CS 3111 -Introduction to Machine Learning

Lab 01 – Feature Engineering

➢ *Introduction*

The task of this lab was to apply our knowledge of data preprocessing, feature selection and engineering for the given data set in order to develop the best-performing machine learning classification model. We were given with a loan default prediction data set and had to develop an **XGBoost Classifier** model using the training data and with feature engineering techniques. Then, using the validation data checked the accuracy of the model. Finally had to find the reduced set of features (minimum number of features) to predict the target label more accurately.

➢ *Data Set*

- Three separate data sets to train, validate and test.
- Dataset name : Loan default prediction data set
- Total number of features : 145
- Number of train records : 517788
- Number of valid records : 172596

➢ *Data Analysis & Preprocessing*

1. Handling missing values

After having a look at the data set, it was clear that there were many missing values and some columns with all null values as well. Those null values needed handling before training the model. So, following steps were taken to handle missing values.

- Dropping columns with all null values. (‘id’, ‘url’, ‘next_pymnt_d’, ‘member_id’)
- Dropping columns with more than 50% missing values. (Now all the columns have less than 13% missing values)
- Selecting numerical features and imputing missing values in the features with median values.
- For the categorical features, imputing missing values in the features with the mode.
- All these transformations were done to all train, validate and test datasets.

2. *Handling outliers*

Existence of outlier values can have a significant impact on the model performance and decision-making process. So, such outliers need to be handled at the earliest moment.

- Calculating **summary statistics** for the numerical features.
- Identifying observations that are 3 standard deviations away from the mean as outliers.
- Trim the outliers in numerical features.
- These transformations also applied to train, validate and test datasets.

3. *Feature Encoding*

Feature encoding is also essential machine learning. Because most machine learning algorithms require numerical input data. So, all the categorical features were subjected to feature encoding process.

- Firstly, categorical features are recognized and all the unique values for each feature in the train, validate and test datasets are analyzed separately.
- Based on those observations, different encoding techniques were used for encoding various data.

- For the features with two possible values such as ‘term’ and ‘initial_list_status’, **One-Hot encoding** technique was applied to convert into numerical values. The original columns were replaced with the encoded values.
- For the features with more than two unique values such as ‘grade’, ‘subgrade’, ‘home_ownership’, ‘verification status’, and ‘application_type’, **label encoding** was used to prevent from increasing the dimensionality by One-Hot encoding. Dictionaries were used to map the values to ensure the consistency of mapping through the train, validate and test datasets.
- The features that contain dates as values were encoded into **Date-time format**. (Separating year and month) Ex:’issue_d’, ‘earliest_cr_line’, ‘last_pymnt_d’, ‘last_credit_pull_d’ etc.
- ‘purpose’ feature contained 14 unique values. So that column was subjected to **frequency encoding** based on its frequency values.
- Some features that contained only a single unique value such as ‘pymnt_plan’, ‘hardship_flag’ were dropped from the dataset. Because, since it contains the same value for all the data records, the model could not use this feature for prediction task.
- There were some features that contained a large number of unique values such as ‘title’, ‘emp_title’ and ‘addr_state’. Those values were hard to encode. So, they were dropped from the dataset.
- Accordingly, all the categorical features were encoded into numerical values.
- Note that all these encodings were done across all training, validate and test datasets.

4. *Feature Scaling*

Feature scaling is the process of transforming the features of a dataset so that they have a similar scale or distribution. Feature scaling is important for the algorithm sensitivity, faster convergence and improved performance.

- In this lab activity, **standardization** technique was used to scale the numerical features.

5. *Correlation Analysis*

The given dataset contains a large set of features. So, it is important to reduce the number of features by avoiding useless features to avoid the curse of dimensionality. Identifying correlations among features helps in feature selection process, identifying redundant features and detecting multicollinearity. Avoiding highly correlated features help to improve model performance and improve the data quality. In this lab activity, correlation analysis is done in the following manner.

- **Numerical correlation matrices** are used to compute the correlation between the numerical features. It gives a numerical value that represents the correlation between the features.
- A **contingency table** was used to calculate the statistical measures of association between categorical variables such as chi-square tests, Cramer's V statistic and contingency coefficients.
- Finally, a **heat-map** is plotted using the calculated correlated values for easy visualization.
- Using the data from correlation matrix, highly correlated feature pairs with more than 90% correlation were recognized and then one of the features from those **correlated pairs** was dropped from the dataset to avoid redundancy.
- These redundant features were dropped from validate and test data as well.

6. *Feature Selection*

Selecting the most relevant features and discarding the irrelevant and redundant features helps in improving the model performance, reduced complexity, noise reduction and addressing the curse of dimensionality.

- During the preprocessing steps so far, the number of features in the dataset reduced from 145 to 73.
- **Mutual Information Classification**: 'mutual_info_classif' function from scikit-learn's 'feature_selection' module was used to compute the mutual information between each feature and the target variable.

- **SelectKBest**: The ‘SelectKBest' class from scikit-learn’s ‘feature_selection’ module was used select the best 20 features based on the computed mutual information scores for each feature.

7. *Building & Training the Model*

In this lab activity, an XGBoost Classifier model is developed to train the model on the given dataset.

- XGBClassifier is imported from the xgboost and initialized the model.
- Then the train data is split into x_train and y_train sets.
- Thereafter the model is trained with x_train and y_train sets with the default values for its hyperparameters.

8. *Hyperparameter Tuning*

Once the initial model is built, hyperparameter tuning is often conducted to optimize the model’s performance.

- A hyperparameter grid is specified including the hyperparameter values and range of values to search over.
- In this lab activity, ‘learning_rate’, ‘max_depth’, ‘n_estimators’, ‘gamma’, ‘subsample’ and ‘colsample_bytree’ hyperparameter values are tuned.
- There are two methods to perform hyperparameter tuning, namely random search and grid search. Here the random search technique is used to find the optimal hyperparameters for which the model accuracy is at a maximum.

9. *Cross-validation*

Cross-validation is used to assess the performance and generalization ability of a machine learning model.

- In this lab task, cross-validation is done in 10 folds and evaluation is done for the classifier on each fold.
- An array of scores is returned where each score represents the performance of the classifier on a particular fold.

- The cross-validation was done on the validation dataset, which is divided into x_validate and y_validate sets.
- In this case, the mean value of the cross-validation score is approximately 98%.

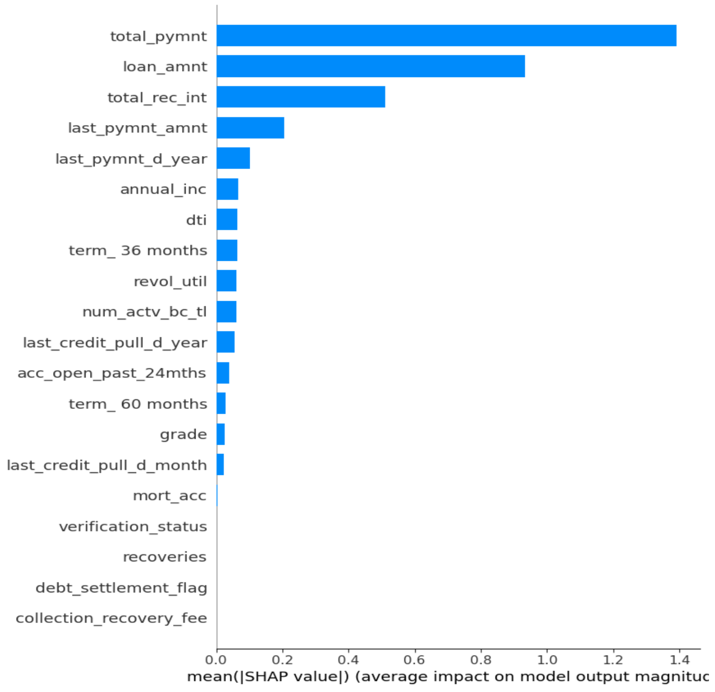
10.Predicting values

After building the best-performing model, the target values can be predicted for the test dataset. Finally, the target values are predicted for the test data with the use of XGBClassifier and the result is appended to the test dataset and the final csv file with predicted results is downloaded.

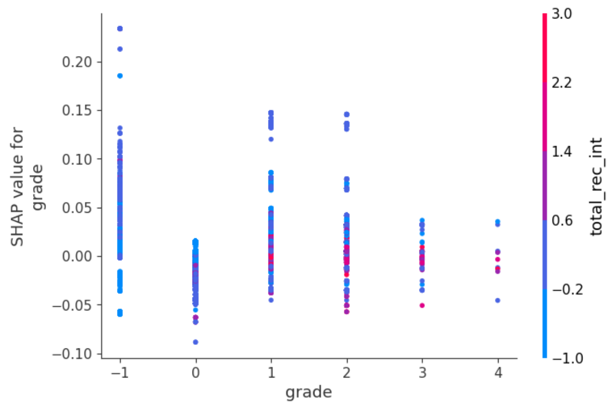
11.Sharply values as an explainable AI technique

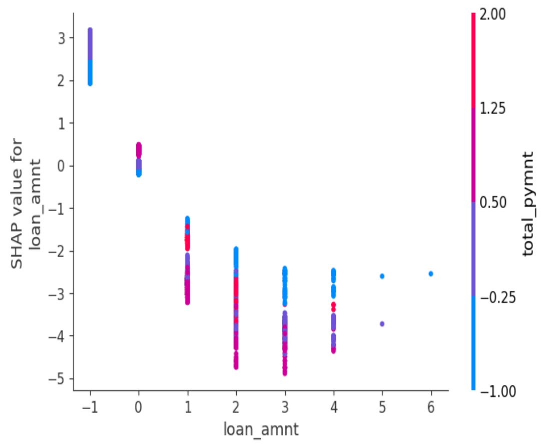
Shapley values quantify the contribution of each feature in the dataset. They quantify the contribution of each feature to the model's prediction for a specific instance.

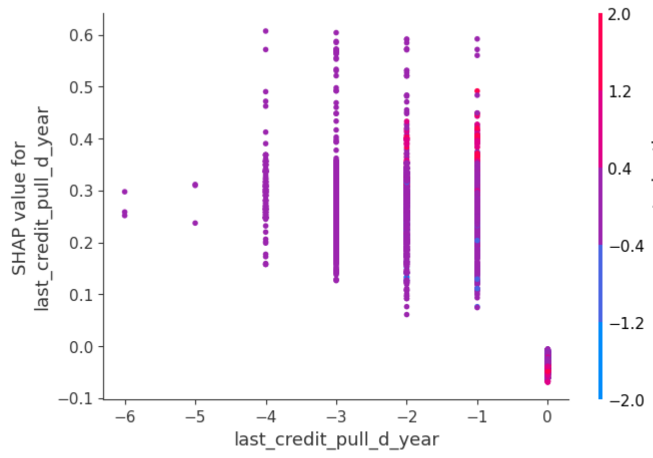
- Mean shapley values are calculated for each column in the train dataset.
- Summary Plot is used to have an overview of the Shapley values for all features in your dataset. The graph shows the distribution of Shapley values for each feature and ranks them based on their importance.

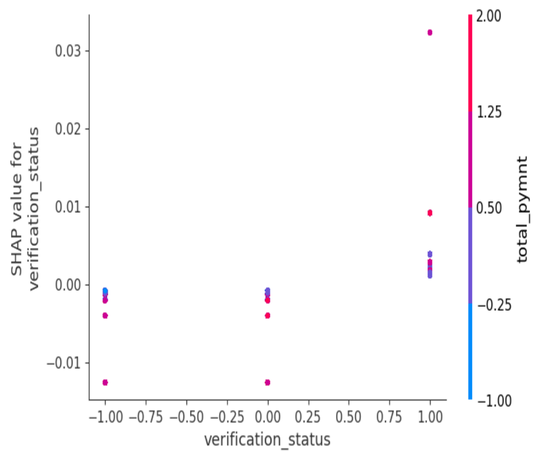


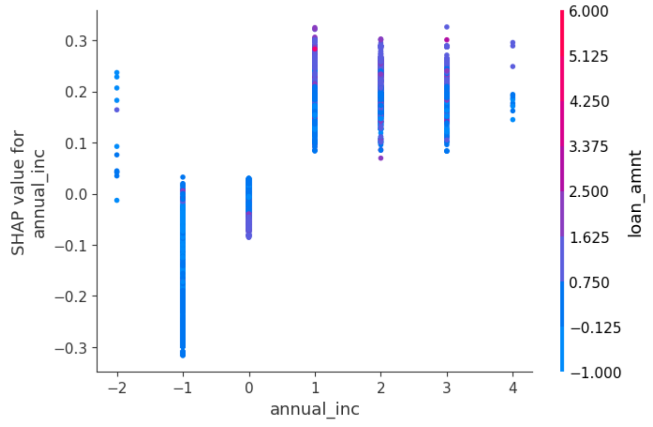
- Individual Feature Importance Plot visualizes the Shapley values for a specific feature across different instances in the dataset. This shows how the feature's value influences the model's prediction for a specific instance.

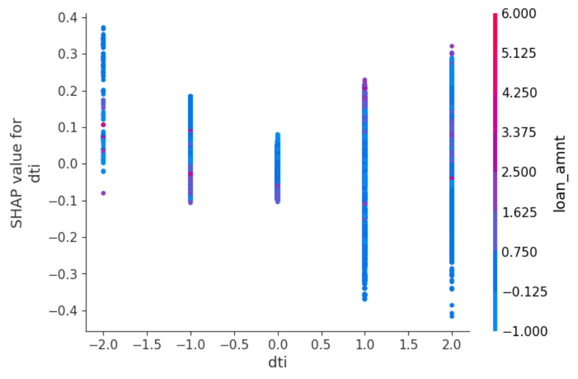


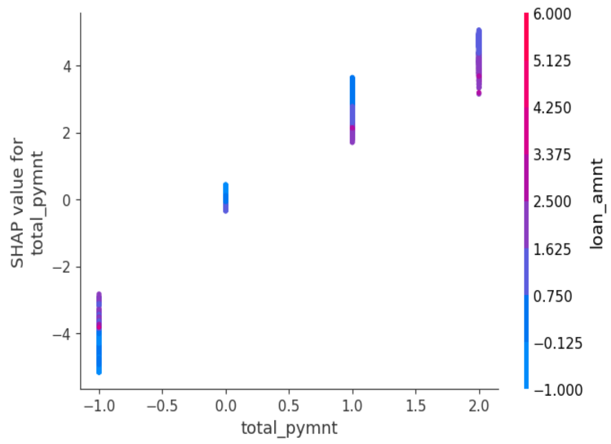


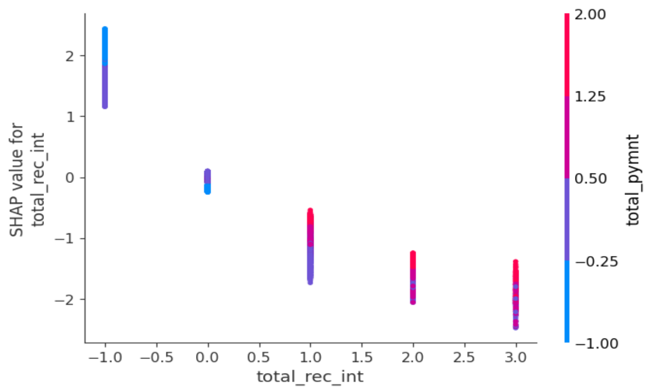












- Positive Shapley values indicate that the feature positively contributes to the model’s prediction, meaning that higher values of the feature tend to increase the model’s prediction.
- Negative Shapley values mean that higher values of the feature tend to decrease the model’s prediction.

12.Conclusion

Based on these Shapley values we can see that the ‘total_pymnt’, ‘loan_amnt’, ‘total_rec_int’, ‘last_pymnt_amnt’ are seem to be the most influential features for the model’s performance. Hence we can conclude that those features have the most significant impact on the model’s predictions and we should prioritize them for further analysis and feature engineering.