Machine Learning Engineer Nanodegree

Capstone Project

Ashique Mahmood

August 17, 2018

I. Definition

Project Overview

Credit history plays an important role in deciding the fate of a consumer's loan application. However, a vast majority of the population do not have extensive credit history and are thus often in a position of disadvantage. Home Credit, a global loan provider, launched a competition on Kaggle to help solve this problem. This project aims to utilize the datasets from the competition to build a predictive model, capable of predicting the credit worthiness of consumers.

Problem Statement

This is a classification problem, where the intended result is to be able to predict if a consumer applying for loan will be able to repay it on time or will delay the repayment. In order to do so, a machine learning model will be used to learn from the available data and predict on new unseen data of consumers. The final result of the model is evaluated on Kaggle by submitting the model's predictions on the test dataset.

The strategy used to solve the problem would involve the following steps:

1. Data exploration – explore and understand the available data
2. Data processing – process the data to be used by a machine learning model
3. Baseline modelling – set benchmark for evaluating against using machine learning models
4. Solution modelling – develop an improved model
5. Refinement – optimize solution model to improve performance
6. Model evaluation and validation – evaluation of the model to ensure solution of the stated problem

Metrics

The model performance will be measured using the area under the ROC curve between the predicted probability and the observed target. The ROC curve is a plot of the true positive rate against the false positive rate. Since the problem asks us to predict the probability of the class, using these two metrics, the ROC curve helps us understand how well the classifier separates the two classes. This is the score used on Kaggle to evaluate the performance as well.

II. Analysis

Data Exploration

A total of seven datasets are available to us for training our model. The relationship between the different datasets is illustrated by the diagram below:

Main Loan Applications Datasets:

This is the main dataset for the project, consisting of static data for all loan applications. The training set consists of a label, TARGET, where 1 represents client with payment difficulties and 0 for those without. We have 120 features in this dataset, with 307,511 observations, each with a unique ID. This ID is used later to join information from the other datasets to the main dataset. The feature space covers essential key information for loan applications such as the amount of credit for the loan, annual income of the client, annuity of the loan. It also includes detailed information about the client's housing, the documents provided by the client during the application etc.

Bureau and Bureau Balance Datasets:

The bureau dataset consists of loan application details of the client’s previous loans that they received from other institutions which were reported to the Credit Bureau. We have numerical features here such as the amount of credit, the debt, amount overdue and categorical features as well, such as the type of loan. Each loan application is identified with a unique ID, SK_ID_BUREAU. The bureau balance dataset gives us more details about each of the loans in the bureau dataset. This dataset gives us the monthly balance for the loans specified by the SK_ID_BUREAU in the bureau dataset and the status of the loan with categorical values such as Active, Closed, unknown or the binned amount of days past due.

Previous Loan Applications Datasets:

This dataset consists of all previous applications for Home Credit loans of clients who have loans in main dataset. There is one row for each previous application related to loans in main dataset. Each unique loan is identified with an ID, SK_ID_PREV. This dataset consists x features, consisting of loan amount, annuity, down payment and client type to name a few.

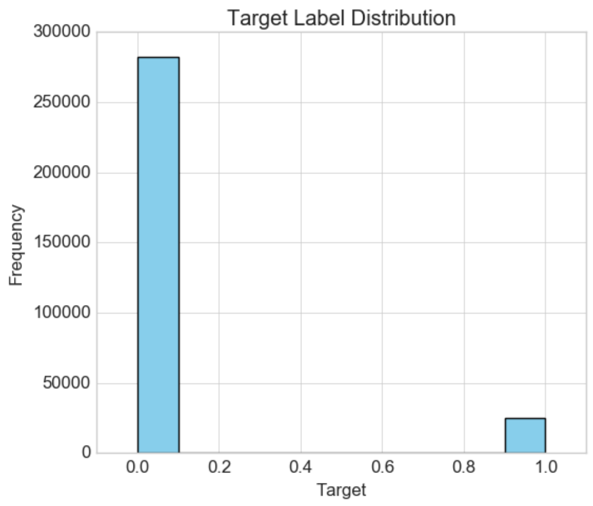
POS Cash Balance, Credit Card Balance and Installments Payments Datasets:

These three datasets give us more details about the loans in the previous application dataset. We have the monthly credit card balance, number of drawings from the ATM, installments left to pay, number of installments to name a few.

Summary of the Datasets

The following table summarizes the total number of features and their data types available to us to learn from across all the datasets:

Dataset	No. of Features	Observations	Categorical	Numerical
application_train	120	307511	16	104
bureau	15	1716428	3	12
bureau_balance	2	27299925	1	1
POS_CASH_balance	6	10001358	1	5
credit_card_balance	20	3840312	1	19
installments_payments	6	13605401	0	6
previous_application	35	1670214	16	19
Total	204	58,441,149	38	166



The class distribution of the training data is imbalanced, as can be seen from the plot above. This poses a challenge for some models due to having a significantly higher number of observations to learn to predict one thing over the other.

Missing Values:

A large number of features in the dataset had missing values, some as much as 74.0%. The baseline models were evaluated with dropping features which had more than 20% of observations missing.

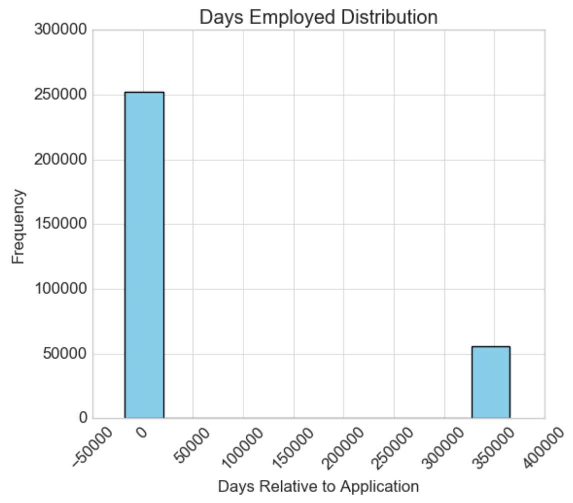
Descriptive statistics of domain features:

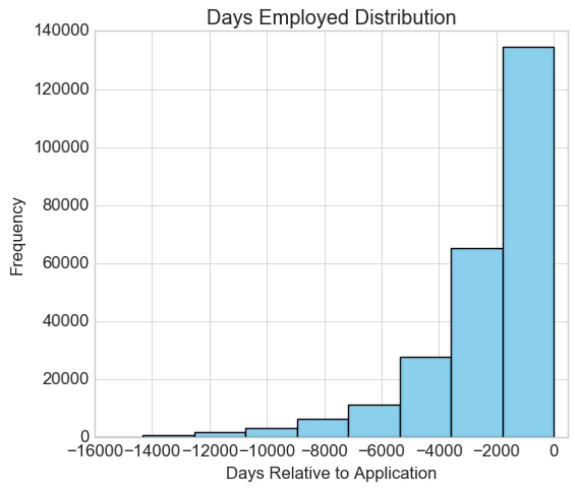
Feature	Mean	Std. Deviation	Min	Max
Credit Amount	599,026	402,490	45,000	4,050,000
Annuity	27,108.5	14,493.7	1,615.5	258,025.5
Amounts Goods Price	538,396	369,446	40,500	4,050,000

Income	168,797	237,123	25,650	117,000,000

The amount of credit, annuity, value of the goods for which the loan was taken, and income of the client are all very important questions to ask for loan applications. Looking at the descriptive statistics for these, we learn that amount of credit for the loans are very widely spread out, with a high standard deviation. The amount of goods price follows the same trend.

Abnormalities:

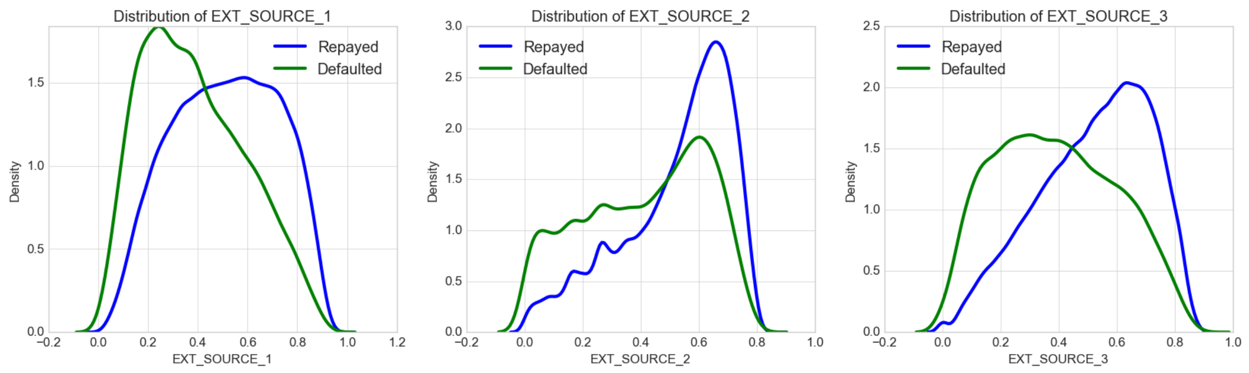




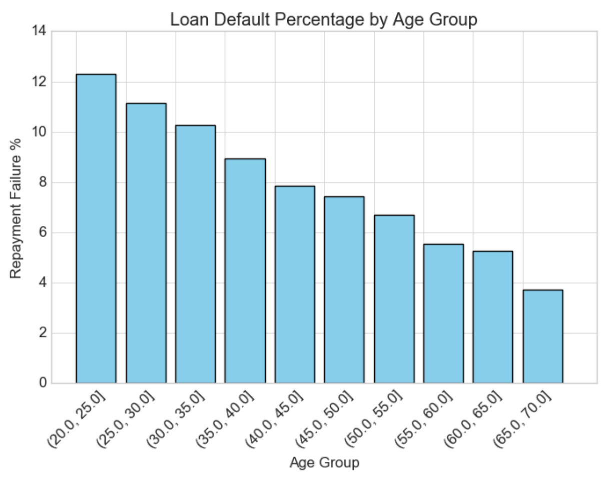
Looking at the distribution, there's a huge group of people whose employed age is 365243 days. This is clearly an anomaly, as no one would be working for a thousand years. The figures above compares the distributions before and after the removal of the anomalies.

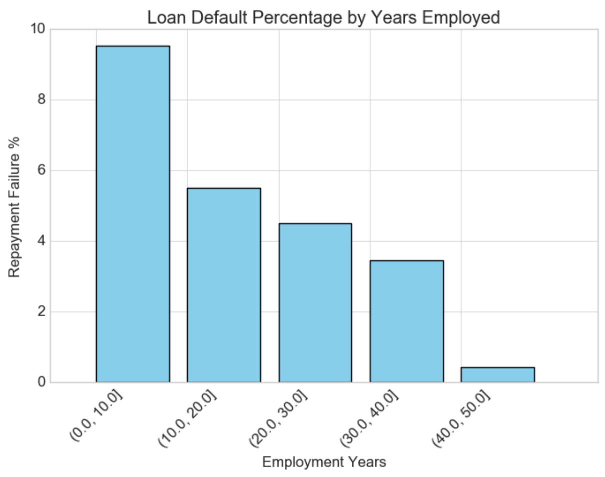
Five other features from the previous applications dataset was found to have the same number 365243 in multiple occasions. They were replaced with nan, to be treated as missing values.

Exploratory Visualization



Amongst the highly correlated features, the top three are the EXT_SOURCE 1,2,3 score features. These normalized scores are a credit rating in the banks sample coming from some external sources. When we look at the distribution of these scores, they show clear separate trends for the two classes. Clients who are likely to repay their loans tend to have higher scores across the three features. The difference in the means between the two classes are a good indication. Especially with EXT_SOURCE_3, the scores for clients who repaid their loans on time center around 0.7 whereas scores for those who faced difficulty with repayment center around a much lower value of 0.3. A mean of the three could therefore be an important feature in predicting the repayment capabilities of clients.





The bar plots above show us the percentage of loan repayment failure in terms of age groups and years employed groups. Younger clients, in the age group of 20 to 25, appears to default the most on their loans. The risk of defaulting decreases gradually with age.

When it comes to the years employed, we see a similar trend. Clients with less number of working years under their belt appears to contribute the most to loan repayment failures. Clients with a hefty working experience of 40 to 50 years are the group least likely to have struggles repaying their loans.

Algorithms and Techniques

We are required to predict the probability of the consumer belonging in one of the two classes, repayment of loan on time or defaulting. Earlier research by Khandani et al 2010 makes use of a generalized classification and regression trees (CART)-like algorithm to solve the credit risk problem. More recent research by Charpignon et al has shown that advanced tree based algorithms such as Random Forests and GBTs outperform CART and logit models previously used. This would be evidenced by the benchmark stage of the project as well.

The solution model is a gradient boosted tree based model. It is an ensemble tree based algorithm which uses boosting instead of bagging to combine many trees. Unlike random forest where we average the outcome of separately built independent trees, in gradient boosting, the trees are built sequentially, each focusing on the previous one's weakness.

Stratified K-fold cross validation was used during training and prediction for all models. Stratified k-fold ensures that each fold has equal number of classes for the model to learn from. Given the class imbalance in our dataset, this is very essential. To prevent the model from over-fitting on the training data set, the model is trained on different parts of the training set and part of the training set is used for validation.

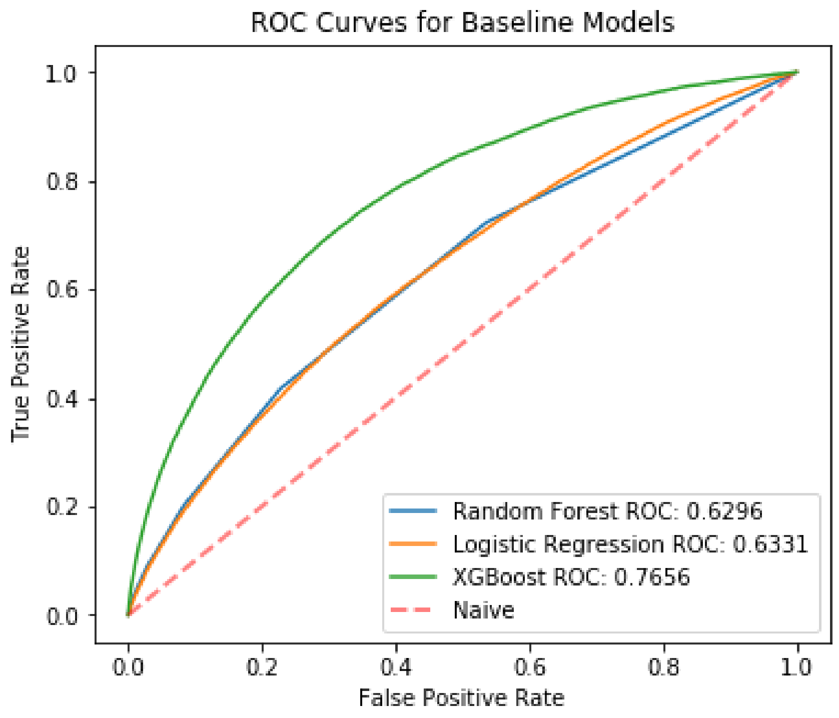
A random search algorithm was implemented to find the optimum set of hyperparameters for the model. Given the size of the dataset and the hyperparameter space for the model, performing grid search would be extremely computationally expensive. Even with random search, using the entire dataset would not be feasible. A small portion, 13% of the entire dataset, was used to perform random search.

Early stopping was used to find the best number of estimators. With early stopping, the training continues until the model's validation score does not improve for a specific number of rounds. The best iteration was then picked to predict on the test set.

Since the problem asks us to predict the probability of the class, using these two metrics, the ROC curve helps us understand how well the classifier separates the two classes. Thus, all the models were evaluated using this metric.

Benchmark

Since this is a classification problem, a naive benchmark score would be would be 0.5. Three models were picked to be tested as our baseline. One ensemble tree based model: random forest, a gradient boosted tree based model and a logistic regression classifier. Features with more than 20% of the data missing was dropped during this benchmarking and missing values were imputed using the mean.



The following table summarizes the benchmark scores from the models:

Model	*Training Score*	*CV Score*	*Test Score*	*Training Time*
Random Forest	0.9997	0.6296	0.6250	4.11
Logistic Regression	0.6354	0.6331	0.6290	37.37
XGBoost	0.7783	0.7656	0.7540	41.38

The random forest model scores the lowest and suffers from overfitting. The logistic regression fares much better. As expected, the gradient boosted model achieves the highest validation and test set score. However, it took the longest amount of time to train. The highest score from the three models would be used as the benchmark for our solution.

III. Methodology

Data Preprocessing

Aggregating and joining the datasets:

One of the challenges for this project is to use the information from all the datasets. This involves extracting information from the datasets in a meaningful way and joining them to the main dataset. The following strategy was used for this purpose:

Aggregate statistics computed:

- Numerical: ['mean','max','min','count','sum']
- Categorical: ['count','mean']

The motivation behind this was to capture the information in a way that adds value. For example, the mean amount of debt across a client's loans in the bureau, sum of the days the client delayed repayment over all the loans, the maximum amount of times a client prolonged the credit for the loans, or the total number of times the client's loans were closed. Initially an attempt was made to specify the type of aggregate statistics to compute for each feature. This strategy was not efficient at scaling to the massive number of features across all the datasets and was limited by my own imagination. Hence, later it was decided to compute all five for all numerical features and two for all the categorical features. Functions were developed for computing these and reused to aggregate data from each dataset.

- Bureau and Bureau Balance: the features in the bureau balance data was aggregated based on each loan using SK_ID_BUREAU. This was further aggregated for each client using their id in the main application dataset and joined to the main dataset.
- POS, Credit Card Data, Installments and Previous Applications: the features for the first three datasets were aggregated by their corresponding previous applications ID and then further aggregated by the current client ID and joined to the main dataset. The data for the previous applications dataset was aggregated by the current client ID and joined to the main dataset.

Encoding Categorical Features:

Out of the 57 categorical features across all the datasets, 10 of them had only two classes. One hot encoding creates a separate feature for each unique category of a feature. When it comes to features with only two unique categories, a simpler solution is to encode each category with a unique number. This helps to prevent the feature space from expanding beyond necessary. While this method works perfectly well for features with more than two unique classes, we would have to sacrifice model interpretability, as we won't know which number is assigned to which label. Hence, categorical features were encoded using one hot encoding where there were more than two categories and label encoding for features with two or less categories. A separate categorical feature was created for each categorical feature containing missing values during the encoding.

Missing Values:

In the aggregated dataset, there were a total of 1260 features missing values. Out of 1800+ features, 1260 of them had missing values, with some features missing as high as 74% of the observations. Features missing a high percentage of values are less likely to add any value. The benchmark models, logistic regression and random forest classifier, both requires data without missing value. Thus, the missing values were imputed using the mean.

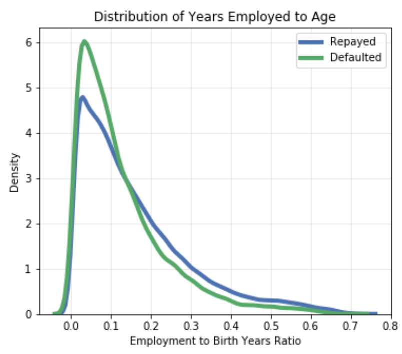
The final solution, a gradient boosted machine, handles missing values without the need for imputation.

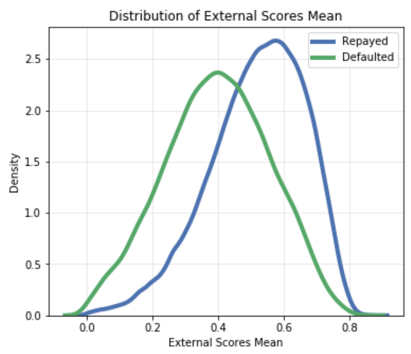
Anomalies:

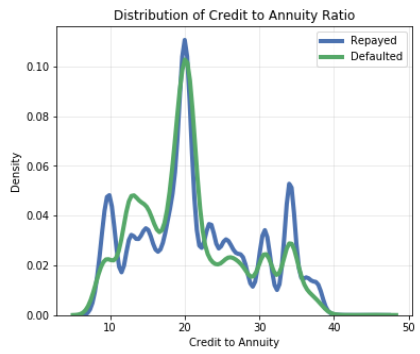
Anomalies found in the previous section, from the applications and previous applications datasets, were replaced with nan to be treated as missing values.

Feature Engineering

One of the great things of using a tree based model as a baseline is being able to get a sense of the feature importance. As indicated by our initial data analysis, the external source scores played an important role in helping the model classify the clients. Days employed, the age of the client, amount of annuity and credit all appear in the top most important features. Using this information, the following new features were created to help us better model the data:







External Sources Features: given how important these scores are, two new features were created combining the three scores – a mean of the three scores and a product of the three scores

Debt to income ratio: lending institutions often use the debt-to-income ratio to judge the credit worthiness of clients. The amount of money you owe compared to the amount of money you're bringing in helps them understand how likely you can pay back your loans. This was the motivation behind creating the annuity-to-income ratio feature.

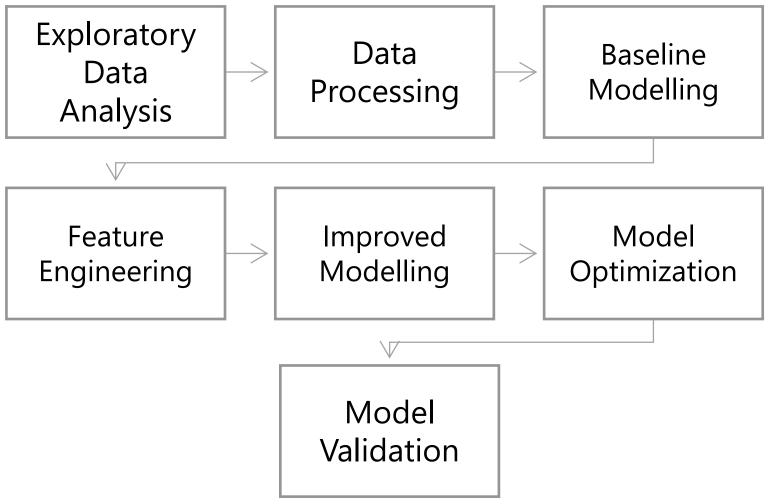
Credit-term: another useful feature would be the length of the loan payment. This can be calculated as the ratio between credit and the annuity.

Credit-to-income: the amount of money borrowed compared to the amount of money the client earns is another good financial indicator. Two new features were created with this motivation, one with the credit amount of the loan another with the price of the goods against which the money borrowed.

Regional Rating: the institutions rating of the client's residence appears in the list of important features. A mean of two such ratings were used to create a new feature.

Implementation

The solution implementation can be summarized in the stages below:



The solution was implemented using python as the programming language, using various open source libraries for data processing, modelling and visualizations. The pandas library, built on top of numpy, was used for data processing and manipulation. Functions for processing the datasets and joining them together were developed and modularized into a class. The matplotlib and seaborn libraries were used for visualizations.

The model used for the solution is an implementation of gradient boosted decision trees. This was implemented using an open source library developed by Microsoft, called LightGBM. The library promises higher performance in terms of speed compared to the XGBoost library used earlier for benchmarking.

Refinement

Adding more engineered features improved both the validation and test score. The effect of including the features in our model is summarized in the table below:

	CV Score	*Training Score*	*Test Score*
Without Engineered Features	0.7807	0.8331	0.775
With Engineered Features	0.7823	0.8363	0.782

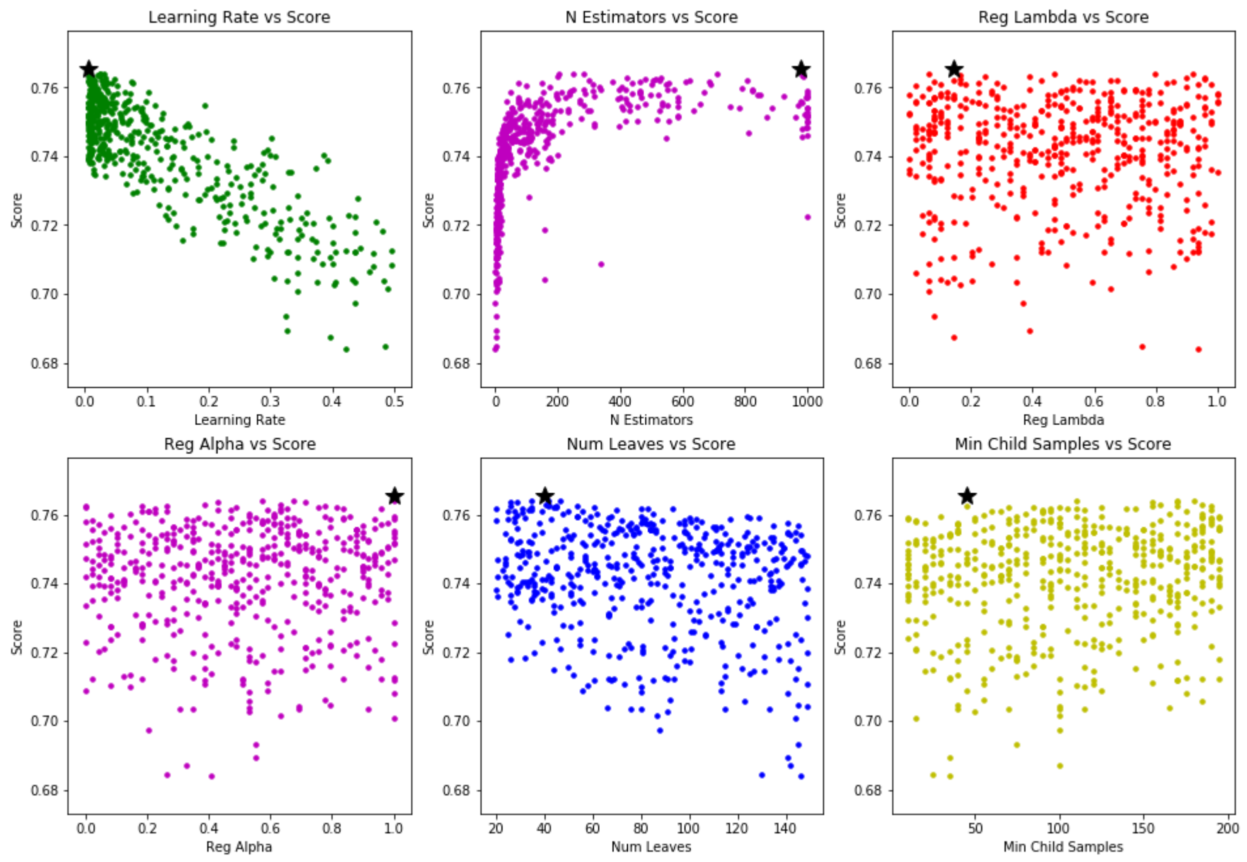
Hyperparameter Optimization

Performance of a machine learning model is always dependent on the model’s design. Hyperparameters let us tune the model to our needs. The model of choice here has multiple hyperparameters that control model complexity and performance. Light GBM has several important hyperparameters that affect model performance. Notable ones are:

n_estimators: the number of boosted trees to be used. This is perhaps the most important parameter for the model. Unnecessarily high number of sequentially boosted trees would tend to overfit the model as it tries to memorize the dataset. Too less would give us an underfitted model.

learning_rate: this provides us with another way of controlling overfitting. The learning rate determines growth rate of the boosted trees, deciding when to add new trees.

Random search was run for 500 iterations using 40,000 observations, which makes up 13% of the entire dataset. The following scatter plots illustrate the relationship between the hyperparameters and the cross-validation scores. Although these do not fully reflect the effect of changing each individual hyperparameter on the score since they were not changed one at a time, it still helps us visualize any possible trends.



There is a visible trend in the learning rate and the number of estimators. There is a trade-off between the number of trees and the learning rate. The lower shrinkage rate acts as a natural way of regularization but requires a higher number of trees ultimately. This leads to longer training times, requiring more computational power. But this eventually leads to a more robust model which does not suffer from high variance or bias. The number of leaves and the minimum samples required for each leaf are both kept to a low number, reducing the risk of overfitting.

The random search was performed on a small portion of the data. Thus, when translated to the entire dataset, the gain in performance was not significant. But the insights gained from the random search helped fine tune the final model's parameters. For the final set of hyperparameters, early stopping with 100 rounds was used to find the optimum number of trees, which resulted in 10,000 estimators. Given the magnitudes of increase in the trees, the learning rate needed to be increased and through an iterative process, the best results were found with a learning rate of 0.02.

The final parameters of the model are summarized below:

Hyperparameter	**Value**
boosting_type	gbdt
colsample_bytree	0.5
'learning_rate'	0.02
'min_child_samples'	45
'min_child_weight'	0.001
'n_estimators'	10,000
'num_leaves'	40
'reg_alpha'	1
'reg_lambda'	0.142857
'subsample'	0.915152
'subsample_for_bin'	200000
'is_unbalance'	True

The following table shows the difference in performance between before and after tuning the model:

	CV Score	*Training Score*	*Test Score*
Unoptimized Model	0.7823	0.8331	0.782
Optimized Model	0.7895	0.8363	0.785

IV. Results

Model Evaluation and Validation

From the three types of models tested during the benchmarking stage, gradient boosted tree based models showed the most amount of promise. Consequently, the solution model is also an implementation of a gradient boosted classifier.

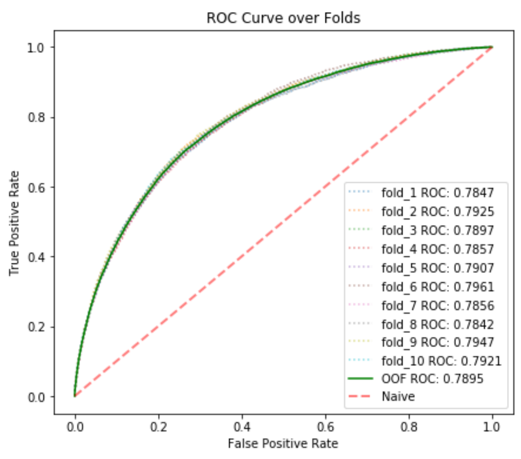
The following table summarizes some key metrics of the final model.

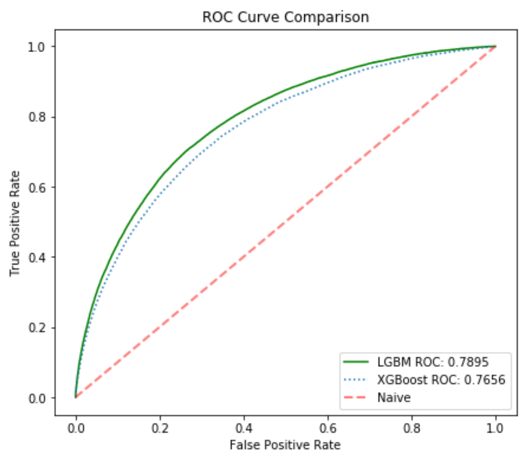
Class	*Precision*	*Recall*	*f1 score*	*Support*
Repaid	0.96	0.76	0.85	282686
Defaulted	0.20	0.68	0.30	24825
Avg/total	0.90	0.75	0.80	307511

Our goal is to be able to detect as many potential loan defaulters as possible. The results from the model could be used to launch further investigation to ultimately decide the level of risk for the loan. The model has an overall high recall rate, which serves our purpose of flagging risky loan applications. The model would perform as an excellent first barrier to risk analysis. Ideally, we would want a high recall rate for defaulters if the model is to be used as an initial risk assessment tool. The solution achieves this with a relatively high score of 0.68 compared to the precision score of 0.20. The high precision for clients who are likely to repay the loan on time is also beneficial towards this goal. However, this model should not be used as the final decision maker, due to the low precision rate for those who are likely to default.

		Prediction	
		Repaid	**Defaulted**
True	**Repaid**	213,470	69,216
	Defaulted	7,932	16,893

The prediction results are summarized in the easily interpretable confusion matrix. One of the key thing is to note here is that regardless of the imbalance between the classes, the model faired quite well at identifying client who has potential to default on their loans.





The figure above shows the receiver operator characteristic curve for the solution and the baseline model. This helps us visualize the gain in performance achieved with the solution. It is interesting to note how the ROC curve for the solution model across the folds remained similar, further indicating stability of the solution.

To test the generalization of the model, it was trained on only the features which accounted for the top 90% of the importance. This lead to a small increase in the cross-validation score, from **0.7895** to **0.7896**, and a very marginal decrease of 0.006 in the test score,

The model was further trained on data with the top 3 most important features removed. This removed the features credit to annuity ratio, mean of the external credit scores and the amount of annuity. All three of these features are extremely important for the domain of this problem. Regardless, there was not a significant decrease in performance of the model. The model achieved a cross validation score of 0.7815, down from 0.7895, a marginal loss, which indicates the robustness of the model.

Justification

The table below illustrates the difference in performance between the baseline model and the solution model.

Model	*Training Score*	*Validation Score*	*Test Score*
Baseline	0.7783	0.7656	0.7540
Solution	0.8671	0.7895	0.7850

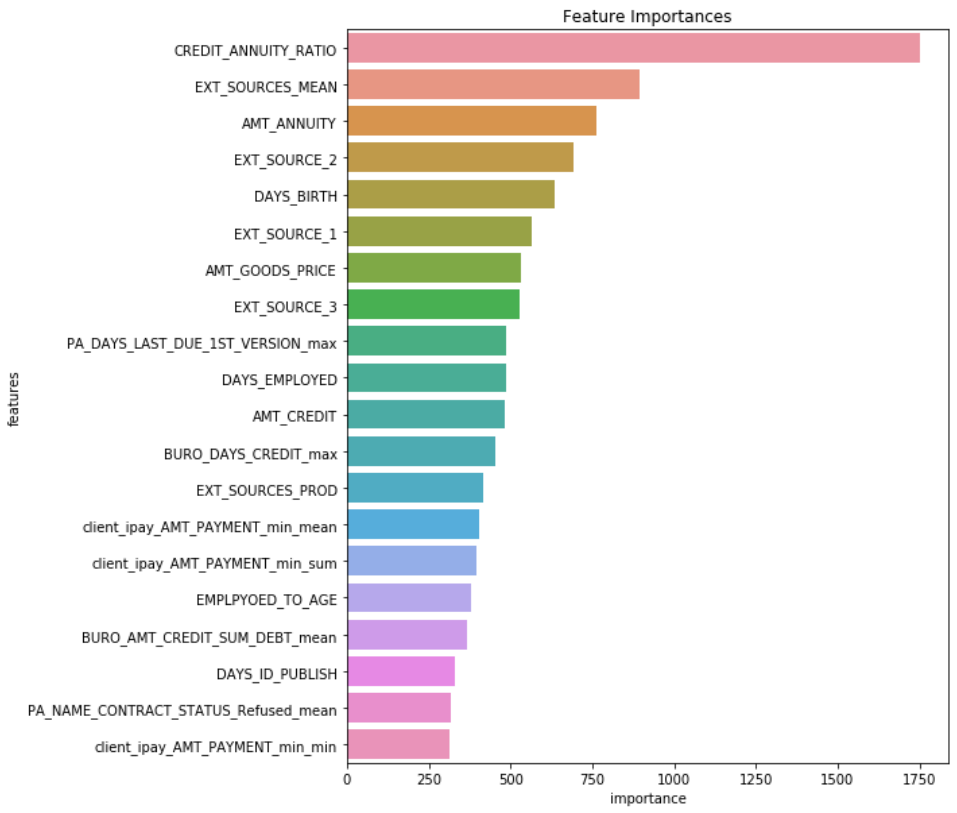
The solution model achieved a 2.39 percent higher score on the validation set and a 3.1% higher score on the test set. According to Kaggle's policy, the test set is scored with a portion of the full dataset.

To summarize, the developed solution is stable, generalizes well and is not susceptible to small permutations in the training dataset. It provides great performance gains over the baseline and is ready to add a significant amount of value in the consumer credit industry.

V. Conclusion

Free-Form Visualization

One of the interesting aspects of the project was feature engineering. Besides choosing the right model and hyperparameters, the biggest gain in performance came from feature engineering, as documented in the refinement section. The plot below illustrates the top 20 features.



The credit to annuity ratio, which is essentially the credit term, tops the chart. Luckily, this is easily translatable to the domain of the problem. The agreed length of repayment for a loan should be used for learning more about consumer credit behavior.

The credit scores from three external sources also heavily influence the classification. This illustrates how important credit scores are and financial institutions should devote resources to calculate and obtain credit scores for clients from using data from reliable sources.

Reflection

The whole process of the problem solution can be summarized in the steps below:

1. Problem identification – the goal was to find a novel problem that machine learning can aid in helping a large number of people. The Home Credit Default challenge on Kaggle resonated well with this.
2. Research – this stage involved learning about the domain of the problem and common techniques used to solve it.
3. Data exploration and analysis: exploring all the datasets learning about the data
4. Data processing: aggregating the data and processing it to be suitable for modelling
5. Benchmarking: setting a benchmark to evaluate final solution against
6. Improved modelling: building a model to outperform the benchmark in an attempt to solve the classification problem
7. Optimizing the solution: this involved feature selection, hyperparameter optimization and engineering features out of the existing ones
8. Testing and validation: analyzing the solution to ensure the integrity of it and how well it actually solves the defined problem

Personally, I found the exploration, optimization and testing of the solution very interesting. Visualizing the data to learn and make assumptions about the problem was exciting. Seeing your efforts leading to a better solution was the most satisfying aspect of the optimization process. Testing and validating the solution to learn about the rigidity and robustness of was also exciting.

The data processing step was quite challenging. Aggregating them, dealing with memory errors and missing values were all quite troublesome.

The final solution does show good performance and reliability and would definitely aid in assessing the credit worthiness of loan applicants.

Improvement

One of the aspects of the solution that can benefit from further optimization would be the searching of the best hyperparameters. In my solution, I have implemented a random search algorithm, using only 13% of the available data. This took a huge amount of time to implement on a virtual machine with 24 CPU cores and 48GB of RAM. The effectiveness of this search translated to the entire dataset has thus been marginal. This illustrates the issues with scaling this method on a much larger dataset. Random search is an uninformed method for searching the best hyperparameters, meaning that it does not use the information gained from trying out the different combinations. An informed method such as Bayesian hyperparameter optimization would narrow down the search space to promising combinations, using previous experience. A solution which uses the entire dataset to perform an informed hyperparameter search would definitely yield better results.

I have found the feature engineering and selection aspects of the project to be mainly empirical. This illustrates the need of domain expertise when solving a machine learning problem and also poses the problem with scaling such solutions. Automated feature discovery and selection would enable the deployment of machine learning solutions at scale for similar problems.