DATA 1030 Midterm Project Report

Bangxi Xiao

Data Science Initiative, Brown University

1. Introduction

1.1. Motivation

The peer-to-peer lending has grown rapidly in recent years and provides loans to serve many purposes, such as consumer credit, small business and student education. The motivation behind our project is to help lenders make better lending decisions to maximize returns while minimize risks. This project provides critical application for investors to predicting future loan status using the Kaggle dataset.

1.2. Data Overview

The dataset is obtained from Kaggle website: https://www.kaggle.com/mishra5001/credit-card, which has a size of over 100 features and 300,000 samples. According to the column "TARGET", the data could be split into two parts – the defaulters and the non-defaulters, where defaulters are identified as people who are rejected by the loan company while the non-defaulters are accepted by the companies. There are two major challenges in our project. The first is the high dimensional data set, which originally contains over 120 features and there are potential collinearity or correlation among a large fraction of them. The highly imbalanced data imposes the second major challenge. We will present some methods to tackle the two problems in the latter part of the report.

The description of the data can be found in the website provided above.

1.3. Goal

The first step of the project is to successfully process the missing values – some of the features suffered from large proportion of missing values and we have to deal with them. Next, we are going to resolve the problem of unbalanced data and the curse of dimension problems. The ultimate goal of the project is to successfully identify the defaulters given their information (a binary classification task). Since the loan company definitely pays more attention to the customers who will potentially default a credit in the future, our work should focus on deriving the characteristics from the defaulters.

1.4. Related Work

The data was obtained from Kaggle and a few previous works were done by the contributors on Kaggle. Their efforts were mainly concentrated on binary classification problem, exploratory data analysis and data structure analysis. Joe Corliss [1] made prediction on loan charge offs from initial listing data and proposed a logistic regression with SGD parameter training algorithm. Also, he performed some other models such as random forest classifier and k-nearest-neighbor. Nathan George [2] from Kaggle made contributions on basic exploratory data analysis on the data by examining the distribution of each class. Mathew [3] explored correlations between features. Luke [4] demonstrated some corrupted / not well-formed rows in the data and he found out that these rows seem to be some sort of summary statistic or addendum referring to previous rows, which contain only a single column, providing us straight help in data cleaning process. Previous research focuses on loan status and credit risk model. Chen and Tsai [5] studied using hybrid

machine learning models for credit risk classification problem. Khandani et.al.[6] focused on using machine learning for consumer credit models.

2. Exploratory Data Analysis

2.1. Target Variable

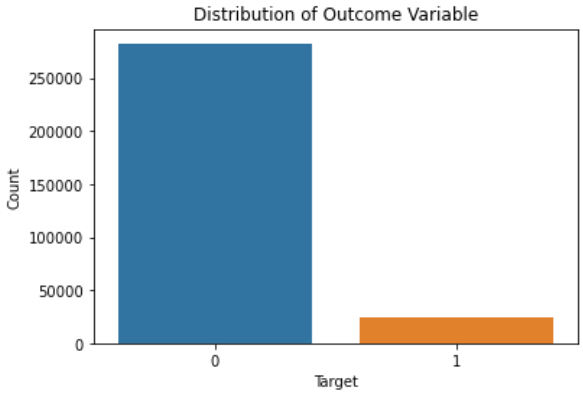

Fig 0: The number of defaulters vs non-defaulters, the samples are highly unbalanced.

The target variable – whether a loan is default or not, as we can see, is highly imbalanced. 0 indicates the non-default status while 1 indicates the default status.

2.2. Selecting from Similar Features

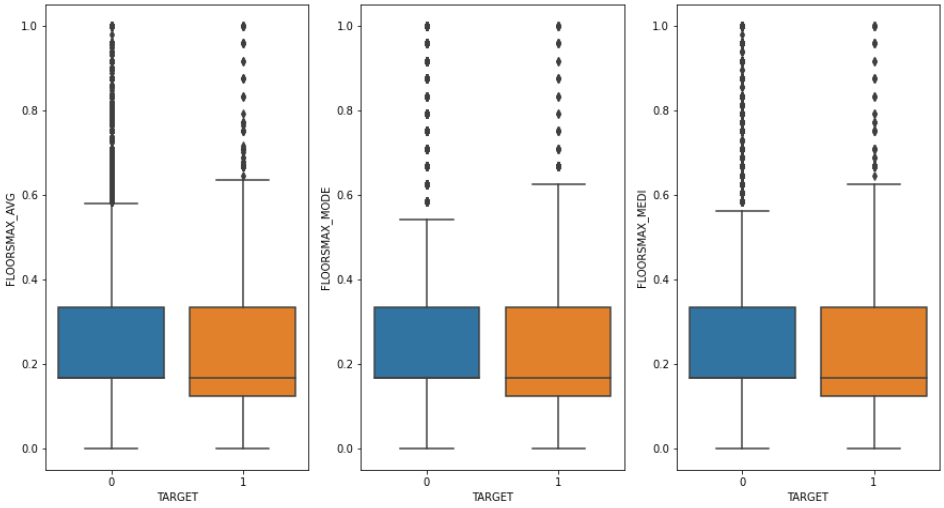

Fig1: Floor-related features refer to the number of floors (normalized) where the client lives has, where AVG indicates the average number of floors, MODE the maximum and MEDI the median. We can see that there are some slight differences between defaulters and non-defaulters: the defaulters have most of the floors lower than the non-defaulters.

We encountered many similar features in the dataset, for example, the feature “FLOORSMAX”, which suggests the maximum number of floors where the client lives, provides 3 measurements – the average, maximum and median. The bar plots suggest that they are quite similar to each other, so we decided to remove the duplicated ones. We kept the median value of this feature.

2.3. Removal of Non-informative Features

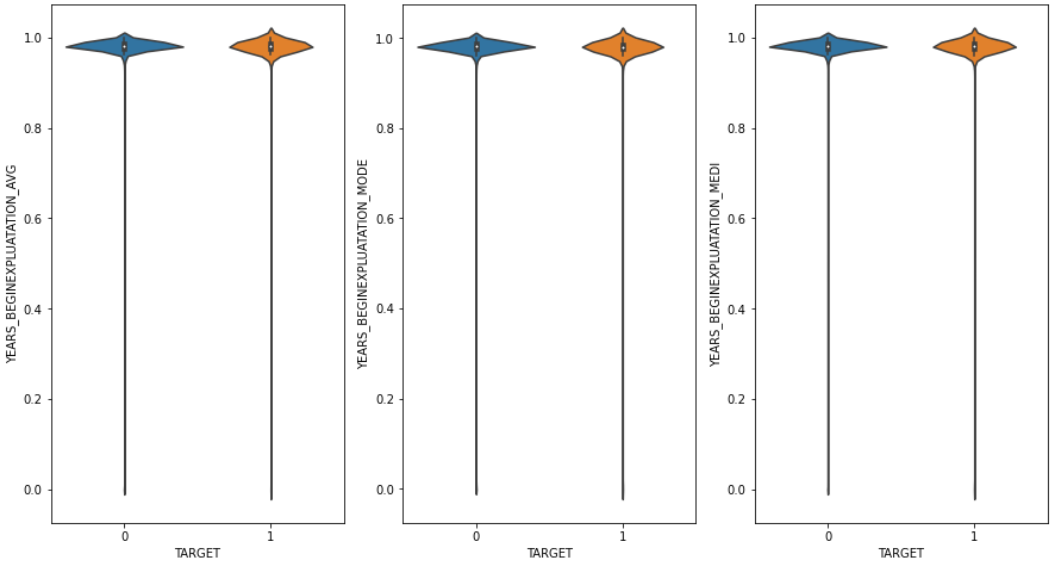

Fig 2: We can hardly tell the difference from the 3 features between defaulters and non-defaulters since their distributions are too close to each other. We might consider delete the 3 features from our model.

We do encounter the features which have no predictive power – no matter in defaulters' data or non-defaulters' data, the distributions are almost identical. We will drop the features if they are not informative.

2.4. Linear Relation Study

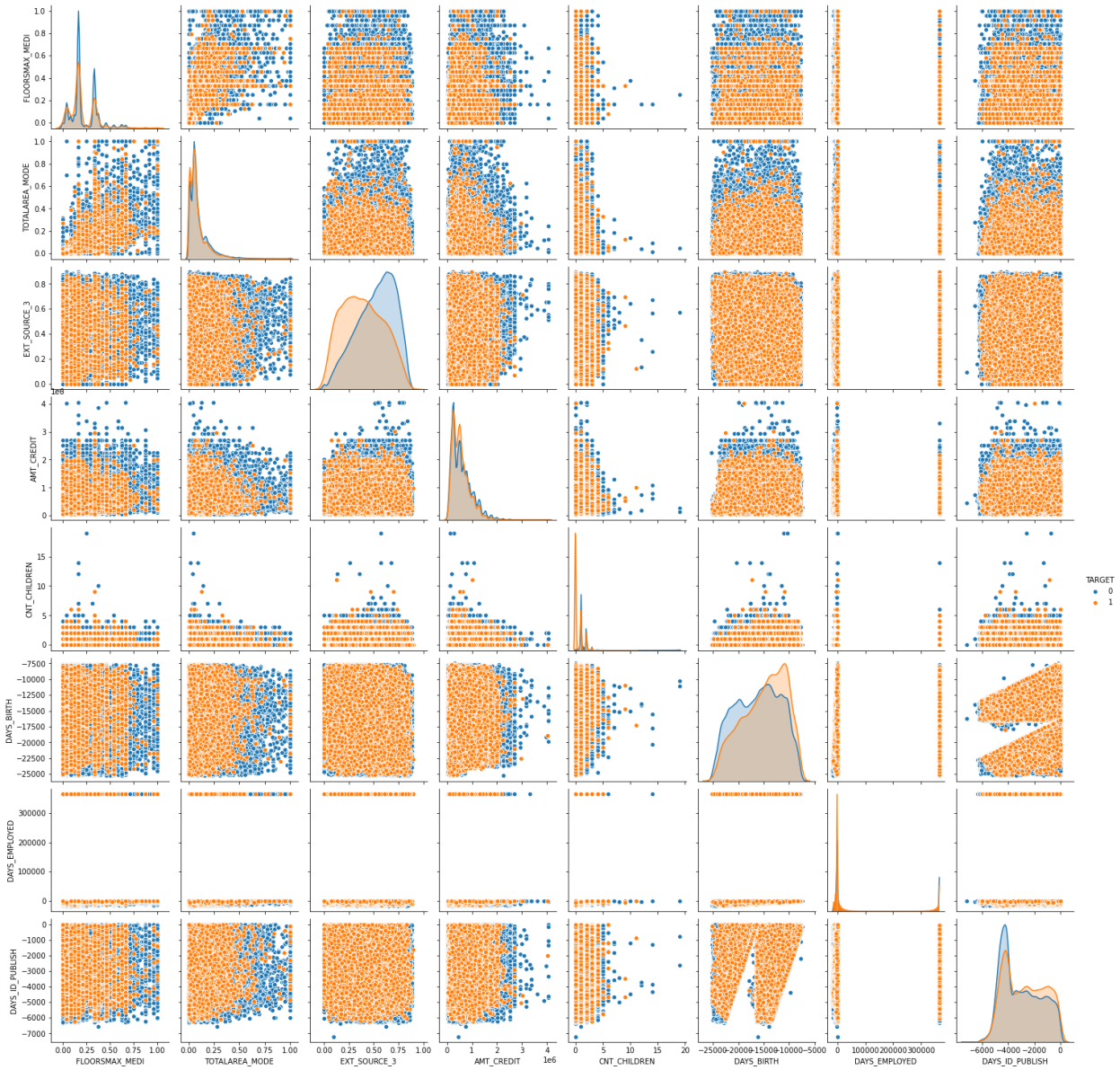

Fig 3: We examined a selection of continuous features by plotting the density curves and scatter plots. To see the discrepancies between defaulters and the non-defaulters.

We also selected a subset of continuous features to examine the linear relationship between defaulters and non-defaulters. However, due to the large number of non-defaulters, some figures are hard to identify. Some discrepancies between defaulters and non-defaulters can still be captured via the density plots (in the middle). For example, in the feature “EXT_SOURCE_3”, which is the outer source of credit score, the density curve of non-defaulters clearly deviates from the other. This implies that the feature might be important during the classification process.

2.5. Categorical Feature Example

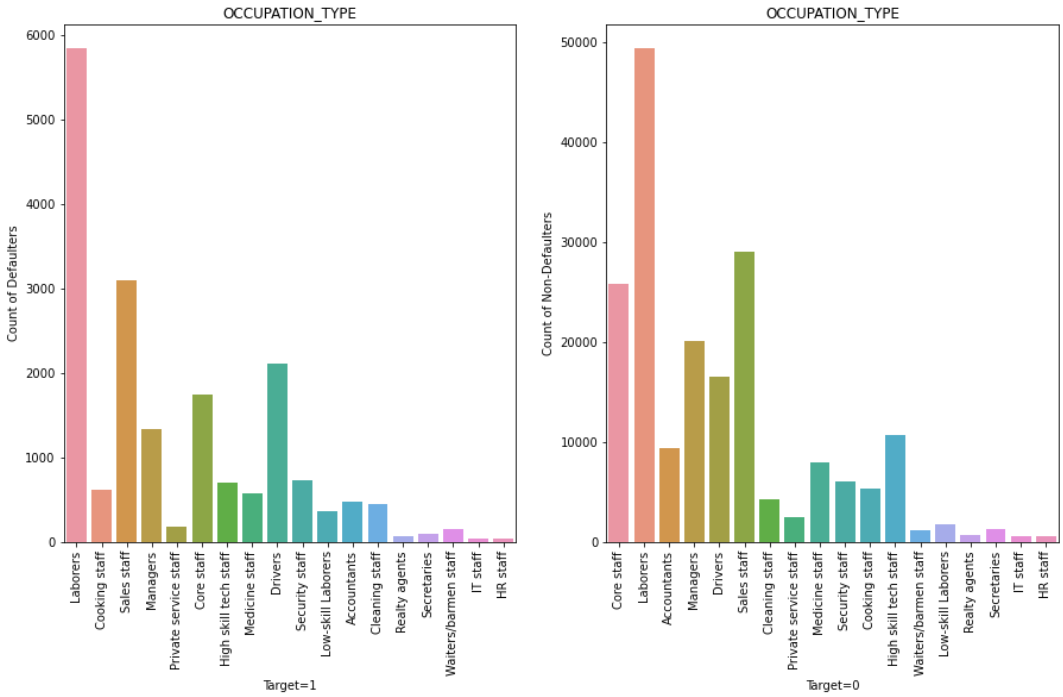

Fig 4: This figure illustrates the customers' occupations. Some discrepancies are observed between the defaulters and the non-defaulters in this plot, for example, the number of managers who claim to default a credit is lower than the non-default ones.

Here is an example categorical feature – the occupation type of the clients. Some small differences between the two targets are revealed.

3. Data Preprocessing

3.1. Handling Missing Value

We first filtered out the features with missing proportion greater than 50.00%. For features with missing proportion less than 10.00%, we directly ruled out the corresponding rows. Then, the features are split into 3 major groups: the continuous features, the categorical features and the ordinal features. Different treatments were applied to different types of features.

In categorical features, we used “Other” and “Unknown” to separately impute the missing values in “OCCUPATION_TYPE” and “EMERGENCYSTATE_MODE”, before which, we have checked that the data is missing at random.

The occupation counts in the data:

```
Laborers                55186
Sales staff             32102
Core staff              27570
Managers                21371
Drivers                 18603
High skill tech staff   11380
Accountants              9813
Medicine staff           8537
Security staff           6721
Cooking staff            5946
Cleaning staff           4653
Private service staff    2652
Low-skill Laborers       2093
Waiters/barmen staff     1348
Secretaries              1305
Realty agents             751
HR staff                  563
IT staff                  526
Name: OCCUPATION_TYPE, dtype: int64
```

We created another classed called “Other” to deal with the missing ones.

The emergency status feature consists of only “Yes” and “No”. For the missing ones, we use ‘Unknown’ as the third class.

Regarding the continuous features, we will use Iterative Imputer to do the imputation. Specifically, we do it after splitting the data into training, validation and test sets.

For the ordinal features, there is no missing value. We do nothing.

3.2. Splitting

The splitting of the dataset is quite simple – the only thing we need to pay attention is the unbalance nature of the data. Thus, a stratified splitting should be performed. The dataset is I.I.D. since each row represents each client and the clients are different from each other. Also, it does not contain any grouping structure or time-series pattern. The splitting results are revealed as following:

Subsets	Proportion	X shape	Y shape	Y Categorical Counts
Train	0.6	(183731, 63)	(183731,)	(0, 1) -> (168878, 14853)
Validation	0.2	(61244, 63)	(61244,)	(0, 1) -> (56293, 4951)
Test	0.2	(61244, 63)	(61244,)	(0, 1) -> (56293, 4951)

Table 1: Train, validation and test split facts

Since we have categorical, continuous and ordinal features, we initialized 3 different scalers and encoders.

The one-hot encoder is used to deal with the categorical features. In our dataset, we applied the one-hot encoder on the 42 categorical features. For example, the feature “CODE_GENDER” indicates the gender of client; after one-hot encoding, for each entry, we will have a [1, 2] vector to represent the gender. Likewise, for occupations, we have 19 categories; after one-hot encoding, each occupation will be represented by a 1 x 19 vector.

The ordinal encoder encodes the level data. For example, in our dataset, we have feature “REGION_RATEING_CLIENT”, which is the rating score from a regional bank or corporation. The rating is leveled from 1 to 5. Thus, an ordinal encoder best suits the situation here. In all, we have 4 ordinal features.

Lastly, the standardized scaler works as a scaler for the continuous data features such as income amount, number of children and so on. Comparing with the Min-Max scaler, the continuous features in our dataset seem have no hard boundaries; thus, all the continuous ones (number of 17) are proceeded with Standard Scaler.

3.3. Balancing

We used SMOTE to balance the training data. SMOTE works with pretty well with high dimensional data. Specifically, take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

4. Github

All the coding works have been uploaded to my personal Github repository:
https://github.com/rchopinw/DATA1030_MIDTERM_PROJECT.git

5. Reference

[1] Joe Corliss. Predicting Charge-off from Initial Listing Data,
https://www.kaggle.com/pileatedperch/predicting-charge-off-from-initial-listing-data

[2] Nathan George. Some basic EDA in R and demo how to load the data,
https://www.kaggle.com/wordsforthewise/eda-in-r-arggghh

[3] Anuradha. Credit EDA Study https://www.kaggle.com/anuradhamohanty/credit-eda-study

[4] Luke. Demonstrating Corrupted/Mal-formed Rows,
https://www.kaggle.com/lukemerrick/demonstrating-corrupted-mal-formed-rows

[5] Tsai, Chih-Fong. Chen, Ming-Lun. (2010). Credit Rating by Hybrid Machine Learning Techniques. Applied Soft Computing. 10 (2010) 374-380.

[6] Khandani, A. E., Kim, A. J., Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. Journal of Banking Finance, 34(11), 2767-2787.