Lab 01 – Feature Engineering

Lab Report

Name – P. D. Lakshan

Index number – 210333K

Table of contents

1) Introduction

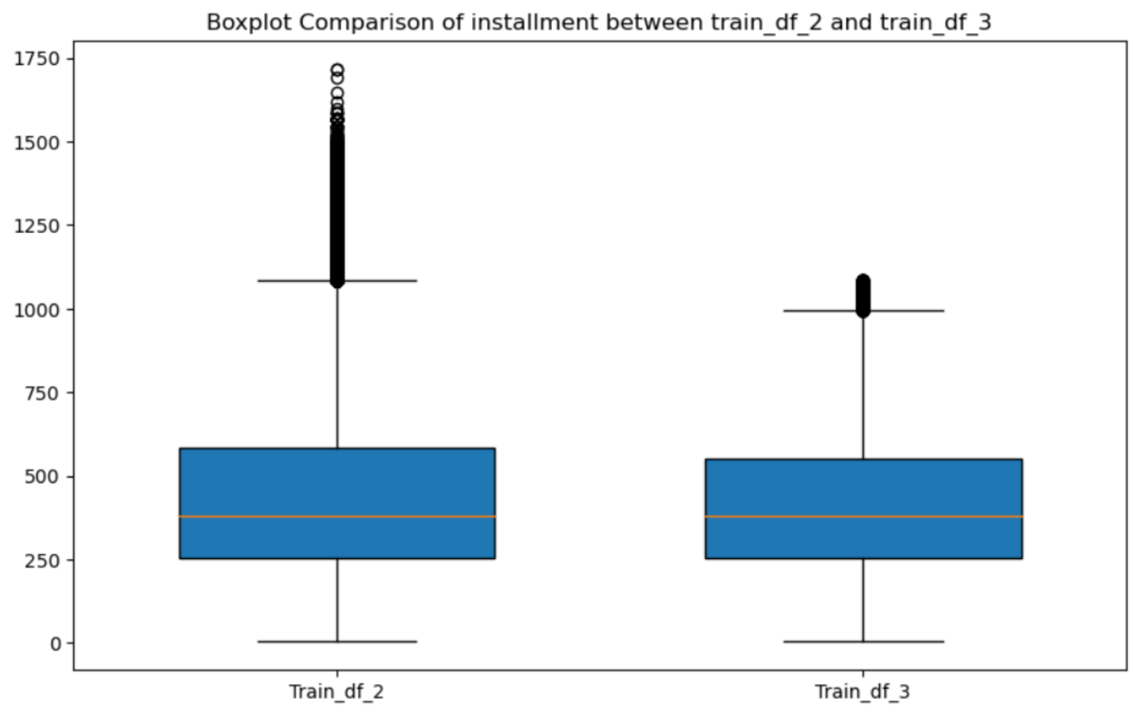
In this lab report, we analyze a loan default prediction dataset provided by a US finance company. The dataset comprises records of previous loan applicants, indicating whether they defaulted or not. Our goal is to identify patterns that signify default likelihood, enabling informed decision-making in loan approval processes. With around 860,000 observations and 150 variables, including credit scores and address details, this dataset presents real-world complexities such as missing values and outliers. By leveraging machine learning techniques, we aim to uncover actionable insights to enhance risk assessment and lending practices.

2) Data preprocessing

➢ Outlier detection and handling

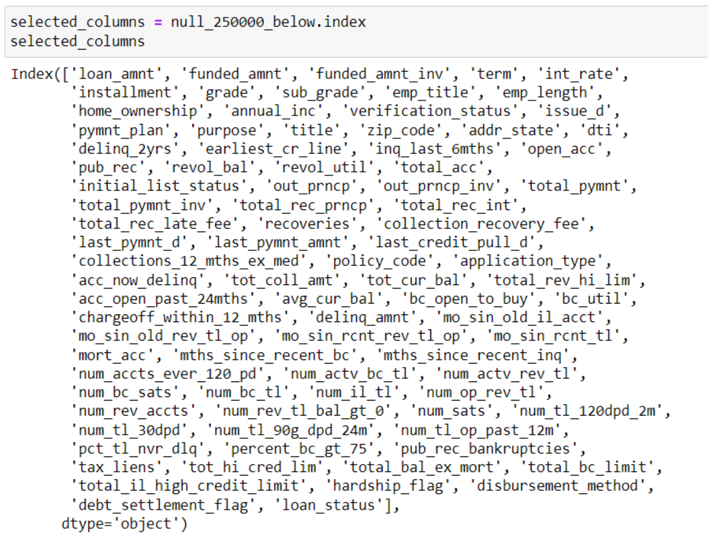
Detecting outliers means finding data points that are very different from the rest. We used boxplots to do this in our dataset, which has many columns of data. Instead of throwing away these unusual data points, which might be important, I decided to replace them with the median. This is because if we remove them completely, we might lose valuable information that could be helpful for our analysis.

Without knowing the business logic exactly, sometimes removing outliers can lead to overfit the model as well. By only adjusting outliers for specific columns, we aimed to keep our data accurate without losing too much important information and overfitting the model.



➢ Handling missing values

There were some columns with considerably large number of null entries. So, in the beginning of the data cleaning process, I remove columns which have more than 250,000 null entries (about 50% of total entries).



```
selected_columns = null_250000_below.index
selected_columns
```

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'issue_d',
       'pymnt_plan', 'purpose', 'title', 'zip_code', 'addr_state', 'dti',
       'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mths', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim',
       'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
       'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
       'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
       'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_inq',
       'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
       'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
       'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats', 'num_tl_120dpd_2m',
       'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'num_tl_op_past_12m',
       'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'pub_rec_bankruptcies',
       'tax_liens', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit', 'hardship_flag', 'disbursement_method',
       'debt_settlement_flag', 'loan_status'],
      dtype='object')
```

Then impute categorical columns with column’s mode and numerical columns with column’s mean.

3) Feature Engineering

➢ Feature encoding

During the lab experiments, we employed different strategies to process our features based on their characteristics:

- One-Hot Encoding:

Used for features with few unique categories and no inherent order. It converts categorical variables into binary vectors, where each category is represented by a binary digit.

- Label Encoding:

Applied to features with a clear order. It assigns a unique numerical label to each category, preserving the ordinal relationship.

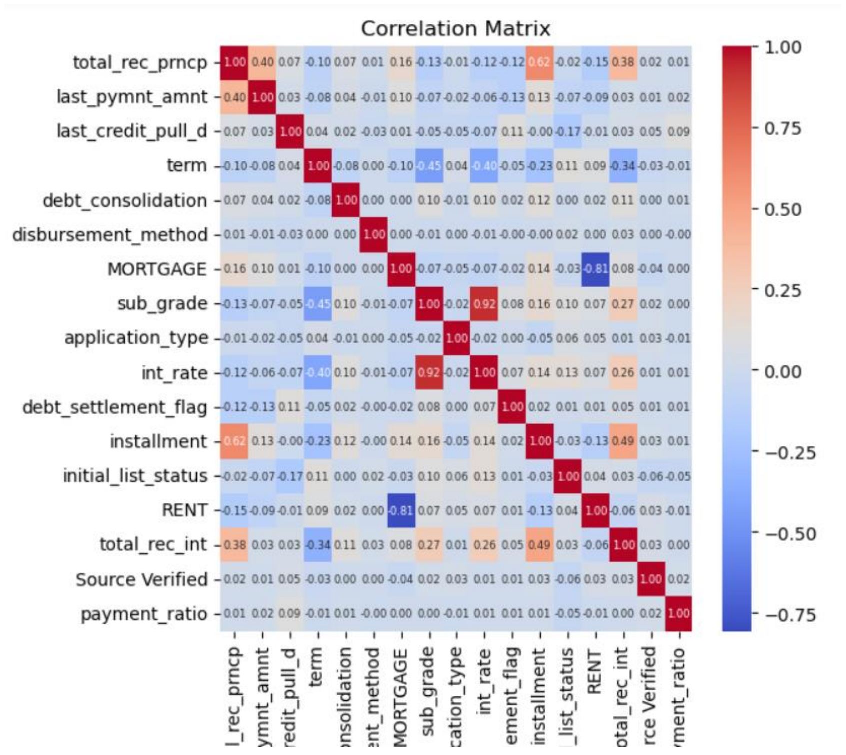
- Target Encoding:

Utilized for features with high cardinality but no specific order. It replaces categories with the mean of the target variable for each category.

These encoding techniques were chosen to ensure effective representation of our data for machine learning models, considering the nature of each feature.

➢ Feature selection

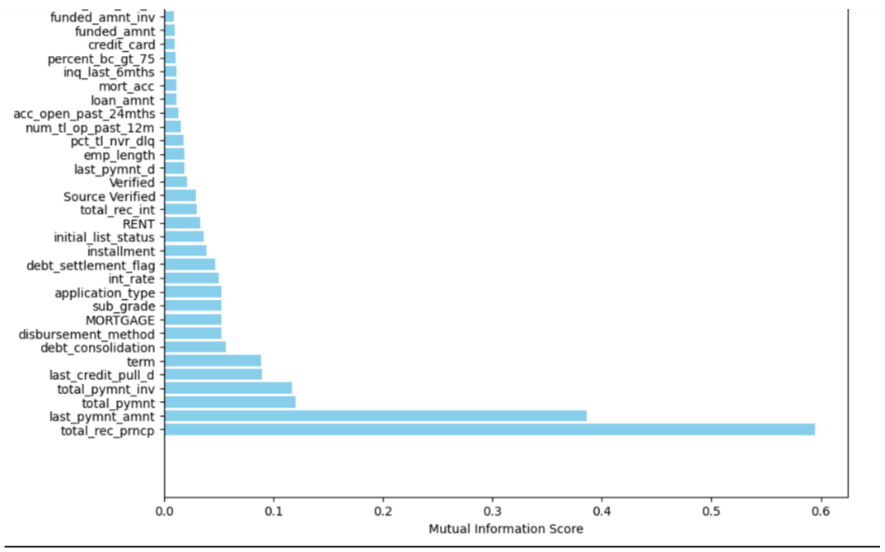
- Correlation-based feature selection



By analyzing the correlation matrix, we observe strong correlations among certain features, such as int_rate and sub_grade. Given this correlation structure, removing int_rate from the feature set could potentially enhance the efficiency of our model while maintaining its current level of accuracy. This strategic feature selection aims to mitigate multicollinearity issues and streamline the model's computational complexity, thereby optimizing its performance without sacrificing predictive accuracy.

- Using mutual information

In feature selection, mutual information gauges how much insight into one feature aids in predicting another. By utilizing the mutual information function from sklearn, we calculated the mutual information values for each feature pair. Subsequently, we sorted the features based on these values. This process enabled us to identify and select the most informative features for training our model, thereby enhancing its predictive accuracy and efficiency.

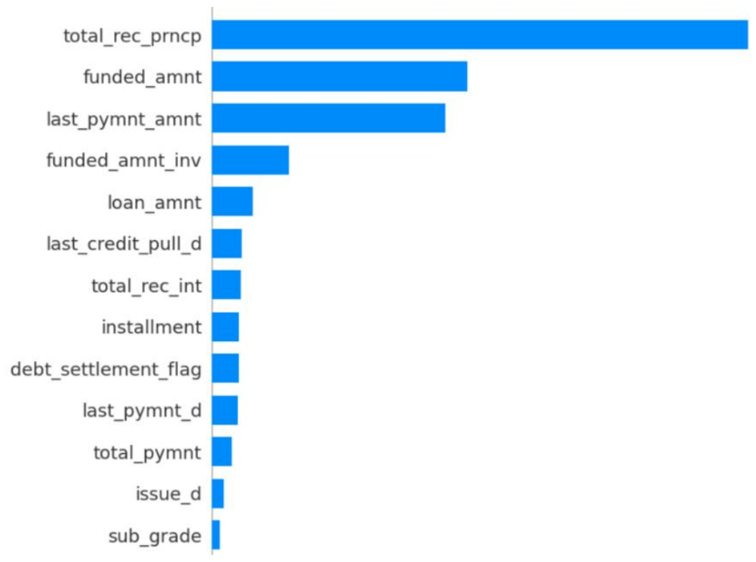


➢ Feature extraction

We noticed a strong correlation between the "total_pymnt_inv" and "total_pymnt" columns. Instead of removing one of these columns, we decided to create a new feature called "payment_ratio" by combining information from both columns. We calculated this ratio and added it as a new feature in our dataset. When comparing the accuracy of our model after either dropping a column or extracting this new feature, we found that adding the "payment_ratio" feature actually improved accuracy. This approach allowed us to maintain valuable information while enhancing the performance of our model.

➢ Other feature engineering techniques

After training the model, we employed SHAP analysis to gain deeper insights into its predictions. SHAP analysis helped us understand the contribution of each feature towards individual predictions, thereby providing valuable explanations for the model's decision-making process.



Note:

Since we plan to use XGBoost for our classification task, we don't need to worry much about scaling our features. That's because XGBoost and similar tree-based algorithms are pretty good at handling features of different scales all by themselves. So, we can focus on other parts of building our model without spending time on scaling our features.

4) Model training and evaluation

we got optimal 16 columns and we used XGBoost classifier as our model. We got over 99% accuracy from it.

5) Conclusion

In this lab, we dealt with a massive dataset containing over 500,000 data objects and 150 features. Our main focus was on feature engineering, where we learned how to manipulate and enhance these features to improve our model's performance. We experimented with different techniques like encoding methods (like one-hot, ordinal, and target encoding), selection methods (like correlation-based, mutual information, and SHAP analysis), and extraction methods.

After applying these techniques, we trained our model using the XGBoost classifier. Impressively, we achieved over 99% accuracy with fewer features, demonstrating the effectiveness of our feature engineering approaches. Overall, this experience provided valuable insights into handling large datasets and optimizing feature engineering for improved model performance.

Link for jupyter notebook – https://github.com/lakshanpd/Feature-Engineering