AIM 511 : Machine Learning Project Report: Loan Default Prediction

AI Avengers
Madhav Sanjay Patil (IMT2022109)
Pranav Kulkarni (IMT2022053)

December 12, 2024

Link To GitHub Repository : `https://github.com/py-xis/AIM-511-Machine-Learning-Project`

1 Problem Statement

The objective of this project is to develop a machine learning model to predict loan defaults, aiding financial institutions in assessing borrower risk. The project employs a dataset containing various financial and demographic features of customers.

2 Dataset Overview

2.1 Source

The dataset is split into training and testing sets, containing features like loan amount, income, credit history, and other demographic details.

2.2 Dataset Details

The dataset consists of `255,347 rows` and `18 columns`. Below is a description of the key features:

- **LoanID:** String, a unique identifier for each loan.
- **Age:** Integer, the age of the borrower.
- **Income:** Integer, the annual income of the borrower.
- **LoanAmount:** Integer, the amount of money being borrowed.
- **CreditScore:** Integer, the credit score of the borrower, indicating their creditworthiness.
- **MonthsEmployed:** Integer, the number of months the borrower has been employed.
- **NumCreditLines:** Integer, the number of credit lines the borrower has open.
- **InterestRate:** Float, the interest rate for the loan.
- **LoanTerm:** Integer, the term length of the loan in months.
- **DTIRatio:** Float, the Debt-to-Income ratio, indicating the borrower's debt compared to their income.
- **Education:** String, the highest level of education attained by the borrower (e.g., PhD, Master's, Bachelor's, High School).
- **EmploymentType:** String, the type of employment status of the borrower (e.g., Full-time, Part-time, Self-employed, Unemployed).
- **MaritalStatus:** String, the marital status of the borrower (e.g., Single, Married, Divorced).

- **HasMortgage:** String, whether the borrower has a mortgage (Yes or No).
- **HasDependents:** String, whether the borrower has dependents (Yes or No).
- **LoanPurpose:** String, the purpose of the loan (e.g., Home, Auto, Education, Business, Other).
- **HasCoSigner:** String, whether the loan has a co-signer (Yes or No).
- **Default:** Integer, the binary target variable indicating whether the loan defaulted (1) or not (0).

3 EDA

3.1 Issues with the Dataset

3.1.1 Dirty Data Issues

- **Missing Values:** The dataset does not contain missing values, which minimizes the need for imputation or related techniques.
- **Outliers:** Significant outliers are present in variables like `LoanAmount`, `Income`, and `InterestRate`, which can distort model performance.
- **Skewness in Numerical Features:** Variables such as `Income`, `LoanAmount`, and `DTIRatio` exhibit slightly right-skewed distributions.
- **Potential Redundancy:** Some categorical variables, such as `HasMortgage` and `HasCoSigner`, are imbalanced

3.1.2 Tidiness Issues

- **Column Names:** Certain column names, like `DTIRatio` and `NumCreditLines`, lack clarity. Renaming them to more descriptive terms (e.g., `DebtToIncomeRatio`) would enhance interpretability.
- **Feature Engineering Opportunities:** New columns such as `Income_to_LoanAmount` and `CreditScore_to_Age` can strengthen predictive power by capturing relationships between existing variables.
- **Unscaled Numerical Data:** Numerical features, such as `Income` and `LoanAmount`, vary greatly in scale. Normalization is essential for models like KNN that rely on distance metrics.
- **Imbalance in Target Variable:** While the target variable (`Default`) might appear balanced, further checks for class imbalance are needed to ensure model fairness.

3.2 Class Imbalance Consideration

Balancing the dataset depends on the evaluation metric and problem context. If accuracy is your metric, balancing is not strictly necessary.

4 Data Cleaning and Data Preprocessing

4.1 Encoding and Column Transformation

- Outlier Removal from the Numerical Columns using the IQR Formula: Outliers in numerical columns are detected using the Interquartile Range (IQR). The IQR is calculated as the difference between the third quartile ($Q3$) and the first quartile ($Q1$):

$$IQR = Q3 - Q1$$

The lower and upper bounds for detecting outliers are defined as:

$$\text{Lower Bound} = Q1 - 1.5 \times IQR$$

$$\text{Upper Bound} = Q3 + 1.5 \times IQR$$

Any data points outside this range, i.e., below the lower bound or above the upper bound, are considered outliers and are removed accordingly in the data cleaning process. This method ensures that extreme values do not disproportionately affect the analysis or model performance.

- Applied One-Hot Encoding to all categorical variables except `Education` column.
- Ordinally-encoded column `Education`.
- The columns `HasDependents`, `HasCosigner`, and `HasMortgage` which were initially having the values Yes or No were changed to boolean
- Numerical Columns are scaled using StandardScaler

4.2 Feature Engineering

- We created a new feature `Income_to_LoanAmount`, calculated as the ratio of `Income` to `LoanAmount`. This feature provides insight into the borrower's ability to repay the loan relative to their income. A higher ratio indicates better financial stability and a lower risk of default.
- The feature `CreditScore_to_Age` was engineered by dividing `CreditScore` by `Age`. This metric helps assess the borrower's creditworthiness over their lifespan, giving a perspective on how well they have maintained a good credit score relative to their age.
- We generated the feature `LoanAmount_InterestRate` by multiplying `LoanAmount` with `InterestRate`. This feature represents the overall interest burden of the loan. It is useful for evaluating how the cost of borrowing varies with the amount of the loan and helps in assessing loan affordability.
- The feature `MonthsEmployed_to_Age` was created by dividing `MonthsEmployed` by `Age`. This feature shows the proportion of a borrower's life they have spent working. It is indicative of job stability and work experience, which can be factors influencing loan repayment likelihood.
- We also calculated `DTIRatio_LoanAmount` as the product of `DTIRatio` and `LoanAmount`. This feature quantifies the borrower's overall debt obligations relative to the loan amount they are applying for. A higher value could indicate that the borrower is taking on a considerable amount of debt relative to their income, signaling a higher risk of default.

5 Checkpoint - 1

Modeling and Experiments

Model Selection Reasoning

Model selection depends on the nature of the dataset and the problem at hand. Below, we discuss the reasoning for excluding certain models and opting for others based on the characteristics of the dataset.

Models Not Used

1. **Multinomial Naive Bayes (MultinomialNB)**:
   This model is suitable for datasets where features represent counts or frequencies, such as text data or document classification problems. Since the current dataset consists of numerical and categorical features related to loan applications rather than count data, MultinomialNB is not appropriate.
2. **Bernoulli Naive Bayes (BernoulliNB)**:
   BernoulliNB works well for binary features (e.g., presence or absence of a word in text classification). While the dataset does contain some binary categorical variables, the majority of the features are numerical. Thus, BernoulliNB is not a good fit for this dataset.
3. **Naive Bayes (General)**:
   The Naive Bayes family of models assumes feature independence, which is rarely the case in real-world datasets. In this dataset, variables such as income, loan amount, and credit score are likely interdependent, making the Naive Bayes assumption unsuitable.
4. **Gaussian Naive Bayes (GaussianNB)**:
   GaussianNB is designed for datasets where features follow a Gaussian (normal) distribution. While some numerical variables in the dataset might approximate normality, others (e.g., LoanAmount, DTIRatio) exhibit skewness and non-linear relationships, reducing the effectiveness of this model.

5. **Logistic Regression**:
   Logistic regression assumes linear separability of classes, which does not hold for this dataset given the complex and potentially non-linear interactions between variables. Moreover, logistic regression can struggle with datasets having a large number of features, especially if they interact in non-linear ways, as is the case here.

Models Used

1. **Random Forest**:
   Random Forest is robust to non-linear relationships and can handle both numerical and categorical features effectively. Its ability to capture feature interactions and importance makes it well-suited for this dataset.

2. **AdaBoost**:
   AdaBoost builds an ensemble of weak learners (typically decision trees) and iteratively focuses on correcting errors, making it effective for complex datasets. It can handle both categorical and numerical features and is resilient to overfitting if tuned properly.

3. **XGBoost**:
   XGBoost is a powerful gradient boosting algorithm that excels in handling large datasets with complex feature interactions. Its regularization mechanisms reduce the risk of overfitting, and it works well with non-linear data.

4. **Decision Tree**:
   Decision trees are intuitive and can handle non-linear relationships between variables. While they can overfit on their own, they serve as excellent base models for ensemble methods like Random Forest and AdaBoost.

5. **K-Nearest Neighbors (KNN)**:
   KNN is a non-parametric method that makes no assumptions about the underlying data distribution. It is effective for capturing local patterns in data and works well for datasets with mixed types of features when scaled appropriately.

Summary of Model Selection

The models used (Random Forest, AdaBoost, XGBoost, Decision Tree, and KNN) were chosen for their ability to handle non-linear relationships, mixed feature types, and complex interactions in the data. In contrast, models like Naive Bayes and Logistic Regression were excluded due to their assumptions and limitations, which do not align well with the characteristics of this dataset.

Models Tested

The following classifiers were evaluated using a Stratified K-Fold cross-validation approach with 10 splits to ensure a balanced representation of classes in each fold:

- **DecisionTreeClassifier**: A simple yet interpretable model using a tree-based approach for classification.
- **KNeighborsClassifier**: A non-parametric method that predicts the class based on the majority class among the k nearest neighbors.
- **RandomForestClassifier**: An ensemble of decision trees that improves model accuracy and reduces overfitting through bagging.
- **AdaBoostClassifier**: An ensemble method that combines multiple weak classifiers to form a strong classifier using the Adaptive Boosting algorithm.
- **XGBClassifier**: A highly efficient and flexible gradient boosting framework that optimizes performance.

Model Evaluation

The performance of each classifier was measured using accuracy as the evaluation metric. Cross-validation was performed with 10 folds, and the mean accuracy was recorded for each model. The results are summarized in Table 1.

Table 1: Model Accuracy Results

Model	**Mean Accuracy**
RandomForestClassifier	0.8853
AdaBoostClassifier	0.8852
XGBClassifier	0.8845
KNeighborsClassifier	0.8735
DecisionTreeClassifier	0.8018

Analysis of Results

From the results, it is evident that the **RandomForestClassifier** and **AdaBoostClassifier** achieved the highest mean accuracy scores, both close to 0.885. The **XGBClassifier** followed closely behind with a score of 0.884. The **KNeighborsClassifier** performed slightly worse with an accuracy of 0.873, and the **DecisionTreeClassifier** had the lowest performance, with a mean accuracy of 0.8018.

These results indicate that ensemble methods like Random Forest and AdaBoost are highly effective for this dataset, likely due to their ability to combine multiple learners to reduce variance and increase robustness. Conversely, the Decision Tree model's relatively lower performance highlights the limitations of using a single tree-based model without ensembling.

5.1 Model Fine-Tuning

We focused on fine-tuning the three top-performing models from the initial round of evaluations: **RandomForestClassifier**, **AdaBoostClassifier**, and **XGBClassifier**. We also incorporated **LightGBM** into the experiments due to its reputation for efficient and accurate performance in classification tasks.

AdaBoost Fine-Tuning

The **AdaBoostClassifier** was fine-tuned by adjusting hyperparameters such as `n_estimators` and `learning_rate`. We evaluated models using a Stratified K-Fold cross-validation approach with 10 splits to ensure that each fold had a similar distribution of the target variable. The goal was to balance the bias-variance trade-off while optimizing performance. The highest accuracy was achieved with `learning_rate` values of 0.9 and 0.99, yielding an accuracy of 0.8861.

Random Forest Fine-Tuning

The **RandomForestClassifier** was fine-tuned by exploring hyperparameters like `n_estimators` and `ccp_alpha`. We experimented with values of `ccp_alpha` ranging from 1e-6 to 10 to control the complexity of the model and mitigate overfitting. Again, we used Stratified K-Fold cross-validation to maintain class balance across the training and validation sets. The best accuracy obtained was 0.8854, showing strong performance but not significantly better than AdaBoost.

XGBoost Fine-Tuning - Approach 1

In the first approach, we used the same preprocessing steps as in the earlier stages of modeling. We maintained the existing encoding strategy, where only specific categorical features were encoded, and the numerical features were used as they were. We focused on hyperparameter optimization using `RandomizedSearchCV` to identify the best model configuration, ensuring efficient exploration of the hyperparameter space.

XGBoost Fine-Tuning - Approach 2

For the second approach, we revised the preprocessing steps to enhance model performance. Specifically, we applied one-hot encoding to all categorical columns, including the `Education` column, which had previously been ordinally encoded. Additionally, we used `StandardScaler` to standardize all numerical columns, ensuring that the features were appropriately scaled for model training. Hyperparameter tuning was conducted using `RandomizedSearchCV`, similar to the first approach, but with the revised preprocessing to improve the robustness and efficiency of the model.

Rationale for Choosing XGBoost

We selected **XGBoost** as the final model for several reasons:

- **Consistent High Accuracy**: XGBoost consistently performed well across different hyperparameter configurations, demonstrating robustness and stability.
- **Efficiency and Scalability**: XGBoost's optimized gradient boosting framework allows for efficient training and inference, which is beneficial for large datasets.
- **Advanced Regularization**: The model's built-in regularization mechanisms (`gamma` and `min_child_weight`) help to prevent overfitting, making it a strong choice for our problem.

Use of Stratified K-Fold Cross-Validation

Throughout the model evaluation and fine-tuning process, we utilized **Stratified K-Fold Cross-Validation** to maintain the distribution of the target variable across training and validation sets. This approach is crucial for imbalanced datasets, as it ensures that each fold has a similar proportion of both classes, reducing the risk of biased model performance. Although this was implemented indirectly through the `cross_validate` and `GridSearchCV` functions, it played a significant role in providing reliable and consistent evaluation metrics.

Final Model Selection

XGBoost-2 was selected as the final model due to its superior performance in handling imbalanced data and scalability.

Conclusion and Insights

6 Checkpoint - 2

Models Used and Hyperparameter Tuning

Multi-Layer Perceptron (MLP) Classifier

Model: The Multi-Layer Perceptron (MLP) Classifier is a feedforward artificial neural network that maps input features to output classes through one or more hidden layers. It supports a variety of activation functions and optimization algorithms for training.

Hyperparameter Tuning: The hyperparameters were optimized using **RandomizedSearchCV** with a **Stratified K-Fold** cross-validation strategy. The key hyperparameters tuned include:

- Hidden Layer Sizes: Number and structure of neurons in hidden layers (e.g., (50,), (100,)).
- Activation Functions: Relu, Tanh, Logistic.
- Solvers: Optimization algorithms like Adam and SGD.
- Regularization: Alpha values for L2 regularization.
- Learning Rate: Constant or adaptive learning schedules.

Best Parameters: The best configuration for the MLP Classifier was found to be:

- **Solver :** Adam
- **Hidden Layer Sizes:** (100, 50)
- **Activation Function:** Logistic
- **Alpha:** 0.001
- **Random State:** 42
- **Learning Rate:** Constant

Accuracy: The accuracy achieved through the Multi-Layer Perceptron Classifier is 0.88825.

Support Vector Classifier (SVC)

Model: The Support Vector Classifier (SVC) is a supervised learning algorithm that finds a hyperplane in a high-dimensional space to separate data points into different classes. It supports various kernels to handle both linear and nonlinear data.

Hyperparameter Tuning: The hyperparameters were optimized using **RandomizedSearchCV** with a **Stratified K-Fold** cross-validation strategy. The key hyperparameters tuned include:

- Regularization Parameter (**C**): Controls the trade-off between maximizing margin and minimizing classification error.
- Kernel: Specifies the type of hyperplane (e.g., linear, RBF, polynomial) used for classification.
- Gamma: Kernel coefficient for RBF and polynomial kernels, determining the influence of individual data points.

Best Parameters: The best configuration for the SVC was found to be:

- **Regularization Parameter (C):** 10
- **Kernel:** RBF
- **Gamma:** Scale
- **Random State:** 42

Accuracy: The accuracy achieved through the Support Vector Classifier is 0.88758.

Logistic Regression

Model: Logistic Regression is a statistical model that predicts the probability of a binary outcome. It uses the logistic function to map input features to class probabilities and supports various regularization techniques.

Hyperparameter Tuning: The hyperparameters were optimized using **RandomizedSearchCV** with a **Stratified K-Fold** cross-validation strategy. The key hyperparameters tuned include:

- Regularization Strength (**C**): Controls the level of regularization (e.g., 0.01, 0.1, 1, 10, 100).
- Penalty: Type of regularization (L1, L2, ElasticNet).
- Solver: Optimization algorithms such as lbfgs, saga, and liblinear.
- Max Iterations: Maximum number of iterations for convergence.

Best Parameters: The best configuration for Logistic Regression was found to be:

- **Regularization Strength (C):** 0.1
- **Penalty:** L2
- **Solver:** liblinear
- **Max Iterations:** 500
- **Random State:** 42

Accuracy: The accuracy achieved through Logistic Regression is 0.88586.

Conclusion

This project successfully developed a machine learning model for predicting loan defaults using a diverse set of financial and demographic features. After rigorous exploratory data analysis and feature engineering, ensemble-based methods such as Random Forest, AdaBoost, and XGBoost emerged as top-performing models, with XGBoost selected as the final model for deployment. This choice was guided by its superior accuracy, efficiency, and robustness against overfitting.

The results highlighted the effectiveness of incorporating domain-driven engineered features, such as `Income_to_LoanAmount` and `CreditScore_to_Age`, to enhance model performance. Additionally, addressing dataset imbalances using Stratified K-Fold Cross-Validation ensured fair evaluation and reliable model performance metrics.

Key Insights

- **Credit Risk Indicators:** Features like `CreditScore`, `DTIRatio`, and `Income_to_LoanAmount` strongly influence loan default predictions. Borrowers with lower credit scores, higher debt burdens, or insufficient income relative to their loan amounts are at greater risk of default.
- **Class Imbalance:** The dataset showed a significant class imbalance, with only 11.63% of loans defaulting. This imbalance necessitated careful model evaluation using Stratified K-Fold Cross-Validation to ensure unbiased performance metrics. Despite the imbalance, the accuracy observed for the minority class (defaults) was reasonable, with an F1-score of [insert value] and a recall of [insert value]. This indicates that the chosen models, particularly Random Forest and XGBoost, were robust in distinguishing between default and non-default cases without requiring additional balancing techniques such as SMOTE. These results highlight the capability of ensemble methods to handle imbalanced datasets effectively.
- **Feature Engineering Value:** Newly engineered features provided meaningful insights into borrower financial behavior and repayment capacity, significantly improving model predictive power.
- **Model Performance:** Ensemble models, particularly XGBoost, demonstrated superior performance due to their ability to capture non-linear interactions and effectively handle both numerical and categorical data.
- **Future Prospects:** Incorporating additional behavioral or transactional features, such as historical loan repayment patterns, could further enhance predictive accuracy. Testing advanced ensemble techniques like Stacking or incorporating neural network architectures for feature learning represents potential directions for improvement.