explore:

- Target Attribute Distribution.
- Number of Missing Values in each column for all the data tables and its Percentage.
- Single Attribute distributions.
- Distribution of Features Vs the Target.

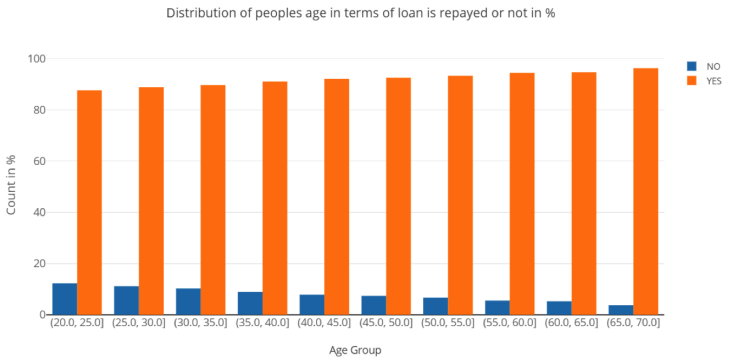

Figure 2: Age VS Loan Repaid

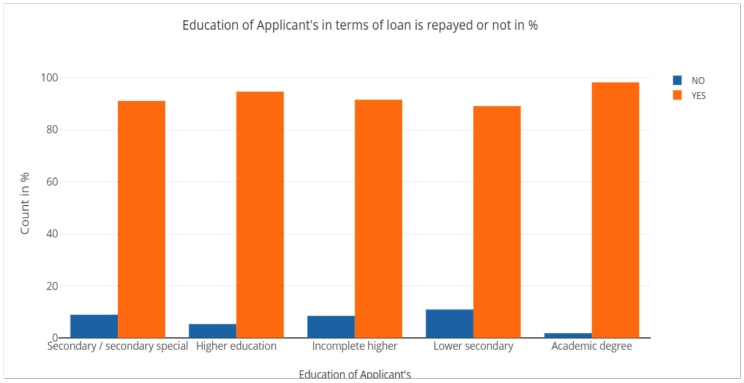

Figure 3: Education VS Loan Repaid

Figure 2 shows how many percent of people in a particular age group are capable of repaying the loan. We observe a clear trend - Older people pay their loan back at a higher rate than younger people.

Figure 3 shows how many percent of people are capable of repaying the loan based on their education. We observe that people with Academic Degrees pay their loan back at the highest rate.

We also find out that the training data set is Imbalanced with the Percentage of Persons Capable of Paying the Loan vs Not Capable of Paying the Loan as 91.9% and 8.07% respectively. This goes to show that even though these people are deemed 'high risk' by lenders, more than 91.9% pay back their loan.

4 Feature Engineering

Feature Engineering is the most critical part of our approach and consequently where we spend most of our time and effort. As the data is provided to us in the form of relational datasets, it falls upon us to create a feature matrix compatible with machine learning models. The approaches taken by us are:

- Manual Feature Engineering
- Automated Feature Engineering

4.1 Manual Feature Engineering

Manual feature engineering can be a tedious process and often relies on domain expertise.

Due to the limited domain knowledge of loans and what makes a person likely to default, we concentrate on getting as much info as possible into the final transformed data. The idea is that the model will then pick up on which features are important rather than us having to decide that.

The approach is to make as many features as possible and then give them all to the model to train on. Later, we can perform feature reduction using the feature importance from the model.

4.1.1 Method

- Convert all categorical features to dummies.
- Take min, max, mean, and var for all numerical features that need aggregation.
- Drop any constant columns.
- Add domain specific features based on literature, domain expertise etc.

4.1.2 Domain Specific Features

The following are the domain specific features that are identified and generated:

- Percentage of days employed
- Income credit percentage
- Income per person
- Annuity income percentage
- Payment rate
- Application credit percentage
- Payment percentage
- Payment difference
- No. of installments accounts
- No. of cash balance accounts
- No. of credit accounts
- No. of credit bureau accounts
- No. of previous applications

4.2 Automated Feature Engineering

"*Coming up with features is difficult, time-consuming, requires expert knowledge.*" - Andrew Ng.

As we did not have much time, nor expert financial knowledge, we use an automated feature engineering tool. The tool we use is Featuretools (2). Featuretools excels at creating features from relational datasets.

The crux of Featuretools are functions called Feature Primitives and an algorithm called Deep Feature Synthesis (DFS).

4.2.1 Feature Primitives

Feature Primitives are the main ingredient of Featuretools. They define functions that can be applied across or within datasets to create new features. There are two types of Feature Primitives:

- Aggregation primitives - They are applied across two related (parent - child) datasets. They take as input multiple values, and output a single value. Some Aggregation Primitives are "mean", "count" and "sum".
- Transform primitives - They are applied within one dataset. They take as input a column and output a new transformed column. Some Transform Primitives are "absolute", "year" and "hour".

4.2.2 Deep Feature Synthesis (DFS)

DFS is the workhorse of Featuretools. It is an algorithm which automatically creates new features. The new features created are specified by the Feature Primitives. In essence, DFS stacks (one or more) Feature Primitives to create new, meaningful features. The maximum number of Feature Primitives stacked is controlled by the "max_depth" parameter.

4.2.3 Feature Sets

We use two sets of Feature Primitives for DFS.

- Default primitives - ["sum", "std", "max", "skew", "min", "mean", "count", "percent_true", "n_unique", "mode"]
- Subset primitives - ["sum", "count", "min", "max", "mean", "mode"]

5 Feature Selection

After Feature Engineering, we end up with three sets of features - Manual Features (Manual), Automated features using subset primitives (Subset) and Automated features using default primitives (Default). Each of these sets contain a lot of useless features, which act as noise for our model. To reduce the number of features we use Random Forest(5). We determine the importance of each features, and keep only features with an importance higher than a threshold (we use a threshold of 0).

Table 1: Number of features before and after feature selection.

	Manual	Subset	Default
Original	727	1171	1984
After	513	879	1218

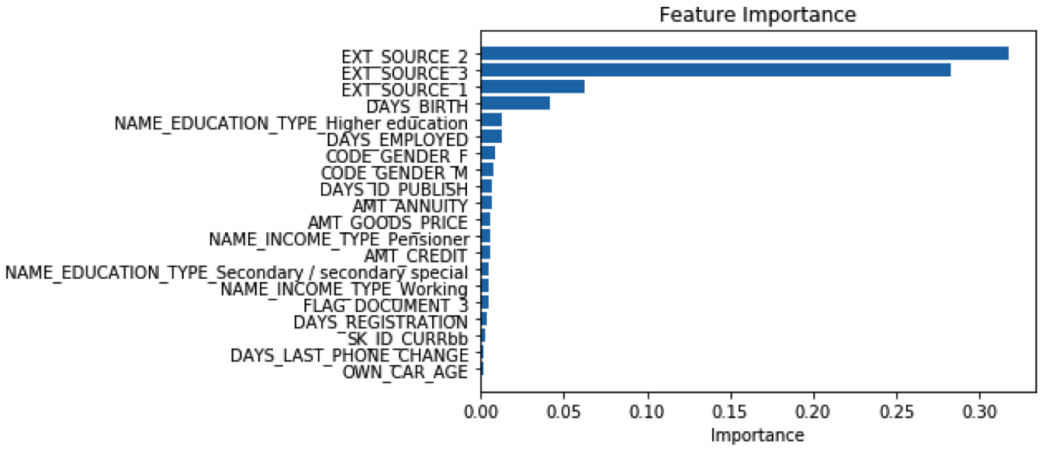

Figure 4: Feature Importance - Manual

Figure 4, 5 and 6 show the top 20 features created by Manual, Subset and Default, respectively. The x-axis is the feature importance score and should be interpreted in relative terms, not absolute.

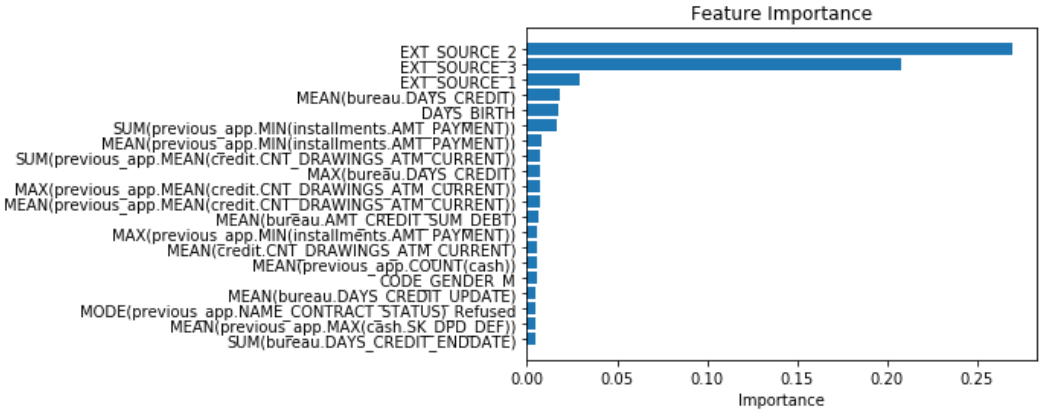

Figure 5: Feature Importance - Subset

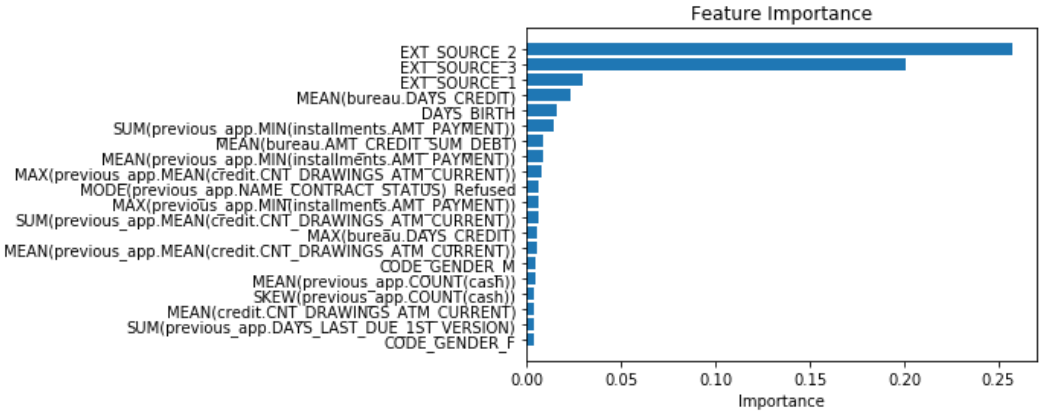

Figure 6: Feature Importance - Default

First, we see that automated feature engineering creates very informative features. 14 of the automatically created features are present in the top 20 features.

Next, we observe that using default primitives creates only 1 more top 20 feature, when compared to the feature's created using subset primitives. This suggests that using feature's created by subset primitives should suffice.

6 Experiments and Results

6.1 Models

6.1.1 Random Forest

Random Forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

This project uses Random forest for feature selection based on an importance threshold and to create the baseline model. (5)

6.1.2 Gradient Boosting Machine

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision

trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

This project uses lightgbm which is a highly efficient gradient boosting decision tree package. (4)

K-fold cross-validation is used to prevent lightgbm from overfitting.

6.2 Results

The metric used is Area under the Receiver Operating Characteristic (AUROC). It is a measure of how well a model can distinguish between two groups. The score measures discrimination, that is, the ability of the model to correctly classify the sample. A score of 1 (best) means the model is perfectly discriminating, while a score of 0.5 (worst) means a model is randomly guessing. (3).

Table 2: Results

Model	**Score (on 100)**
Random Forest, Manual Features, No feature selection (Baseline)	73.681
LightGBM, Manual Features, No feature selection	74.545
LightGBM, Manual Features with Domain features, Threshold = 0	76.216
LightGBM, Default automated features, Threshold = 0	77.048
Random Forest, Subset automated features, Threshold = 0	77.182
LightGBM, Subset automated features, Threshold = 0	77.362
LightGBM, Subset automated features, Threshold = 0.01	72.598
LightGBM, Subset automated features, Threshold = 0.001	**77.450**

6.3 Interpretation of Results

- We set our Baseline model as one using Random Forest to classify, with Manual Features and no feature selection. It achieves a score of 73.681.
- Switching to LightGBM gives an improvement of 0.864.
- A big jump of 1.671 is achieved once we add Domain Features and Feature selection (with a threshold of 0). This reinforces the fact that features engineering is one of the most important part in an ML pipeline.
- Using subset primitives gives a slightly higher score than default primitives. This could be because using default primitives creates a lot of unimportant noisy features.
- Using subset primitives and a threshold of 0.01 gives a score of 72.598, which is lower than our baseline. This threshold proves to be too high and is destroying several important features.
- Lowering the threshold to 0.001 gives our best score of 77.450. This threshold seems to keep the most optimal features and removes the noisy ones.

7 Conclusion

In this project we express the importance of feature engineering. In many cases, it can prove more effective to spend time creating useful features, instead of just focusing on model tuning. Without much model tuning, we achieve a score of 77.450, which is just 3.12 points below the top score of 80.570.

Secondly, we demonstrate the usefulness of automated feature engineering. Without expert knowledge it is almost impossible to create highly effective features by hand. A package such as Featuretools, helps us create features we would have missed. It is easy to use and highly beneficial.

8 Acknowledgments

- This project was completed as a part of the Data Science Practicum 2019 course at the University of Georgia.

- Dr. Shannon Quinn for unparalleled support through the course of the semester.
- Kaggle and Home Credit for providing us with the data sets.

.

References

[1] Home Credit. (2018) Home Credit Default Risk,
https://www.kaggle.com/c/home-credit-default-risk

[2] JM Kanter. (2015) Deep Feature Synthesis: Towards Automating Data Science Endeavors
www.jmaxkanter.com/static/papers/DSAA_DSM_2015.pdf

[3] Tom Fawcett. (2006) An Introduction to ROC Analysis
https://www.sciencedirect.com/science/article/pii/S016786550500303X

[4] Guolin Ke, Qi Meng, Thomas Finley. (2017) LightGBM: A Highly Efficient Gradient Boosting Decision Tree,
http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradi

[5] Liaw, Andy Wiener, Matthew. (2001) Classification and Regression by RandomForest,
cogns.northwestern.edu/cbmg/LiawAndWiener2002.pdf