Predicting Credit Default by Gradient Boosting Trees

CS 5824 Advanced Machine Learning Term Project Report

Sha Li, Xiao Guo, Yi Lu

ABSTRACT

Credit default prediction is a critical task for financial institutes to access default risks before approving loan subjects to potential customers. However, even with the large available data and advanced prediction techniques, it's still challenging to obtain accurate default predictions, since (1) customers' behaviors are usually dynamic and evolving, (2) available real-world data is usually dirty with missing, noisy and abnormal values and thus exerting negative impacts on predictive models, (3) traditional basic models are too weak to obtain a clear decision boundary for complicated tasks, and (4) results come out from ML models usually lack interpretability and its hard to convert them to insights. To address these challenges, we employ ensemble learning techniques to construct gradient-boosting tree models for more precise credit default prediction, and enable model results to be interpretable. Specifically, we use a public industrial-scale dataset that tracks millions of customers' historical financial behaviors for our study. To get promising predictions, we perform exploratory data analysis to investigate the data, conduct extensive feature engineering to capture and abstract useful features, refine models by parameter tuning, and interpret results by feature exploration and Shapley additive explanations. The experiment results demonstrate that, our proposed solution, LightGBM outperforms other baseline methods including Xgboost, CatBoost, Adaboost, Random forest, Extratrees and vanilla gradient-boosting trees.

Contribution

All three team members discussed and determined the project topic, models and report writing. The specific contributions are listed below:

- For the EDA part, Sha Li was responsible for the analysis of categorical variables, Delinquency, Spend and Payment variables; Xiao Guo and Yi Lu completed the analysis of Balance and Risk variables. Each member prepared the corresponding codes. After obtaining the EDA results, we discussed and summarised observations and each member wrote his/ her part of analysis.
- For feature engineering, members discussed and made plans on techniques/ steps we would use. Sha Li completed the programming and report writing.
- For model construction, Sha Li implemented LighGBM and baseline models. Yi Lu implemented CatBoost and Xiao Guo Implemented Xgboost.
- For the paper writing, apart from the above-mentioned parts who are directly responsible for, Sha Li wrote the abstract, methodology, results and evaluation sections. Xiao Guo and Sha Li wrote the problem formalization, data source introduction, and evaluation metrics sections. Yi Lu and Sha Li wrote the introduction and conclusion sections.
- All team members reviewed and polished the final report.

PREDICTING CREDIT DEFAULT BY GRADIENT BOOSTING TREES

Sha Li, Xiao Guo, Yi Lu
Department of Computer Science
Virginia Tech
Blacksburg, VA

ABSTRACT

Credit default prediction is a critical task for financial institutes to access default risks before approving loan subjects to potential customers. However, even with the large available data and advanced prediction techniques, it's still challenging to obtain accurate default predictions, since (1) customers' behaviors are usually dynamic and evolving, (2) available real-world data is usually dirty with missing, noisy and abnormal values and thus exerting negative impacts on predictive models, (3) traditional basic models are too weak to obtain a clear decision boundary for complicated tasks, and (4) results come out from ML models usually lack interpretability and its hard to convert them to insights. To address these challenges, we employ ensemble learning techniques to construct gradient-boosting tree models for more precise credit default prediction, and enable model results to be interpretable. Specifically, we use a public industrial-scale dataset that tracks millions of customers' historical financial behaviors for our study. To get promising predictions, we perform exploratory data analysis to investigate the data, conduct extensive feature engineering to capture and abstract useful features, refine models by parameter tuning, and interpret results by feature exploration and Shapley additive explanations. The experiment results demonstrate that, our proposed solution, LightGBM outperforms other baseline methods including Xgboost, CatBoost, Adaboost, Random forest, Extratrees and vanilla gradient-boosting trees.

1 Introduction

Credit Default Prediction (CDP) refers to a task in the financial field that leverages statistical and/or machine learning models to forecast the possible default risk of credit loan subjects [1]. With the purpose of reducing financial risks, CDP is adopted by almost every financial institution to proactively identify customers who are willing and able to pay back their charges on time before approving loans.

A precise CDP model prevents financial institutes from potential losses by estimating the possible default risk and eliminating the new credit proposals if the model identifies it as "default". Traditionally, CDP is done by two steps, (1) extracting and abstracting characteristics of "default" and "not default" customers, (2) constructing models that can automatically distinguish these two types of

customers, and (3) dynamically analysing customers' credit profiles and historical financial behaviors to update the risk probability. However, the process of CDP is challenging since the behavior of defaulting customers is highly uncertain and volatile, and customers' data for establishing predictive models are usually dirty and contain missing, noisy, abnormal and out-of-distribution data.

Our project aims to investigate such a challenging task by leveraging machine learning techniques to accurately predict whether a user will cheat on his/ her credit subjects or not. We explore and identify the critical features that are closely related to the target behavior - default or not, by leveraging an industrial-scale dataset that tracks millions of customers' historical financial behaviors. Many solutions such as decision trees, support vector machines [2], and classifiers consensus system [3] have been proposed to predict credit default rates in the past few years. However, instead of using general ML models, we resort to ensemble strategies in the project to tackle this complicated problem. Aside from simply developing ML models for the credit default prediction, we will also answer the following questions: 1) how and why ensemble models can gain improvements over existing solutions; 2) what are the most correlated/ contributing factors driving defaults; 3) can feature engineering help improve prediction performance; and 4) to what extent our results can help financial institutes to prevent consumer default.

Specifically, we utilize American Express (Amex) as a case study. Here are the main reasons: 1) Amex posted their anonymized data on Kaggle [4], which is accessible for the pubic; 2) Amex is one of the biggest and most well-known card issuers in the United States, it has a great number of representative customers and corresponding data as well.

Contributions. In summary, our project has the following contributions:

- Before constructing machine learning models, we perform a thorough exploratory data analysis (EDA) on the dataset and generate insights for further data processing and feature engineering.
- Considering that the dataset is imbalanced and contains uneven missing value distributions, we adopt multiple feature engineering techniques to transform and generate new features to improve the model accuracy.
- We develop and compare different gradient boosting trees (GBDT) models to address the complicated prediction task.
- We enable the outputs from the model to be interpretable and explainable.

The remainder of the report is organized as follows: we formalize the problem in §2, explain our methodology - including the system design, essential phrases and selected models in §3. Next, we detail our exploratory data analysis, feature engineering, model construction and implementation steps in §4, §5 and §6 respectively. We provide performance evaluation and report the results in §7. Finally, we sum up the report and conclude in §8.

2 Problem Formalization

Binary Classification. We formalize the credit default prediction problem as a binary classification task. Let $< \mathcal{D} =< X_i, y_i > |0 < i < N, y_i \in [0, 1] >$ represents a given dataset of N customers' statement records. One customer may have multiple statements obtained from different periods. X_i is a set of features for customer i that describes his/ her financial behaviors/ characteristics from M aspects. y_i is a binary value indicating whether the customer i will default or not, $y_i = 1$

indicates that the customer defaults, while $y_i = 0$ indicates not. The solution we propose is to construct and learn a mapping function f so that predictors X_i can be transformed to the target y_i for $i-th$ customer, precisely. Thereafter, the trained function f can be utilized to predict a potential customer's future default risk.

Optimization Objective. Suppose *y* is the ground truth of customer *i* and $\hat{y}$ is its corresponding model output, we minimize the categorical cross-entropy loss function [5] (Eq. (1)) to get the estimated parameters of our predictive model.

$$\ell(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i) \tag{1}$$

where *n* is the total number of training samples.

3 Methodology

In the project, we harness machine learning techniques to investigate default prediction by using an industrial-scale dataset. We treat the problem as a binary classification problem. The binary predicted targets - whether customers will pay back their credit card balance amount or not in the future are made based on their profiles and historical behavior data. This section details the methodology we use to address the problem.

3.1 System Design

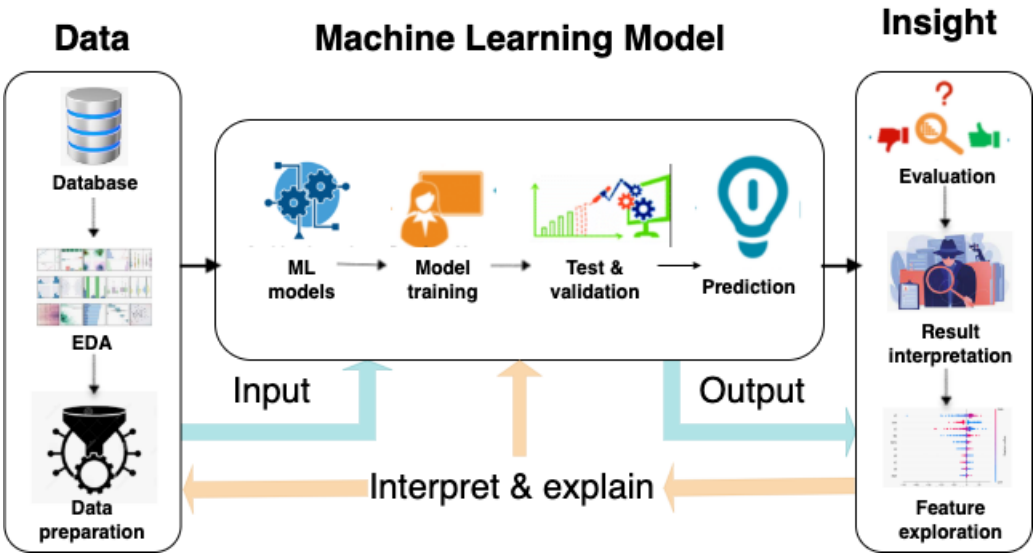

Figure 1: The systematic overview of our solution.

We provide a systematic framework for our solution in Fig. 1. Typically, the key phases being involved can be described as follows.

- **Exploratory data analysis.** As an initial inspection of the data, we employ EDA to preliminarily investigate and understand the data at hand. This step incorporates a series of

statistical summarization and visualisation techniques to uncover patterns, check correlations, spot anomalies, test hypotheses and assumptions [6].

- **Data pre-processing**. Data cleaning and imputation are two major tasks in this step. The objectives are to check and drop duplicate records, remove variables that contain a great portion of missing values, and conduct imputation on some variables.
- **Feature engineering**. The dataset offers time-series statement records for customers, and both categorical and numerical features are included. As such, feature engineering techniques are employed to encode, transform and generate features that guide the ML model for accurate prediction.
- **Model construction, training and refinement.** We construct Gradient boosting decision trees (GBDT) models for the classification tasks (reasons are detailed in §3.2).
- **Evaluation and result interpretation.** Model performances are assessed by a series of evaluation metrics. Results and features exploration are followed to discuss and extract insights for the topic - default prediction and future research.

3.2 Selected Model

Customer default prediction is a complicated and challenging task since multiple aspects of customers should be taken into consideration. And the prediction requires high accuracy and generalization capability. To achieve these goals, we will explore ensemble strategy to better address the problem. Specifically, we focus on gradient-boosting decision trees (GBDT) in the project. Gradient boosting decision trees (GBDT) have been proven to achieve state-of-the-art performance for a wide-range of ML problems [7]. GBDT is an ensemble strategy to construct an accurate and powerful classifier by composing a series of weak classifiers sequentially [8].

Formally, given a training dataset $\mathcal{D} = \{(x_i, y_i)\}(|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, the objective of gradient boosting is to find a function $\hat{F}(x)$ to approximate the output targets y from input values x by minimizing the given loss function $\ell = (y, F(x))$, as a weighted of sum of a series weak classifiers [8]:

$$F_t(x) = F_{t-1}(x) + \gamma_t h_t(x) \tag{2}$$

where t is a specific iteration step, $h_t(x)$ is the weak leaner at *t*-iteration with a corresponding weight γ_t. Gradient boosting learns in an iterative manner. At each step, a new weak model is trained to predict the "error" of the current model. To illustrate, in the first iteration, the approximation function is obtained by:

$$F_0(x) = arg\min_{\theta} \sum_{i=1}^{n} \ell(y_i, \theta) \tag{3}$$

Where θ is a set of model parameters. Recursively, subsequent models are expected to minimize [7]:

$$(\gamma_t, h_t(x)) = arg\min_{\gamma,h} \sum_{i=1}^{n} \ell(y_i, F_{t-1}(x_i) + \gamma h(x_i) \tag{4}$$

However, solving the above optimization directly is computationally infeasible in general. Instead, gradient boosting treats each h_t as a greedy step in a gradient descent optimization. As such, h_t is trained on a new dataset $\mathcal{D} = \{x_i, r_{ti}\}$ where the pseudo-residual [9] r_{ti} are obtained by:

$$r_{ti} = [\frac{\partial \ell(y_i, F(x))}{\partial F(x)}]_{F(x)=F_{t-1}(x)} \tag{5}$$

which can be taken as "F_t is attempting to correct the errors of its predecessor F_{t-1}" [10].

Regarding the default prediction problem, here are some reasons why we choose GBDT:

- The boosting process allows the weak classifier learns from the prediction errors of the previous iteration, which reduces the bias; and the final classifier takes a weighted average of many weak models, which reduces the variance of prediction.
- Upon our primary checking, the dataset we use contains multiple missing data in rows and columns, so it is not a good idea to delete them directly. Fortunately, most of GBDT models have intelligent algorithms to automatically handle missing values through the learning process.
- Tree-based models have special algorithms to calculate feature importance, which helps verify and determine what kind of features would be useful for the prediction.

Specifically, we try three GBDT algorithms in our project: eXtreme Gradient Boosting (XGBoost) [11], Light Gradient Boosting Machine (LightGBM) [12], and Categorical Boosting (CatBoost) [13]. In summary, XGboost is a scalable GBDT algorithm. It contains a sparsity-aware algorithm to find the best split, employs a weighted quantile sketch to approximate tree learning, and incorporates a regularized model to prevent overfitting [11]. LightGBM utilizes Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to speed up training and improve accuracy. In particular, GOSS is a subsampling technique to increase the importance of features with higher uncertainty, while EFB bundles sparse features into a single feature [12]. Catboost constructs symmetric trees to overcome overfitting automatically handles categorical features, and leverages the ordered boosting, which is a permutation-driven approach to avoid target leakage and reduce prediction shifts that occur during training [13].

3.3 Evaluation Metrics

General performance metrics. As the credit default prediction is formalized as a binary classification problem, general metrics for evaluating classification tasks, such as *accuracy, precision, recall, F1_score, classification report, AUC scores* are used to assess the performance of our models.

Rank-aware metric. In addition, we also consider rank-aware metrics since we want the model can rank high-risk customers with a higher priority as a warning. We include a specialized ranking evaluation metric $R = 0.5 * (G + D)$ for performance assessment. Specifically, G is Normalized Gini Coefficient [14] and D is the default rate (positive rate) captured within the highest-ranked 4% of the predictions. Since the dataset is imbalanced, sub-metrics G and D assign a weight for negative labels to adjust down-sampling [15].

4 Exploratory Data Analysis

In this section, we first provide an overview of the dataset we use for the project, followed by an extensive exploratory data analysis, and detailed observations that we obtain from the analysis.

4.1 Data Source

Our dataset is provided by American Express, which is published on Kaggle [4]. The dataset contains over 50 GB data, around 5.5 million rows of records (5,531,451 in total). Each record

represents one statement for a customer in a specific period. To sum up, the dataset has the following key characteristics:

- The dataset contains statement records for 458913 customers.
- There are 191 columns, describing 191 aspects of the customers' profiles.
- Each customer has an unique 'customer_ID'.
- Each customer has multiple statements that are associated with different periods. Therefore, this dataset tracks historical behaviours of customers.
- The dates of statements spanned from 2018-01-01 to 2018-03-31.
- The dataset consists of customer features from five aspects: delinquency, spend, payment, balance and risk. Each aspect has multiple anonymized variables. Specially, all features are anonymous with the first capital of 5 aspects plus an integer.
- The target feature of customers is represented with binary values, 1 means default while 0 means normal (not default).
- Among all 189 features (except 'customer_ID'), 11 of them are categorical variables, while the rest are numerical.

To illustrate, Figure 2 shows two statement records of a customer at different dates.

customer_ID	S_2	S_3	B_17	D_114	P_3	R_2
0000099d6bd597052cdcda90ffabf56573fe9d7c79be5f...	2017-03-09	0.124035	NaN	1.0	0.736463	0.006204
0000099d6bd597052cdcda90ffabf56573fe9d7c79be5f...	2017-04-07	0.126750	NaN	1.0	0.720886	0.006206

Figure 2: Data sample. These are 2 records for a specific customer who's labeled as *1 - default*. Specifically, *S_2* is the statement date, *S_3, B_17, D_114, P_3*, and *R_2* are variables related to his spend, balance, delinquency, payment and risk aspect respectively. *D_114* is categorical variable, *S_2* is date data, and others are normalized numerical variables. Features are anonymous (without explicit meanings) and missing data is exist.

4.2 Preliminary Checking

Data size. The dataset contains 5531451 rows, 191 columns. Among them, 185 variables are in $float64$ format, 2 variables are $int64$ and 4 are $object$. It costs $7.9+$ GB memory for the sever to load the whole dataset. Considering that the dataset is quite large, special data compression techniques should be applied to reduce its size, memory cost and speed up calculation.

Missing values. The problem of missing value is quite common in real-world datasets. Our dataset is not an exception. Missing value can bias the results of the machine learning models and/or reduce the accuracy of the model [16]. Therefore, before going deep into the EDA steps, we first figure out how many variables have missing values, and report their percentages. Typically, for variables with over 70% missing values, we drop such variables; for variables having fewer than 20% missing values, we do data imputation. Upon checking, there are 29 variables having over 50% missing values. The statistical distribution of missing values for all variables is shown in Fig 3.

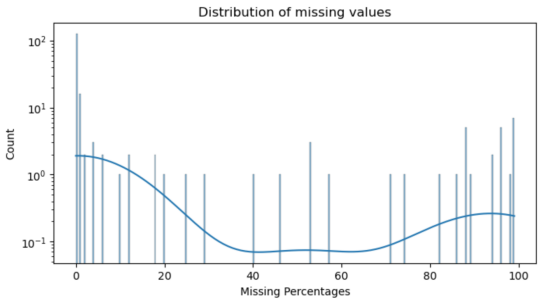

Figure 3: The dataset contains varying degrees of missing values among 191 variables.

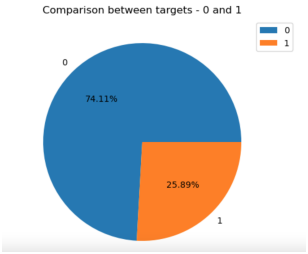

Figure 4: The dataset is highly imbalanced with 74.11% 'not default' customers and 25.89% 'default' customers.

4.3 Categorical Variables Analysis

4.3.1 Target

For the target variable, there are two categories, i.e., 'not default' and 'default', represented by '0' and '1' respectively. However, upon checking the dataset, the target has imbalanced distribution with 118,828 (25.8934%) customers are labeled as "default" while 340,085 (74.1066%) are labeled as "normal" (shown in Fig. 4).

Insight: to handle the imbalance of dataset, we apply Synthetic Minority Oversampling Technique [17] (a.k.a SMOTE) to oversample the dataset.

4.3.2 Categorical - Target Relationship

For the categorical variables, we plot their distributions with different targets to preliminarily check their relationships. From Fig.5 we can observe that, (1) most variable have uneven distributions; (2) variable *B_30* has dominant values of 0, (4) missing values exist, but only occupy small portions (except *D_66*), (4) when values are less than 4.0, *B_38* has more samples with $target = 0$, which is in contrary with values that are larger than 3.0.

4.4 Variables Analysis by Aspects

In this section, we focus on numerical variables in each feature group. Generally, we conduct (1) ***Quantitative-quantitative analysis*** to identify highly correlated variables. These variables can be considered redundant and partially removed by feature selection techniques such as principal component analysis (PCA) [18]. (2) ***Quantitative-categorical analysis*** to identify variables that have favorable relationships with the target. These variables should get more attention since they might have more predictive power with respect to our task. Specifically, we measure such "correlation" by Pearson correlation coefficient (PCC) [19]. We consider variables with $|PCC| \geq 90$ are "highly correlated", and $|PCC\_with\_target| \geq 0.5$ are "favorable related". Note that, we are aware of the limitations of PCC, but we believe this works well for our purpose to conduct simple and preliminary investigation on the dataset at this moment. We provide all visualizations discussed within this section in the Appendices A, but only list major observations and insights here.

4.4.1 Delinquency Variables ('D_*')

Detailed plots are shown in Appendix A. Fig. 16 shows the distribution of variables, Fig.24 depicts the correlations among features.

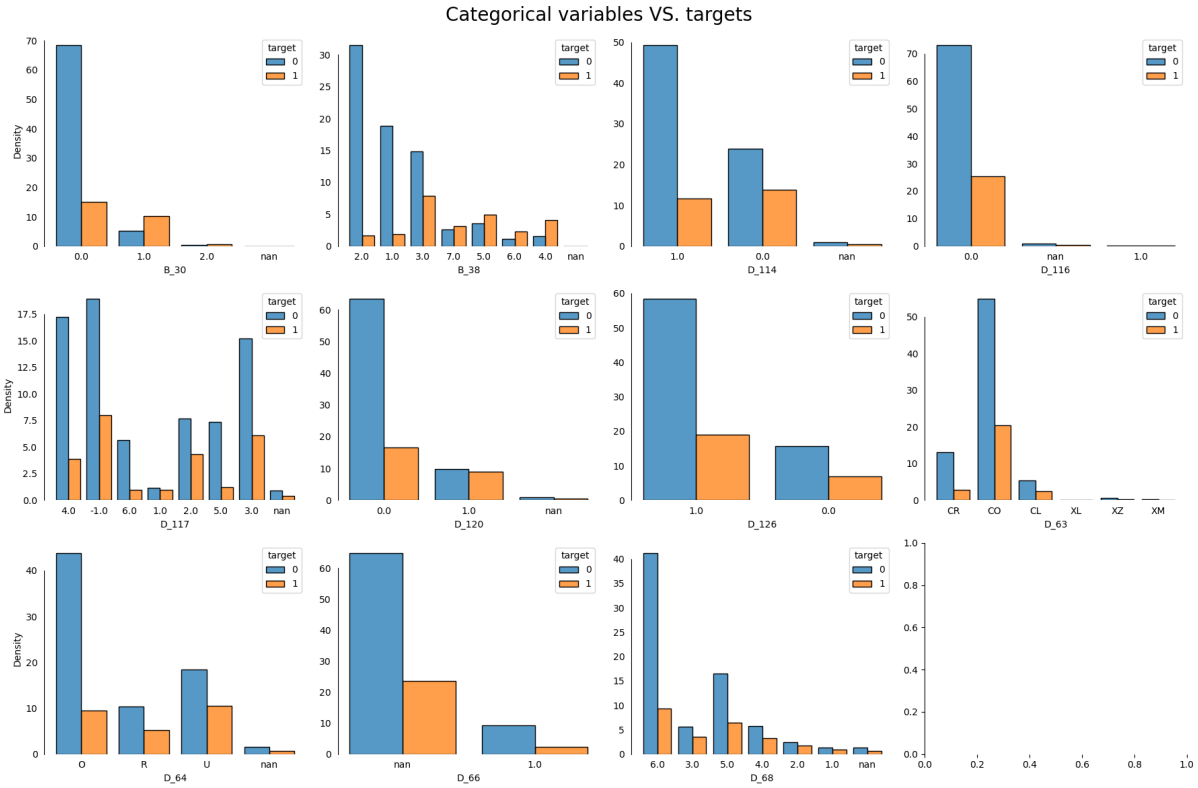

Figure 5: Distributions of categorical variables versus targets.

Observations:

- Many 'D_*' variables have missing values.
- Most of 'D_*' variables exhibit skewed distribution
- The following variables have high PCC ($\geq$ 90) with others: *D_58, D_74, D_75, D_62, D_77, D_73, D_136, D_62, D_104, D_103, D_119, D_108, D_139, D_141, D_143*
- The following variables have favorable relationships with the prediction targets: *D_48, D_55, D_61, D_44*

4.4.2 Spend Variables 'S_*'

Detailed plots are shown in Appendix A. Fig. 17 shows the distribution of variables, Fig.23 depicts the correlations among features.

Observations:

- 'S_2' is a date-type variable, representing the statement dates, so it's excluded for analysis in this step
- All variables display skewed distribution
- Almost all variables have extreme end of value (towards 0)
- Variables *S_3, S_7, S_24, S_22* have high correlations
- No variables have favorable relationships with targets, but *S_3* and *S_7* have $|PCC\_with\_target| \geq 0.3$

4.4.3 Payment Variables 'P_*'

Fig.21 depicts the correlations among features. There are only three variables in the "payment" group. Fig. 6 shows the distribution of these variables, from which we can find that *P_2* is slightly skewed, *P_3* follows normal distribution and *P_4* appears to be bimodal distributed. Only *P_2* has favorable correlation ($PCC\_with\_target = -0.67$) with the target.

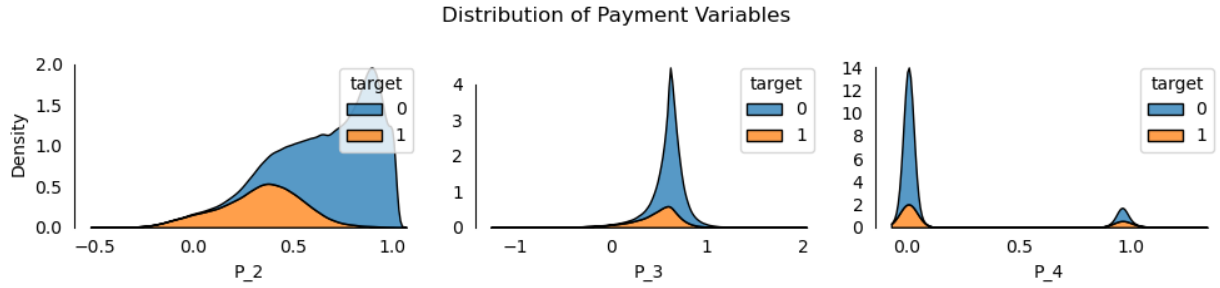

Figure 6: Distribution of "payment" variables.

4.4.4 Balance Variables 'B_*'

Fig. 20 shows the distribution of variables.

Observations:

- *B_33, B_8, B_20* have similar distribution patterns: values are either close to 0 or 1.
- Over half variables have dominant values near 0
- Almost all variables have extreme end of value (towards 0)
- Variables *B_1, B_11, B_33, B_37, B_2, B_13, B_12, B_14, B_15, B_8, B_23* have high correlations
- Variables *B_2, B_3, B_7, B_9, B_18, B_33* have favorable correlations with target

4.4.5 Risk Variables 'R_*'

Fig. 19 shows the distribution of variables, Fig.22 describes the correlations.

Observations:

- All variables exhibit skewed distribution
- Except for variables *R_27, R_12* whose values are close to 1, the remaining variables have values near 0
- No variables have high correlation; no variables have favorable correlation with target

5 Feature Engineering

Feature engineering [20] is a process to transform raw data into features that can be used for predictive models. The ultimate goal of feature engineering is to prepare features that best fit the ML model so that enhancing the predictive performance. Especially, for complicated tasks like our project, where the dataset contains a considerable quantity of anonymous variables with missing values in various degrees. This exerts adverse effects on both model learning and prediction. A reasonable feature engineering process can pull out useful and meaningful features that favour

precise predictions. Additionally, since the dataset records statements for different time periods, ***aggregated*** features should be generated in a correct manner to represent a customer. In this section, we illustrate the extensive feature engineering we conduct in our project.

5.1 Missing Value Handling

Based on the preliminary checking mentioned in §4, different portions of missing values exist in 191 variables. To mitigate their adversarial effect on the model training process, we drop variables that have more than 90% missing values, and impute variables with lower than 25% missing values. For the rest, we limit their impacts by (1) feasible feature engineering, and (2) choosing an ML model that is able to handle missing values and less vulnerable to missing values.

Regarding the data imputation, we try two methods, *k*NN imputation [21] with 5 neighbors, and iterative imputation [22] with random forest trees as the estimator. Note that, numerical and categorical variables are imputed by separate imputers. However, from the prediction results, we find both methods lead to ML models with comparable performance, but iterative imputation costs a much longer time to process.

5.2 Feature Generation

The dataset tracks historical behaviors of customers, therefore, each customer has one or more records. To get an integrated and more concise representation for a customer, all features from different periods should be summarized and aggregated. As such, we group records for each customer according to his/ her unique "customer_ID", and then rank statements by date. Based on a customer's all records, we generate a one-row aggregated feature by the following steps.

Numerical/ continuous variables. For a customer, we compute the corresponding maximum, minimum, average, and standard deviation for all numerical variables.

Categorical variables. We first transform categorical variables into numeric representations. There are 11 categorical variables, we divide them into two groups and use different encoding methods according to their values: (1) definitely no ordinal information exists among different categories, for example, the variable 'D_63' whose values are 'CR', 'CO', 'CL', etc. We transform such variables by *one-hot encoding* [23]. (2) variables that are obscure and hard to tell whether some kind of order exists, e.g, variable 'D_116' whose values are 4, -1, 6, 2, 1, 3, 5, we convert them by *label encoding* [24]. Then, for a customer's all statements, we count the total number of occurrences, and the total number of unique values for different categorical features.

5.3 Time-based Features

As mentioned before, the dataset we use is time-series based. Therefore, specific evolutional trends, seasonality and serial correlation might affect customers' behaviors. But these properties might be complicated and obscure. As such, we generate time-based aggregated features to capture such temporal characteristics. This step is majorly applied to numeric variables.

Last values. The most recent value in a time series usually has the highest importance for the next value's prediction since they are the most correlated. Thus, we keep the values for all variables in the newest statement.

Lag features. Lag features are values at a prior time period that are considered useful because they are created on the assumption that what happened in the past can influence or contain a sort of intrinsic information about the future [25]. Specifically, we generate two lag features for all numeric variables. Suppose S_0^i and S_t^i represent the values for a numeric feature S in customer i's first and last statement, we take their difference and scale by computing $(S_t^i - S_0^i)$ and (S_t^i / S_0^i).

Difference. We take the quantitative difference between two consecutive statements for a numeric feature. For example, given a feature "F" with 3 values f_{t-1}, f_t, f_{t+1} representing 3 statements for different periods for a customer, we take the differences $f_t - f_{t-1}, f_{t+1} - f_t$, and then compute their maximum (max), minimum (min), average (avg) and standard deviation (std). Besides, we compute the difference between avg and f_t to mitigate the influence of outliers.

Combination. After the completion of generating all the above-mentioned features, we combine them together. The number of features increases from 189 (excluding 'Customer_ID' and 'date' variables) to 949.

5.4 Data Type Conversion

To further save memory, we reduce data bytes for numeric variables by converting $float/int64$ to $float/int16$.

6 Model Construction

The major model we use in our project is Light Gradient Boosting Machine (LightGBM) [12]. For comparison, we also implement eXtreme Gradient Boosting (XGBoost) [11] and Categorical Boosting (CatBoost) [13] since they are two other state-of-the-art gradient boosting techniques that have comparable performance with LightGBM. Besides, several popular ensemble tree models, including vanilla gradient boosting trees, Random Forest, Adaboosttrees and Extratrees serve as baseline models to be compared with LightGBM, CatBoost and XGBoost. In this section, we provide details about experiment setup and implementation.

6.1 Experiments

We conduct the following experiments and record the corresponding evaluation metrics specified in §3.3. We have three major objectives regarding the ablation experiments: (1) examine which model has the best performance based on the evaluation metrics; (2) compare how the performance changes (improves or degrades) with and without feature engineering; and (3) investigate how features contribute to the prediction. Therefore, we conduct three series of experiments:

1. All above-mentioned models with fully featured engineered data.
2. All above-mentioned models with raw data (without feature engineering but with categorical encoding and imputation).
3. Feature exploration by models' internal mechanisms and SHAP.

6.2 Implementation

SMOTE. Considering the dataset is highly imbalanced, we employ the Synthetic Minority Oversampling Technique (SMOTE) [17] to synthesizes new examples for the minority class (i.e., 'default'

class) and downsample the majority class (i.e., ‘not default’ class).

Cross validation. We apply 5-fold cross-validation in our experiments to avoid data split bias and obtain “fair" performance estimation.

Train, validation and test split. We use 70% data for training models and leave 30% data for testing. In each training round, the training set (70% of all data) is divided into 5 subsets, 4 of them are used for training and the rest one is left out for validation.

Parameter Tuning. Parameter tuning is of great significance for building a more precision classifier. Table 1 lists the tuned parameters and corresponding trail values, which are guided by LightGBM document [26]. We use random search with 5-fold cross-validation to complete the parameter tuning since it has been empirically and theoretically proven that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid [27].

Table 1: Parameter tuning for LightGBM

Parameter	Values for trail
Boosting type	[‘gbdt’, ‘dart’]
learning_rate	[0.01, 0.02, 0.03, 0.04, 0.05, 0.06]
num_leaves	[10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65]
min_data_in_leaf	[20, 30, 40, 50, 60, 70, 80, 90]
feature_fraction	[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
subsample	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
bagging_fraction	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

7 Results and Evaluation

In this section, we first summarize the performance of different models evaluated by a rank-aware metric and a set of general metrics, with a highlight of LightGBM. Then we present further feature investigation and result interpretation.

7.1 Performance Summarization

We summarize the performances of all models that are trained by data with complete feature engineering procedure in Table 2. Specifically, our solution - LightGBM outperforms all other models for 5 evaluation metrics (Note that, according to the document [28], balanced accuracy is equal to recall, also known as sensitivity and the true positive rate). We also provide a bar plot in Fig.14 for a more intuitive comparison.

Table 2: Performance summary for different models trained by data with complete feature engineering procedure

Model	Rank-aware metric	Balanced accuracy	Precision	Recall	F1-score
Lightgbm	**0.79812**	**0.90377**	**0.90018**	**0.90377**	**0.90055**
Xgboost	0.77914	0.89935	0.899	0.89935	0.89917
CatBoost	0.78431	0.89977	0.90016	0.89977	0.89995
RandomForest	0.48367	0.85795	0.88924	0.85795	0.86439
ExtraTrees	0.48367	0.85223	0.88852	0.85224	0.85936
GradientBoostTrees	0.4939	0.86465	0.89097	0.86465	0.87033
AdaboostTrees	0.49376	0.86598	0.88927	0.86598	0.87128

7.2 Training Convergence

After using random search with 5-fold cross-validation, we obtain the best parameter combination of LightGBM model is: $`boosting' : `dart', `min\_data\_in\_leaf' : 39, `num\_leaves' : 110, `learning\_rate' : 0.01, `feature\_fraction' : 0.20, `bagging\_fraction' : 0.50, `lambda\_l2' : 2$. This parameter setting reaches 0.0.79812 for the rank-aware metric. We show the training progress of LightGBM in Fig. 7.

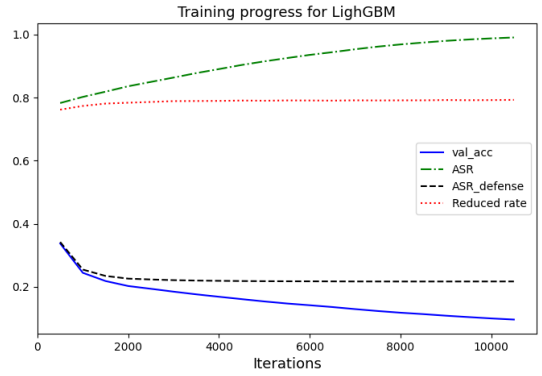

Figure 7: Training progress for LightGBM

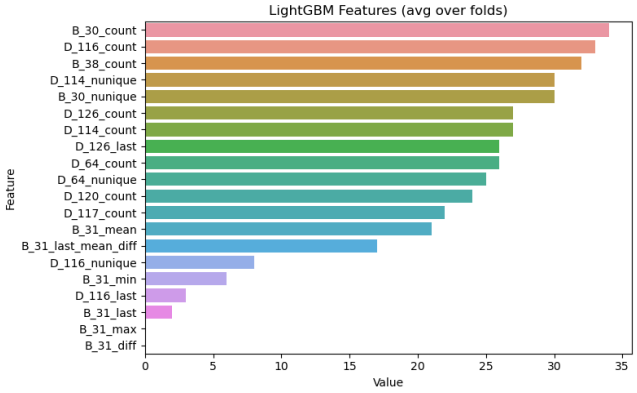

Figure 8: Feature Importance extracted from LightGBM

7.3 Feature Interpretation

7.3.1 Feature Importance

LightGBM, Xgboost and CatBoost have their internal mechanisms to rank features' importance. We extract top 20 from there three trained models and shown in Fig.8, Fig. 9 and Fig.10. However, the "important" features selected by different models are also different. So we resort to one more tool - SHapley Additive exPlanations (SHAP) [29] to further investigate them.

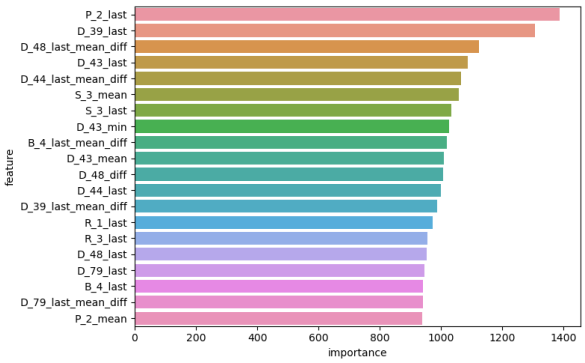

Figure 9: Feature Importance extracted from Xgboost

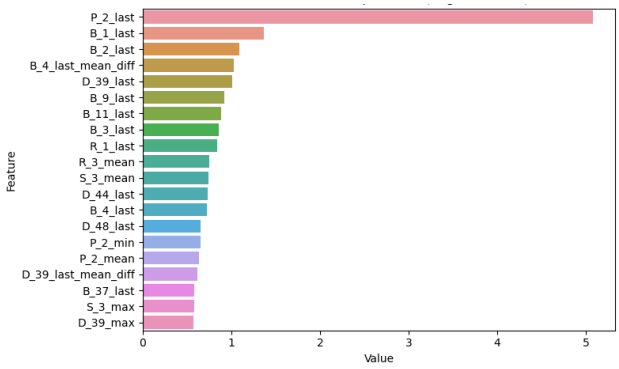

Figure 10: Feature Importance extracted from CatBoost

7.3.2 Results Interpretation

We resort to Shapley values [29] to further explore features and outputs from the trained LightGBM model and enable results to be interpretable and explainable. Shapley values represent a feature's responsibility for a change in the model output.

Summary Plot. We plot the top 20 important features ranked by Shapley values in Fig 11. The figure reflects feature importance and also feature effects on the prediction. Each row represents a feature, a single point refers to a data sample. The upper rows means more important features. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in blue [29]. Here are some interesting insights gained from Fig. 11 (only list a few insights for illustration, similar patterns tell similar insights):

- P_2_last has the most significant influence on the default prediction. Most of red points are located in $SHAP > 0$ area, which means as P_2_last increases, a customer is more likely to be default.
- For features like $B\_1\_last, D\_44\_last, B\_9\_last$, whose blue points are centred in right while red points are located in left, if these values increase, a customer is less likely to be default.

Dependent Plot. We can also understand how a single feature interacts with another one and how they affect the prediction by dependent plots. For example, Fig.12 shows the interactions among P_2_last, S_6_mean and the prediction target. From this figure we can find that, when $P\_2\_last = 0.0$, a customer who has smaller S_6_mean is more likely to be default. Another example Fig. 13 indicates when $B\_1\_last = 1$, a customer has smaller D_45_mean value is more likely to be default.

7.4 Feature Engineering Boosts performance

We perform 2 experiments - using raw data (explained in §6.1) and data generated by the complete feature engineering procedure to train LightGBM, Xgboost and CatBoost, and compare their performance by rank-aware metric. The results are presented in Fig. 15), from which we can find that all three models trained by feature engineered data have higher performance score. This illustrates the significance of a feasible and reasonable feature engineering procedure: (1) it helps ML models to improve their performance towards the intended tasks, and (2) it mitigates the negative impacts caused by the missing, noisy and abnormal data in training set.

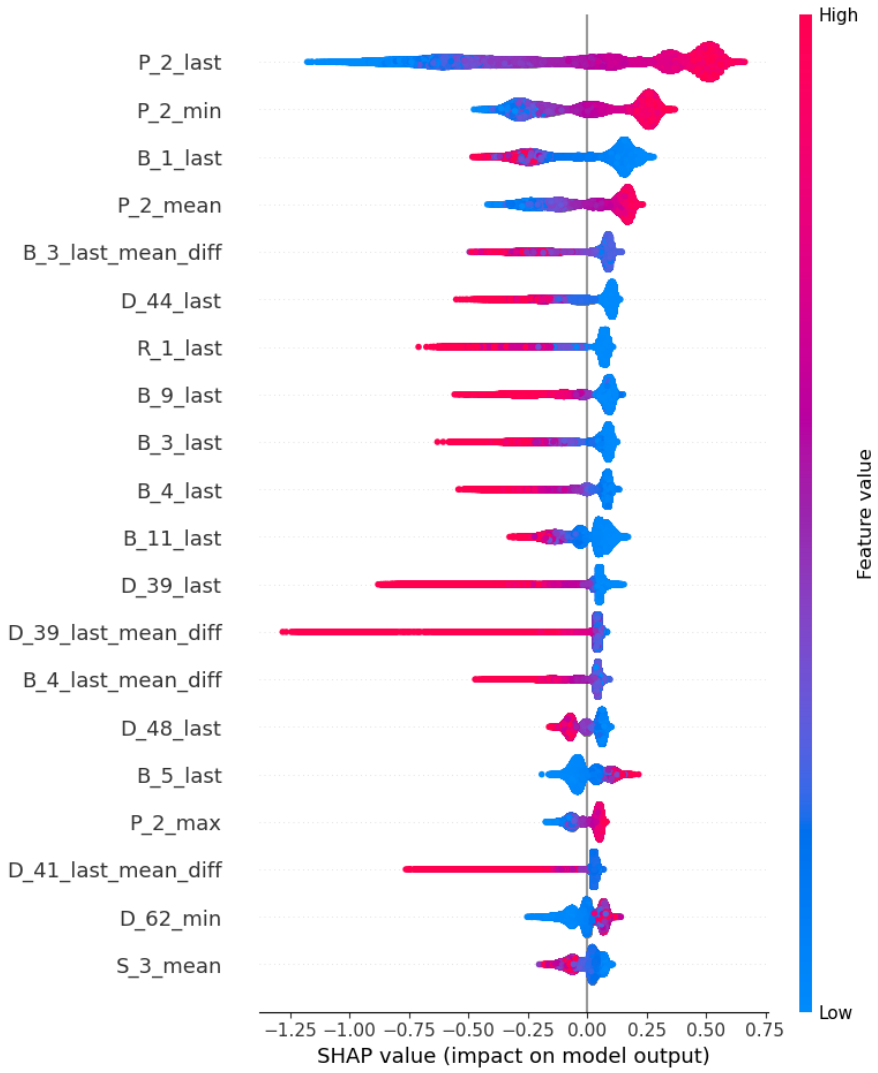

Figure 11: Summary plot for top 20 most important features. This reflects the feature importance and influence on the prediction. Upper rows means more important features. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in blue.

8 Conclusion

Aiming at more precise credit default prediction, this project employs ensemble learning techniques to construct gradient boosting tree models, and implements a complete machine learning pipeline from exploratory data analysis, feature engineering, model training, evaluation and result interpretation. Our solutions, majorly LighGBM, CatBoost and Xgboost outperform baseline models including AdaboostTrees, RandomForests, ExtraTrees and vanilla gradient boosting trees. In addition, our experiment result demonstrates that, a feasible and reasonable feature engineering improves ML predictions and mitigates the negative impacts caused by dirty data. What's more, we enable the results to be interpretable and explainable by the Shapley additive explanation tool.

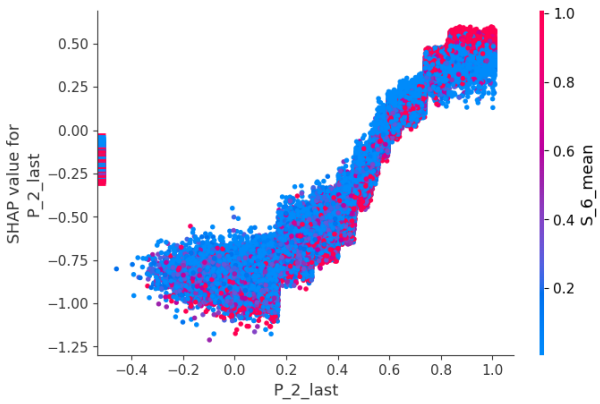

Figure 12: When $P\_2\_last = 0.0$, a customer who has smaller S_6_mean is more likely to be default

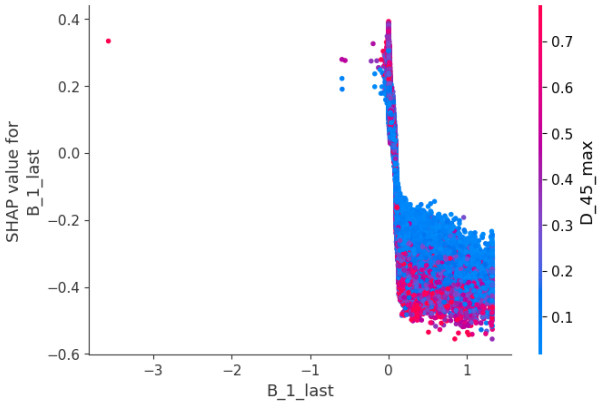

Figure 13: When $B\_1\_last = 1$, a customer has smaller D_45_mean value is more likely to be default

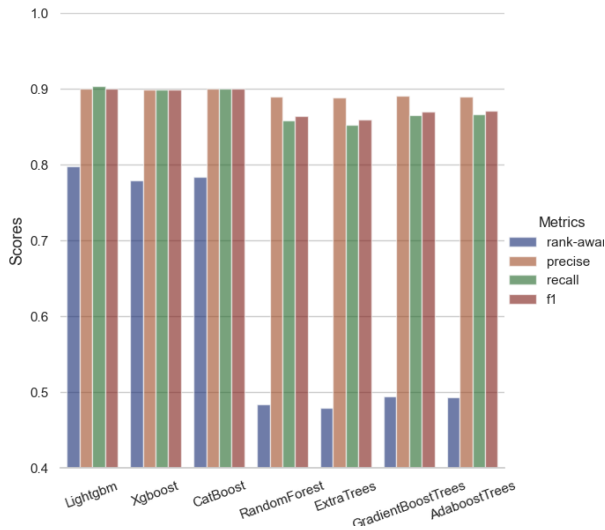

Figure 14: Performance comparisons for different models

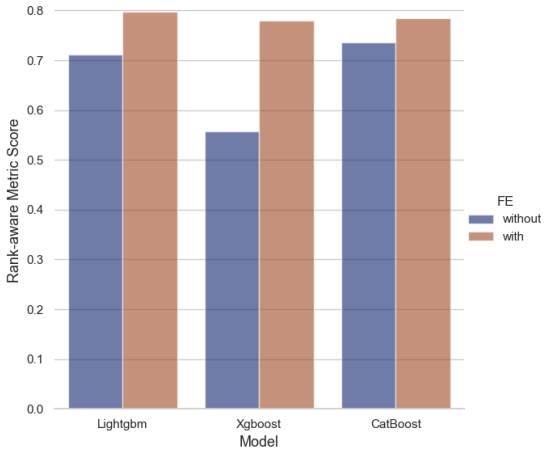

Figure 15: Performance boosted by feature engineering

References

[1] Ronghua Xu, Hefeng Meng, Zhiqiang Lin, Yonghui Xu, Lizhen Cui, and Jun Lin. Credit default prediction via explainable ensemble learning. In *5th International Conference on Crowd Science and Engineering*, pages 81–87, 2021.

[2] Fahmida E Moula, Chi Guotai, and Mohammad Zoynul Abedin. Credit default prediction modeling: an application of support vector machine. *Risk Management*, 19(2):158–187, 2017.

[3] Maher Ala'raj and Maysam F Abbod. Classifiers consensus system approach for credit scoring. *Knowledge-Based Systems*, 104:89–105, 2016.

[4] American Express Team on Kaggle. Aggregated customer profile, 2022. data retrieved from Kaggle competition, https://www.kaggle.com/competitions/amex-default-prediction/data.

[5] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212, 2022.

[6] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

[7] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, 2021.

[8] Andreea Anghel, Nikolaos Papandreou, Thomas Parnell, Alessandro De Palma, and Haralampos Pozidis. Benchmarking and optimization of gradient boosting decision tree algorithms. *arXiv preprint arXiv:1809.04559*, 2018.

[9] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[10] Gradient boosting — Wikipedia, the free encyclopedia. [Online; accessed 29-September-2022].

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[13] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[14] Chau-Nan Chen, Tien-Wang Tsaur, and Tong-Shieng Rhai. The gini coefficient and negative income. *Oxford Economic Papers*, 34(3):473–478, 1982.

[15] Amex Team. American express - default prediction, May 2022.

[16] Alan C Acock. Working with missing values. *Journal of Marriage and family*, 67(4):1012–1028, 2005.

[17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[18] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[19] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[20] C Reid Turner, Alfonso Fuggetta, Luigi Lavazza, and Alexander L Wolf. A conceptual basis for feature engineering. *Journal of Systems and Software*, 1999.

[21] Scikit-learn: Knnimputer. https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html. Accessed: 2022-12-02.

[22] Scikit-learn: Iterativeimputer. https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html. Accessed: 2022-12-02.

[23] Scikit-learn: Onehotencoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html. Accessed: 2022-12-02.

[24] Scikit-learn: Labelencoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed: 2022-12-02.

[25] Introduction to feature engineering for time series forecasting. https://medium.com/data-science-at-microsoft/introduction-to-feature-engineering-for-time-series-forecasting-620aa55fcab0#:~:text=Lag%20features%20are%20values%20at,intrinsic%20information%20about%20the%20future. Accessed: 2022-12-02.

[26] Parameters tuning for lightgbm. https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html. Accessed: 2022-12-04.

[27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[28] Scikit-learn: Balanced accuracy score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html. Accessed: 2022-12-04.

[29] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Appendices

A EDA Results

This section presents EDA results for §4.

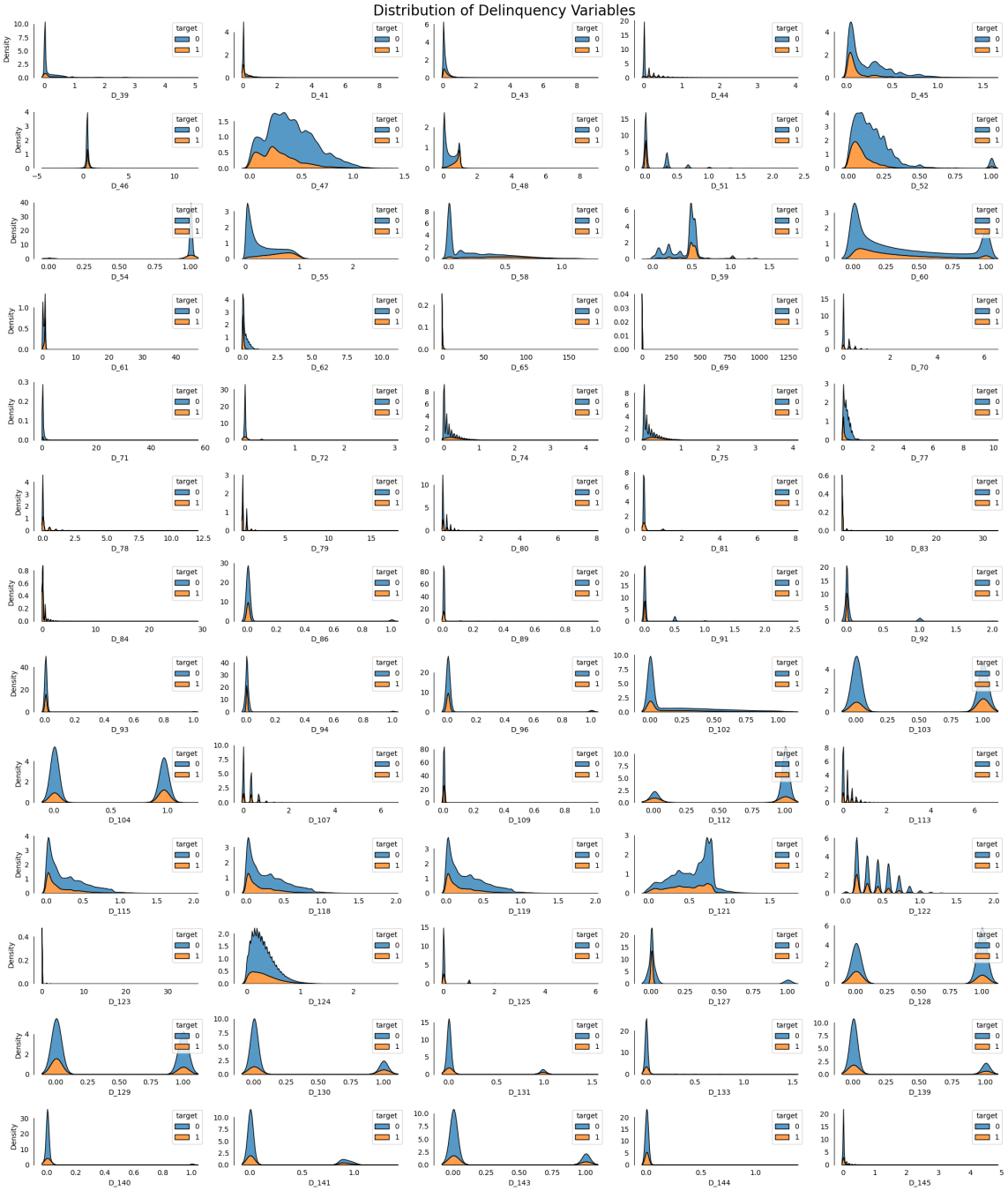

Figure 16: The distribution of Delinquency Variables ('D_*'). Most of variables in this group exhibit skewed distribution.

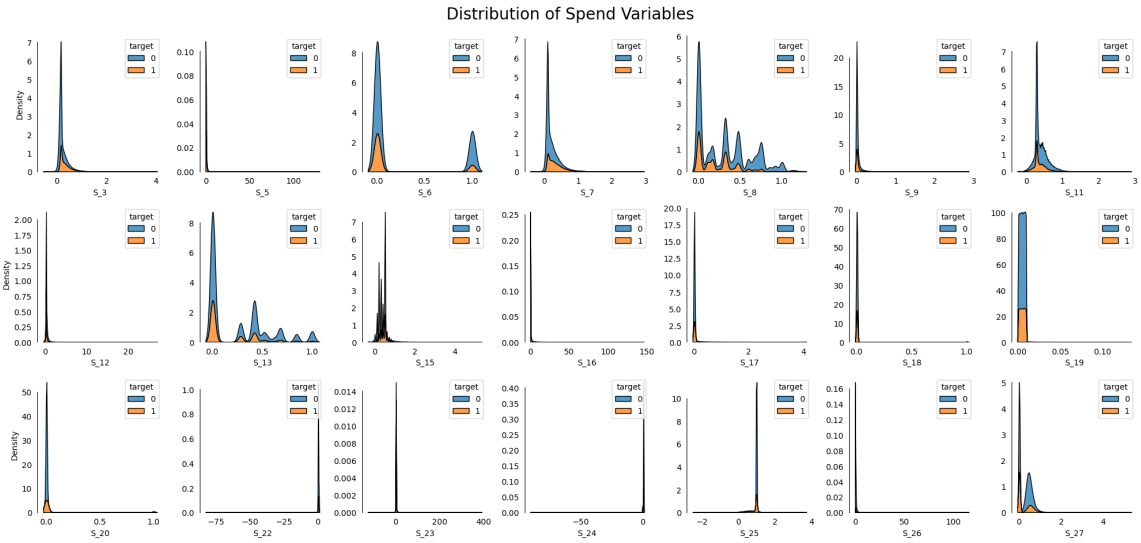

Figure 17: The distribution of Spend Variables (‘S_*’). Almost all variables are very close to 0.

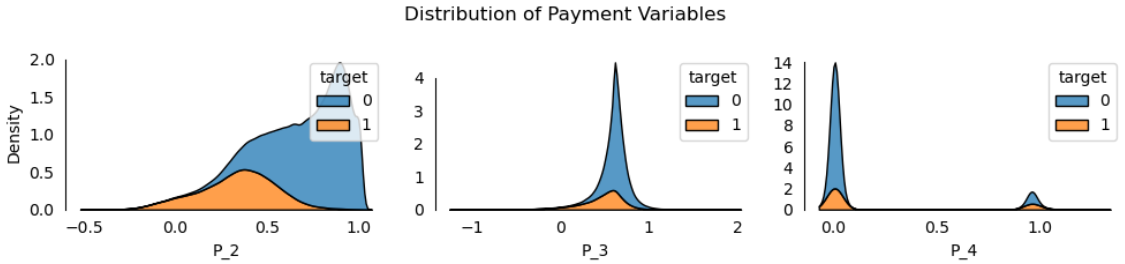

Figure 18: The distribution of Payment Variables (‘P_*’). P_2 is slightly skewed, P_3 is normal distributed, P_4 has values towards 0.

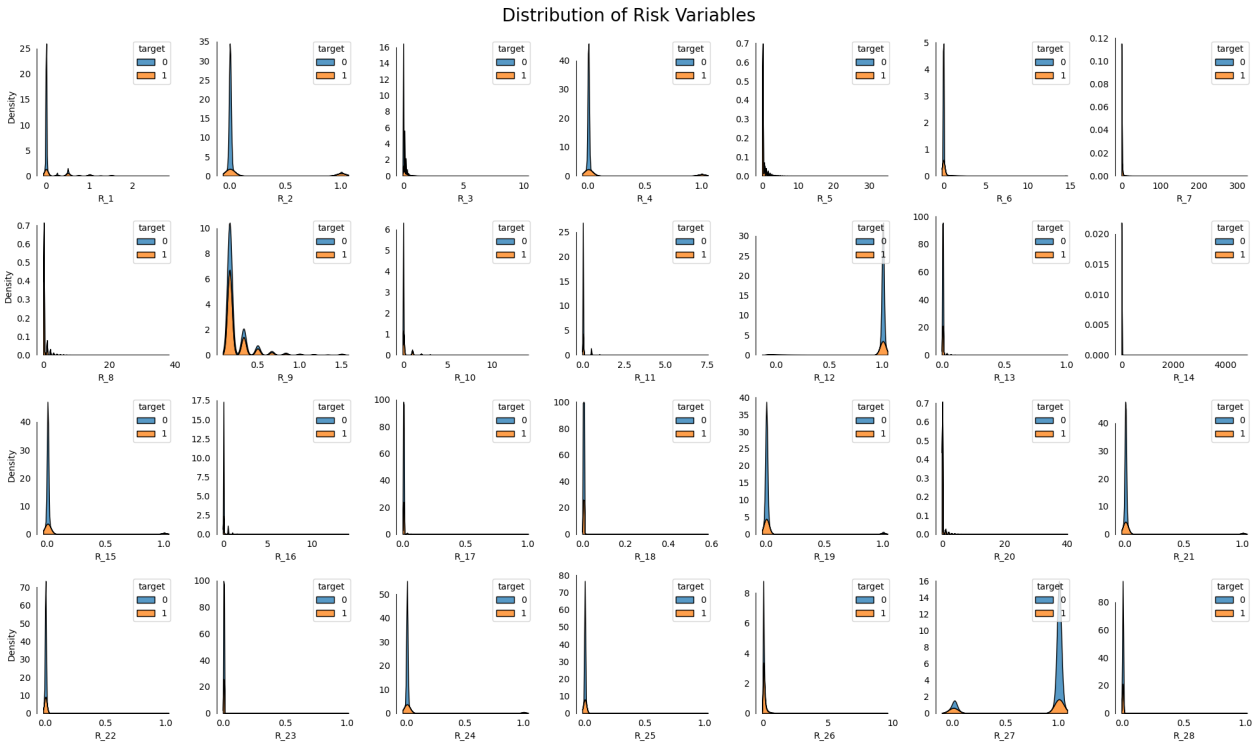

Figure 19: The distribution of Risk Variables (‘R_*’).All variables exhibit skewed distribution.

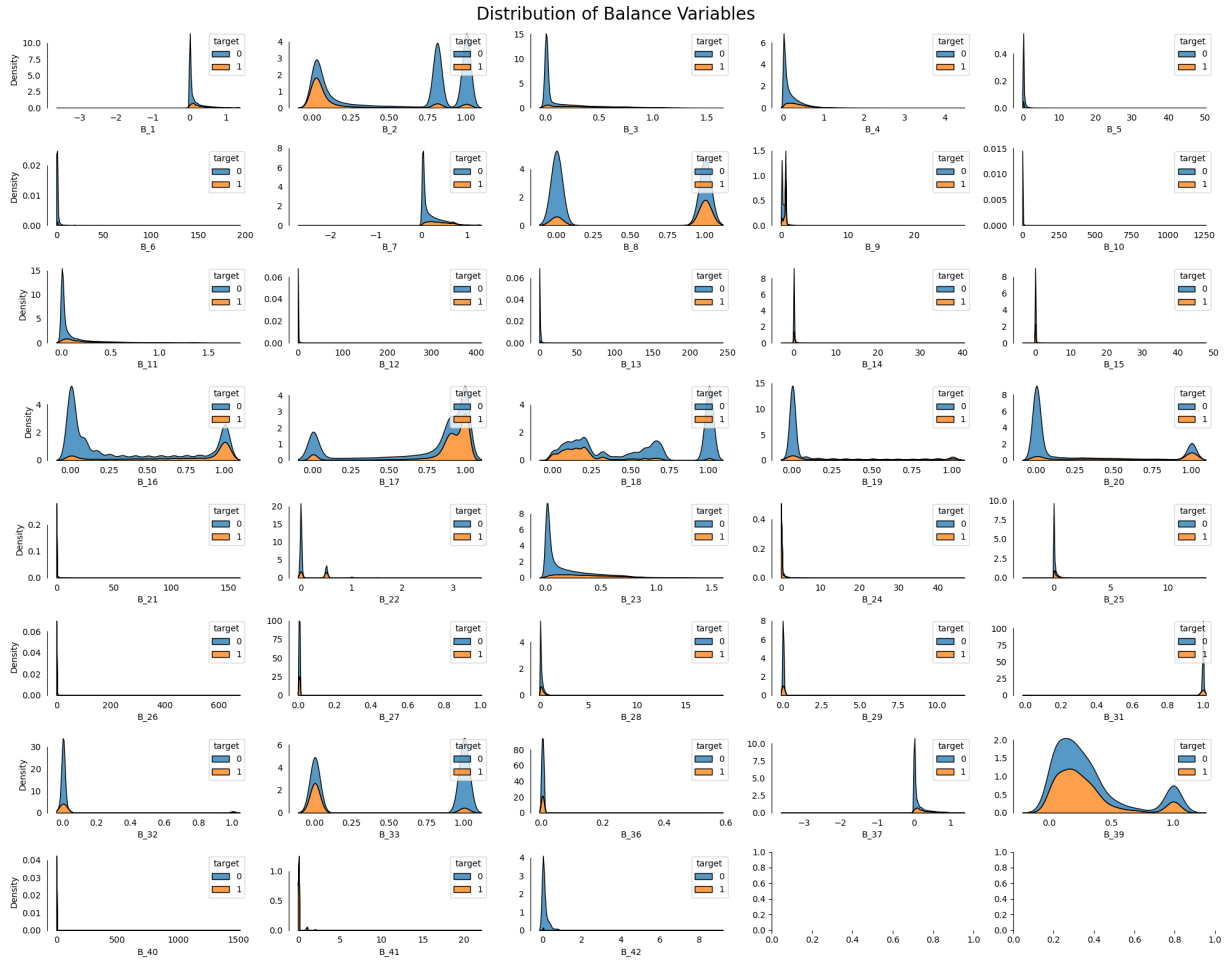

Figure 20: The distribution of Balance Variables (‘B_*’). Over half variables have dominant values near 0. Almost all variables have extreme end of value (towards 0).

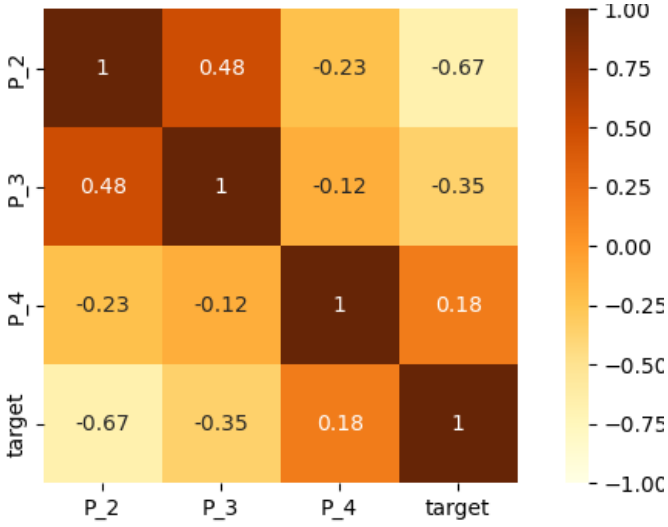

Figure 21: Correlations among Payment Variables (‘p_*’)

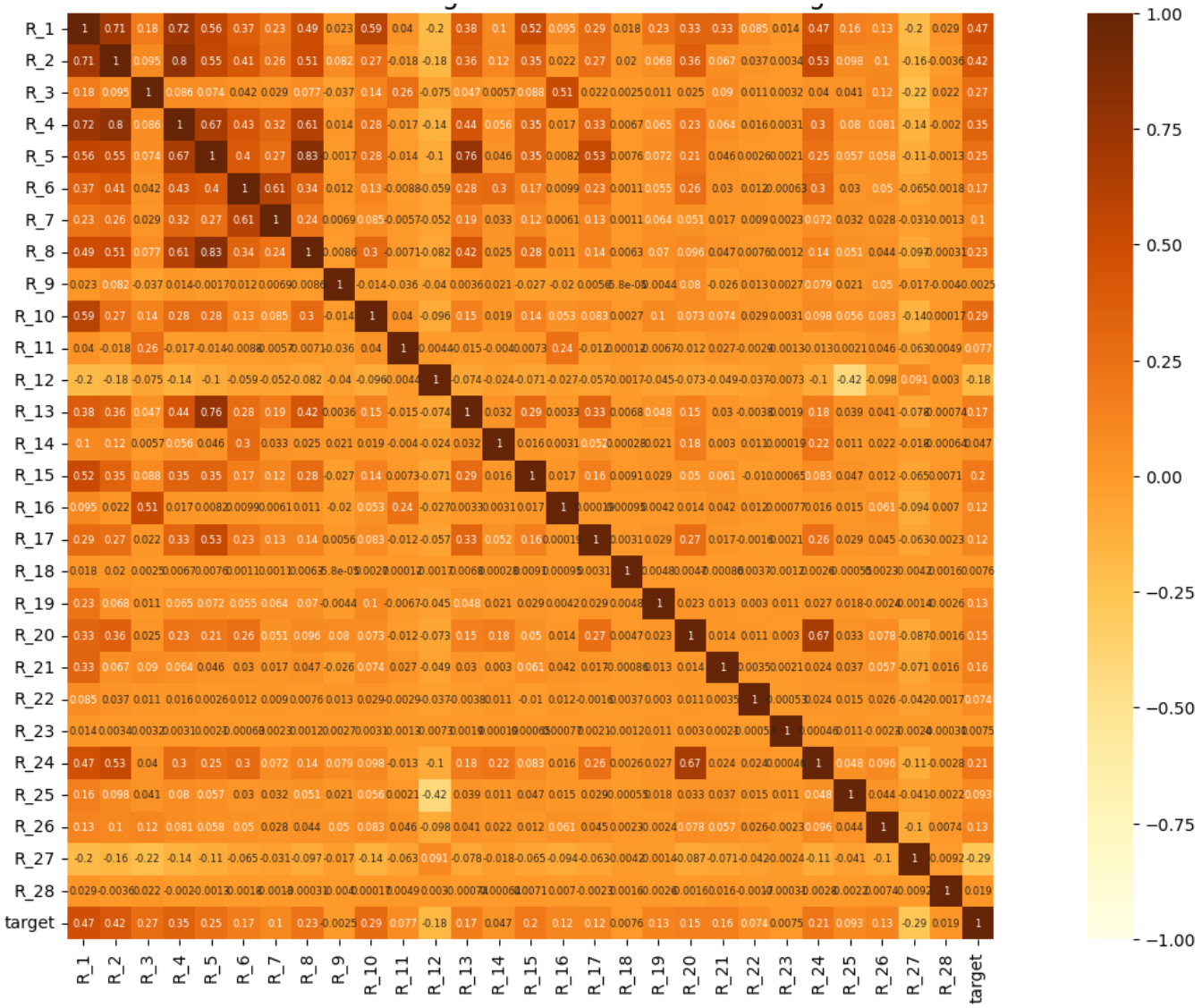

Figure 22: Correlations among Risk Variables ('R_*')

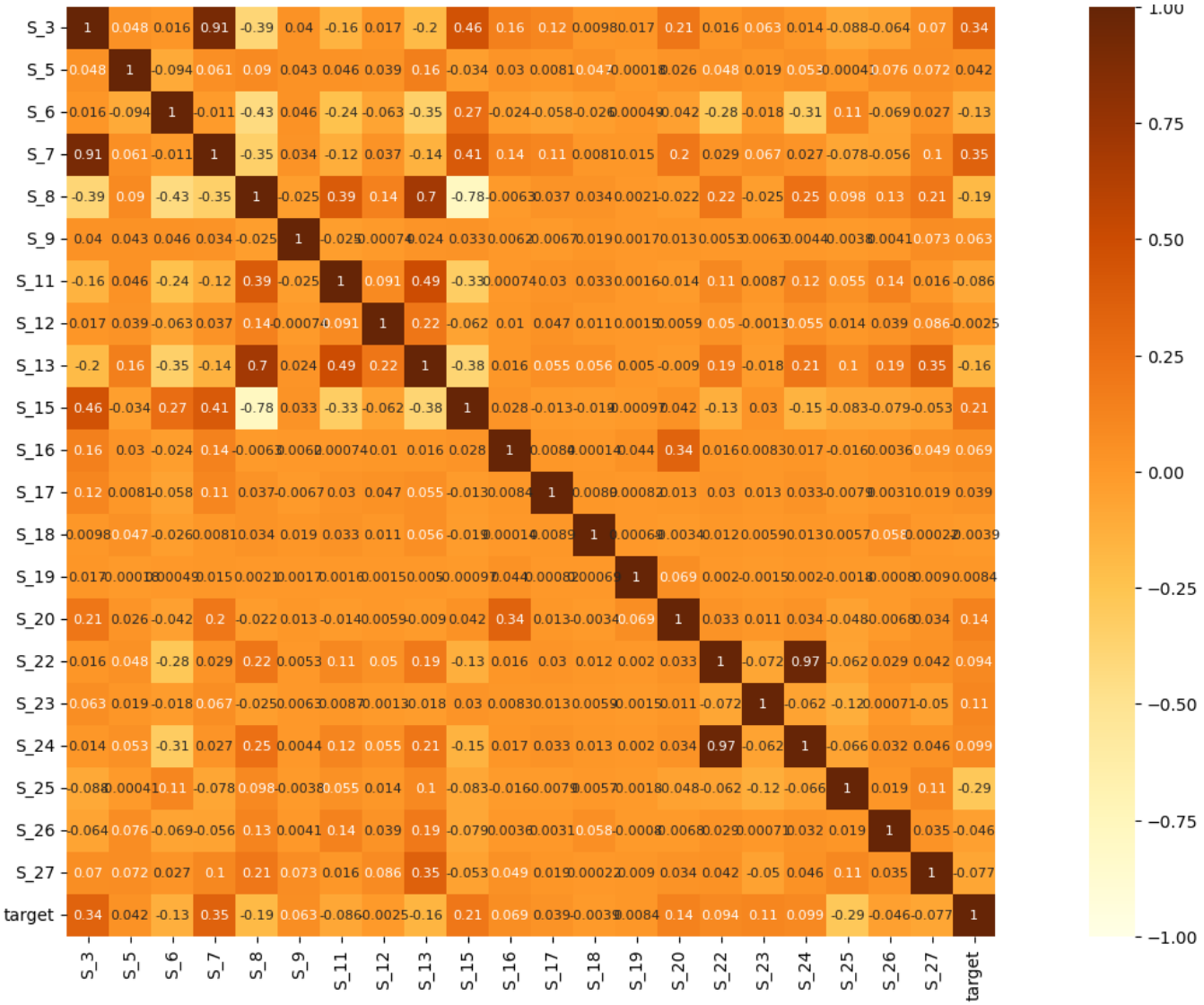

Figure 23: Correlations among Spend Variables (‘S_*’)

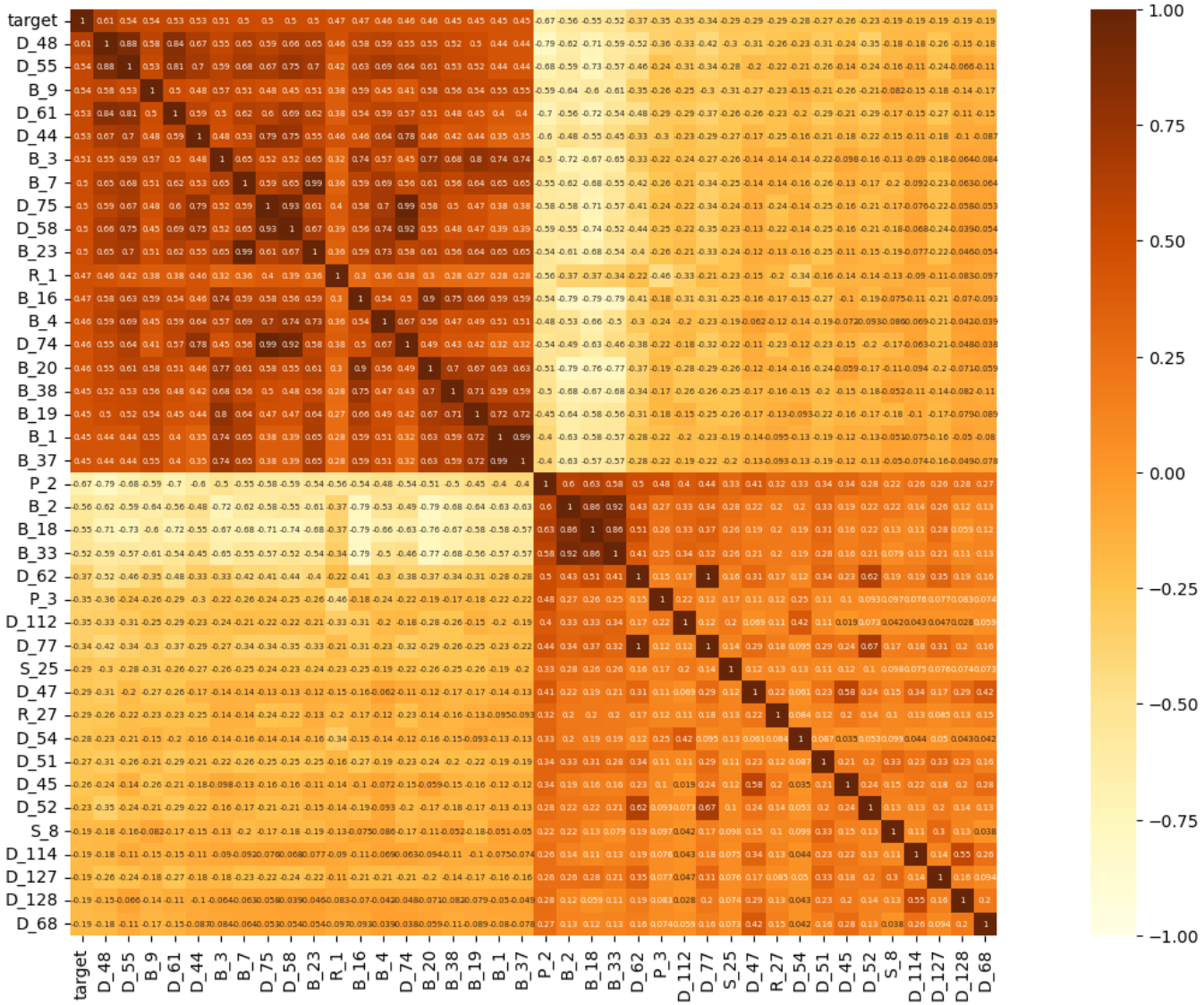

Figure 24: TOP 20 variables that have high correlations with the target variable