Exploratory Data Analysis and Data Preparation Report

Objective

The objective of this analysis was to clean and preprocess a dataset related to loan default prediction. The preprocessing steps included handling missing data, encoding categorical variables, converting data types, imputing missing values, and preparing the dataset for further modeling. This report outlines the steps taken during the exploratory data analysis (EDA) and data preparation process.

1. Data Loading and Initial Inspection

The dataset was initially loaded from a CSV file, and a preliminary inspection was conducted to understand its structure. The inspection revealed the dataset's columns, data types, and potential issues, including missing values and incorrect data types. This initial analysis was crucial for determining the subsequent data preparation steps.

2. Missing Data Handling

2.1 *Null Value Analysis*

A custom function was used to assess missing values across all columns. Two key metrics were calculated for each column:

- NullSum: The total number of missing values.
- NullMean%: The percentage of missing values.

This step helped in identifying columns with a significant proportion of missing data, enabling informed decisions on how to handle them.

2.2 *Dropping Rows and Columns*

Rows with missing values in the critical [member_id] column were dropped to maintain the integrity of the dataset. Additionally, columns with more than 50% missing data were removed to avoid including unreliable or incomplete data in the analysis. This step ensured that the remaining data would be reliable and usable for modeling.

3. Handling Categorical Variables

3.1 *One-Hot Encoding*

Categorical variables such as [home_ownership] and [purpose] were transformed into binary features using one-hot encoding. This approach created separate binary columns for each category within the original categorical variables. By retaining all categories in the encoding (i.e., not dropping the first category), a more comprehensive representation of each feature was maintained. This transformation ensured the dataset was in a suitable format for machine learning algorithms, which require numerical input.

4. Data Type Conversion

4.1 *Conversion of Percentage Columns*

Certain columns in the dataset, such as [int_rate] and [revol_util], contained percentages in string format. These percentage values were cleaned by removing the "%" symbol and converting the values to float type. This step ensured that the percentage columns were in a usable numerical format, appropriate for further analysis or modeling.

4.2 *Extracting Numeric Values*

Other columns like [term] and [emp_length] required conversion from string format to numeric types. The values were extracted as numbers and cast into the appropriate data types: integers for [term] and floats for [emp_length]. This ensured that all columns had the correct data type, preventing errors during model training.

5. Imputation of Missing Values

5.1 *Imputation Based on Skewness*

Missing values in numerical columns were handled based on the skewness of each column's distribution:

- If a column showed high skewness (greater than 0.5), the median was used for imputation to avoid distorting the data with extreme values.

- If a column had low skewness, the mean was used to fill in the missing values. This approach ensured that imputation was done in a way that was consistent with the distribution of the data.

This method helped preserve the overall structure and distribution of the data while filling in missing values in a meaningful way.

6. Saving the Cleaned Data

After performing all the necessary preprocessing steps, the cleaned dataset was saved to a CSV file. This allowed the data to be stored in a format that can be easily used for subsequent analysis or machine learning model training. By excluding the index column from the saved file, the data was saved in a cleaner format without unnecessary row labels.

7. Duplication Check

Duplicate records were identified and removed to ensure that the dataset contained only unique entries. The duplicated() function was used to check for and eliminate duplicate rows, improving the dataset's quality and ensuring that the model is trained on non-redundant data.

Conclusion

The dataset was successfully cleaned and preprocessed through several steps:

- Missing data was handled by removing rows with critical missing values and columns with excessive missing data.
- Categorical variables were encoded using one-hot encoding to make them suitable for machine learning models.
- Percentage columns were converted into numeric values, and other columns were converted to their appropriate data types.
- Missing values in numerical columns were imputed based on skewness, ensuring that the imputation method was appropriate for the distribution of the data.
- The cleaned dataset was saved for future use in machine learning modeling.

These steps ensure that the dataset is now prepared for use in building predictive models for loan default prediction, with a focus on providing accurate and reliable data to improve model performance.

EDA and Feature Selection for Loan Default Prediction

Objective

The goal of this analysis is to perform exploratory data analysis (EDA) on a loan default prediction dataset, which includes both numerical and categorical features, missing values, and potential outliers. The target variable is binary, representing whether a loan defaults (1) or not (0).

Distribution of the Target Variable (bad_flag)

The target variable, bad_flag, indicates whether a loan defaults (0) or not (1). A pie chart was created to visualize the distribution of this binary variable, providing insights into potential class imbalances. The results showed a smaller proportion of non-default loans (1) compared to defaults (0). This imbalance is important to note as it may affect model performance, necessitating techniques such as oversampling the minority class to prevent bias toward the majority class.

Distribution of Numerical Features

Several key numerical features were analyzed using histograms overlaid with kernel density estimation (KDE). Features such as loan_amnt, int_rate, annual_inc, and dti were included in the analysis. The histograms revealed that features like annual_inc (annual income) and loan_amnt (loan amount) exhibited significant right-skewness. This skewness suggests that most loans are relatively small, with a few large loans acting as outliers.

Correlation Analysis

A correlation matrix was calculated to examine the relationships between numerical features. This matrix was visualized with a heatmap, which helped identify any strong correlations between pairs of features. By carefully examining the correlations between variables, redundant features can be identified and removed, ensuring a more efficient and effective model.

Feature Importance Using Random Forest

A Random Forest classifier was trained on the dataset to assess the relative importance of each feature in predicting loan defaults. The model's feature importance scores were visualized, showing that 15 features were the most influential features. These features are essential as they provide key financial information. Features with lower importance scores can be excluded from the modeling process to reduce complexity and improve model efficiency.

Top 15 Features Selection

Based on the feature importance analysis from the Random Forest model, the top 15 most relevant features were selected. Focusing on these top features improves model performance by training on the most critical variables, reducing both computational complexity and the risk of over fitting. This step ensures that the model is not distracted by irrelevant or noisy features, which could otherwise degrade its predictive power.

Comparison of Feature Selection Methods: Chi-squared Test vs. Random Forest

To further validate the robustness of the feature selection process, a comparison was made between the features selected using the Random Forest model and the Chi-squared test. The Chi-squared test was applied to select the top 15 features. When comparing the features selected by both methods, several common features were identified, reinforcing their importance. Additionally, features unique to each method were explored to identify additional patterns or relationships in the data. This comparison helped ensure that the selected features are indeed the most significant, whether from a statistical or machine learning perspective.

Anomaly Detection Using Isolation Forest

Anomaly detection was performed using the Isolation Forest algorithm, which identifies outliers by isolating observations in the feature space. Anomalies were detected by calculating anomaly scores, and the instances were classified as normal (1) or anomalous (-1). The analysis found that a small proportion of the instances were classified as anomalies, which may represent errors or unusual loan conditions. These anomalies need

to be addressed carefully to ensure they do not skew the model's performance. Visualization of the anomaly scores showed a clear separation between normal instances and outliers, highlighting the importance of treating these outliers before training the model.

Conclusion

The exploratory data analysis provided valuable insights into the dataset, including the distribution of features, relationships between variables, and the identification of critical features for predicting loan defaults. The feature importance analysis highlighted the most influential predictors, while the anomaly detection identified outliers that could distort the model's learning process. By selecting the top 15 features using both the Random Forest model and the Chi-squared test, we ensured that only the most relevant features were retained for training. The insights gathered during this phase will guide the creation of a robust predictive model for loan default prediction.

Key points:

- The dataset exhibited an imbalance in the target variable, highlighting the need for techniques like resampling to address this issue.
- Key features such as were identified as crucial predictors of loan defaults.
- Outliers detected using the Isolation Forest algorithm will be further analyzed to ensure they are appropriately handled in the modeling process.
- The comparison between the Random Forest and Chi-squared test feature selection methods confirmed the robustness of the selected feature set.

XGBoost Model Development and Evaluation for Loan Default Prediction

Objective:

The objective of this section is to develop a robust machine learning model for loan default prediction, using XGBoost to classify loan default status based on historical loan data. This section covers the handling of class imbalance, model training using cross-validation, performance evaluation, and prediction on a test dataset.

1. Class Imbalance Handling with SMOTEENN

To address the issue of class imbalance in the dataset, where the majority of the loans were non-defaults (bad_flag = 1), SMOTEENN (Synthetic Minority Over-sampling Technique and Edited Nearest Neighbors) was applied. SMOTEENN combines SMOTE, which generates synthetic samples for the minority class, and Edited Nearest Neighbors (ENN), which removes noisy samples from both classes. This helped balance the dataset and improved the model's ability to detect the minority class (loan defaults).

2. Stratified K-Fold Cross-Validation for Model Training

Stratified K-Fold cross-validation was used to evaluate the model's performance. This technique ensures that each fold of the data contains a proportional representation of both classes (bad_flag = 1 and bad_flag = 0), improving the fairness of the evaluation, especially when dealing with class imbalance. The dataset was split into 5 folds, and each fold was used as both a training and validation set. This cross-validation process helps in obtaining more generalized performance metrics.

The model was trained using **XGBoost**, which is a powerful gradient boosting algorithm. The model incorporated specific hyper-parameters to handle the class imbalance effectively. In particular, the scale_pos_weight parameter was adjusted to balance the contribution of the minority and majority classes during training.

3. Model Evaluation Metrics

F1 Score: The F1 score was used to evaluate the model's performance, especially its ability to balance precision and recall for the minority class. This metric is crucial when dealing with imbalanced datasets, as it emphasizes both false positives and false negatives.

AUC Score: The Area under the Receiver Operating Characteristic (ROC) Curve (AUC) was calculated to assess the model's ability to distinguish between the two classes. A higher AUC score indicates a better ability to classify loans correctly as defaults or non-defaults.

ROC Curve: The ROC curve was plotted for each fold to show the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity). The AUC for each fold was used to analyze how well the model performed across different subsets of the data.

4. Performance Summary

After the cross-validation process, performance metrics such as the mean F1 score and mean AUC score were calculated to provide an overview of the model's effectiveness. Additionally, the standard deviation of these scores was computed to assess the model's consistency across different folds. These metrics offer insights into how well the model generalizes to unseen data.

5. Test Data Prediction and Thresholding

After training the model, it was used to predict the loan default probability on a separate test dataset. The predicted probabilities were converted into binary predictions (bad_flag = 0 or bad_flag = 1) using a threshold of 0.6. This threshold was chosen to balance the trade-off between false positives and false negatives, ensuring that the predictions were both accurate and useful for decision-making.

6. Class Distribution in Test and Train Data

The class distribution of bad_flag (defaults vs. non-defaults) in both the training data and the predicted test data was examined. This ensured that the model's predictions were not overly biased towards the majority class. The ratio of bad_flag = 1 to bad_flag = 0 was calculated for both the original training data and the predicted test data to evaluate how well the model maintained the class balance.

Conclusion:

The steps outlined in this report describe the process of handling class imbalance, training the model using Stratified K-Fold cross-validation, evaluating the model's performance using F1 and AUC scores, and making predictions on a test dataset. The model demonstrates good generalization and balanced performance, providing valuable insights into loan default prediction.

The approach employed ensures that the model is robust and capable of accurately predicting loan defaults, with a focus on fairness and generalization, especially when dealing with imbalanced classes.

Neural Network Implementation for Loan Default Prediction

Introduction

This report outlines the design, implementation, and evaluation of a neural network model built using PyTorch to classify loan default predictions. The dataset encompasses various features related to loan applications, such as applicant information and loan status, with the target variable [bad_flag] indicating whether a loan will default (0) or not (1). The task aims to predict the likelihood of a loan default using the features provided in the dataset.

Model Design & Architecture

Network Design

The neural network model is structured with a single hidden layer, a commonly used architecture for binary classification tasks. The model allows flexibility in the number of neurons in the hidden layer, facilitating fine-tuning to improve performance based on the dataset's complexity. The design follows object-oriented principles, making the model modular and easy to maintain.

Activation Functions

- **ReLU Activation:** The hidden layer employs the ReLU (Rectified Linear Unit) activation function. ReLU is effective in avoiding issues like vanishing gradients and is known to work well in deep learning models, especially with larger datasets.
- **Sigmoid Activation:** The output layer uses the Sigmoid activation function, ideal for binary classification. It squashes the model's output to a range between 0 and 1, which can be interpreted as the probability of loan default.

Object-Oriented Design

The model is implemented using an object-oriented approach, encapsulating all components into a class structure. This promotes code reusability, scalability, and clear separation of concerns. The core of the design is the forward method, which defines how data flows through the network from input to output. The model consists of:

- **Input Layer:** Accepts the feature vector with a configurable size.

- **Hidden Layer:** A fully connected layer with a configurable number of neurons, followed by a ReLU activation.
- **Dropout:** A dropout layer is applied after the hidden layer to mitigate overfitting by randomly setting some neurons to zero during training.
- **Output Layer:** A fully connected output layer with one neuron, passed through the Sigmoid activation for binary classification.

Model Training

Loss Function

The loss function employed for this binary classification task is Binary Cross-Entropy Loss (BCELoss). This function calculates the discrepancy between predicted probabilities and actual binary labels. BCELoss is well-suited for models predicting probabilities in binary classification scenarios.

Optimization

The Adam optimizer is used to update the model weights. Adam is a robust optimization algorithm that adapts the learning rate for each parameter, making it particularly effective for deep learning tasks and large datasets. It combines the advantages of both momentum and RMSProp, leading to faster convergence.

Training Loop

The training process involves passing the data through the network, calculating the loss, and updating the model weights using back-propagation. The training loop iterates over a predefined number of epochs, with the optimizer adjusting the weights after each pass through the training data.

Epochs and Learning Rate

The model is trained for 50 epochs, with a learning rate of 0.003. These parameters are selected based on empirical results to balance training speed and model performance.

Model Evaluation

Performance Metrics

- **F1 Score:** The F1 score, which is the harmonic mean of precision and recall, is used to evaluate the model's ability to balance false positives and false negatives. This metric is particularly important in imbalanced datasets, where predicting the minority class is crucial.
- **ROC AUC Score:** The Receiver Operating Characteristic (ROC) curve is used to evaluate the model's performance across different threshold values. The area under the curve (AUC) quantifies the ability of the model to distinguish between classes, with higher AUC values indicating better model performance.

Cross-Validation

To ensure that the model generalizes well and is not over-fitting, Stratified K-Fold Cross-Validation is implemented. This technique ensures that each fold has an equal distribution of class labels, providing a more reliable estimate of the model's performance across various data splits. The model is trained and evaluated on 5 different splits of the data.

ROC Curve

The ROC curve is plotted to visually represent the tradeoff between sensitivity (True Positive Rate) and 1-specificity (False Positive Rate) at various decision thresholds. The ROC AUC score quantifies the overall performance of the model, with values closer to 1 indicating better classification capability.

Inference

After training, the model is used to make predictions on unseen test data. The inference process follows these steps:

Data Scaling: The test data is scaled using the same scaler applied during training to maintain consistency.

Thresholding: The model outputs probabilities, which are then converted into binary predictions (0 or 1) based on a threshold value. In this case, the threshold is set to 0.75, which is selected to maximize the precision and recall balance.

The model generates predicted labels (bad_flag = 1 or 0), which are stored and can be further analyzed or used in decision-making processes.

Conclusion

The neural network model effectively classifies loan defaults, achieving robust performance metrics such as the F1 score and ROC AUC score. The architecture, consisting of a hidden layer with ReLU activation, dropout for regularization, and a Sigmoid output layer for binary classification, follows best practices for deep learning model design. Stratified K-Fold Cross-Validation ensures that the model generalizes well, and the use of performance metrics like F1 score and AUC provides a reliable evaluation of model effectiveness.

Neural Network Design and Architecture

1.1 *Objective*

The goal of this project was to develop a neural network model capable of predicting binary outcomes, specifically whether a loan application (or similar data point) would be classified as "bad" or "not bad" based on its features. This binary classification task was to be approached by implementing a neural network with one or more hidden layers, applying an appropriate activation function, and training the model using a binary cross-entropy loss function.

1.2 *Network Architecture*

The neural network architecture was designed to address the binary classification problem effectively:

- *Input Layer:* The number of features in the dataset determined the size of the input layer.
- *Hidden Layers:* The network included two hidden layers, where the number of neurons in each layer was treated as a tunable hyper-parameter. ReLU (Rectified Linear Unit) was chosen as the activation function for these hidden layers to introduce non-linearity and mitigate issues like the vanishing gradient problem.
- *Output Layer:* The output layer consisted of a single neuron with a sigmoid activation function, which maps the output to a probability between 0 and 1, suitable for binary classification tasks.
- *Dropout Layers:* Dropout layers were introduced after each hidden layer to prevent over-fitting, ensuring that the model could generalize well to unseen data.

The architecture was designed to balance performance and model complexity while ensuring robustness through the use of dropout and ReLU activations.

1.3 *Training Strategy*

The training process was guided by the following considerations:

- *Loss Function:* Binary Cross-Entropy (BCELoss) was employed as the loss function, as it is standard for binary classification tasks. This function measures the performance of the classification model by comparing predicted probabilities with the actual labels.
- *Optimizer:* The Adam optimizer was selected for its adaptive learning rates and efficiency in handling sparse gradients, which are common in deep learning models.
- *Epochs & Batch Processing:* The model was trained over 30 epochs, and the optimizer adjusted the weights after every forward pass. The network utilized full-batch processing for simplicity.

1.4 *Hyper-parameter Tuning*

To ensure optimal performance, several hyper-parameters were tuned, including:

- The number of neurons in each hidden layer.
- The learning rate of the Adam optimizer. These hyper-parameters were optimized through manual grid search. The grid search aimed to find the combination of parameters that maximized the model's performance, particularly its AUC score and F1 score.

2. Model Evaluation and Performance Metrics

2.1 *Cross-Validation*

Stratified K-Fold Cross-Validation was implemented to assess the model's generalizability. This technique ensures that the class distribution is similar across each fold, allowing the model to perform well across various subsets of the data. The dataset was split into 5 parts, and each fold was used as a validation set while the remaining folds were used for training.

2.2 *Evaluation Metrics*

The performance of the model was evaluated using the following metrics:

- *F1 Score:* The F1 score is the harmonic mean of precision and recall. It was chosen to balance both false positives and false negatives, especially in the case of imbalanced datasets.

- *AUC* (Area under the Curve): The AUC score provides an aggregate measure of the model's ability to distinguish between the positive and negative classes. A higher AUC indicates a better-performing model.
- *Class Distribution*: The ratio of predicted positive (bad_flag = 1) and negative (bad_flag = 0) classifications was monitored to ensure a balanced prediction distribution.

The model achieved:

- A mean F1 score of 0.7802, indicating strong performance in balancing precision and recall.
- A mean AUC score of 0.7948, confirming that the model was highly effective at distinguishing between the two classes.

2.3 *Inference on Test Dataset*

Once the model was trained and hyper-parameters optimized, it was used to make predictions on a separate test dataset, where the target variable ("bad_flag") was initially missing. The model outputted probabilities, which were converted into binary predictions using a threshold of 0.85.

The test set predictions were evaluated by comparing the predicted class distribution with the original distribution of the target variable in the training set. The predicted ratios of "bad" and "not bad" were consistent with expectations, and the results were added to the test dataset for further analysis.

3. Results and Interpretation

3.1 *Model Inference*

Upon applying the trained model to the test dataset, the following results were observed:

- The predicted class distribution of bad_flag in the test dataset was similar to the training dataset, maintaining a balanced ratio of predicted 1's and 0's.
- The predicted classes were added to the test dataset, allowing for further exploration and analysis of model behavior.

3.2 *Comparison with Training Dataset*

The ratio of predicted "bad" instances (bad_flag = 1) in both the training and test datasets was analyzed. The distribution in the test dataset was found to be consistent with the training set, demonstrating that the model generalized well across the two datasets.

4. Conclusion

The neural network model successfully addressed the binary classification task with a high level of accuracy. The architecture, consisting of two hidden layers with ReLU activation and a sigmoid output, provided a robust solution to predict the "bad_flag" category. The use of cross-validation ensured that the model was not over-fitting to a specific data split, and the hyper-parameter tuning process helped optimize the model's performance.

The model achieved excellent results, with an F1 score of 0.7802 and an AUC of 0.7948, indicating that it performed well across various evaluation metrics. The test dataset predictions were consistent with expectations, confirming the model's ability to generalize to unseen data.