Home Credit Default Risk

Capstone Project

Omar Alqasem
November 3rd, 2018

I. Definition

Project Overview

Credit risk is the risk of not meeting the legal obligations from a borrower not making the payments required. Banks uses strict credit risk management to minimize lending borrowers who fail to repay the obligated payments upon agreed terms based on applicant credit history. Therefore, many applicants with insufficient or non-existing credit history struggle to get loans. As a result, these applicants are taken advantage of by untrustworthy lenders.

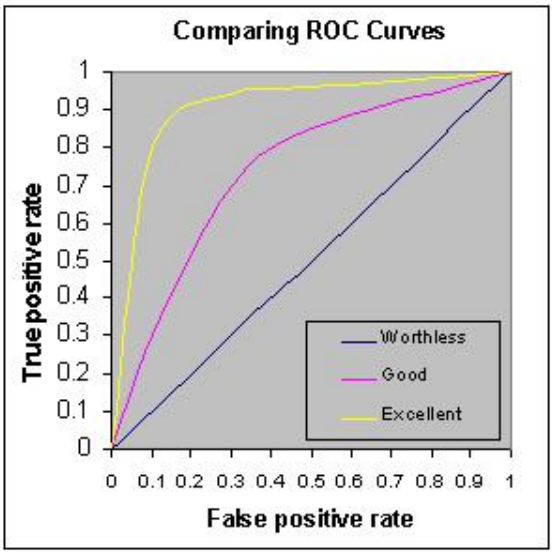
Home Credit Group (HCG) goal is to help applicants with non-existing or insufficient credit history with cash or revolving loans. The project solution shall help HCG to take the right decision whether to give a loan to an applicant based on their application information. The input data of applicants' information has been given to the public in a competition hosted by Kaggle to select the best solution with the highest evaluation score. This project will use the same dataset given for the competition.

Problem Statement

Home Credit Group's (HCG) goal is to open their doors for applicants with insufficient or non-existing credit history to apply for loans and receive a positive loan experience. At the same time, HCG wants to eliminate applicants who fail to repay their obligated payments. Therefore, in order to help eliminating rejections for capable borrowers of repaying their loans, Machine Learning will be applied to this problem to predict new applicants' ability to repay their loans. The process will involvr many steps such as (1) data exploration to gain deep understanding about the dataset, (2) data preprocessing to prepare the dataset for model training, (3) model training on classification algorithms e.g. Logistic Regression, AdaBoostClassifier, XGBoost, LightGBM, (4) model selection based on highest evaluation metrics and finally further improvement through hyper-parameter tuning. The dataset of past applicants is taken from Kaggle website that's part of a previous competitions. The final trained model shall classify new applicants' ability to repay for the request loan.

Metrics

The dataset given by Home Credit Group (HCG) is an imbalanced dataset. This means that the dataset contains much more records applicants that were capable to repay the loan than applicants whom defaulted. Therefore, accuracy measure shall not be reliable to evaluate the classification models since the majority of the dataset belongs to only one class. In this project, we will mainly use Area Under the Receiver Operating Characteristic Curve (ROC AUC) score for evaluating the model performance since it calculates the sensitivity and specificity across a continuum of cutoffs. To calculate the AUC score of a model, an ROC curve must be drawn which is produced by thresholding the model's predicted probabilities at various places between zero and one, and taking the points scored above the threshold as predicted members of the positive class, and those below the threshold as predicted negative classes. Note that that the true positive rate of a thresholding refers to the proportion of positive classes correctly predicted while the false positive rate of a thresholding refers to the proportion of negative classes incorrectly predicted.



The graph above shows an example of an ROC curve with three plotted lines excellent, good and worthless plotted. Note that an area of 1 means a perfect test result, while 0.5 or below means a poor test result.

II. Analysis

Data Exploration

The dataset used for this project has been taken from one of Kaggle's past competitions in CSV format. It constructs of 122 features and over 300K records.

These records represent historical loan applications and their target labels of whether the applicant has repaid the full loan installments. Like any other real-world scenario, the number of positive class (default applicants) weighs only 8% of the dataset and the rest is marked as negative class (paid successfully). To get a sense of how the dataset looks like, Table II.I below shows the first 5 records of the dataset along.

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRED
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000

Table II.I: First 5 records of Home Credit Default Risk taken from Kaggle competitions

The dataset contains many features related to the loan application and applicant's personal information. Features related to the loan application include loan type whether cash or revolving loan, credit total requested by client, final credit total given for client, down payment …etc. Likewise, features related to applicant's personal information include gender, number of children, education level, family status, duration of employment …etc.

During the data exploration, it has been encountered some abnormalities within the dataset. As an example, one of the applicant's employment duration exceeded 1000 years. Likewise, another applicant's gender is marked as XNA. These abnormalities have been mutated accordingly. Furthermore, the dataset suffers from lots of missing values for some features where they exceeded 60% of the dataset missing. Those missing features have been addressed to mutate with the average values. Finally, categorical variables do exist in the dataset and they have been one-hot encoded for the classification models to understand.

Exploratory Visualization

It's a common practice to visualize some of the characteristics in the dataset used for any machine learning problem in order to make deep understanding of the dataset. As for this project, we begin with the question, how many applications have paid their loan successfully. Figure 2.1 shows that more than 250K loan applications have been successfully paid while less than 30K applications have defaulted. This figure indicates that HCG data is an imbalanced dataset.

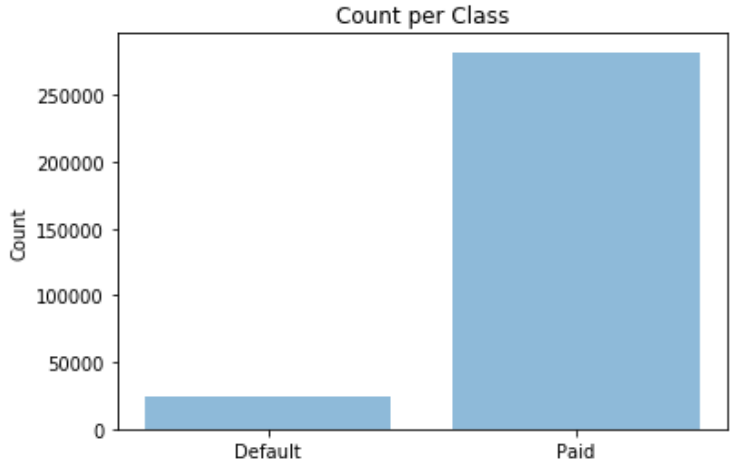
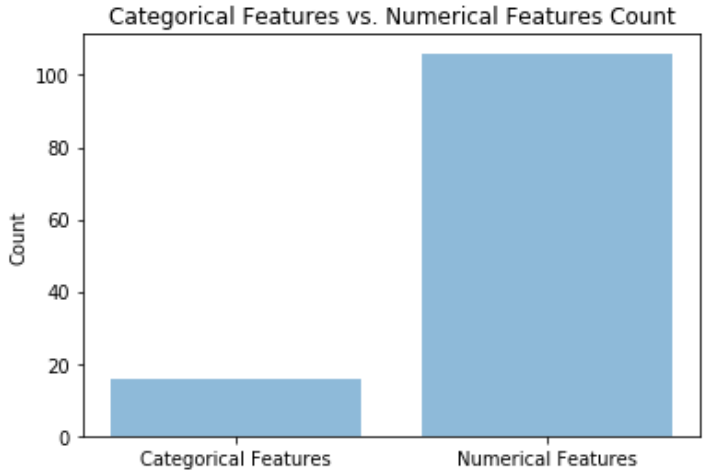


Figure 2.1: Number of loan applications that have been paid versus loan applications that have been defaulted

Figure 2.2 shows that the dataset contains more than 100 numerical features and 18 categorical features.



Next, categorical features are good starting point to find any relevance with the target variable. We begin with features related to applicants' personal information such as gender, family status, children count etc. This will provide good insights whether these features have any relationship with the ability of repaying the loan installments. As shown in Figure 2.2, we can identify that cash loans applications happen to have more default than revolving loans.

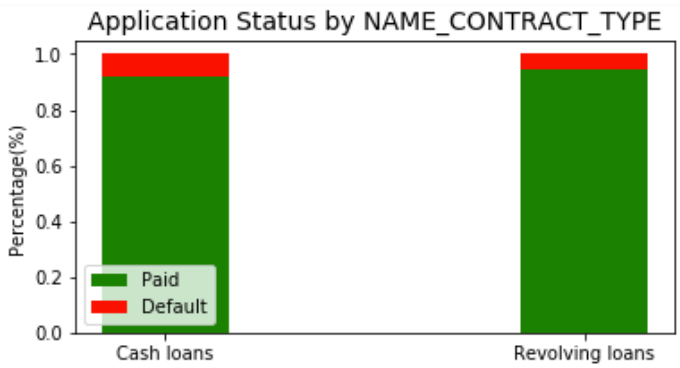


Figure 2.2: % of paid application versus % of default application by contract type.

Furthermore, Figure 2.3 shows that 85% of male applicants have successfully paid the full loan installments and 90% of female applicant have successfully paid the full loan installments. This shows that male applicants are more likely to default than female applicants.

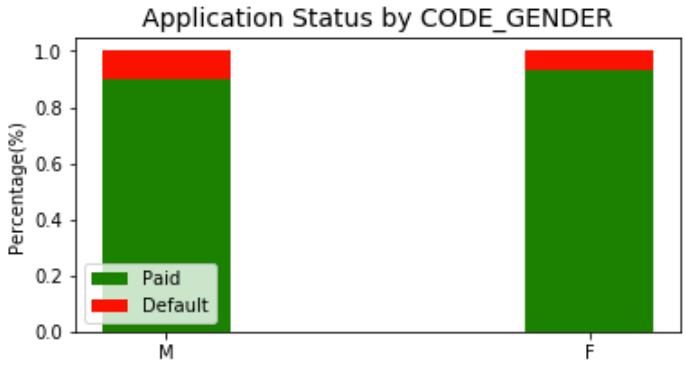


Figure 2.3: % of paid application versus % of default application by applicant gender.

In addition, income type is an important factor in identifying whether an applicant is willing to repay the loan. Figure 2.4 shows that students and businessmen are more likely to successfully pay the loan installments. On the other hand, unemployed and maternity leave applicants have a higher risk to default than any other applicants.

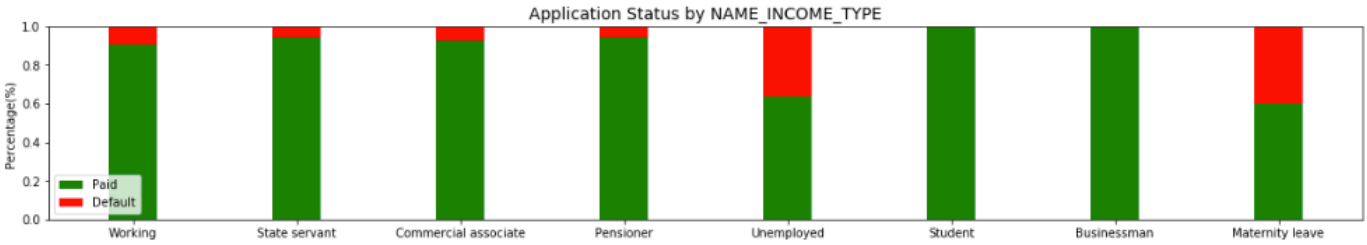


Figure 2.4: % of paid application versus % of default application by applicant income type.

Besides categorical features, numerical features provide meaningful insights as well. One of the important numerical feature in identifying applicant's ability to repay the full loan amount is the applicant age. Figure 2.5 shows that younger applicants are more likely to default than older applicants.

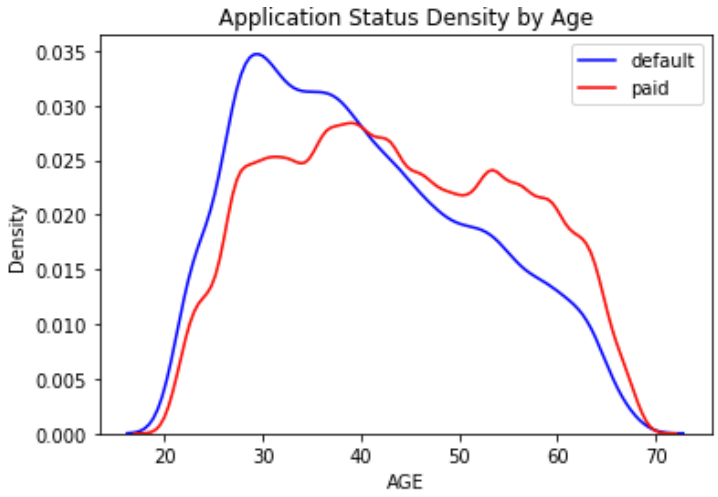
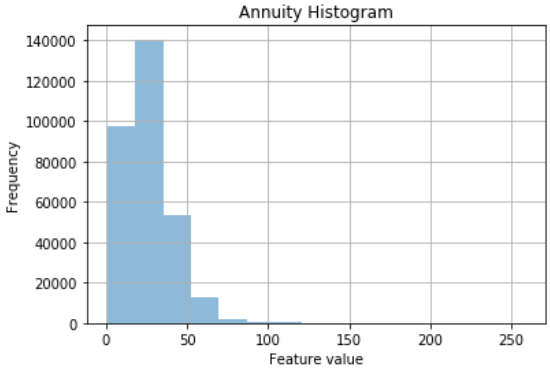
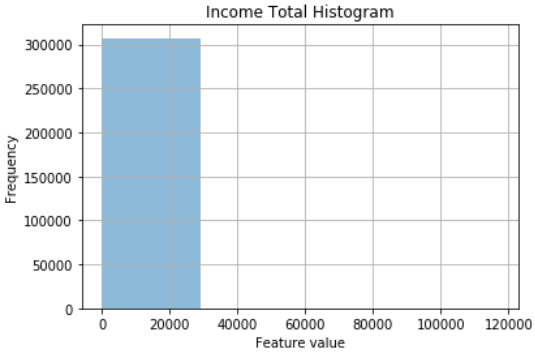
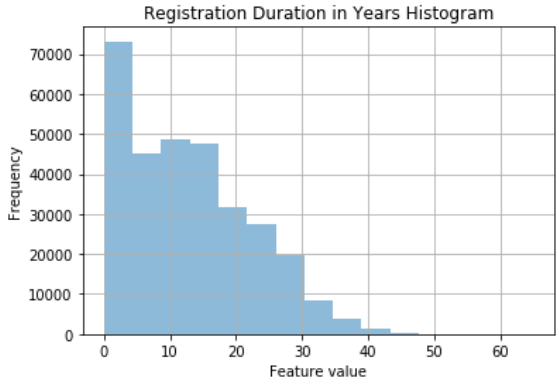


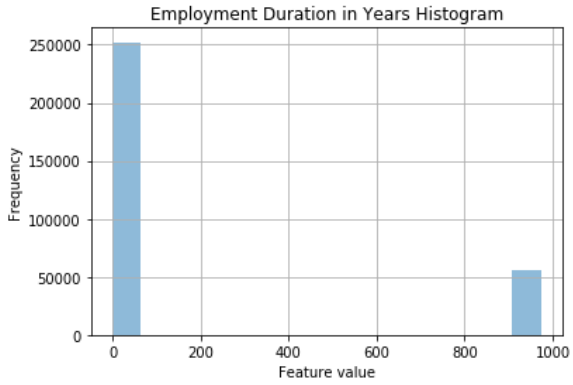
Figure 2.5: Density of paid applications versus Density of default application by applicant age.

Furthermore, it is very important to check the skewness of the features to make sure those numerical features do not affect the performance of the classification model. Are they the features' values normally distributed? Do they have very large and very small values that may affect the performance of the classification model? The following figures shows that some of the most important features such as credit amount, total income, etc… are highly skewed and require further normalization during data preprocessing prior to model training.









Nevertheless, statistical analysis of the numerical features can help in understanding the dataset deeply. The below figure shows the most important numerical features with their mean, std, min, and max values.

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	CNT_CHILDREN	DAYS_LAST_PHONE_CHANGE	CNT_FAM_MEMBERS	DAYS_BIRTH	DAYS_EMPLO
count	3.075110e+05	3.075110e+05	307499.000000	307511.000000	307510.000000	307509.000000	307511.000000	307511.00
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	1.687979e+05	5.990260e+05	27108.573909	0.417052	-962.858788	2.152665	-16036.995067	63815.04
std	2.371231e+05	4.024908e+05	14493.737315	0.722121	826.808487	0.910682	4363.988632	141275.76
min	2.565000e+04	4.500000e+04	1615.500000	0.000000	-4292.000000	1.000000	-25229.000000	-17912.00
25%	1.125000e+05	2.700000e+05	16524.000000	0.000000	-1570.000000	2.000000	-19682.000000	-2760.00
50%	1.471500e+05	5.135310e+05	24903.000000	0.000000	-757.000000	2.000000	-15750.000000	-1213.00
75%	2.025000e+05	8.086500e+05	34596.000000	1.000000	-274.000000	3.000000	-12413.000000	-289.00
max	1.170000e+08	4.050000e+06	258025.500000	19.000000	0.000000	20.000000	-7489.000000	365243.00

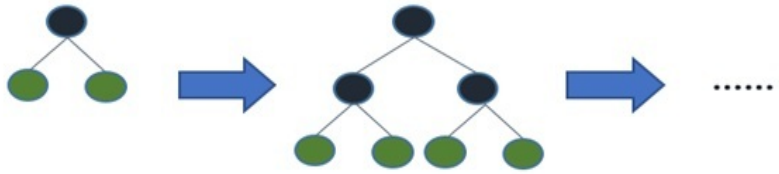
Algorithms and Techniques

As a starting point, it is unclear which classification model will best solve the problem of predicting home credit default risk. Therefore, we will use different classification models for training and testing then a final classification model with the best evaluation metric will be selected. In this project, we will use classification algorithms such as Logistic Regression, Adaptive Boosting, Naïve Bayes, XGBoost and Light GBM classifiers for predicting home credit default risk.

Logistic Regression is a type of classification algorithm involving a linear discriminant. logistic regression outputs a probability that the given input point belongs to a certain class. The output of Logistic Regression is a sigmoid curve (AKA S-curve). Where the value on the x-axis (independent variable) would determine the dependent variable on the y-axis. In logistic regression there are only two possible outcomes. 0 and 1. That something occurs, or it doesn't. We use a threshold value to make our prediction easier. If the x-axis' corresponding y-value (probability) is lesser than the threshold value, the outcome is taken as 0. If it is greater than the value, the outcome is taken as 1.

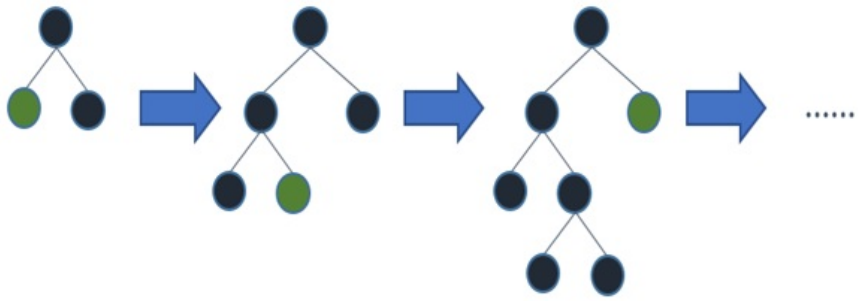
Naive Bayes Classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. [1]

Boosting algorithms such as Adaboost and XGBoost fit a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy. [2]



Level-wise tree growth

LightGBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm [3].



Leaf-wise tree growth

Recall that the dataset used for this project is imbalanced where the number of applications that have been successfully paid weighs more than 90% of the dataset. Therefore, we will scale the positive class to control the balance of positive class and negative class in the initial run for applicable classification models such as Logistic Regression, XGBoost and LightGBM models.

Below table contains the pros and cons for each classifier used in this project.

Classifier	Pros	Cons
Logistic Regression	- Low variance. - Provides probabilities for outcomes. - Works well with diagonal feature decision boundaries	- High Bias
AdaBoostClassifier	- Automatically handle missing values - It does not overfit easily - It can leverage many different weak-learners	- Sensitive to noisy data and outliers
Naïve Base	- Computationally fast - Simple to implement - Works well with high dimension	- Relies on independence assumption and will perform badly if this assumption is not met
XGBoost	- It can leverage many different weak-learners	- Slow Performance
LightGBM	- Faster training speed and higher efficiency. - Lower memory usage - Better accuracy - Compatibility with Large Datasets - Parallel learning supported. -	Minimum documentation and user base

[1] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[2] https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/
[3] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Benchmark

There are many existing research studies that predict whether a loan will default using different datasets. In this project, we will use one of these studies' final AUC score as a benchmark for our classification model. One of these studies is The LendingClub's Loan [1] and their best AUC score was 0.711. The aim for this project is to produce a model that results with a better AUC score than 0.711.

[1] http://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html

III. Methodology

Data Preprocessing

Prior to training a classification model, it is important to do some preprocessing required to avoid problems during model training. The preprocessing steps for this project include handling missing values in specific features, dealing with abnormalities in the dataset, converting categorical features to binary format, and normalizing the numerical features.

Missing values in the dataset's features cause runtime errors during model training. The dataset contains 128 features and 67 out of these features have missing values. The number of missing values ranges between less than 1% of the dataset and up to 70% for some of the features. For categorical features, missing values have been mutated with the most repetitive feature. For example, NAME_TYPE_SUITE is a categorical feature that's missing 42% of the dataset. Figure 3.1 shows that "Unaccompanied" is the most repetitive feature value. Missing values were mutated with Unaccompanied value.

```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

Figure 3.1: NAME_TYPE_SUITE feature values count.

For numerical features, a different approach has been taken to handle missing values. Feature like OWN_CAR_AGE has more than 65% missing values. This feature is interrelated with FLAG_OWN_CAR in a way such that if the applicant does not own a car, OWN_CAR_AGE will not show a value. Therefore, we replaced missing data with a zero if FLAG_OWN_CAR is true. For other numerical missing values, we replaced them with -999 to represent a missing value instead of NaN.

Abnormalities do exist in the dataset as discussed in previous sections. CODE_GENDER that represents the applicant's gender contain values either man or

female. The dataset happens to have a third value of XNA which has been replaced with Female as it's the most common value for that feature. Another abnormality occurs in the dataset is DAYS_EMPLOYED which represents the duration of employment in days. One record has 365243 as a value for DAYS_EMPLOYED which is about 1000 years. This is not a realistic value and therefore it has been replaced with a zero.

Many machine learning algorithms cannot understand categorical directly. They require input data to be in numerical format. To handle this matter, one-hot encoding is performed on all 16 categorical features to convert to binary format. Furthermore, it is a common practice to perform logarithmic transformation on highly skewed numerical features so that large and small data do not negatively affect the performance of the learning algorithm. We applied the logarithmic transformation to numerical features in the dataset with care to logarithm of 0 to avoid undefined values. Finally, in addition to performing logarithmic transformation, we normalized the numerical features in the dataset to limit the values between 0 and 1. This is to ensure that all features are treated equally when applying a supervised learner.

Implementation

AdaBoostClassifier, GaussianNB, and LogisticRegression classification models are implemented using scikit-learn Python module, while XGBoost and LightGBM are implemented using XGBoost and LightGBM module respectively. The Home Credit Default Risk dataset has been split into training and testing dataset with a test size of 30% of the dataset. The function train_predict has been created to accept a classification model, training features. training labels, testing features, and test labels. The function fits the training data into the classification model then predicts the testing features. Finally, it returns the evaluation metrics for each classification model as illustrated below.

```
#Split data into training and testing eturn
X_train, X_test, y_train, y_test = train_test_split(data,
                                                    target,
                                                    test_size = 0.3,
                                                    random_state = 48)

# Return evaluation metrics scores
def get_scores(y_train, y_test, y_pred_train, y_pred_test):
    res = {}
    res['acc_train'] = accuracy_score(y_train, to_binary(y_pred_train.copy()))
    res['acc_test'] = accuracy_score(y_test, to_binary(y_pred_test.copy()))

    res['f_train'] = fbeta_score(y_train, to_binary(y_pred_train.copy()), beta=0.5)
    res['f_test'] = fbeta_score(y_test, to_binary(y_pred_test.copy()), beta=0.5)

    res['rec_train'] = recall_score(y_train, to_binary(y_pred_train.copy()))
    res['rec_test'] = recall_score(y_test, to_binary(y_pred_test.copy()))

    res['pre_train'] = precision_score(y_train, to_binary(y_pred_train.copy()))
    res['pre_test'] = precision_score(y_test, to_binary(y_pred_test.copy()))
```

```
21. 
22.     res['auc_train'] = roc_auc_score(y_train, y_pred_train)
23.     res['auc_test'] = roc_auc_score(y_test, y_pred_test)
24.     return res
25. 
26. # convert probability to binary for LighGBM
27. def to_binary(y_pred):
28.     for i in range(0, len(y_pred)):
29.         if y_pred[i] >= .5:
30.             y_pred[i] = 1
31.         else:
32.             y_pred[i] = 0
33.     return y_pred
34. 
35. def train_predict(learner, X_train, y_train, X_test, y_test):
36.     '''''
37.     inputs:
38.        - learner: the learning algorithm to be trained and predicted on
39.        - X_train: features training set
40.        - y_train: income training set
41.        - X_test: features testing set
42.        - y_test: income testing set
43.     '''
44.     res = {}
45. 
46.     # Special training for LightGBM as it does not follow the standard .fit
47.     if learner[0] == 'LightGBM':
48.         lgb_train = lgb.Dataset(data=X_train,label=y_train)
49.         learner = lgb.train(learner[1], lgb_train, 10)
50. 
51.     # Fit the learner to the training data using .fit(training_features, training_l
    abels)
52.     else:
53.         learner = learner[1].fit(X_train, y_train)
54. 
55.     predictions_train = learner.predict(X_train)
56.     predictions_test = learner.predict(X_test)
57. 
58.     res = get_scores(y_train, y_test, predictions_train, predictions_test)
59.     # Success
60.     print("{} trained on {} samples.".format(learner.__class__.__name__, len(X_trai
    n)))
61.     tn, fp, fn, tp = metrics.confusion_matrix(y_test, to_binary(predictions_test.co
    py())).ravel()
62.     print("TN: {:}, FP: {:}, FN:{:}, TP:{:}".format(tn, fp, fn, tp))
63.     # Return the results
64.     return res
```

The classification algorithms are initialized with the default parameters with a random state of ‘48’. Each model is trained on the training dataset by using .fit function. All classification models accept a Pandas object for the dataset except LightGBM which requires an LGBM dataset by using the below line of code.

lgb.Dataset(data=X_train,label=y_train)

Since the home credit default risk dataset is an imbalanced dataset, all classification algorithms predict a negative class for each input data. Therefore, we addressed this issue by scaling the positive class to equal the total number of the training dataset

divided by the total number of positive class of the training dataset as illustrated below.

```
pos_class = target.count()//np.sum(target)  # positive class scale
classifiers = [
    ('ABoost', AdaBoostClassifier(random_state=48)),
    ('GNB', GaussianNB()),
    ('LR', LogisticRegression(random_state=48, class_weight='balanced')),
    ('XGBoost', XGBClassifier(random_state=48, scale_pos_weight=pos_class)),
    ('LightGBM', { 'objective': 'binary',
                   'metric': 'auc',
                   'scale_pos_weight': pos_class,
                   'random_state': 48}) # passing parameters only for LightGBM
]

results = {}

# for each classifier, perform train_predict and show the evaluation metrics
for classifier in classifiers:
    print("Started training", classifier[0], "model")
    res = train_predict(classifier, X_train, y_train, X_test, y_test)
    results[classifier[0]] = res
    print("Train - AUC: {:.04}, Accuracy: {:.04}, f1 score: {:.04}, Recall: {:.04},
           Precision: {:.04}".format(
        res["auc_train"], res["acc_train"], res["f_train"], res["rec_train"], res["
pre_train"])
         )
    print("Pred - AUC: {:.04}, Accuracy: {:.04}, f1 score: {:.04}, Recall: {:.04},
Precision: {:.04}".format(
        res["auc_test"], res["acc_test"], res["f_test"], res["rec_test"], res["pre_
test"])
         )
```

After scaling the positive class, the classification models started to automatically adjust weights inversely proportional to class frequencies in the input data. After performing training and testing, the resulted confusion matrix and evaluation metric is captured below for each classification model. Note that all classification models return a binary prediction for each test data input except LightGBM which returns a probability. LightGBM probability will be converted to 1 if the probility is greater than or equal to 0.5, and 0 if the probability is less than 0.5. This is required since confusion matrix, accuracy, recall, and F1 evaluation metric functions expect binary inputs.

Model	TP	FP	TN	FN
AdaBoostClassifier	4	9	84596	7557
GaussianNB	7266	78774	5831	295
LogisticRegression	4680	30370	54235	2881
XGBClassifier	5261	28051	56554	2300
LightGBMClassifier	2361	7588	77017	5200

Figure 3.2: Confusion matrix for each classification model

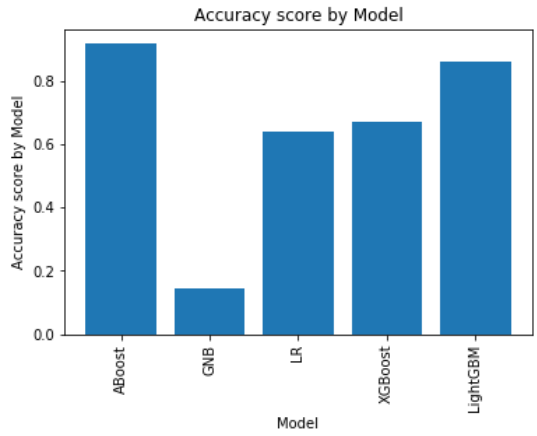


Figure 3.3: Accuracy score by classification model

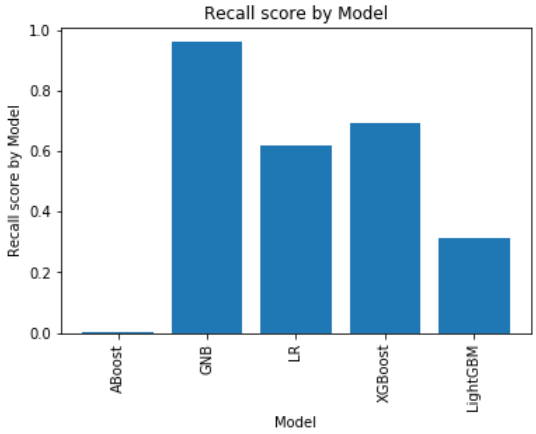


Figure 3.4 Recall score by classification model

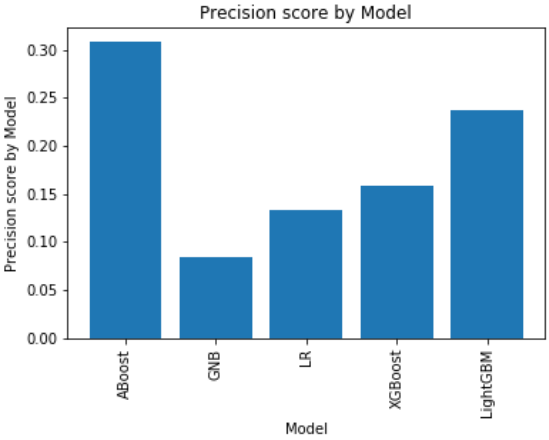


Figure 3.5: Precision score by classification model

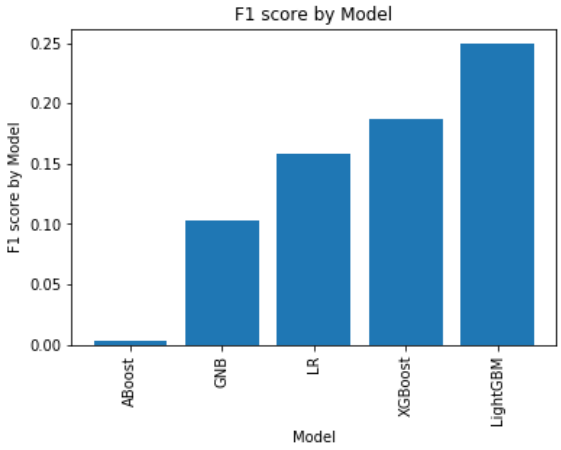


Figure 3.6: F1 score by classification model

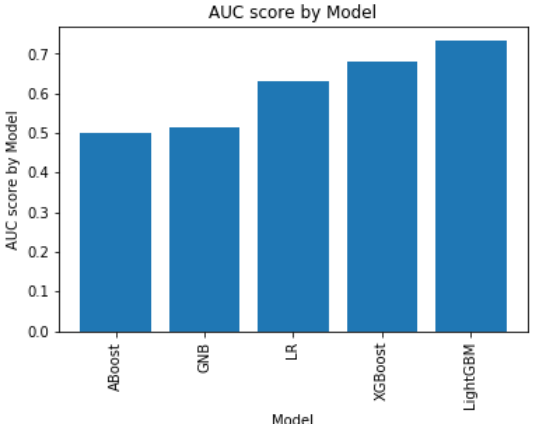


Figure 3.7: AUC score by classification model

After visualizing the evaluation metrics, we can clearly see that LightGBM has the highest F1 and AUC evaluation metrics. Therefore, we choose LightGBM for further improvements which will be detailed in the next section.

Refinement

In order to maximize optimization of classification models, hyper-parameter tuning is performed to achieve this goal. Since LightGBM performed the best compared to the other selected models, we will try to find its best hyper-parameters that can optimally predict home credit default risk on the given dataset. LightGBM has many fine parameters to be tuned which are shown below.

Parameter	Final Tuned Parameter Value
boosting_type	gbdt, dart
objective	binary
metric	auc, f1
num_leaves	np.random.randint(64, 128)
max_depth	np.random.randint(6, 12)
min_data_in_leaf	int(2 ** (np.random.rand() * 3.5 + 9))
feature_fraction	np.random.rand()*0.35+0.65
bagging_fraction	0.768
bagging_freq	1
lambda_l1	10 ** (np.random.rand() * 4)
lambda_l2	10 ** (np.random.rand() * 3 + 2)
min_gain_to_split	0.0
min_sum_hessian_in_leaf	0.1
num_threads	16
verbose	0
scale_pos_weight	pos_class

We have set the number of boosting iterations to 5000 and an early stopping rounds 250 so that the model will train the model until the validation score stops improving for 250 rounds.

```
1.  evals_result = {}  # dict to store evaluation results of all the items in `valid_se
    ts`
2.  dev_X, val_X, dev_y, val_y = train_test_split(X_train,
3.                                                y_train,
4.                                                test_size = 0.2,
5.                                                random_state = 42) # split training d
    ataset into train and validation
6.  pos_class = dev_y.count()//np.sum(dev_y) # recalculate positive class weight after
    CV split
7.
8.  lgb_train = lgb.Dataset(data=dev_X,label=dev_y) # training dataset
9.  lgb_val = lgb.Dataset(data=val_X, label=val_y) # validation dataset
10. lgb_test = lgb.Dataset(data=X_test,label=y_test) # testing dataset
11.
12. gbm = lgb.train(params, lgb_train, 5000, valid_sets=[lgb_val], \
13.                 valid_names = ['cv'], evals_result=evals_result, \
14.                 early_stopping_rounds=250, verbose_eval=100,
15.                 categorical_feature='auto')
16. predictions_test_tuned = gbm.predict(X_test)
```

During training, the model has improved AUC score to 0.76 on the cross-validation dataset as shown below.

```
Training until validation scores don't improve for 250 rounds.
[100]   cv's auc: 0.753214
[200]   cv's auc: 0.758614
[300]   cv's auc: 0.760037
[400]   cv's auc: 0.760226
[500]   cv's auc: 0.760255
[600]   cv's auc: 0.760183
[700]   cv's auc: 0.76004
Early stopping, best iteration is:
[510]   cv's auc: 0.760331
```

Model	TP	FP	TN	FN
LightGBMClassifier	2394	7364	77074	5062

Table 3.8: Confusion Matrix before hyper-parameter tuning

Model	TP	FP	TN	FN
LightGBMClassifier	5177	25976	58734	2279

Table 3.9: Confusion Matrix after hyper-parameter tuning

	Before Hyper-Parameters Tuning		After Hyper-Parameters Tuning	
	Train	**Test**	**Train**	**Test**
Accuracy	0.863	0.862	0.863	0.693
Precision	0.244	0.238	0.244	0.166
Recall	0.330	0.321	0.330	0.694
F1	0.257	0.251	0.257	0.195
AUC	0.748	0.736	0.748	0.759

We can clearly see in table 3.10 a comparison of the evaluation metrics on LightGBM classifier before and after hyper-parameter tuning. We can clearly see an excellent improvement on AUC score from 0.736 to 0.759 for the test dataset. The final hyper-parameters that resulted in AUC improvement to 0.759 are shown in below table.

Parameter	Final Tuned Parameter Value
boosting_type	gbdt, dart
objective	binary
metric	auc, f1
num_leaves	116
max_depth	7
min_data_in_leaf	2394
feature_fraction	0.848
bagging_fraction	0.768
bagging_freq	1
lambda_l1	134.297
lambda_l2	3105.441
min_gain_to_split	0.0
min_sum_hessian_in_leaf	0.1

num_threads	16
verbose	0
scale_pos_weight	12

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

Initially, we selected many classification models to predict home credit default risk on a given dataset which are AdaBoostClassifier, Naïve Bayes Classifier, LogisticRegression, XGBoostClassifier, and LightGBM. Since the dataset is an imbalanced dataset, accuracy evaluation metric does not really tell us how the model is performing. Instead, the primary evaluation metric used for this problem is AUC score. After training and testing the dataset, we have collected each model's evaluation metrics and plotted them in below figures.

Model	**TP**	**FP**	**TN**	**FN**
AdaBoostClassifier	4	9	84596	7557
GaussianNB	7266	78774	5831	295
LogisticRegression	4680	30370	54235	2881
XGBClassifier	5261	28051	56554	2300
LightGBMClassifier	2394	7364	77074	5062

Figure 4.1: Confusion matrix for each classification model

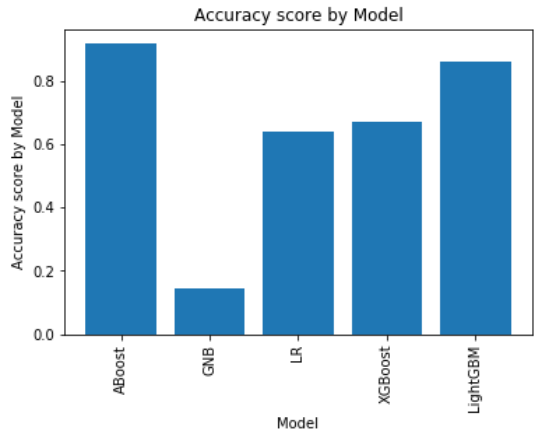


Figure 4.2: Accuracy score by classification model

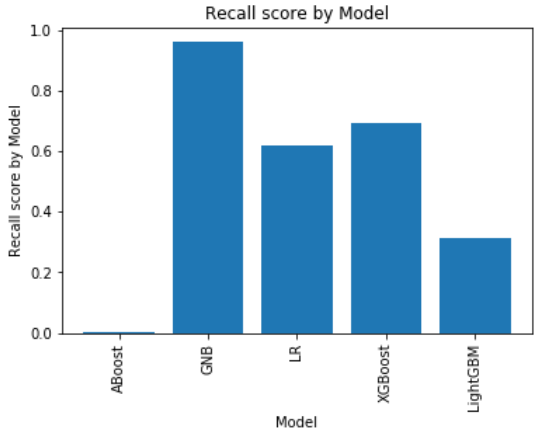


Figure 4.3 Recall score by classification model

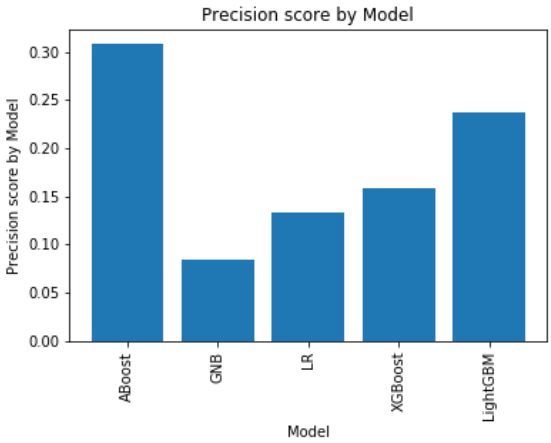


Figure 4.4: Precision score by classification model

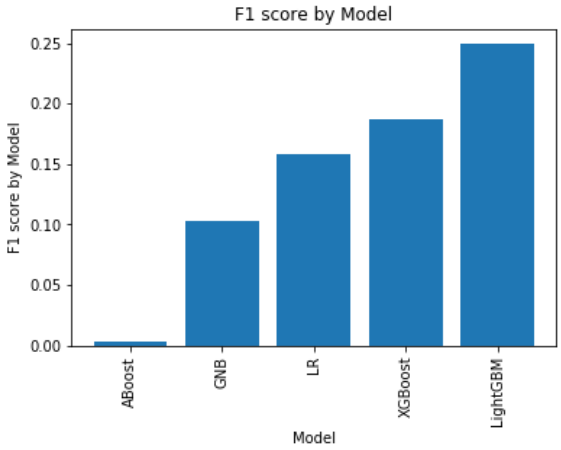


Figure 4.5: F1 score by classification model

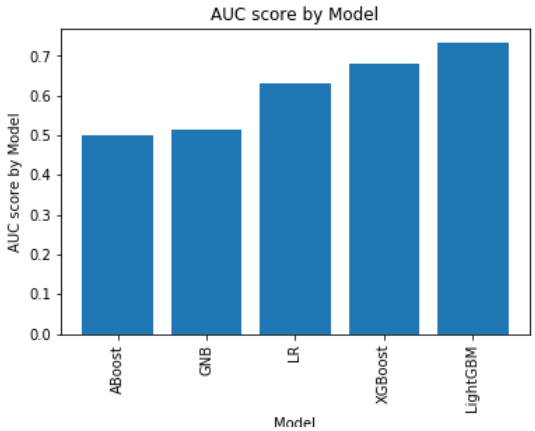


Figure 3.7: AUC score by classification model

It turned out that LightGBM has the highest AUC score of 0.731 and therefore been selected for further improvements. After tuning LightGBM classifier hyper-parameters, we gained an improvement of AUC score to 0.759. The parameters that increased model evaluation is shown below.

```
Best Parameters that improved our final model LightGBM:
{'boosting_type': 'gbdt', 'objective': 'binary', 'metric': ['auc', 'f1'], 'learning_rate': 0.1, 'num_leaves': 116, 'm
ax_depth': 7, 'min_data_in_leaf': 2394, 'feature_fraction': 0.8480511816917773, 'bagging_fraction': 0.768216153214347
9, 'bagging_freq': 1, 'lambda_l1': 134.29783711684772, 'lambda_l2': 3105.4418845715327, 'min_gain_to_split': 0.0, 'mi
n_sum_hessian_in_leaf': 0.1, 'num_threads': 16, 'verbose': 0, 'is_training_metric': 'True', 'scale_pos_weight': 12}
```

Model	TP	FP	TN	FN
LightGBMClassifier	5177	25976	58734	2279

Table 3.9: Confusion Matrix after hyper-parameter tuning

	Before Hyper-Parameters Tuning		After Hyper-Parameters Tuning	
	Train	Test	Train	Test
Accuracy	0.863	0.862	0.863	0.693
Precision	0.244	0.238	0.244	0.166
Recall	0.330	0.321	0.330	0.694
F1	0.257	0.251	0.257	0.195
AUC	0.748	0.736	0.748	0.759

Finally, after random parameter search process, the model has gained an improvement of AUC score to 0.76 on the cross-validation test set and 0.759 on the final test dataset. This suggests that the model is sufficiently robust for its intended application and does not show overfitting.

Justification

Recall that the aim is to benchmark our model with existing studies on similar application of predicting loan default on LendingClub dataset. Comparing our model evaluation metric with the benchmark, we can see that our model received an AUC score of 0.759 while the benchmark AUC score is 0.713.

While this performance comparison highlights the advantage of using a complex model over a simple mode in this particular application, it is good to assess the final model on its standalone metrics and consider its limitation of a low precision score of 0.166 is low for a precision-optimized model which highlights a difficulty of minimizing false positive cases.

Despite the limitation of this model, it still can serve its purpose of identifying home credit default risks. This model provides a higher level of confidence in identifying applicants whom will result in failing to repay their loan installments than a naïve prediction. The model results in minimizing home credit default risk.

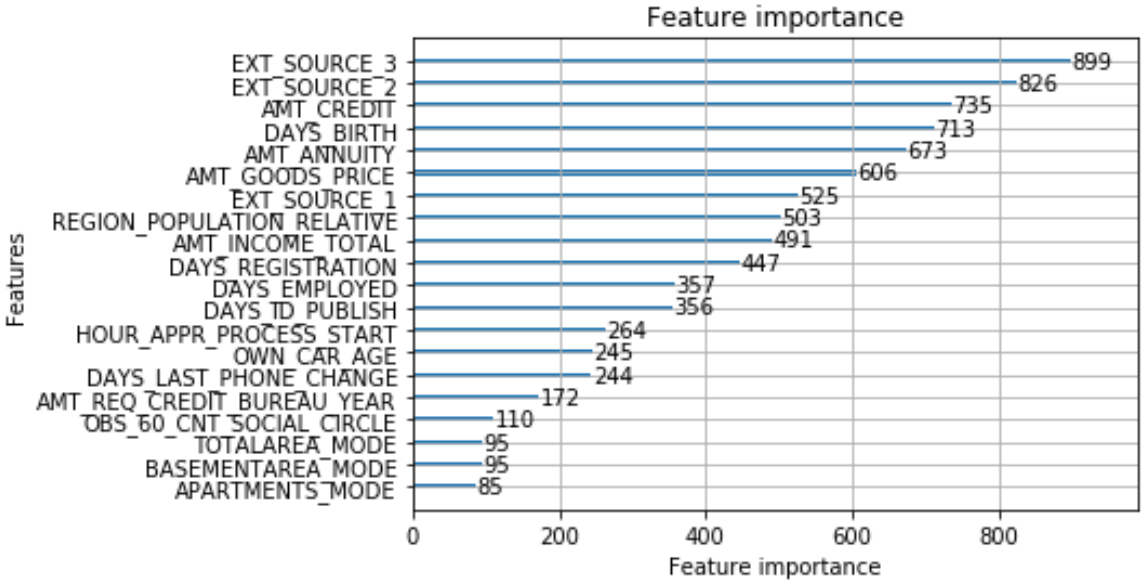
In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

After model training, it is good to know the top contributing features that help to predict whether an applicant is likely to default on home credit repayment. These features importance have the highest weight to the final prediction on each data input. The feature importance figure below details the top 20 important features for the given dataset of predicting home credit default risk



Reflection

The most difficult task faced in implementing this project was data cleansing and preprocessing. The dataset has many missing values which I was not sure whether to remove or mutate record with missing feature values. I started off with removing as this is the most convenient approach however, I ended up having a poor model performance. Another challenge was dealing with the imbalanced dataset. I noticed that all models were predicting applicants to repay the loan since over 90% of the dataset is negative class. I tried to balance the dataset myself which turned out to be a greater challenge which led me to scale the positive class instead that worked out successfully.

Improvement

Although the final model's evaluation metrics are acceptable, further improvements can be done to gain higher metrics score. There is a high potential of improvement in performing feature engineering of the home credit default risk dataset such as credit to income ratio, credit to annuity ratio, income per child ratio …etc. Feature engineering also include reduction of unnecessary features that do not add a value in constructing a final classification model should be removed to minimize model complexity.

Another potential improvement can be made by removing outliers from the dataset. Although the aim is to encounter any applicants whom at default risk, the model can greatly benefit from removing these outliers as they may affect the prediction of such risks. Finally, although scaling positive class in an imbalanced dataset helps to avoid naïve prediction, adding more positive class data in the dataset can greatly help the model to predict default risks.