Loan Default Risk Prediction Model Project

Big Data Section D, 2024 Spring

Team member: Zeren Gesang(zg2442), Chelsea Liu(ql2547), Yuewei Shi(ys5795)

05/05/2024

NYU

Content

1. Brief Introduction
2. Data Cleaning
3. EDA
4. Feature Selection
5. Model Training and Prediction
6. Results and Comparison

1. Brief Introduction

Project statement: Our project is designed to assess default risk for people with little or no credit history. Using data provided by financial companies, we use machine learning methods to predict a person's ability to repay, which can make it easier for people without a credit history to get a loan.

Objective: Explore the relationship between different variables, find important features, and build models. Finally, we choose the best model to predict the default risk.

1. Brief Introduction

Data Source: kaggle(Home Credit - Credit Risk Model Stability)

Technologies and Programming Languages:

Data Preparation and Processing: Python 3, NumPy, Pandas, Polars

EDA: python packages(matplotlib, seaborn, statistics,etc.)

Machine Learning Models: XGBoost, LightGBM, CatBoost

CASE_ID

	Internal source	External source
DEPTH 0*	static_0	static_cb_0
DEPTH 1*	applprev_1, other_1, deposit_1, person_1, debitcard_1	tax_registry_a_1, tax_registry_b_1, tax_registry_c_1, credit_bureau_a_1, credit_bureau_b_1, tax_registry_c_1
DEPTH 2*	applprev_2, person_2,	credit_bureau_a_2, credit_bureau_b_2

*
depth=0 - These are static features directly tied to a specific case_id.
depth=1 - Each case_id has an associated historical record, indexed by num_group1.
depth=2 - Each case_id has an associated historical record, indexed by both num_group1 and num_group2.

	Internal Files	External Files
DEPTH 0	train_static_0_0.csv train_static_0_1.csv	train_static_cb_0.csv
DEPTH 1	train_applprev_1_0.csv train_applprev_1_1.csv train_other_1.csv train_deposit_1.csv train_person_1.csv train_debitcard_1.csv	train_tax_registry_a_1.csv train_tax_registry_b_1.csv train_tax_registry_c_1.csv train_credit_bureau_a_1_0.csv train_credit_bureau_a_1_1.csv train_credit_bureau_a_1_2.csv train_credit_bureau_a_1_3.csv train_credit_bureau_b_1.csv
DEPTH 2	train_applprev_2.csv train_person_2.csv	train_credit_bureau_a_2_0.csv train_credit_bureau_a_2_1.csv train_credit_bureau_a_2_2.csv train_credit_bureau_a_2_3.csv train_credit_bureau_a_2_4.csv train_credit_bureau_a_2_5.csv train_credit_bureau_a_2_6.csv train_credit_bureau_a_2_7.csv train_credit_bureau_a_2_8.csv train_credit_bureau_a_2_9.csv train_credit_bureau_a_2_10.csv train_credit_bureau_b_2.csv

Feature Groups

P - Transform DPD (Days past due)
M - Masking categories
A - Transform amount
D - Transform date
T - Unspecified Transform
L - Unspecified Transform

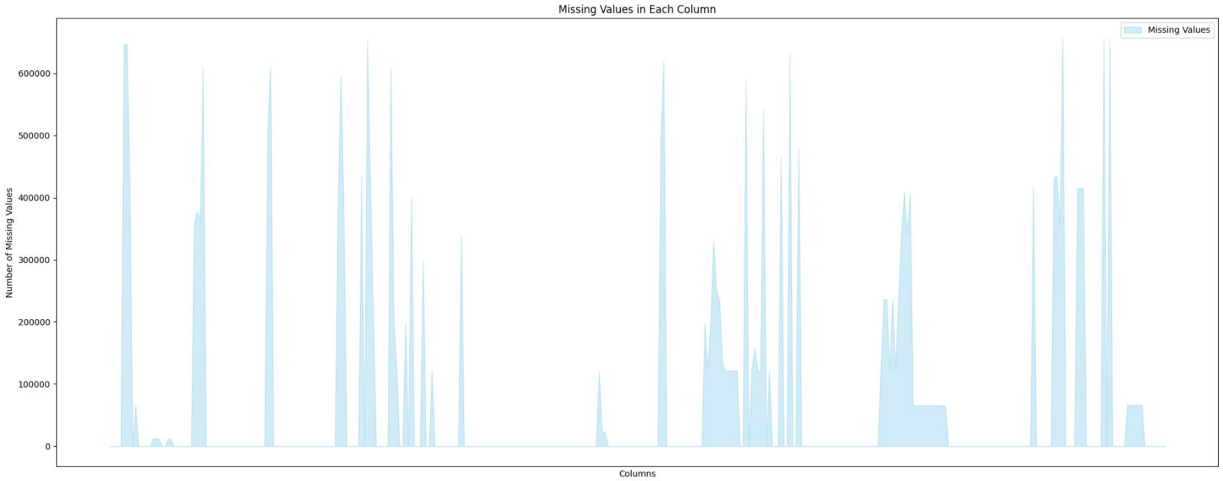
Columns Example

P - 'actualdpd_943P'
M - 'maritalst_385M'
A - 'pmtssum_45A'
D - 'dateofbirth_337D'
T - 'riskassesment_940T'
L - 'pmtcount_4955617L'

2. Data Cleaning

(1) Missing values: Set the threshold to 95%

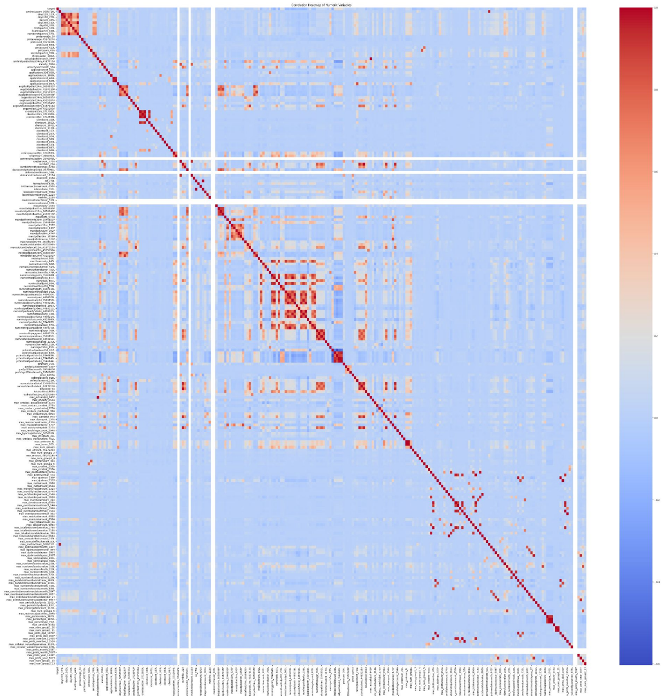
(2) Outlier detection: IsolationForest



2. Data Cleaning

(3)Unrepresentative classification features: only one category or > 200 categories

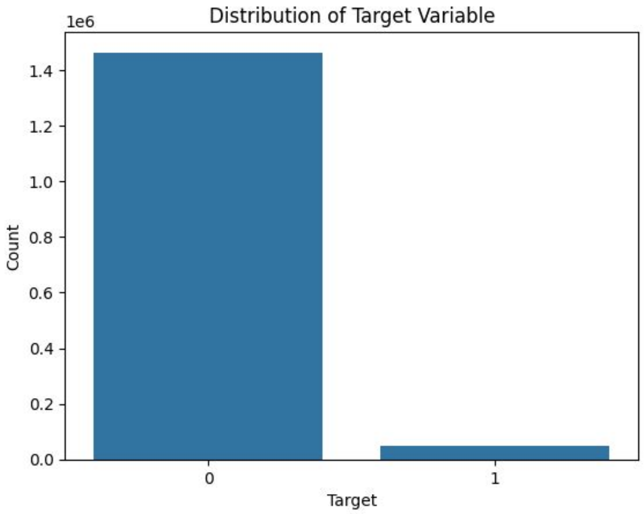
(4) Independent variables: Correlation coefficient

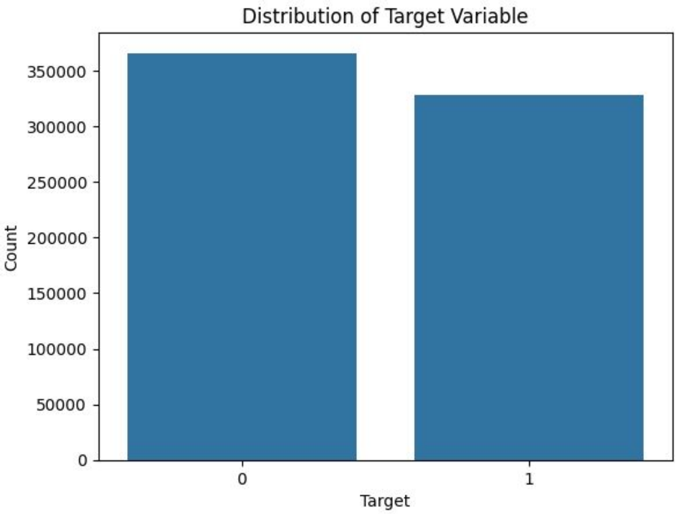


NYU

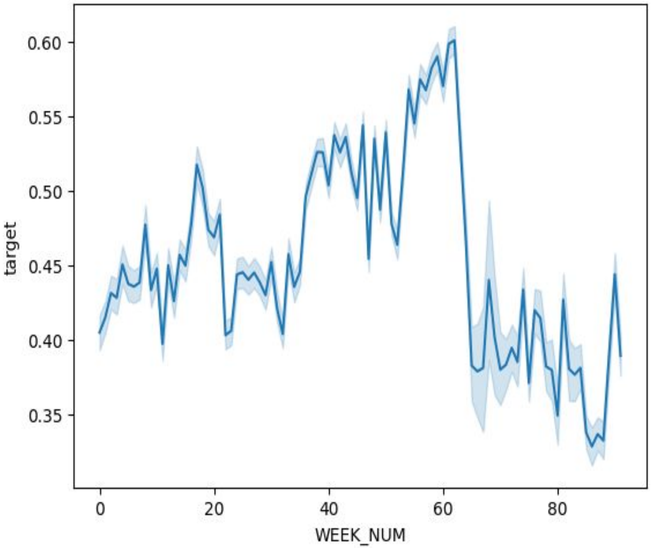
2. Data Cleaning

(5) Sample unbalance: oversample and undersample

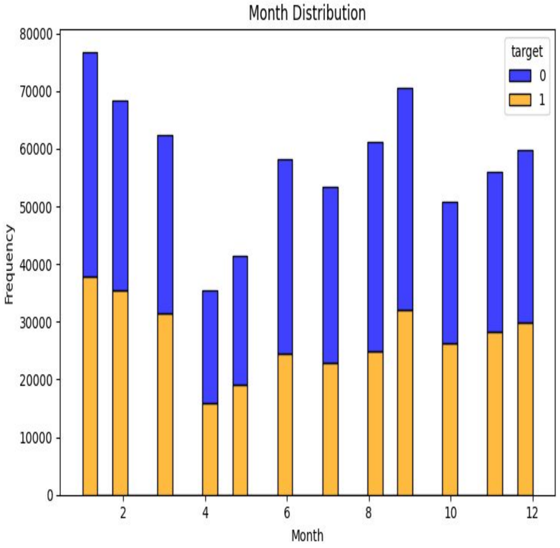




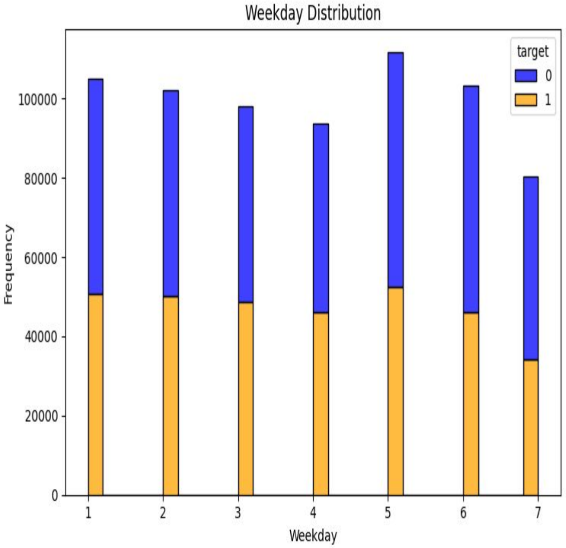
3. EDA



0.60
0.55
0.50
0.45
0.40
0.35
target
0
20
40
60
80
WEEK_NUM



Month Distribution
target
0
1
Frequency
Month

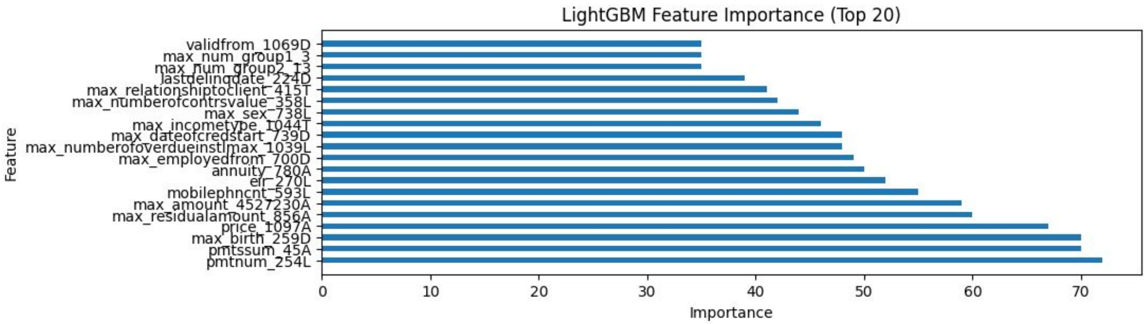


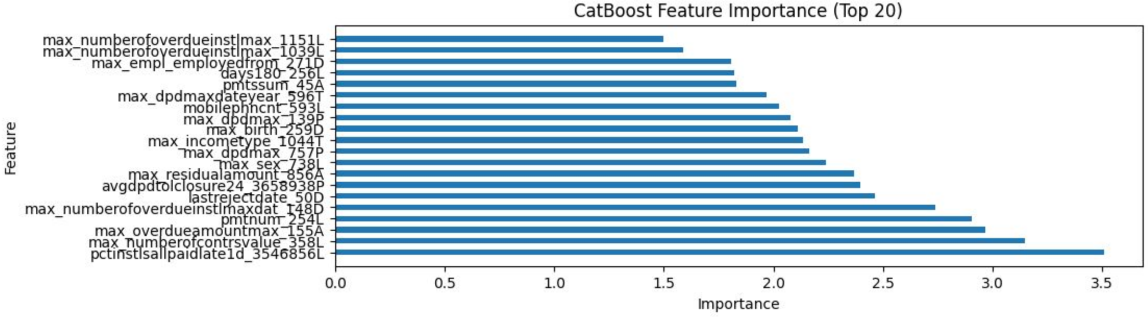
Weekday Distribution
target
0
1
Frequency
Weekday

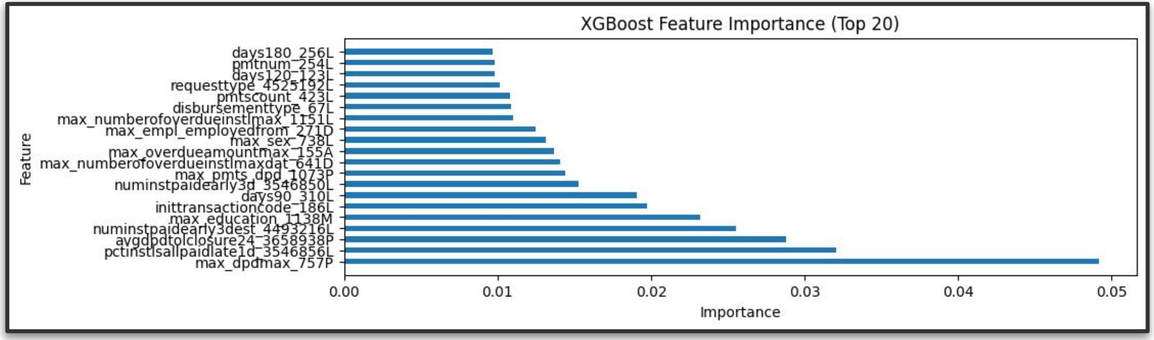
4. Feature Se

- Recursive Feature Elimination:
  More accurate,
  But time-consuming

- [Feature Importance]():
  Fast,
  But unstable

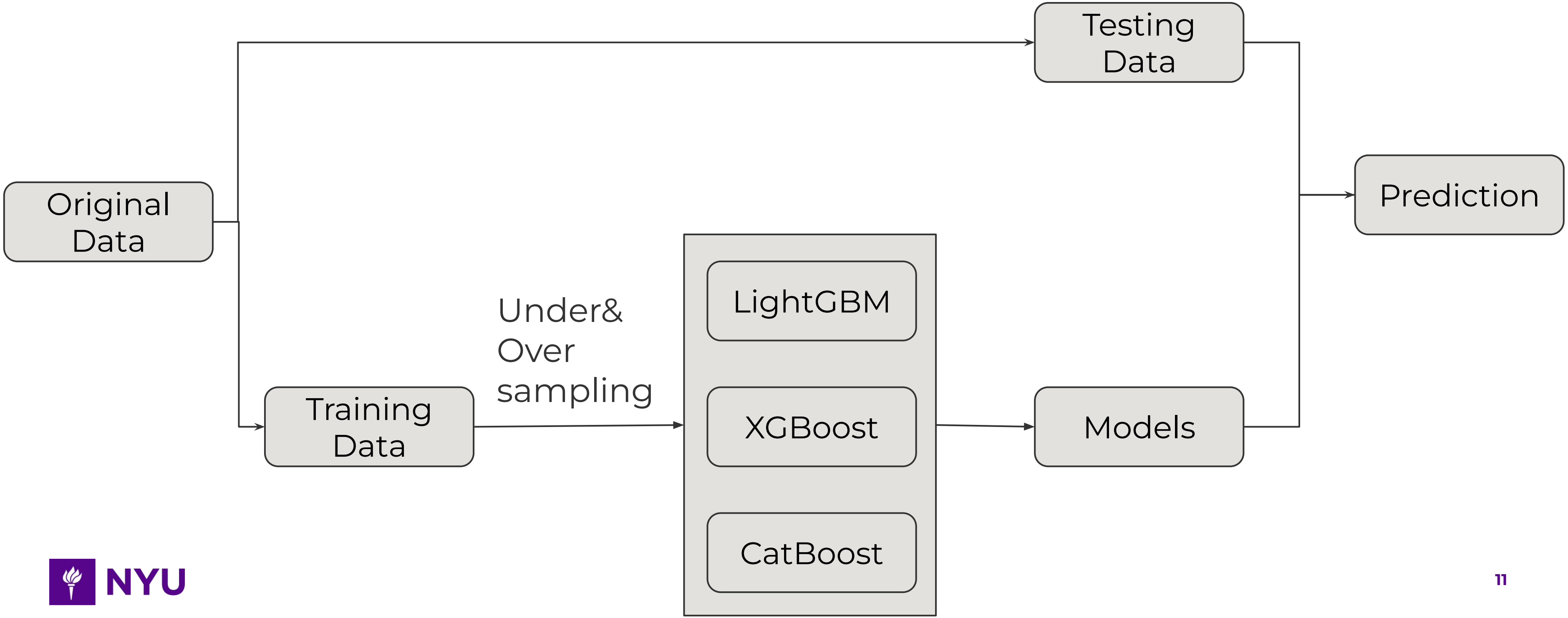






NYU

5. Model Training



Training and Prediction progress

1. **Model Training**: features (X) and their corresponding target values (y).
2. **Evaluation Metric**: "AUC" (Area Under the Receiver Operating Characteristic Curve)
3. **Model Fitting**: The fit method .Parameters: learning rate, depth of trees, and the number of trees (iterations) .
4. **Prediction**: the predict_proba method. Returns a two-column matrix , 1st column is the probability of the negative class (usually 0) and the 2nd column is the probability of the positive class (usually 1).
5. **Probabilities and Scores**:
   - The scores are the output from predict_proba.
   - High probability values indicate a higher confidence in the positive class and vice versa.
6. **Decision Threshold**: By default, threshold of 0.5. Greater than 0.5, the predicted class label will be 1 (positive), smaller than 0.5, it will be 0 (negative).

NYU

6. Conclusion Results

Models	Score of Validation Data	Score of Testing Data
LightGBM	0.85885	0.79597
XGBoost	0.85684	0.81964
CatBoost	0.85478	0.79013

Training Results

LightGBM Model:

Training data:

```
Training until validation scores don't improve for 100 rounds
[100]   valid_0's auc: 0.839801
[200]   valid_0's auc: 0.850962
[300]   valid_0's auc: 0.855293
[400]   valid_0's auc: 0.857417
[500]   valid_0's auc: 0.85808
[600]   valid_0's auc: 0.858279
[700]   valid_0's auc: 0.85863
[800]   valid_0's auc: 0.858827
Early stopping, best iteration is:
[778]   valid_0's auc: 0.858847
```

Testing data:

```
Out of Sample Accuracy: 0.7959659943323887
```

CatBoost Model:

Training data:

```
Default metric period is 5 because AUC is/are not implemented for GPU
0:      test: 0.7238580 best: 0.7238580 (0)    total: 72.4ms  remaining: 1m 12s
100:    test: 0.8297901 best: 0.8297901 (100)  total: 6.7s    remaining: 59.6s
200:    test: 0.8402348 best: 0.8402348 (200)  total: 13.3s   remaining: 52.8s
300:    test: 0.8457661 best: 0.8457661 (300)  total: 19.9s   remaining: 46.1s
400:    test: 0.8487443 best: 0.8487499 (398)  total: 26.4s   remaining: 39.5s
500:    test: 0.8509343 best: 0.8509343 (500)  total: 33s     remaining: 32.9s
600:    test: 0.8522608 best: 0.8522778 (599)  total: 39.5s   remaining: 26.2s
700:    test: 0.8531803 best: 0.8531803 (700)  total: 45.9s   remaining: 19.6s
800:    test: 0.8537449 best: 0.8537516 (799)  total: 52.4s   remaining: 13s
900:    test: 0.8543310 best: 0.8543333 (899)  total: 58.9s   remaining: 6.47s
999:    test: 0.8547843 best: 0.8547843 (999)  total: 1m 5s   remaining: 0us
bestTest = 0.8547842503
bestIteration = 999
```

Testing data:

```
Out of Sample Accuracy: 0.7901316886147691
```

XGBoost Model:

Testing data:

```
Out of Sample Accuracy: 0.8196366061010169
```

Training data:

```
[0]     validation_0-auc:0.74571
[100]   validation_0-auc:0.84104
[200]   validation_0-auc:0.85079
[300]   validation_0-auc:0.85411
[400]   validation_0-auc:0.85557
[500]   validation_0-auc:0.85646
[600]   validation_0-auc:0.85680
[700]   validation_0-auc:0.85696
[800]   validation_0-auc:0.85682
[900]   validation_0-auc:0.85682
[999]   validation_0-auc:0.85684
Completed training fold 5/5
```

Sample Case Prediction

LightGBM :

```
case_id
1911587      0.028370
1679928      0.098218
149421       0.181099
992962       0.049547
1812637      0.293403
dtype: float64
```

XGBoost:

```
case_id
1911587      0.020822
1679928      0.105364
149421       0.135711
992962       0.047066
1812637      0.249646
dtype: float32
```

CatBoost:

```
case_id
1911587      0.052600
1679928      0.165412
149421       0.270684
992962       0.082798
1812637      0.260636
dtype: float64
```

Model Comparison

Feature	XGBoost	LightGBM	CatBoost
Algorithmic Approach	Level-wise tree growth	Level-wise tree growth	Ordered boosting, handles categorical features natively
Categorical Feature Support	Requires preprocessing	Converts categories to integers, uses them directly	Excellent support for categorical features without preprocessing
Speed and Scalability	Highly efficient on CPU/GPU; good for moderate to large datasets	Faster on large datasets due to histogram optimizations	Competitive but can be slower due to complex calculations
Ease of Use	Extensive parameter tuning required for optimal performance	Easier to use with defaults; less flexible for categorical data	Easy to use with categorical data; provides detailed prediction explanations
Model Performance	Excellent, with proper tuning can achieve great results	Comparable or better on large datasets; very efficient	Comparable, excels with datasets that have complex categorical features
Handling Overfitting	Regularization features (L1, L2) to help prevent overfitting	Similar parameters to control overfitting, e.g., `max_depth`	Uses algorithmic approaches to minimize overfitting risk
Community and Documentation	Very well-documented with a large user community	Well-supported with growing documentation and community	Well-documented, though newer, with a growing user base

Link to project code

Main Project :
https://github.com/ChelseaLiu0822/LDRPM-Loan-Default-Risk-Prediction-Model/blob/main/LDRPM_Loan_Default_Risk_Prediction_Model_Project_FinalVersion1.ipynb

Main Branch(with README file):

https://github.com/ChelseaLiu0822/LDRPM-Loan-Default-Risk-Prediction-Model/tree/main

Thank you!

Team member: Zeren Gesang(zg2442), Chelsea Liu(ql2547), Yuewei Shi(ys5795)

05/05/2024

NYU