Loan Default Prediction FOR PROFIT MAXIMIZATION

Semester 20212 — Machine Learning

Prof. Than Quang Khoat

Team 12

NGUYEN NGOC DUNG	NGUYEN HUY HAI	DUONG VU TUAN MINH	NGUYEN HAI LONG
20204905	20200194	20209705	20204920

Outline

OUTLINE OF DISCUSSION TOPICS

1. Problems approaches

2.Data overview and epxloration

3. Evaluation metric, imbalance handling and pipeline

4. Model using and result

5. Conclusion

Problems Approaches

Our approach is to build a machine-learning classification model that can quantify the credit risk of a bank. Specifically, we build and evaluate classifiers that predict whether a given loan will be fully repaid by the borrower.
In addition, there are two types of risks that are associated with the bank's decision:

- If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company
- If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company

II

Data exploratory analysis and feature engineering

Data overview

Our data is taken from kaggle.com/datasets/wordsforthewise/lending-club

The original data set has over 2.2 million rows and 151 columns. Customers provided the characteristics that make up this data, some of which include annual income, job longevity, and credit histories. Ultimately, a loan decision will be based on these factors.

Determining whether a regular consumer is a defaulter or not is our responsibility. To strengthen this process, rigorous models and approaches will be used.

Data exploration and engineering

STEP 1: REMOVE FEATURES

= leaking (e.g. funded_ amnt)

- having many missing values

STEP2: DEALING WITH CONTINUOUS FEATURES

- remove features having high correlation

STEP3: DEALING WITH CATEGORICAL FEATURES

- remove single valued fearures

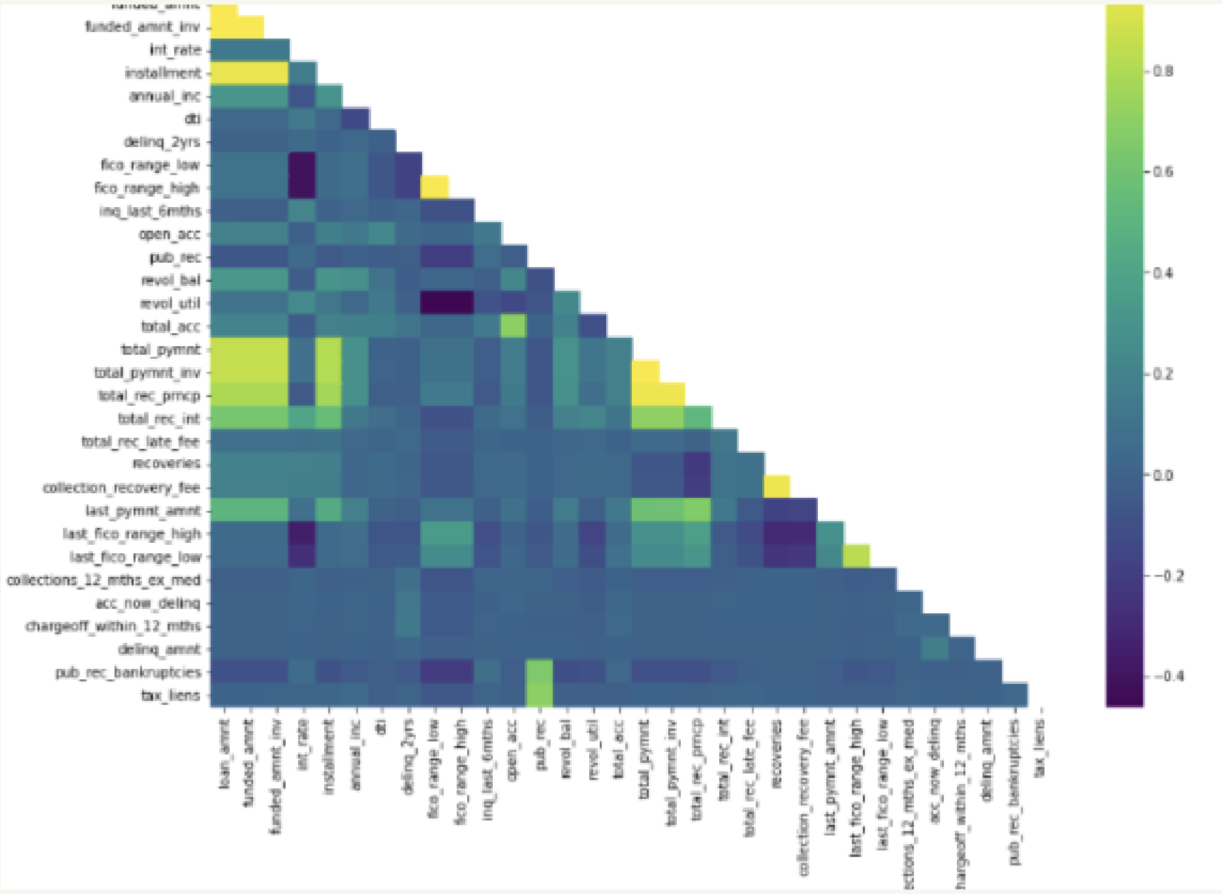
- remove duplicated information

- Remove features having

STEP4: SPLITTING DATA

- train: 0.8

- test: 0.2



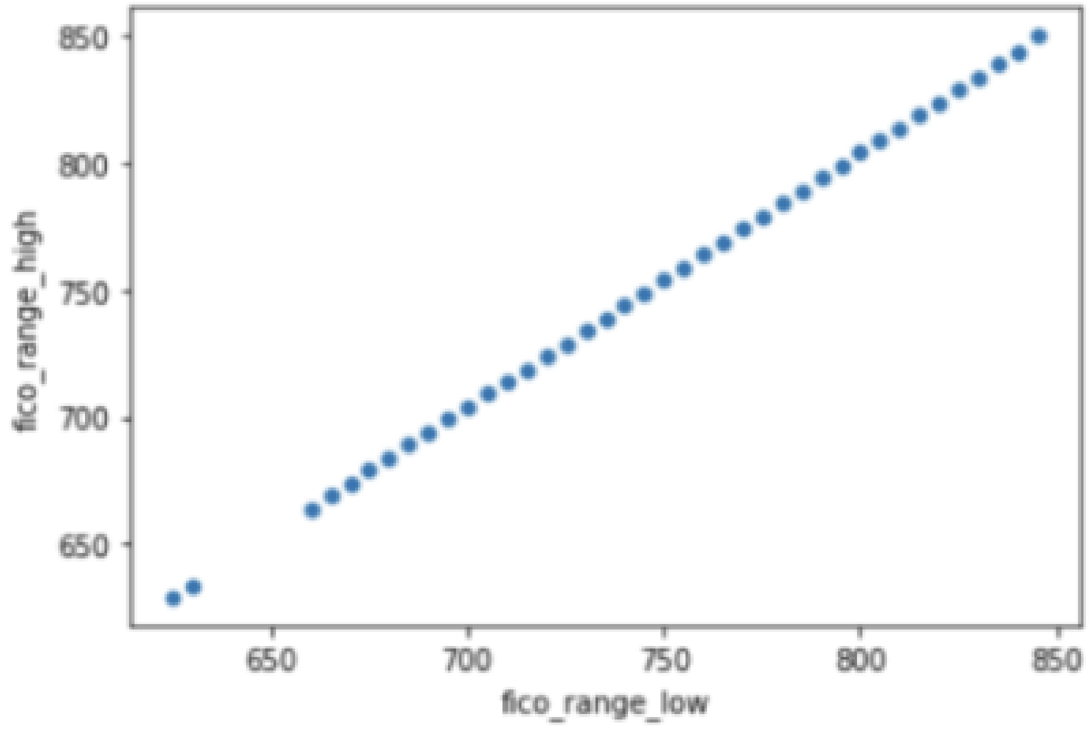


Figure 2: Correlation between 2 features

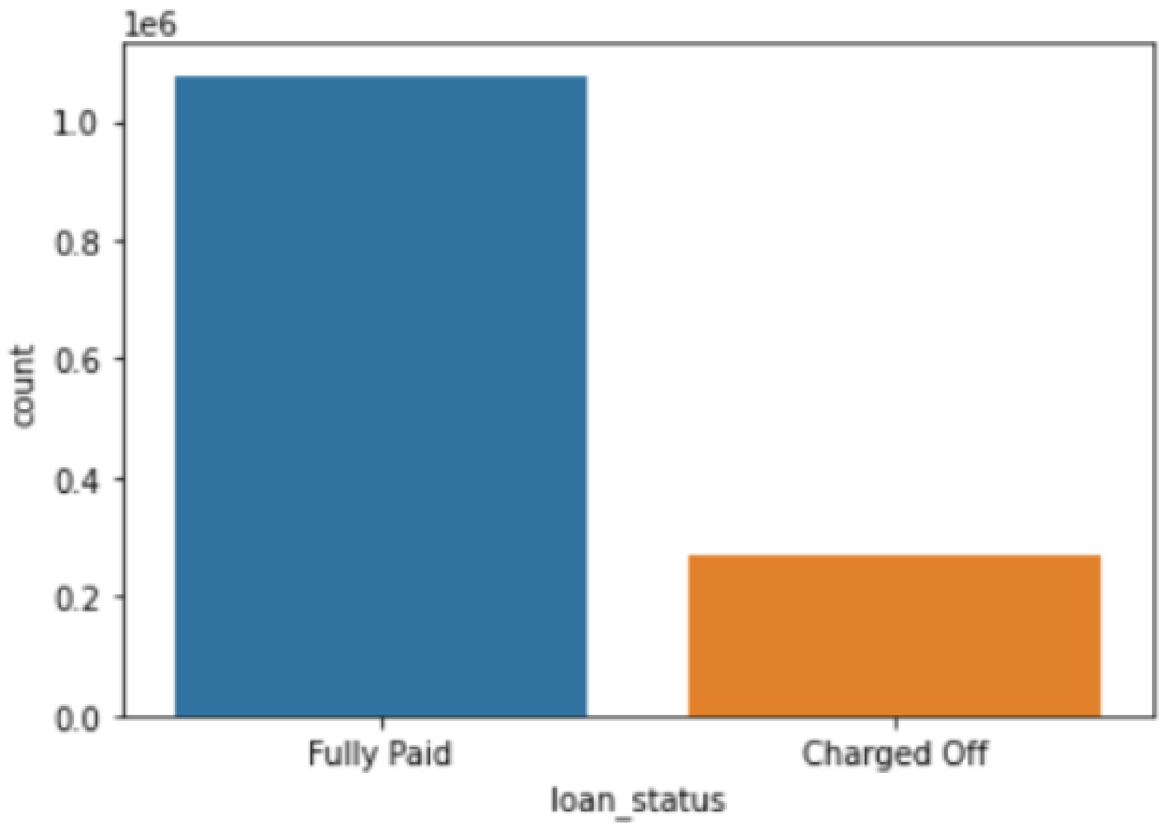
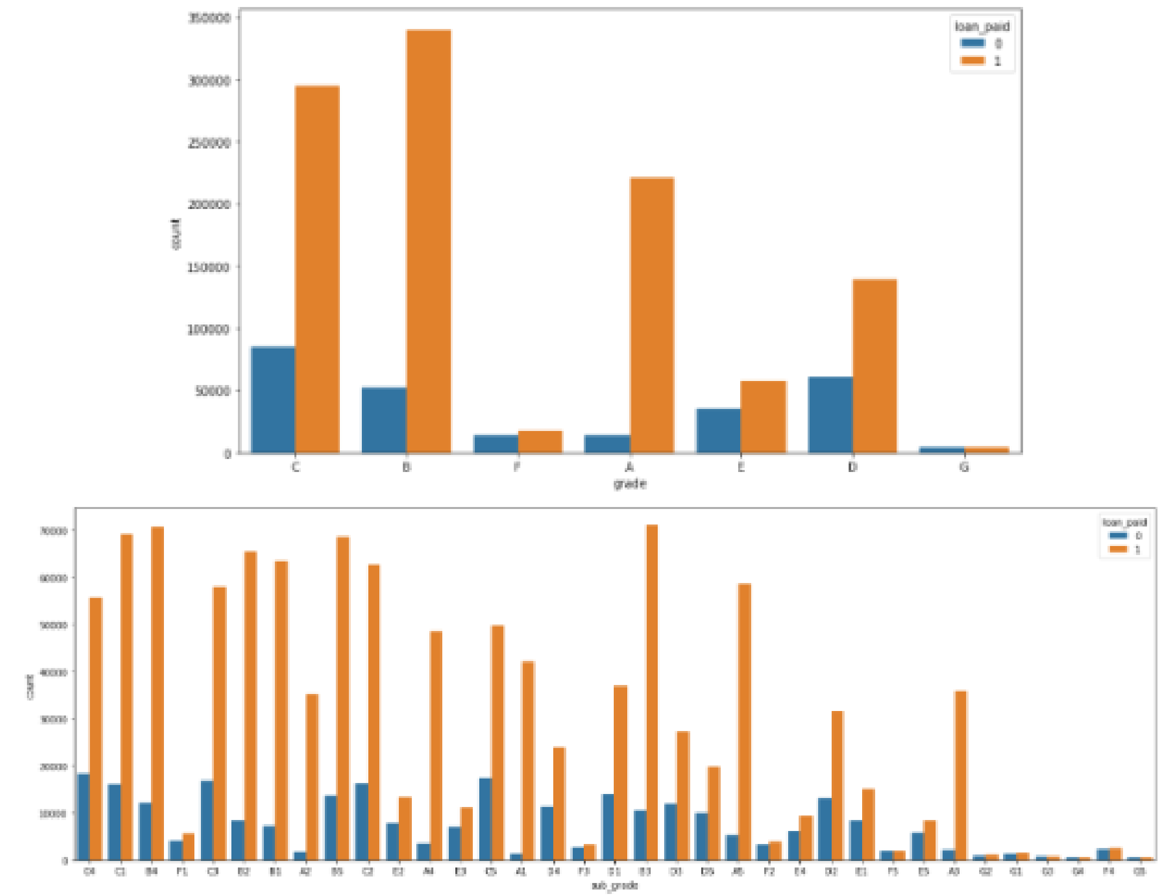


Figure 4: Distribution of target variables



III

Evaluation metric, Imbalance handling and Pipeline

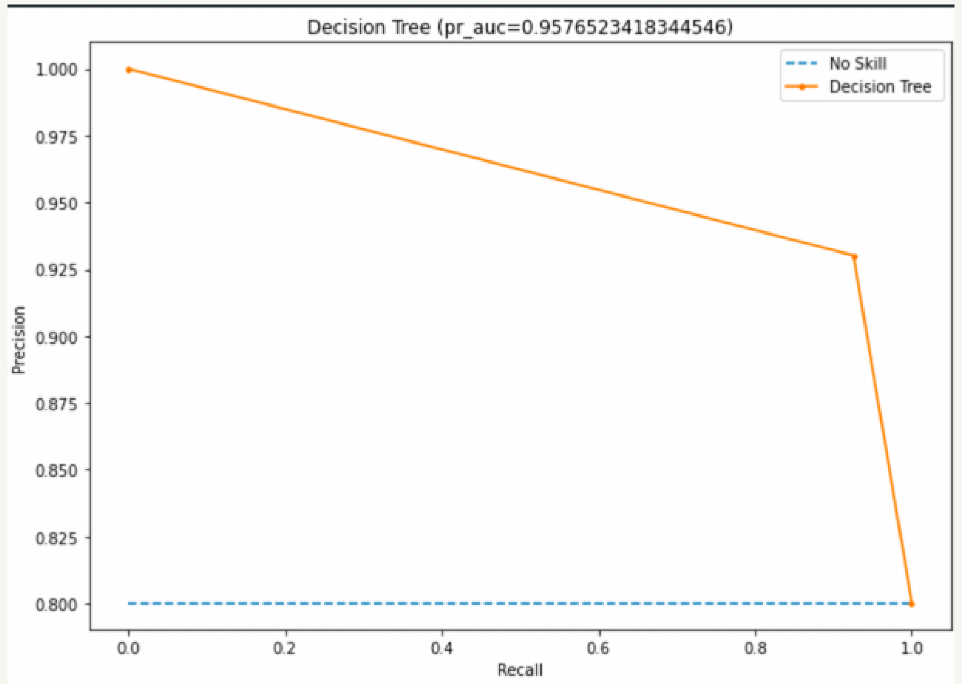
Evaluation Metric

BINARY CLASSIFICATION PROBLEMS

REDUCING FALSE POSITIVE

Eval metric

PRECISION RECALL AUC



F BETA- MEASURE WITH BETA = 0.5

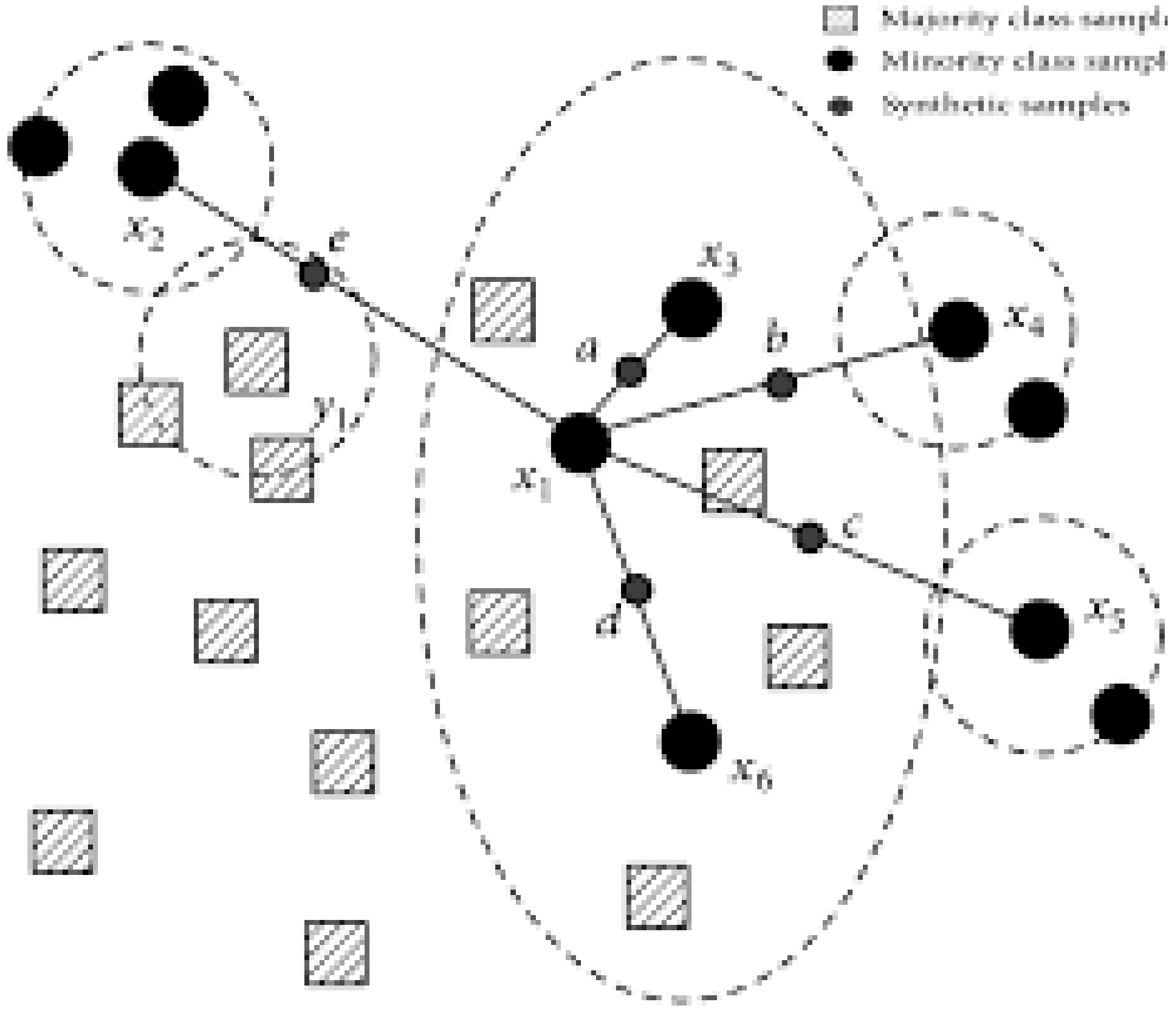
$$F_\beta = \frac{1+\beta^2}{\frac{\beta^2}{Recall} + \frac{1}{Precision}}$$

Imbalance handling

Smote

Choosing examples that are close in the feature space

Drawing a line between the examples in the feature space and drawing a new sample as a point along that line



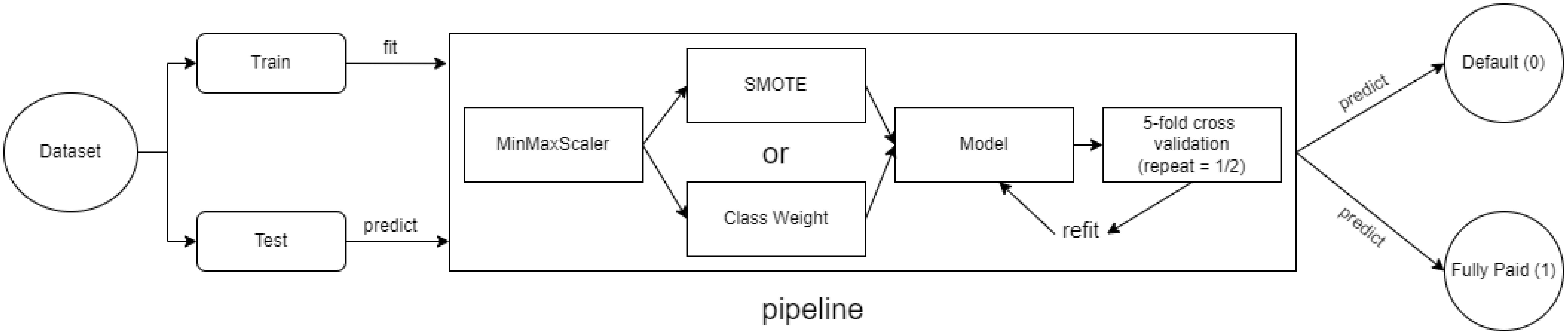
Class Weight

Help the model focus on minority class rather than the majority one by assigns the class weights to their classes.

'balance' class weight: assigns the class weights inversely proportional to their respective frequencies, followed by this formula

$$w_j = n\_samples / (n\_classes * n\_samples_j)$$

Pipeline



Dataset
Train
Test
fit
predict
MinMaxScaler
SMOTE
or
Class Weight
Model
5-fold cross validation (repeat = 1/2)
refit
pipeline
predict
predict
Default (0)
Fully Paid (1)

III

Model Using And Results

Logistic Regression

It learns a linear relationship from the given data set and then introduces a non-linearity in the form of the Sigmoid function where output is probability and input can be from -infinity to +infinity.

Tuning hyperparameters:

- C : Inverse of regularization strength, smaller values specify stronger regularization
- max_iter : Maximum number of iterations taken for the solvers to converge
- solver: Algorithm to use in the optimization problem

Logistic Regression	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		0.875301	0.00145	0.875276	0.0011	26.256346	1.61895	0.872807	0.974491
MinMaxScaler + Base		0.904226	0.000221	0.904201	0.000741	24.521906	1.83919	0.903247	0.981305
MinMaxScaler + SMOTE + Base		0.931736	0.000175	0.931689	0.000418	710.393442	14.557333	0.931779	0.981423
MinMaxScaler + Base (Tuning)	C=100, max_iter=1000, solver='sag'	0.905356	0.000745	0.905214	0.001680	12.082988	2.275316	0.904805	0.981496
MinMaxScaler + (SMOTE + Base) Tuning	sampling strategy=0.8, C=100, max_iter=500, solver='newton-cg'	0.933286	0.000268	0.933425	0.000410	133.473057	24.914387	0.933435	0.981504
MinMaxScaler + (class weight + Base) Tuning	C=100, class_weight= {0: 3.2, 1: 1.0}, solver='saga'	0.933255	0.000107	0.933256	0.000357	34.047845	8.343945	0.933325	0.981580

Perceptron

A perceptron is the simplest neural network, one that is comprised of just one neuron.

Tuning hyperparameters:

- perceptron_penalty

Perceptron	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score
Base		0.874516	0.036919	0.874598	0.037038	13.454261	0.521713	0.846232
MinMaxScaler + Base		0.887178	0.019047	0.887251	0.019246	13.86338	0.106149	0.836828
MinMaxScaler + SMOTE + Base		0.897841	0.013538	0.897772	0.01365	527.701653	17.297408	0.899689
MinMaxScaler + Base (Tuning)	perceptron penalty='l1'	0.894408	0.008845	0.864685	0.008909	329.646469	43.775573	0.858140
MinMaxScaler + (SMOTE + Base) Tuning	perceptron penalty='l1', smote sampling strategy=0.8	0.914760	0.005980	0.915667	0.006923	12.414894	0.674265	0.916345

Support Vector Machine

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space

Tuning hyperparameters:

- C: Regularization parameter. The strength of the regularization is inversely proportional to C
- kernel: Specifies the kernel type to be used in the algorithm.

SVM	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base	max_iter=1500, cache_size=2000	0.793601	0.089995	0.793832	0.089089	331.514857	2.654696	0.821429	0.830260
MinMaxScaler + Base		0.846169	0.013284	0.845901	0.013504	333.654148	6.134611	0.852462	0.933826
MinMaxScaler + SMOTE + Base		0.846906	0.016294	0.846967	0.016419	1220.819262	64.310314	0.834872	0.921614
MinMaxScaler + Base (Tuning)	C=0.001, kernel=rbf’	0.846906	0.008845	0.864685	0.008909	329.646469	43.775573	0.858140	0.942260
MinMaxScaler + (class weight + Base) Tuning	C=0.1, class_weight= 0: 3.6, 1: 1.0, kernel=’linear’	0.889730	0.019498	0.887921	0.017702	6.506483	0.173380	0.8394214	

Decision Tree

Decision Tree is a non-parametric supervised learning method which is very powerful for classification problems.

Tuning hyperparameters:

- max_depth: the maximum depth of the tree
- min_samples_leaf: the minimum number of samples required to be at a leaf node
- criterion: function to measure the quality of a split

Decision Tree	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		1.0	0.0	0.927076	0.000584	29.713296	1.736299	0.929246	0.957652
MinMaxScaler + Base		1.0	0.0	0.927115	0.000697	28.542332	1.72978	0.929127	0.957564
MinMaxScaler + SMOTE + Base		1.0	0.0	0.927115	0.000697	28.542332	1.72978	0.918826	0.951853
MinMaxScaler + Base (Tuning)	max depth=20, min samples leaf=20, criterion='gini'	0.939233	0.000545	0.929216	0.000783	22.88438	0.365039	0.930866	0.984321
MinMaxScaler + (SMOTE + Base) Tuning	max depth=5, min samples leaf=50, criterion='gini'. smote sampling strategy=1.0.	0.937240	0.000814	0.937158	0.000997	186.085667	10.462585	0.937954	0.980921

Random Forest

Random Forest uses several decision trees. It attempts to produce a stochastic forest of trees whose prediction by the whole is more accurate than that of any individual tree by using bagging and feature randomness.

Tuning hyperparameters:

- n_estimators: number of trees
- max_features: number of features that each tree is allowed to train

Random Forest	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		0.99999	0.00001	0.920658	0.00051	376.3120	25.9211	0.921326	0.9864
MinMaxScaler + Base		0.99999	0.00001	0.920769	0.00046	439.7531	46.2193	0.921319	0.98641
MinMaxScaler + SMOTE + Base		0.99999	0.00001	0.933361	0.00051	1381.733	43.2558	0.934771	0.98560
MinMaxScaler + Base (Tuning)	max features='sqrt'; n_estimators=500	0.999966	0.000009	0.920785	0.000713	174.502827	4.308237	0.921831	0.98586

XGBoost

XGBoost is a powerful ensemble learning technique that uses two key technologies: Gradient Boosting and Decision Trees.

Tuning hyperparameters:

- max_depth: maximum allowed depth of the trees
- n_estimators: number of trees
- reg_alpha: L1 regularization on the weights.
- reg_lambda: L2 regularization on the weights.

XGBoost	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		0.920363	0.000241	0.920223	0.000504	178.637775	2.997294	0.920278	0.985985
MinMaxScaler + Base		0.920363	0.000241	0.920223	0.000504	171.318313	5.226494	0.920278	0.985985
MinMaxScaler + SMOTE + Base		0.94018	0.00022	0.940009	0.000499	1237.307	29.86738	0.940456	0.984511
MinMaxScaler + Base (Tuning)	max depth=6, n_estimators=500, reg_alpha=1.0, reg_lambda=1.0	0.958749	0.000345	0.9483265	0.000595	813.649615	4.073549	0.949691	0.993086

AdaBoost

AdaBoost is an ensemble learning method, it uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones

Tuning hyperparameters:

- learning_rate
- n_estimators: number of trees

AdaBoost	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		0.917582	0.000472	0.917464	0.000908	119.126291	6.713171	0.917203	0.983849
MinMaxScaler + Base		0.917582	0.000472	0.917464	0.000908	115.364578	0.348798	0.917203	0.983849
MinMaxScaler + SMOTE + Base		0.932709	0.000752	0.932554	0.001039	943.144256	29.594005	0.9340511	0.981976

Multi Layer Perceptron

A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together

Tuning hyperparameters:

- hidden_layer_sizes: number of hidden layers.

MLP	Parameters	mean train score	std train score	mean test score	std test score	mean fit time	std fit time	f0.5 score	pr auc score
Base		0.876214	0.005335	0.876115	0.005016	646.127894	194.734635	0.884666	0.973610
MinMaxScaler + Base		0.914479	0.002467	0.913958	0.002515	645.883598	44.239567	0.917118	0.984435
MinMaxScaler + SMOTE + Base		0.93672	0.000185	0.935511	0.000599	2029.161879	32.893923	0.935723	0.984072
MinMaxScaler + Base (Tuning)	hidden layer sizes = (50,50,50)	0.924010	0.002080	0.922807	0.002384	374.981939	2.476638	0.922871	0.987012

Conclusion

So far, we have introduced and tackle our classification problem by using different pipeline, both in data analysis and machine-learning model. Several techniques for handling imbalance and leaky data are conducted inside our solution. And the results, as you can see in the tables, the performance of the best model, the XGBoost, satisfies our demand pretty well.