LOAN STATUS CLASSIFICATION

Loan status classification is a task within the domain of machine learning and predictive analytics that involves developing models to predict the status of loan applications. The goal is to automatically categorize whether a loan application will be approved or denied based on various input features. These features typically include financial indicators such as income, credit score, loan amount, and other relevant factors.

By leveraging historical data, machine learning models can learn patterns and relationships that help in assessing the creditworthiness of applicants, ultimately contributing to more efficient and accurate loan approval processes.

This is a Binary Classification problem in which we need to predict our Target label which is “Loan Status”.

Loan status can have two values: Yes or NO.
Yes: if the loan is approved
NO: if the loan is not approved

So, using the training dataset we will train our model and try to predict our target column that is “Loan Status” on the test dataset.

Dataset

The dataset comprises 100,514 entries and 19 columns, offering valuable insights into loan-related information

Loan ID: Unique identifier for each loan.

Customer ID: Unique identifier for each customer.

Loan Status: The status of the loan (target variable). It indicates whether the loan was approved or denied.

Current Loan Amount: The current amount of the loan.

Term: The term (duration) of the loan (e.g., short-term, long-term).

Credit Score: The credit score of the customer.

Annual Income: The annual income of the customer.

Years in Current Job: The number of years the customer has been in their current job.

Home Ownership: The type of home ownership (e.g., own, mortgage, rent).

Purpose: The purpose of the loan (e.g., debt consolidation, home improvement).

Monthly Debt: The monthly debt payments of the customer.

Years of Credit History: The number of years of credit history.

Months since Last Delinquent: The number of months since the customer's last delinquent payment.

Number of Open Accounts: The number of open credit accounts.

Number of Credit Problems: The number of credit problems the customer has.

Current Credit Balance: The current balance on the customer's credit.

Maximum Open Credit: The maximum credit limit for all credit sources.

Bankruptcies: The number of bankruptcies the customer has had.

Tax Liens: The number of tax liens on the customer's records.

DATA CLEANING

In the process of preparing the dataset for analysis, several cleaning steps were implemented to ensure the integrity and quality of the data. Below is a summary of the key data cleaning steps:

Handling Null Values:

- Checked for the presence of null values across all columns.
- Identified columns with null values, particularly the "Months since Last Delinquent" column.
- Dropped the "Months since Last Delinquent" column due to a significant number of missing values.

Removing Unnecessary Columns:

- Dropped columns that were considered unnecessary for the analysis, namely "Loan ID" and "Customer ID."

Handling Duplicates:

- Checked for and removed duplicate entries to ensure data consistency.

DATA PREPROCESSING

The data preprocessing phase played a crucial role in enhancing the dataset's quality and ensuring its compatibility with machine learning models. Here is an overview of the key data preprocessing steps:

Conversion of 'Years in Current Job': Extracted numerical values from the 'Years in Current Job' column, converting it to a numerical type. This facilitates better numerical representation for analysis.

Handling Missing Values: Utilized the KNNImputer to fill missing values in specific columns ('Credit Score,' 'Annual Income,' 'Years in Current Job') based on neighboring values. This approach enhances data completeness.

Label Encoding for Categorical Variables: Applied label encoding to convert categorical values into numerical representations. This is crucial for models that require numerical inputs.

Scaling with Robust Scaler: Utilized the Robust Scaler to normalize numerical features such as 'Current Loan Amount,' 'Annual Income,' 'Monthly Debt,' 'Current Credit Balance,' and 'Maximum Open Credit.' Scaling ensures that the features contribute uniformly to model training, enhancing its stability.

Data Splitting:

Features (X):

- Extracted features from the dataset by removing the 'Loan Status' column.
- Comprises all columns except the target variable, forming the feature set (X).

Target Variable (Y):

- Isolated the 'Loan Status' column as the target variable (Y).
- Represents the class labels for machine learning model prediction.

EDA

Loan Status Distribution:

- 25.2% of loans are charged off, while 74.8% are fully paid.

Job Tenure:

- A substantial 40% have job tenures of 1-5 years, indicating a younger workforce.
- About 30% exhibit moderate job stability with tenures of 6-10 years.
- Another 31% have more established careers with job tenures of 11-15 years.
- No individuals in the sample have job tenures of 26-30 years.

Loan Purposes:

- Top reasons for loans: home improvement, debt consolidation, and buying a house, accounting for over 60%.
- Indicates a preference for short-term loans.

Home Ownership:

- 48.7% of individuals in the dataset have their homes as mortgages.

Credit Score and Loan Repayment:

- Individuals with high credit scores tend to fully pay their loans.

Outliers:

- Outliers present in the data, requiring attention during preprocessing.

Model Training and Evaluation

Models used:

- Logistic Regression
- SVM
- K-Nearest Neighbours
- Random forest
- AdaBoost

Logistic Regression

Hyperparameter Tuning:

- Logistic Regression model is initially fitted without specific solver (reaches maximum iteration).
- Grid Search is employed for hyperparameter tuning using cross-validation.
- Parameters tested: 'C' (regularization strength) and 'penalty' (type of regularization).
- Best Parameters found: {'C': 1, 'penalty': 'l1'}.

Model Training with Tuned Parameters:

- A new Logistic Regression model (LR1) is created using the best parameters.
- LR1 is trained on the training dataset.

Model Evaluation:

Accuracy Scores:

- Training Accuracy: 0.796
- Testing Accuracy: 0.799

Classification Report:

- Precision, Recall, and F1-Score for both classes (0 and 1).

Insights from Classification Report:

- The model exhibits high accuracy, especially in correctly identifying instances of class 1.
- Class 0 (loan charge-off) identification shows lower precision and recall.

K-Nearest Neighbours

Initial Model Fitting:

- KNN model is trained with 3 neighbors (n_neighbors=3) on the training dataset.

Results:

- Training Accuracy: 0.852
- Testing Accuracy: 0.748

Identifying Overfitting:

- A significant difference between training and testing accuracies indicates potential overfitting.
- Hyperparameter tuning is necessary to optimize the model.

Hyperparameter Tuning:

- Grid Search is utilized to find the optimal number of neighbors (n_neighbors).
- Values tested: [1, 3, 5, 7, 9, 11, 13].
- Best Hyperparameters: {'n_neighbors': 13}.

Model Training with Tuned Parameters:

- A new KNN model is created using the best hyperparameters (n_neighbors=13).
- The tuned model is trained on the training dataset.

Model Evaluation:

Accuracy Scores:

- Training Accuracy: 0.805
- Testing Accuracy: 0.790

Insights from Classification Report:

- The model performs exceptionally well in identifying class 1 instances.
- Class 0 predictions show lower recall, indicating challenges in correctly identifying instances of class 0.
- Overall accuracy is 79%, with balanced macro and weighted average F1-Scores, considering class imbalances.

SVM

Initial Model Training:

- SVM was trained with C=0.1 and a radial basis function (RBF) kernel.
- Attained a Training Accuracy of 79.8% and Testing Accuracy of 80.2%.

Performance Evaluation:

- The model exhibits a balanced performance, avoiding overfitting.
- Noteworthy success in predicting class 1 (Fully Paid) with high precision, recall, and F1-score.
- Challenges arise in predicting class 0 (Not Fully Paid) with relatively low recall and F1-score, indicating difficulty capturing instances of class 0.

RANDOM FOREST

Initial Model Training:

- The initial Random Forest model was trained without hyperparameter tuning, resulting in perfect Training Accuracy (1.0) but a noticeable gap with Testing Accuracy (80.2%), indicating overfitting.

Hyperparameter Tuning:

- Employed GridSearchCV to identify optimal hyperparameters.
- Best Hyperparameters: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}.

Post-Tuning Model Training:

- After hyperparameter tuning, Training Accuracy reduced to 97.4%, addressing overfitting.
- Testing Accuracy remained similar at 80.3%.

Data Balancing with SMOTE:

Originally imbalanced data distribution: Class 0 (22587 instances), Class 1 (67006 instances).

Utilized SMOTE to balance the dataset: Class 0 (53558 instances), Class 1 (53558 instances).

Performance Evaluation:

- Despite hyperparameter tuning and data balancing, the model struggles with instances of class 0, as reflected in lower recall.
- High precision and recall for class 1 indicate success in predicting instances where the loan status is 1.

CROSS VALIDATION

Support Vector Classifier (SVC):

- Average Accuracy: 79.86%

Logistic Regression:

- Average Accuracy: 79.49%

K-Nearest Neighbors (KNN):

- Average Accuracy: 76.22%

Random Forest Classifier:

- Average Accuracy: 79.83%

Overall Observations:

- SVM, Logistic Regression, and Random Forest exhibit comparable and relatively high average accuracy.
- Class imbalance, especially in instances with 'Not Fully Paid' status (class 0), poses challenges for all models.
- Random Forest shows robustness but still struggles with class imbalances, impacting the prediction of 'Not Fully Paid' instances.

Using AdaBoost as:

- Loan dataset have imbalanced classes, AdaBoost can mitigate the impact of class imbalance by assigning higher weights to misclassified instances, helping the model focus on the minority class.
- AdaBoost can boost weak learner's performance
- AdaBoost is less prone to overfitting compared to complex models.

Training Accuracy on Resampled Data:

- AdaBoost achieved a training accuracy of approximately 73.6% on the resampled data. This indicates how well the model has learned from the oversampled (resampled) training data.

Testing Accuracy on Original Data:

- The testing accuracy on the original (unbalanced) test data is around 72.9%. This metric reflects how well the model generalizes to unseen data.
- AdaBoost successfully handles class imbalance, maintaining a balanced trade-off between precision and recall on resampled training data.

Conclusion

Logistic Regression:

- Achieved moderate accuracy with balanced performance on both classes.
- Precision and recall for both classes are satisfactory.

K-Nearest Neighbours (KNN):

- Demonstrated high accuracy but showed signs of overfitting.
- Optimal hyperparameter tuning mitigated overfitting but with a slight decrease in accuracy.

Support Vector Machine (SVM):

- Balanced performance with similar training and testing accuracy.
- Exceptional precision, recall, and F1-score for Class 1, but challenges with Class 0.

Random Forest:

- Initially overfitted, with perfect training accuracy.
- Hyperparameter tuning improved the model, achieving high accuracy.
- Struggled with precision and recall for Class 0.

AdaBoost:

- Effective in handling class imbalance but with moderate overall accuracy.
- Balanced precision and recall on the resampled training data.