Home Credit - Credit Risk Model Stability

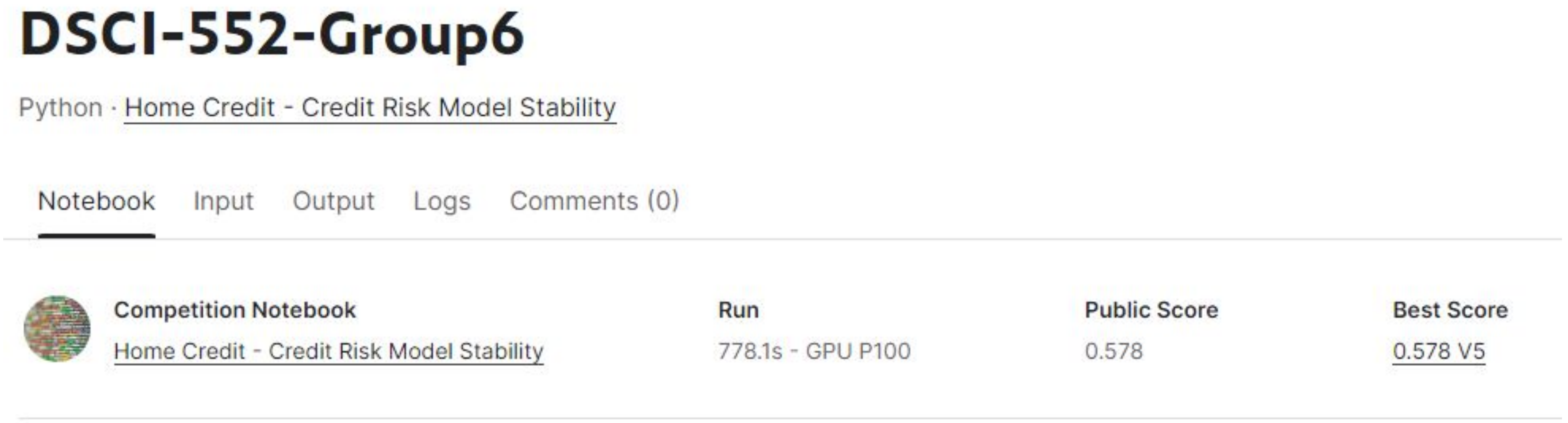
Group 6: Aanandhi Sonduri Panthangi,
Prasad Malwadkar,
Shivani Rajesh Shinde,
Shreya Padmanabhan

INTRODUCTION

1. The absence of a credit history can lead to loan denials, affecting younger individuals or those who prefer cash transactions.
2. Consumer finance providers rely on data to accurately determine clients' repayment capabilities, using statistical and machine learning methods like scorecards to predict loan risk.
3. Regular updates to scorecards are necessary due to the dynamic nature of client behaviors, ensuring stability and performance over time.
4. Home Credit, established in 1997, focuses on responsible lending to individuals with limited or no credit history, promoting financial inclusion.
5. The competition aims to predict default risks, favoring stable solutions over time, which could enhance loan accessibility for historically underserved populations.

Key Accomplishments

Created a model to predict which clients are more likely to default on their loans with an accuracy of 57.8%.



DSCI-552-Group6

Python · Home Credit - Credit Risk Model Stability

Competition Notebook	Run	Public Score	Best Score
Home Credit - Credit Risk Model Stability	778.1s - GPU P100	0.578	0.578 V5

Description of Dataset

1. Base Tables: These tables store fundamental information about each observation, including a unique identifier called case_id.
2. Unique Identification: The case_id serves as a unique identifier for every observation, enabling seamless integration with other tables in the dataset.
3. Foundation for Analysis: Base tables form the foundational layer of the dataset, providing essential context for each observation.
4. Joining Other Tables: They are crucial for joining with other tables to incorporate additional information from diverse data sources.
5. Facilitating Prediction: The base tables play a pivotal role in facilitating the analysis and prediction of default risks for clients based on the available internal and external information.

Our Approach - Data Wrangling

Define Classes for Data Processing: Pipeline and Aggregator

1. **Define Functions for Data Reading and Feature Engineering**: Several functions are defined to read data files, perform feature engineering, and convert data to a more memory-efficient format. These functions include `read_file`, `read_files`, `feature_eng`, `to_pandas`, and `reduce_mem_usage`.
2. The code defines two classes: `Pipeline` and `Aggregator`. These classes encapsulate methods for data preprocessing and feature engineering, respectively.
3. **Pipeline**->
   a. *`Drop unnecessary columns.`*
   b. *`Drop columns with more than 70% missing values`*
   c. *`Drop columns with only one unique value or more than 200 unique values`*
4. **Aggregator->**
   a. *`Method to aggregate numerical features,date features,string features,count features,other features`*

Data Wrangling

1. Feature Engineering
   a. `Add month and weekday features based on "date_decision"`
   b. `Join additional depth DataFrames`
   c. `Handle dates using Pipeline method`
2. Reduce Memory Usage
   a. `Iterate through all the columns of a dataframe and modify the datatype to reduce memory usage.`

Model Comparison: LightGBM vs. CatBoost

LightGBM

- - Requires extensive feature engineering
- - Vulnerable to overfitting
- + Fast training time
- + High memory efficiency

CatBoost

- + Native categorical feature support
- + Robust to overfitting
- - Slower training time
- - Higher memory usage

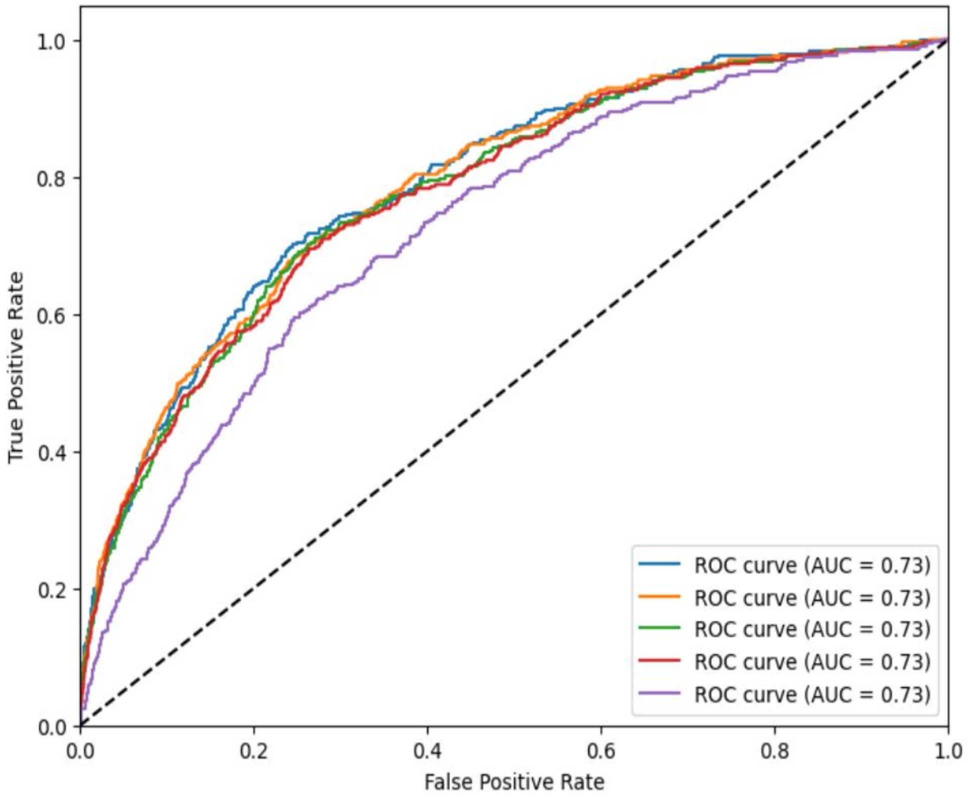
Overall trade-off: While LightGBM is faster and more efficient, CatBoost has higher accuracy and more predictive power.

Our Approach - Model and Parameter Tuning

1. Use of CATBoost Classifier.

   Evaluation Metrics used: Area Under the Receiver Operating Characteristic (ROC) Curve (AUC)

   Learning Rate: 0.03



Ablation Studies

1. Calculate Time Differences, Convert to Total Days, and Drop Unnecessary Columns. Converted time differences to total days for better representation. Dropped unnecessary columns led to removal of redundant or irrelevant information, reducing the complexity of the dataset.
2. Drop Columns with More Than 70% Missing Values. Investigated the impact of dropping columns with a high percentage of missing values. Compared the model performance with and without this step to understand its contribution.
3. Drop Columns with Only One Unique Value or More Than 200 Unique Values.
4. Explored the impact of dropping columns with very low or high cardinality. Evaluated how dropping such columns affects model performance, especially in terms of reducing overfitting or noise.
5. Compare model performance metrics (e.g., accuracy, AUC-ROC, stability metric) before and after each data wrangling step.
6. Used feature selection techniques to identify which features are most informative after each data wrangling step.
7. Analyzed any changes in the distribution of features or target variable after data wrangling.

Conclusion & Future Plans

Avenues for Further Research

- Combining/stacking multiple models to improve performance
- Feature engineering for more informative parameters