CS3111 - Introduction to Machine Learning

Lab 01 -Feature Engineering

Index: 210170G
Name: W.A.R.T. Fonseka

Introduction

We have a loan default dataset from a finance company in the United States. Our task is to build a machine learning model to predict whether a customer is suitable for a loan by identifying patterns in our dataset. In this lab, we used the XGBoost algorithm to build our model. We have three separate files with training data, validation data, and test data, each containing 145 features , including the predicted label. The training dataset is large, with 517,788 data rows.

In this report, I will describe how I built the machine learning model, including data preprocessing techniques, feature selection techniques, feature encoding, dimensionality reduction techniques, and SHAP analysis.

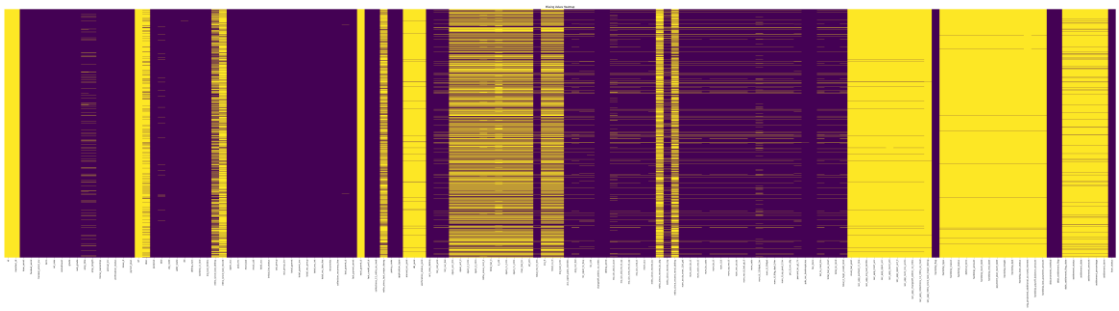
1. Data cleaning

1.1. Remove duplicate data rows

Removing duplicate data rows from a machine learning training dataset is essential for ensuring the model's accuracy and reliability. Duplicates can lead to overfitting, where the model learns to predict the training data too well but fails to generalize to new, unseen data. By eliminating duplicates, we reduce the risk of overfitting and ensure the model is trained on a more balanced and representative dataset. This not only improves the model's performance but also simplifies its interpretation and reduces training time and computational resources.
But luckily, we do not have any duplicates in our training dataset.

1.2. Remove features with more missing values

Features with more missing values do not contribute to the creation of a high-accuracy machine learning model. On the other hand, such useless features can increase our training time, consume more resources, and potentially decrease our model's accuracy. Therefore, it is essential to remove them. I used a heatmap to graphically visualize the missing values.



We can observe that many features have a large number of missing values. Therefore, I will remove features with missing values that exceed a threshold of 0.5, meaning those with half or more missing values.
It reduces our feature size from 145 to 87.

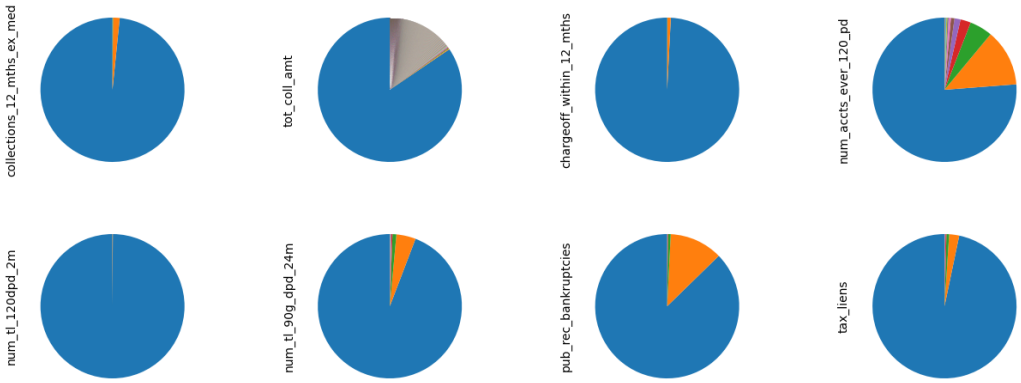
1.3. Fill missing values

I will not fill all features with missing values, as some features are correlated with each other, and I will remove them later. Instead, I will focus on the more important features. Additionally, I can calculate values for some missing values using these correlated features.

- We can calculate missing values for 'dti' using other features (dti = monthly debt payment / annual income). Some 'dti' values are null because the annual income is 0. In reality, these values go to infinity. Since we cannot store infinity, we fill null values with the (max(dti) + 1000).

- I calculated missing values for 'revol_util' using the formula (revol_bal / total_rev_hi_lim) * 100. However, there are still missing values because the 'total_rev_hi_lim' value is 0. For these values, we fill with the (max(revol_util)+1000)

- Now, I plot charts for some features and examine what kind of data they have.



In the above graphs blue color shows count of 0 values they have. We can see that most of the values are 0. Therefore, filling missing values with 0 should not cause

much harm. My assumption was that null values represent the value 0. Sometimes, when we do not have certain data, we leave it empty instead of putting 0.

- Since some data is not mentioned, it may lead to the rejection of loan requests and there exists null value because the customer never had that data. It is important to represent that kind of data in a specific way. Filling missing values with a specific value could harm the dataset(because the missing value count is too high). I fill missing values with a non-existent data value, such as -1, for such features. The advantage is that we can represent null values as -1, which may help the model perform better.
  Features that I fill missing values with -1 :

tot_cur_bal	acc_open_past_24mths	bc_open_to_buy	bc_open_to_buy
mort_acc	mths_since_recent_inq	mo_sin_rcnt_tl	total_bc_limit
num_bc_tl	mo_sin_rcnt_rev_tl_op	num_actv_bc_tl	total_bal_ex_mort
num_il_tl	mths_since_recent_bc	pct_tl_nvr_dlq	mo_sin_old_il_acct

- Assume that missing values in datetime features indicate that the datetime is far in the past. Therefore, fill them with a value older than the oldest value of that feature.(use Jan-1960)
  'last_pymnt_d' and 'last_credit_pull_d' missing values fill with using this method.

1.4. Data transformation

- Change the 'issue date' feature to 'issue year', because in banking, the most important aspect is the issue year, not the actual day. In a similar manner and for similar reasons, we can convert the 'zip_code' values.
- In features with datetime, I replaced the dates with how many months older they are compared to the current date. Otherwise, the model may think it depends on the year, but it actually depends on how much time has passed from then to now.

1.5. Remove data that is not useful

In 'pymnt_plan', 'hardship_flag', 'policy_code', 'out_prncp', and 'out_prncp_inv', we can see that all the data in the feature are the same. This type of data is not useful for training the machine learning model. Therefore, we should remove it.

2. Feature encoding

I use Label Encoding and Ordinal Encoding because we want to minimize the number of features. Instead, using techniques like One-Hot Encoding increases the number of features. In this situation, it's not harmful to use Label Encoding and Ordinal Encoding in a suitable way.

2.1. Label encoding

I applied label encoding for following features:

term	home_ownership
purpose	addr_state
initial_list_status	application_type
disbursement_method	debt_settlement_flag

2.2. Ordinal encoding

I applied ordinal encoding for following features:

sub_grade	emp_length	verification_status

2.3. Other data specific method

In this situation, the employee title is not very important. On the other hand, the data dictionary states that 'Employer Title' replaces 'Employer Name' for all loans listed after 9/23/2013. Therefore, the most important thing is whether the employee title is mentioned or not. I encode this feature such that if the employee title is mentioned, then it is assigned a value of 1; otherwise, it is assigned a value of 0.

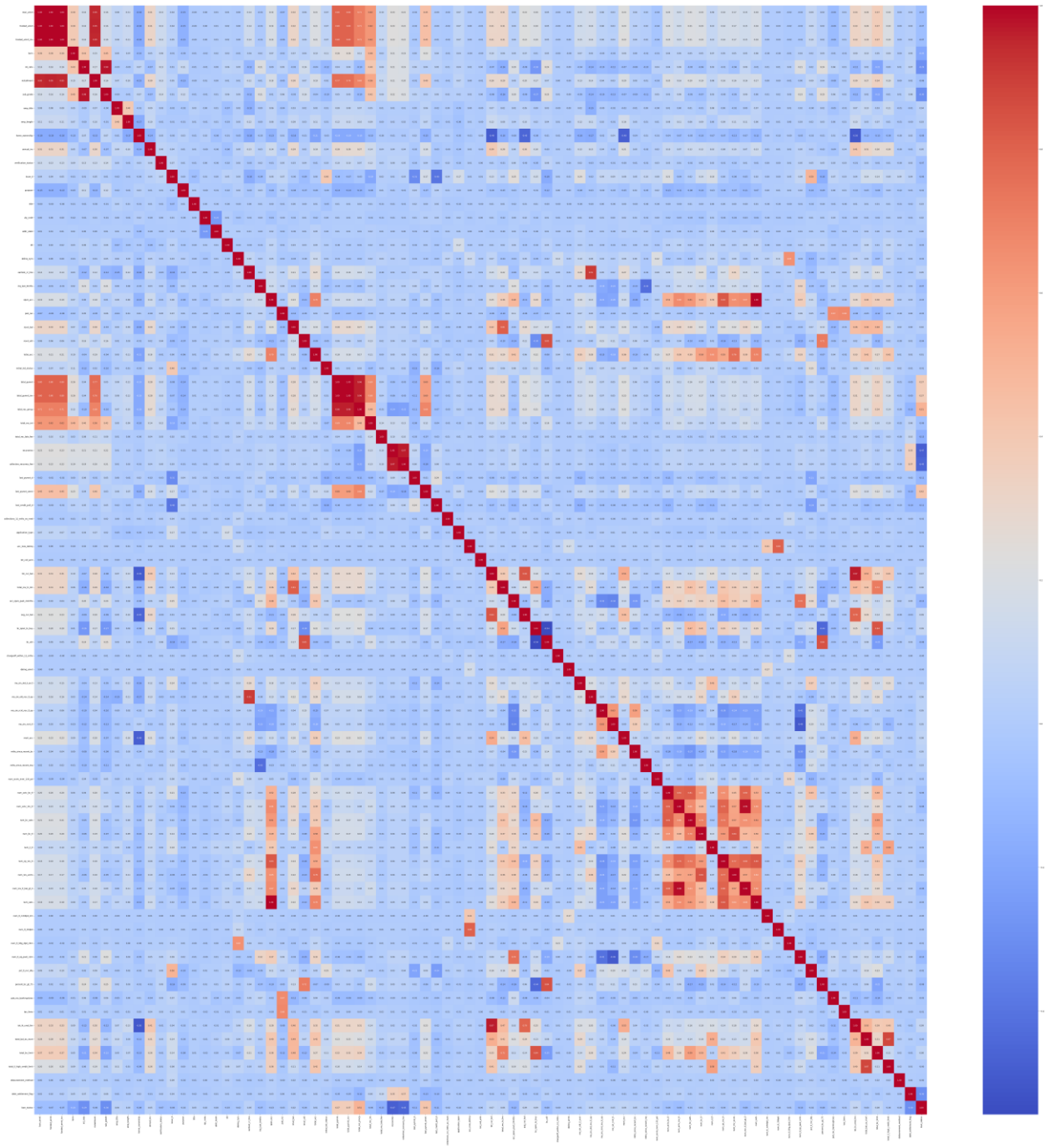
Similarly, the 'title' can be anything that the customer provided. The important thing is whether the title exists or not.

3. Remove correlated features

We can clearly see that some features are highly correlated with each other, while others are not clearly see.

'grade' and 'sub_grade' are correlated, so we can remove 'grade' (but not 'sub_grade', as it provides more information than 'grade')

Now, I created a correlation matrix and plot it as a heatmap, so that I can recognize correlated features.



We can see that there are some positively correlated features, but there are no highly negatively correlated features. Therefore, I will remove them by selecting a threshold value of 0.75.

After applying all the techniques our number of features was reduced to 57.

4. Feature selection

I chose the Recursive Feature Elimination (RFE) method to select the best features. I ran the algorithm several times to identify the minimum number of features that still yielded high accuracy. After multiple iterations, I determined that the optimal number of features was 8, and they are as follows:

1) loan_amnt
2) term
3) issue_d
4) total_rec_late_fee
5) recoveries
6) last_pymnt_d
7) last_pymnt_amnt
8) debt_settlement_flag

Also, I used cross validation method to calculate accuracy and I used 5 number of folds.

5. Dimensionality reduction

When I used PCA as a dimensionality reduction technique, my model's accuracy decreased because reducing more features can lead to missing some important attributes. So, I did not apply PCA for our model. On the other hand, I don't want much because our feature count is low, which is 8.

6. Hyperparameter tuning

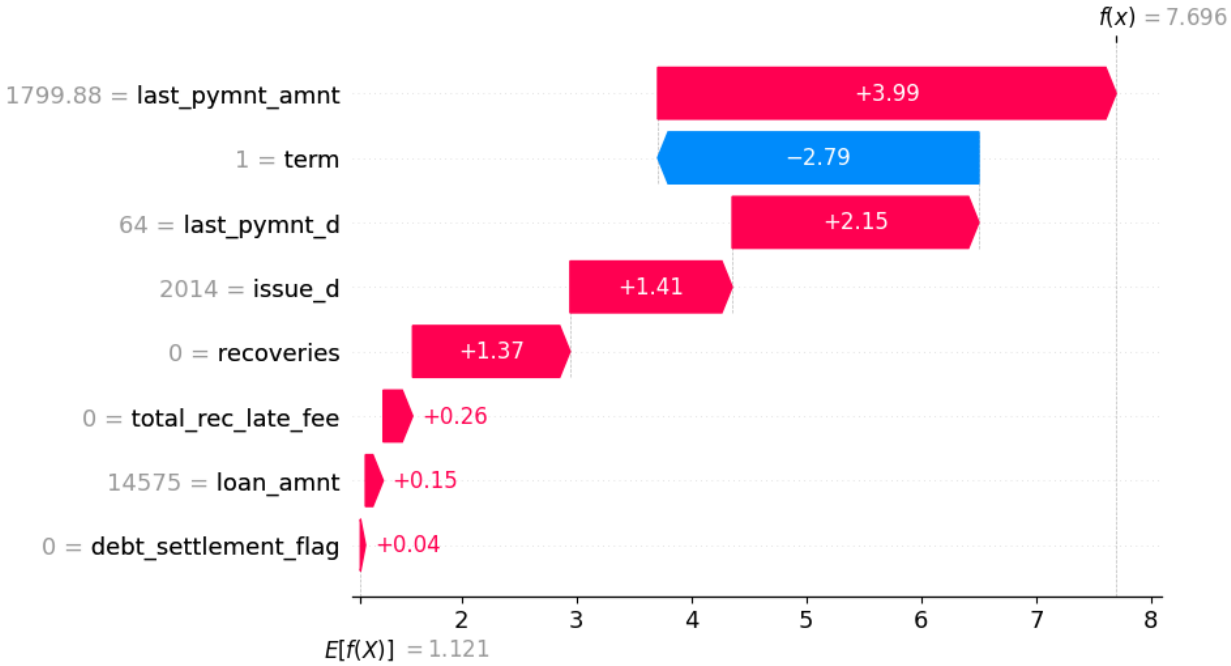
I tuned the hyperparameters of the XGBoost model using GridSearchCV. After tuning, the optimal values for the parameters are as follows:

1) colsample_bytree: 0.9
2) learning_rate: 0.3
3) max_depth: 6
4) n_estimators: 200
5) subsample: 0.9

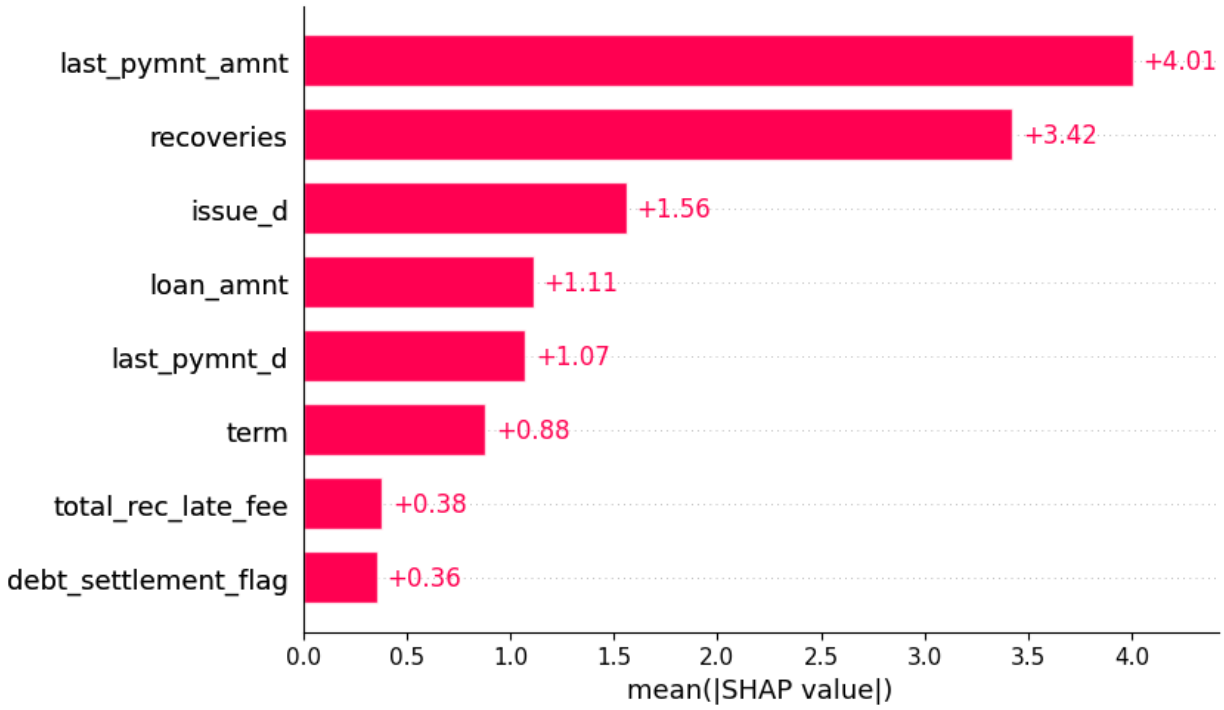
7. Train and validation

After training our model, we need to evaluate its performance on a validation dataset. To do this, we transform the validation data in the same way we did for the training data, ensuring that the model is tested on data it has not seen before. In this case, the accuracy of our model on the validation dataset was 0.99, indicating that it performs very well on unseen data. This high accuracy suggests that our model is robust and generalizes well to new data.

8. SHAP analysis for explainable AI



The waterfall plot shows how each feature contributes to the final prediction for a single instance. It starts with the base value (the average prediction for the entire dataset) and adds or subtracts the SHAP value of each feature to arrive at the final prediction. Features that contribute positively to the prediction are shown in red, while those that contribute negatively are shown in blue. The length of each bar represents the magnitude of the feature's contribution.



The bar plot shows the average SHAP value for each feature across the entire dataset. This plot provides a global view of feature importance. Features with larger average SHAP values are more important in making predictions, while those with smaller values are less important. The bar plot can help identify which features have the most significant impact on the model's predictions overall.

Conclusion

In this lab, I have successfully built a machine learning model to predict loan suitability for customers based on patterns in a dataset provided by a finance company in the United States. We employed the XGBoost algorithm and utilized a range of techniques for data preprocessing, feature selection, encoding, dimensionality reduction, and SHAP analysis.

Preprocessing techniques such as data cleaning, encoding, and data transformation are crucial for increasing our model's accuracy and reducing features. They offer benefits such as reduced training time and lower resource consumption.

PCA or other techniques do not always increase our model's accuracy. It is important to choose the correct techniques for a specific model or dataset.

A graphical representation of data is beneficial as it provides us with more information about the data.

By adjusting hyperparameters, such as learning rates and tree depths, we can fine-tune the model to achieve better accuracy and generalization. This process is crucial for ensuring that the model is robust and performs well on unseen data, ultimately leading to more reliable predictions.

SHAP analysis is crucial for explainable AI as it provides insights into how the model makes predictions. By understanding the contribution of each feature to the final prediction, we can gain valuable insights into the model's decision-making process. This transparency is essential for building trust in the model and ensuring that it is making predictions based on meaningful and interpretable patterns in the data.