Credit Card Default Prediction

Applied Machine Learning - Group 13

Soheil: sf3116 **Allan C:** ac5439 **Fernando:** fc2672 **Jahnavi M:** jm5667 **Kevin Y:** zy2582

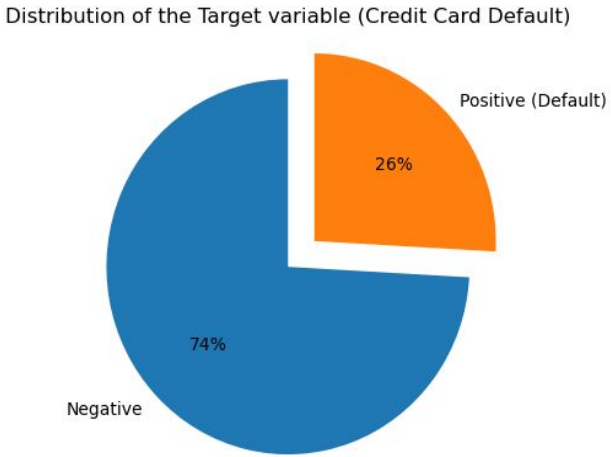
Project Overview

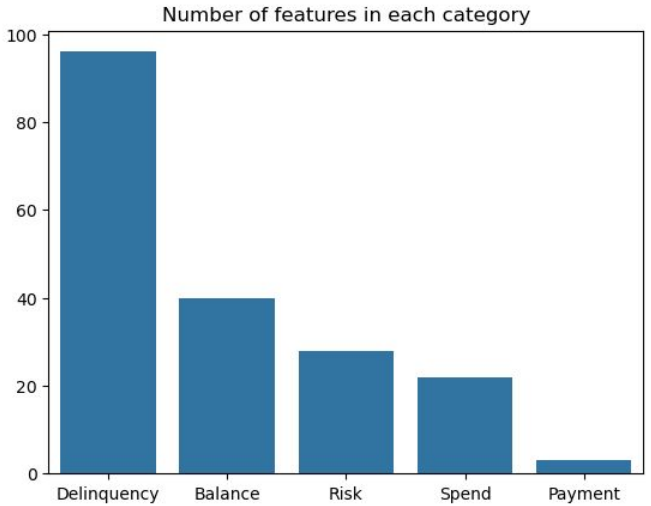
- **Aim :** Binary prediction of customer credit card default.
- **Data** sourced from 2022 Kaggle competition hosted by American Express.
- All features are anonymized but each feature belongs to one of these five feature profile: **Delinquency, Spend, Payment, Balance, Risk**. E.g. D_3 means this anonymized variable is related to delinquency.
- 190 features and 5.53m rows across 459k customers.
- One customer might have up to 13 rows (13 months of data), therefore data aggregation is needed.
- Tree-based classifications (Random Forest, XGboost, Decision Trees) will be used for modeling.

Dataset Size Reduction:

- The dataset that we are using (train_data) is 16.39 GBs. To run this on a local computer will require massive amounts of memory and processing power. Hence, while loading the csv file, we read it in chunks and manipulate the datatypes of the columns to reduce the size of the dataset to just 1.5 GBs.
- The steps we perform are:
    - Reduce the storage of the 'customer_ID' column from 64 bytes to 4 bytes by converting the hexadecimal string to int64, using the last 16 characters and converting from base16 to base10.
    - Reduce the storage of the 'S_2' column using pd.to_datetime()
    - Reduce the size of 11 categorical columns with a maximum of 8 values each ('B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68') from 8 bytes to 1 byte per row by converting them to int8.
    - Reduce the size of 177 numeric columns from 8 bytes to 2 bytes per row by converting from float64 to float16, assuming uniform noise addition.

Initial Data Exploration



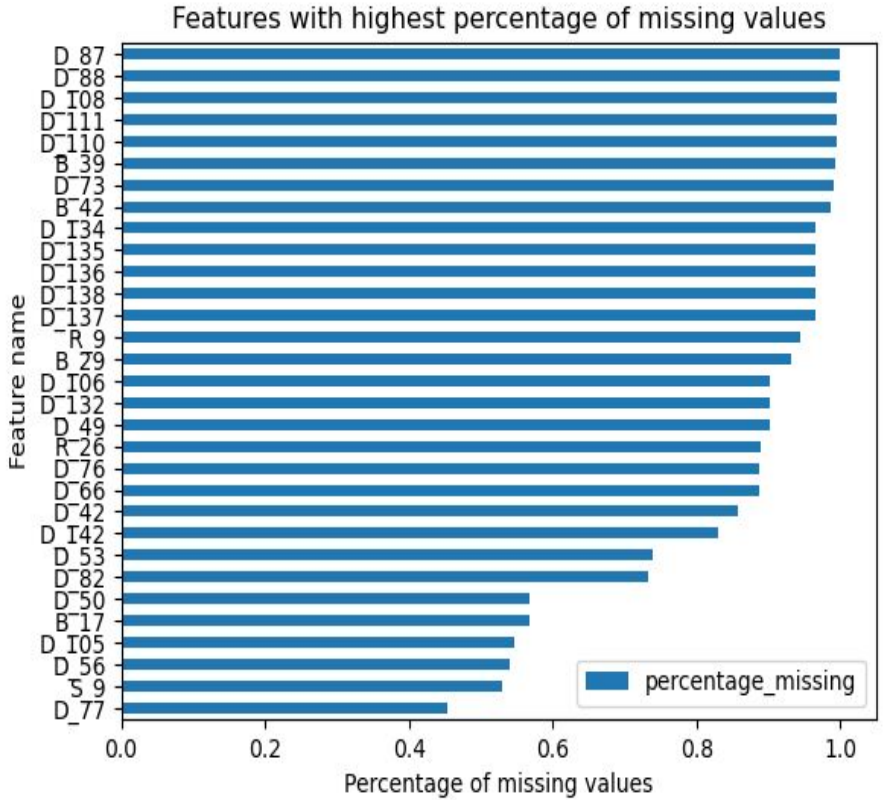


The data was already undersampled in the Kaggle dataset to facilitate predictive capabilities for the positive class. Since the data is not very imbalance resampling the data might not be necessary..

The vast majority of features are related to delinquency, while extremely few are payment variables. It is likely that variables in payment, for being so few, will be highly important decision boundaries for tree based models.

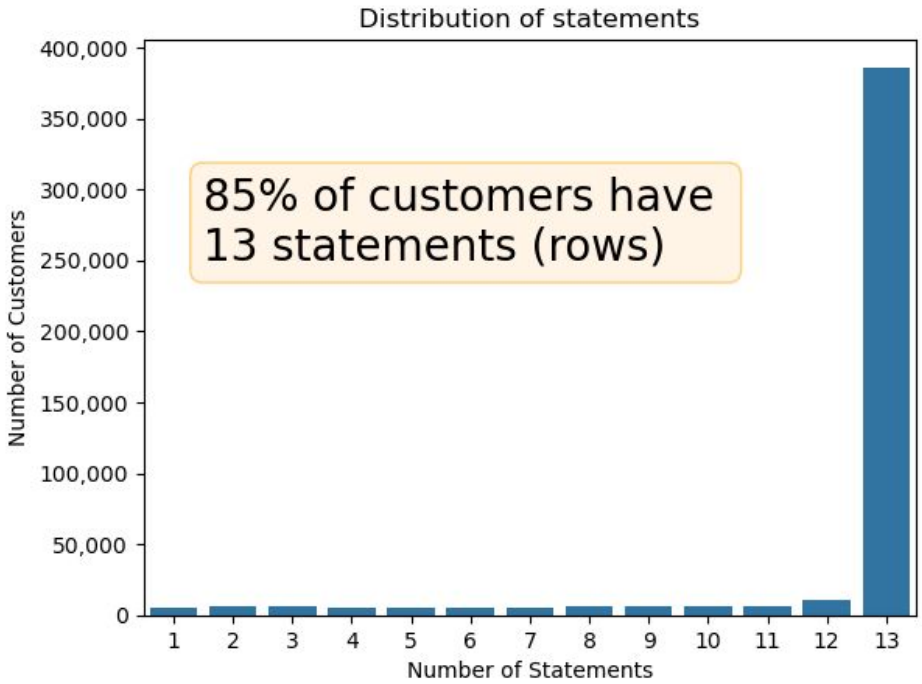
Handling Missing Values

- Out of the 190 features in the dataset, 122 of them have at least one missing value.
- On average each feature has 15% of their values missing.
- Out of the 122 features with missing values, 30 are missing 50% of their entries or more. These features will be disregarded.
- For the rest of missing values, we use mean to impute the missing values.



Data Preprocessing

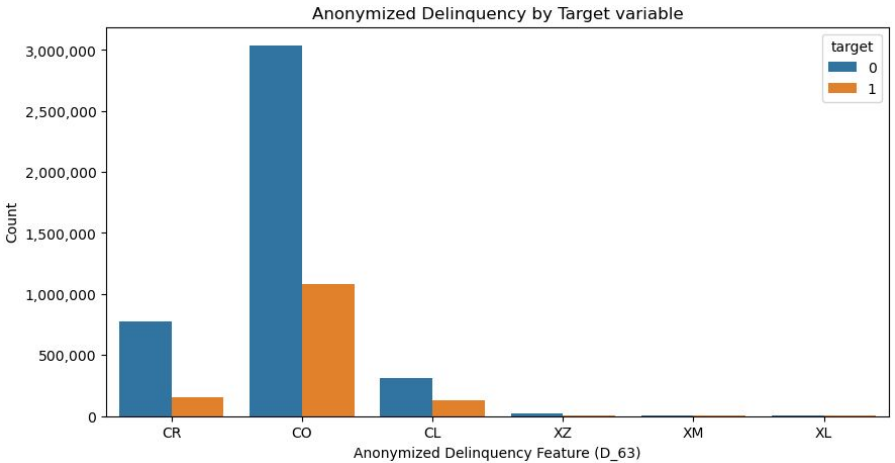
- Each customer has up to 13 statements but we only have the target variable at the customer level. To solve for this we will aggregate the data at the customer level.
- For each numerical variable of each unique customer ID, we create 5 aggregates: mean, standard deviation, min, max, and last statement value.
- For each categorical variable, we create 2 aggregates: mode, and the value of for the last statement
- This multiplies the number of features of the dataset by 5, but allows us to collapse the ~13 rows per customer into one.



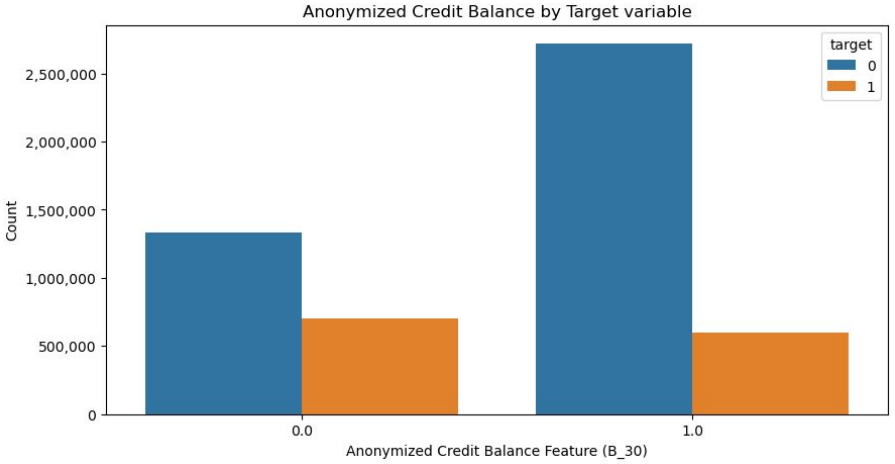
Data Preprocessing

- Data standardization is not needed. The source data is already scaled and normalized.
- In credit card default prediction, resampling is typically necessary to handle the target variable imbalance. In our case, however, the negative class was already undersampled. The negative class has been subsampled for this dataset at 5%.
- There are 11 categorical features. These features are handled by a mix of one-hot encoding and ordinal encoding.

Data Exploration - Categorical Features

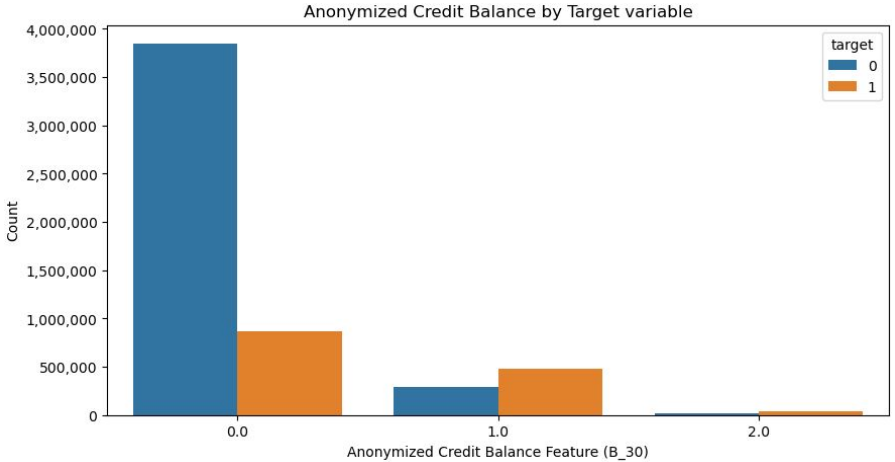


By plotting the side-by-side bar chart for a Delinquency related feature (D_63), we can observe that the target variable is more likely to be positive for values 'CL' and 'CO' compared to 'CR' (bigger blue bars compared to the orange bar). This might indicate that this feature has good predictive properties. Further examination will be done by plotting the feature importance after the modeling is completed.

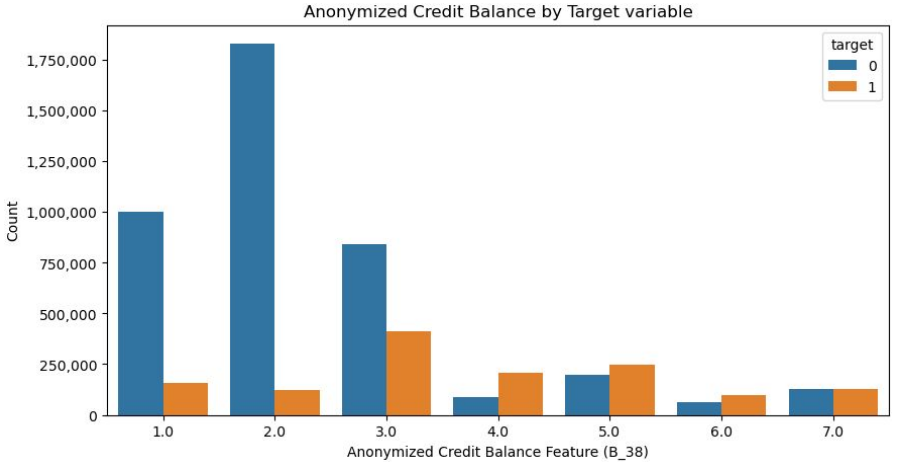


By plotting the side-by-side bar chart for a Balance related feature (B_30)), we can observe that the target variable is more likely to be positive when the feature is 0 compared to 1. This might indicate that this feature has good predictive properties. Further examination will be done by plotting the feature importance after the modeling is completed.

Data Exploration - Categorical Features (ordinal encoded)

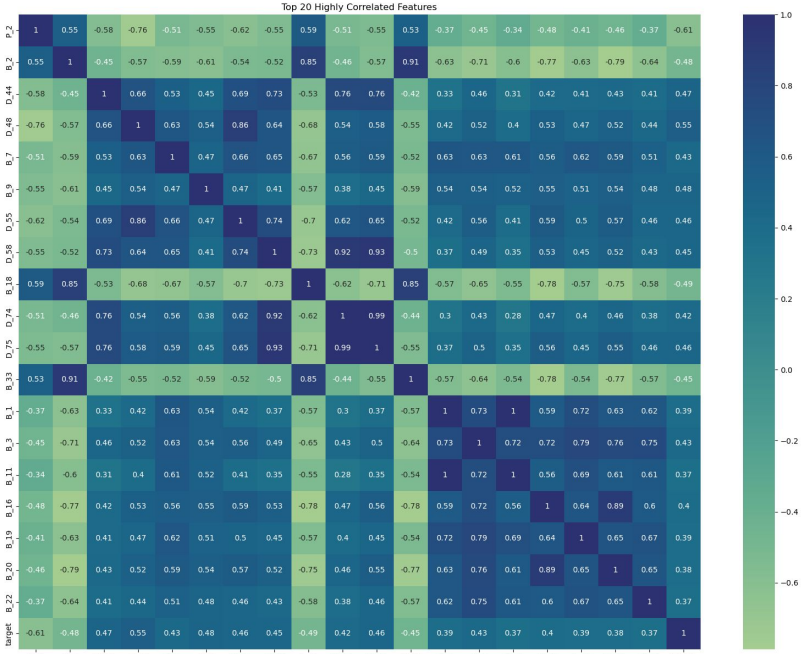


From the side-by-side bar chart for a Balance related feature (B_30), we can observe a very strong pattern. When the feature is 1 or 2, there are more positive hits than negative.

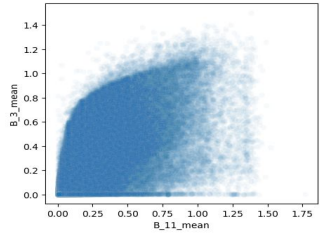


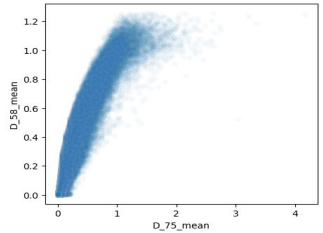
Another predictive pattern is emerged from this side-by-side bar chart for a Balance related feature (B_30). When the value of the feature is higher than 4, there are more positives for the target variable.

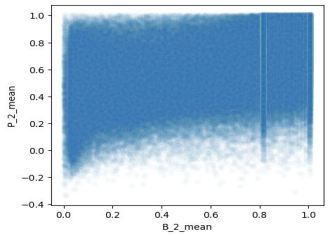
Data Exploration - Numerical Features



Here is the correlation matrix heatmap of the top 20 highly correlated features. As intuitively expected features in the same category are correlated. For instance, Balance related features (B_1, B_3, B_11) all have a strong positive correlation. Our proposed tree-based techniques are robust to these highly correlated features, but keeping track of them is important for interpreting the model. Furthermore, when measuring feature importance, it might be worthwhile to note which features are highlighly correlated to others to avoid double-counting.







By plotting two Balance related numerical features B_3 and B_11 we can observe a non-linear positive relationship. It is intuitively expected that the features in the same category are correlated.

Two Delinquency features (D_75 and D_58) show a strong positive correlation.

The relationship between two features from two separate category (P_2 and B_2) cannot be easily interpreted. Although we can see some non-linear patterns. We hope tree based models can help us to gather more insight on this.

Proposed ML Techniques

- Our baseline model will be a standard decision tree
- In order to improve upon the baseline, we will make sure of the following ensemble and boosting models:
  - Random Forest
  - Adaboost
  - HistGradientBoosting
  - XGBoost
- Other Binary Classification Models:
  - Regularized Logistic Regression
  - RBF Kernel SVM

Elaboration upon proposed techniques:

- Our primary models will be trees because they handle numerical/categorical variables and missing values well
- We improve by training on boosted trees because they focus on the error of misclassified points, which is important to avoid costly false negatives.
- XGBoost requires us to encode categorical data.
- HistGradientBoosting supports categorical variables and works fast on large datasets.
- Logistic Regression to see if the data can be linearly separable.
    - We will use a regularized version of this model to better handle highly correlated features.
- RBF SVM allows us to separate our large dataset into infinite dimensions.