Enhancing Loan Approval Precision: A Machine Learning Approach to Identify and Mitigate Default Risks in Financial Institutions

By Group: CS9

Aditya Gupta 210101009 Swagat Sathwara 210101102 Gautam Juneja 210101041 Shivam Gupta 210101114

Abstract

This study explores loan approval prediction using machine learning on a Kaggle dataset featuring parameters like income, age, and profession. Preprocessing methods including encoding and dataset balancing were applied to address data imbalance. Logistic Regression, Naive Bayes, and Random Forest algorithms were tested, showing varying performance levels. Challenges like data imbalance and feature engineering were identified, suggesting strategies for improvement such as model fine-tuning and exploring ensemble methods. Despite challenges, the study underscores machine learning's potential in loan approval prediction, hinting at avenues for further optimization to enhance predictive accuracy and robustness.

1. Introduction

1.1. Problem defintion

Our problem statement is to predict loan approval based on various parameters like income, age, experience etc, using machine learning techniques which involves creating a model that can effectively predict whether a loan application should be approved or denied based on various factors. This task is crucial for financial institutions to automate and streamline their loan approval processes, reducing manual effort and minimising the risk of default.

1.2. Motivation

The motivation behind solving the problem of loan approval prediction using machine learning techniques lies in the pursuit of efficiency, fairness, and risk mitigation within the lending process. By automating and streamlining loan approval procedures, financial institutions can reduce manual effort, improve risk assessment accuracy, and enhance customer experiences. Additionally, employing machine learning models fosters fairness and transparency by minimising biases in decision-making, while also aiding in regulatory compliance and ultimately facilitating business growth through increased operational efficiency and reduced costs.

1.3. Challenges

Several challenges exist in the domain of loan approval prediction using machine learning techniques. Firstly, ensuring data quality and adequacy is paramount; incomplete or biased datasets can lead to inaccurate model predictions and potential discrimination. Secondly, addressing class imbalance—where approved loans significantly outnumber denied ones—poses a challenge, as it can skew model performance metrics.

1.4. Existing Methods

Several existing methods and techniques are commonly employed in predicting loan approval using machine learning. These include:

1.4.1. Logistic Regression

Advantages:
i. Simple and interpretable model.
ii. Fast training and prediction times, suitable for large datasets.
iii. Outputs probabilities, allowing for flexible threshold tuning.
Disadvantages:
i. Limited ability to capture complex relationships in data.
ii. Assumes linear decision boundaries, may underperform with nonlinear data.
iii. Sensitive to outliers and multicollinearity.

1.4.2. Naive Bayes

Advantages:
i. Simple and efficient, especially for text classification tasks.
ii. Performs well with high-dimensional data and small training sets.
iii. Handles missing values gracefully.
Disadvantages:
i. Strong independence assumption between features, which

may not hold in real-world data.
ii. Limited expressiveness, may not capture complex relationships.
iii. Prone to the ”zero-frequency” problem with unseen feature-label combinations.

1.4.3. Random Forest

Advantages:
i. Highly accurate and robust, less prone to overfitting.
ii. Handles high-dimensional data well and can handle both numerical and categorical features.
iii. Provides feature importance scores for interpretability.
Disadvantages:
i. Computationally expensive, especially with large datasets and many trees.
ii. May be difficult to interpret compared to simpler models like logistic regression.
iii. Not suitable for real-time applications due to longer training times.

1.4.4. Support Vector Machine (SVM)

Advantages:
i. Effective in high-dimensional spaces, suitable for complex data.
ii. Versatile, as it allows the use of different kernel functions to handle nonlinear data.
iii. Robust to overfitting, especially with appropriate regularization.
Disadvantages:
i. Memory-intensive, especially with large datasets.
ii. Can be sensitive to the choice of kernel and its parameters.
iii. Interpretability can be challenging, especially with nonlinear kernels.

1.4.5. Gradient Boosting Machines (GBM)

Advantages:
i. High predictive accuracy, often outperforming other algorithms.
ii. Handles different types of data and can capture complex relationships.
iii. Robust to outliers and noisy data.
Disadvantages:
i. Prone to overfitting, especially with complex models and insufficient regularization.
ii. Computationally expensive and may require tuning of several hyperparameters.
iii. Can be challenging to interpret due to the ensemble nature of the model.

For this project we chose this 3 models:

i. Logistic Regression
ii. Random Forest
iii. Naive Bayes

2. Dataset Explanation

2.1. Features

90 percent of the people are single and rest 10 percent is married. From this we figure out that data is imbalanced with respect to this feature with most of the people belonging to one class.
There is equal distribution of people among various age groups. Also the ratio of married to single people in each age group is similar.
We can see that we have people from various income groups and more or less they are equally distributed.
The frequency of engineering profession being the least with 1.61% and physician profession being the maximum with 2.36% percent. The frequency of all other professions lies between 1.61% to 2.36%.

2.2. Preprocessing

2.2.1. Logistic Regression

a. Encoding Techniques:
i. Utilized Ordinal Encoding initially, but found it unsuitable for non-ordinal categorical features like ”STATE” and ”CITY”.
ii. Adopted One-Hot Encoding, which proved effective in transforming categorical variables into a binary format, enhancing model performance.
b. Data Preparation:
i. Created an additional table through One-Hot Encoding, ensuring each category was represented as a binary column.
ii. Merged the encoded table with the original dataset after removing columns designated for encoding. Diligently removed any duplicated columns during the concatenation process to maintain data integrity.
c. Standardization:
i. Applied standardization to the dataset to ensure consistent scales across all features.
ii. Standardization mitigated potential biases arising from varying measurement units, enhancing model robustness.
d. Balanced Dataset:
i. Addressed data distribution concerns by balancing the dataset to ensure equal representation of both target classes.
ii. Equal representation of classes minimized bias and improved the model’s ability to generalize across diverse scenarios.

2.2.2. Naïve Bayes:

a. Data Cleanliness Check:
A check was performed to ensure that the dataset is clean and does not contain any missing or duplicate values. This

step is essential for ensuring the integrity of the data.

b. Target Variable Distribution:
The distribution of the target variable, 'Risk_flag', was analyzed to understand its balance and imbalance. This analysis helps in determining the appropriate evaluation metrics for the model.

c. Univariate Analysis of Numerical Features:
The distribution of numerical features was analyzed using graphs. This analysis helps in deciding between using categorical or Gaussian Naive Bayes, based on the distribution of features.

d. Distribution of Categorical Features:
Similarly, the distribution of categorical features was analyzed using graphs to understand their distribution and impact on the target variable.

e. Feature Correlation:
A correlation matrix was computed to identify if any features are heavily dependent on each other. It was found 'CURRENT_JOB_YRS' and 'Experience' are correlated.

f. Data Encoding convert data from in string format to int

g. Conversion of Continuous Features to Categorical Features:
Continuous features such as 'Age', 'Experience', and 'CURRENT_JOB_YRS' were converted to categorical features. For example, 'Age' was divided into 6 categories.

2.2.3. Random Forest

Handling Null Values: Since our dataset does not contain any null values, there's no need for imputation or removal of missing data.

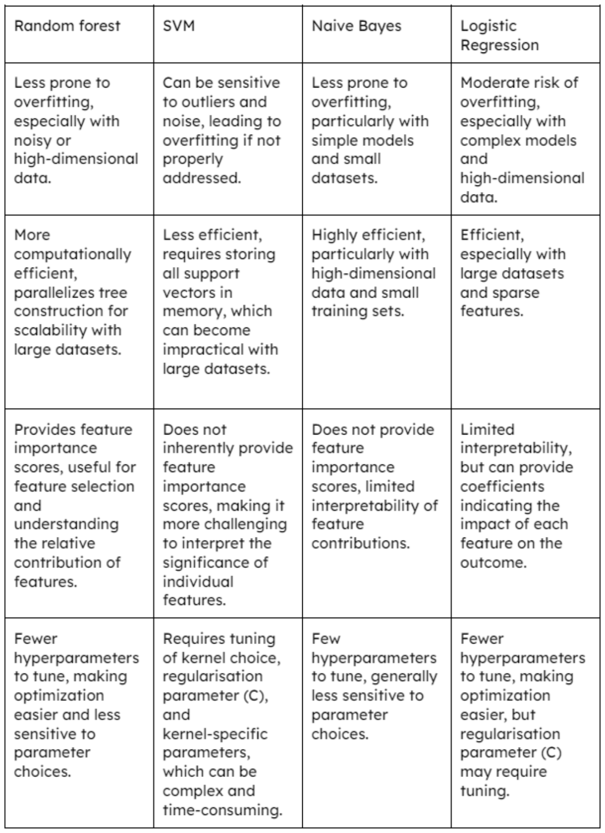
Column Removal :
CITY and STATE Columns:- One-hot encoding these columns would lead to a significant increase in the number of features without adding substantial predictive power. Additionally, ranking cities and states may not provide meaningful insights for our prediction task.

Risk_Flag Column:- This column serves as the target variable for prediction. Therefore, it is excluded from the input data.

Feature Engineering :

Feature engineering transforms Marital Status, House Ownership, and Car Ownership columns to understand financial behavior and risk, utilizing binary-encoded data, profession rankings, and the Risk Flag column isolated as target variable, facilitating effective modeling.

3. Methods:



Random forest	SVM	Naive Bayes	Logistic Regression
Less prone to overfitting, especially with noisy or high-dimensional data.	Can be sensitive to outliers and noise, leading to overfitting if not properly addressed.	Less prone to overfitting, particularly with simple models and small datasets.	Moderate risk of overfitting, especially with complex models and high-dimensional data.
More computationally efficient, parallelizes tree construction for scalability with large datasets.	Less efficient, requires storing all support vectors in memory, which can become impractical with large datasets.	Highly efficient, particularly with high-dimensional data and small training sets.	Efficient, especially with large datasets and sparse features.
Provides feature importance scores, useful for feature selection and understanding the relative contribution of features.	Does not inherently provide feature importance scores, making it more challenging to interpret the significance of individual features.	Does not provide feature importance scores, limited interpretability of feature contributions.	Limited interpretability, but can provide coefficients indicating the impact of each feature on the outcome.
Fewer hyperparameters to tune, making optimization easier and less sensitive to parameter choices.	Requires tuning of kernel choice, regularisation parameter (C), and kernel-specific parameters, which can be complex and time-consuming.	Few hyperparameters to tune, generally less sensitive to parameter choices.	Fewer hyperparameters to tune, making optimization easier, but regularisation parameter (C) may require tuning.

4. Implementation

4.1. Logistic Regression

Logistic regression class: Implements logistic regression model training and prediction functionalities.

4.1.1. Methods included:

- **fit:** Trains the logistic regression model using gradient descent or other optimization algorithms.
- **predict:** Makes predictions using the trained logistic regression model.
- **_sigmoid:** Computes the sigmoid function used to convert raw model outputs to probabilities.
- **_initialize_weights:** Initializes the model's weights and bias terms.
- **compute_loss:** Computes the logistic loss function to optimize during training.
- **_update_weights:** Updates model weights and bias terms based on gradients during optimization.

4.1.2. Challenge:

Multicollinearity among features can destabilize coefficient estimates and reduce model interpretability.

- **Solution:** Implemented feature scaling and regularization to tackle multicollinearity. Feature scaling standardizes feature values, while L2 regularization penalizes large coefficients, encouraging more judicious feature selection.
- **Benefits:** Mitigated multicollinearity enhances model stability and interpretability. Feature scaling ensures all features contribute equally, while regularization controls complexity and prevents overfitting.
- **Performance Improvement:** Enhanced model robustness and generalization lead to improved accuracy and stability. Additionally, better interpretability aids informed decision-making based on predictions.

4.1.3. L2 Regularization

- In the model training phase, a logistic regression model with L2 regularization was employed to predict the risk flag.
- L2 regularization, also known as ridge regularization, was chosen to prevent overfitting by penalizing large coefficient values.
- The regularization parameter (lambda_reg) controlled the strength of regularization, balancing between fitting the training data well and keeping model coefficients small to generalize better to unseen data.
- L2 regularization was incorporated into both the loss function and gradient calculation during model training, ensuring that the model learned from both the training data and the regularization term.
- The model was trained using gradient descent optimization, with hyperparameters such as learning rate (alpha) and the number of iterations tuned to optimize model convergence and performance.

4.2. Naïve Bayes

In the context of classification, Naive Bayes calculates the probability of a class label given the features using the following formula:

1. $P(y|x_1, x_2, ..., x_n)$ is the posterior probability of class y given features $x_1, x_2, ..., x_n$.

2. $P(y)$ is the prior probability of class y.

3. $P(x_i|y)$ is the likelihood of feature x_i given class y.

4. $P(x_i)$ is the prior probability of feature x_i.

4.2.1. Steps for Implementation:

1. Calculate the prior probabilities $P(y)$ for each class in the training set.
2. Calculate the likelihoods $P(x_i|y)$ for each feature x_i given each class y in the training set.

3. For a new data point with features $x_1, x_2, ..., x_n$, calculate the posterior probability of each class y using the Naive Bayes formula:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y) \times \prod_{i=1}^{n} P(x_i|y)}{P(x_1) \times P(x_2) \times ... \times P(x_n)}$$

4. Assign the data point to the class with the highest posterior probability.

5. Evaluate the performance of the Naive Bayes model using metrics such as accuracy, precision.

4.2.2. Laplace Smoothing:

Purpose: Prevent zero probabilities for unseen feature values by adding a small value (usually 1) to the count of each feature value in each class.
Technique: Adjust probability calculation using a formula that includes a smoothing parameter α to control the degree of smoothing.
Formula:

$$P(X_i = value_j | class_k) = \frac{count(X_i = value_j, class_k) + \alpha}{count(class_k) + \alpha * N}$$

where:

- **count**$(X_i = value_j, class_k)$: Number of times feature i has value j in class k.
- **count**$(class_k)$: Total instances in class k.
- **N**: Total possible feature values.
- α**:** Smoothing parameter (typically 1).

4.2.3. Challenges Faced:

- Implementing Laplace smoothing required careful handling of probabilities and counts to ensure that the smoothing parameter was applied correctly. Additionally, managing the data structures for storing class priors and feature likelihoods for each class and feature required attention to detail.
- One significant challenge was encountered due to the initial imbalance in the data regarding the target variable. Initially, the data had an 87:13 ratio, with the majority class (0) dominating the dataset. As a result, the model tended to predict all instances as the majority class, leading to an artificially high accuracy of 87%.
- To address this issue, the data was balanced to a 50-50 ratio, which caused the accuracy to drop to around 53%. This drop indicated that the original data, while imbalanced, did not provide much useful information for the

model. To improve the model's performance, changes were made to the data to associate high income with low risk (0) and low income with high risk. This transformation helped the model make more meaningful predictions and improved its overall performance.

4.3. Random Forest

4.3.1. Decision Tree Node (Node class)

Represents a node in the decision tree.
- feature_index: Index of the feature used for splitting.
- threshold : Threshold value for the split.
- left and right: Left and right child nodes.
- info_gain : Information gain at this node.
- Value: Value at leaf node (class label).

4.3.2. Decision Tree Classifier

Implements the decision tree building process and related functions. -build_tree: Recursively builds the decision tree.
-get_best_split: Finds the best split based on info_gain.
-split: Splits the dataset based on a feature and threshold.
-information_gain: Computes information gain (can use Gini impurity or entropy).
-calculate_leaf_value: Computes the leaf node value.

4.3.3. Random Forest Classifier

Constructs an ensemble of decision trees (Random Forest) using bootstrap aggregation (bagging).Take number of trees required as user input.Initializes with parameters:
- trees: Number of decision trees in the forest.
- min_samples_split: Minimum number of samples required to split a node.
- max_depth: Maximum depth of each decision tree.

4.3.4. Challenges and Considerations

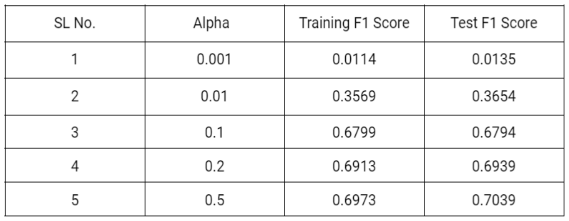
Challenge Faced: Some features may have a large number of unique values, potentially leading to an extensive list of possible split thresholds. Calculating information gain for each possible threshold is computationally expensive and memory-intensive, especially if the number of unique values is very large.
Solution Implemented:I introduced a conditional check (if num_unique_values $>$ 100) to manage features with a high number of unique values. If the number of unique values exceeds a threshold (e.g., 100), I randomly select a subset of unique feature values (possible_thresholds) to use as potential split thresholds. This approach reduces the number of thresholds considered for splitting, thereby improving computational efficiency and reducing memory usage.
Benefits of the Solution: By limiting the number of possible thresholds using random sampling, we efficiently manage features with a large number of unique values without compromising the quality of the split selection. We can also change random sampling so as to keep the mean or variance of feature values selected as close as possible to mean and variance of taking whole dataset.
Improved Performance: The optimized approach reduces computational overhead (time complexity) and memory usage during the tree-building process, contributing to overall performance improvements in the Random Forest implementation.

5. Result

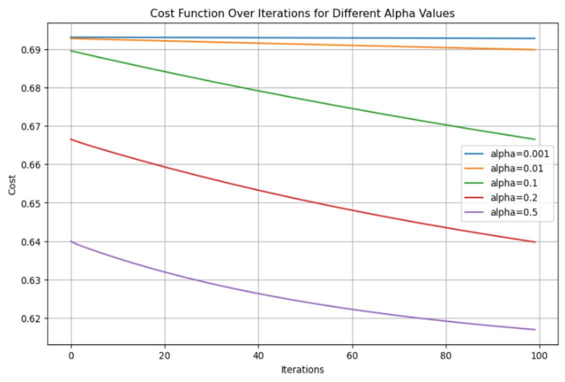
5.1. Logistic Regression

5.1.1. Hyperparameter Tuning and Analysis:

To optimize the model's performance, various combinations of alpha (learning rate) and lambda (regularization parameter) were explored with 100 iterations. The F1 scores were calculated for different alpha and lambda values, and the results are summarized below:



SL No.	Alpha	Training F1 Score	Test F1 Score
1	0.001	0.0114	0.0135
2	0.01	0.3569	0.3654
3	0.1	0.6799	0.6794
4	0.2	0.6913	0.6939
5	0.5	0.6973	0.7039

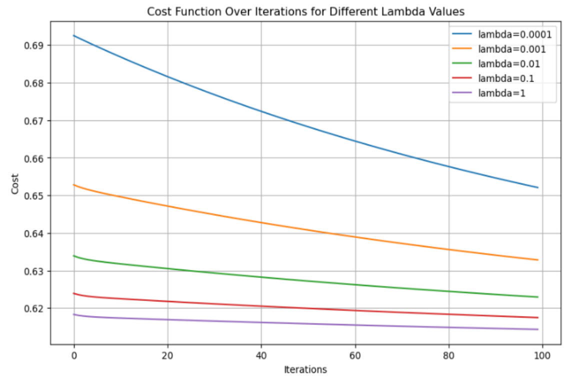


- **Alpha Tuning:**
  The model was trained using different values of alpha to observe its impact on performance. The following alpha values were tested: [0.001, 0.01, 0.1, 0.2, 0.5]. F1 Scores for Different Alpha Values are in above table. Based on the F1 scores, an alpha value of 0.5 was selected as it yielded the highest F1 score on the test set.

- **Lambda Tuning:**
  Similarly, different values of lambda were tested to analyze their effect on model performance. The following lambda values were explored: [0.0001, 0.001, 0.01, 0.1, 1]. F1 Scores for Different Lambda Values:

SL No.	Lambda	Training F1 Score	Test F1 Score
1	0.0001	0.6886	0.6905
2	0.001	0.6937	0.6969
3	0.01	0.6961	0.7024
4	0.1	0.6973	0.7040
5	1	0.6973	0.7044

The F1 scores remained consistent across different lambda values, indicating that the regularization parameter had minimal impact on the model's performance in this case.



5.1.2. Reasoning:

After a certain number of iterations during model training, the value of the cost function may reach a plateau or stabilize. This phenomenon occurs because the gradient descent optimization algorithm converges to a local minimum, where further adjustments to the model parameters do not significantly decrease the cost.
When the cost function stops changing or changes very slowly, it indicates that the model has reached a point where it is making incremental improvements to fit the training data better, but these improvements are not substantial enough to justify further iterations.
At this stage, the model parameters may have settled into a stable configuration, and additional iterations may not lead to significant improvements in performance. Therefore, it is reasonable to observe that the value of the cost remains relatively constant after a certain point during training.

5.1.3. Result Explanation:

Our logistic regression model with L2 regularization achieved promising results, with a training F1 score of 0.7011 and a test F1 score of 0.7074 after 100 iterations. These scores indicate effective risk flag prediction, aligning with the project's objectives of informed decision-making in various sectors.
Observing data trends, certain categorical variables like 'Profession' and 'House_Ownership' demonstrated significant predictive power, offering insights into risk assessment. While our model performed well, there's room for improvement, potentially through feature refinement and deeper data analysis.

5.2. Naïve Bayes

1. Result from Implementation: Running the model on our implementation is time-consuming due to the large number of features and data samples. To mitigate this, we experimented with two scenarios: reducing the number of data samples and reducing the number of features.

(a)Reducing the Number of Data Samples(2000):
Accuracy:0.68 (Train-Test Split: 80:20)

Class	Precision	Recall	F1-Score	Support
0	0.69	0.60	0.64	190
1	0.68	0.75	0.71	210

Note: As the number of data samples is reduced, the accuracy declines slightly.

(b)Reducing the Number of Features(9):
Accuracy:0.679 (Train-Test Split: 80:20, n=34,000)

Class	Precision	Recall	F1-Score	Support
0	0.71	0.61	0.66	3503
1	0.66	0.74	0.70	3478

2. Result from External Library:
I conducted an analysis by running both the categorical and Gaussian Naive Bayes models on our dataset. This comparative study aimed to determine which model aligns better with the characteristics of our data, aiding in the selection of the most suitable algorithm for our classification task.
(a)Categorical Naive Bayes Model:
Accuracy:0.696 (Train-Test Split: 80:20, n=34,000)

Class	Precision	Recall	F1-Score	Support
0	0.72	0.64	0.68	3510
1	0.67	0.75	0.71	3471

(b)Gaussian Naive Bayes Model:
Accuracy:0.536 (Train-Test Split: 80:20, n=34,000)

Class	Precision	Recall	F1-Score	Support
0	0.52	0.89	0.66	3510
1	0.61	0.18	0.28	3471

As we have seen in the EDA that our data is discrete, not continuous thats why Naive Bayes Categorical Model works better than the Gaussian Model.

5.2.1. Categorical vs. Gaussian Naive Bayes:

- The Categorical Naive Bayes model from the external library outperforms the Gaussian Naive Bayes model, as expected due to the discrete nature of the data.
- The Categorical Naive Bayes model achieves an ac-

curacy of 0.696 and an F1 score of 0.710, indicating better performance compared to the Gaussian Naive Bayes model, which has an accuracy of 0.536 and an F1 score of 0.281.

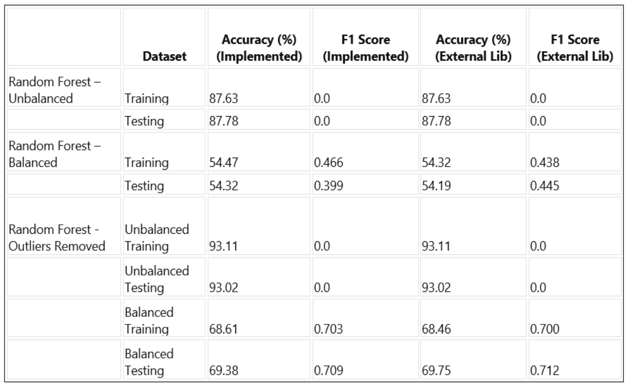
- This highlights the importance of selecting the appropriate Naive Bayes variant based on the data distribution, with categorical Naive Bayes being more suitable for discrete data like the one in this project.

5.2.2. Performance of Our Implementation:

Our implementation of the Naive Bayes model, while computationally intensive due to the large number of features and data samples, yields comparable results to the external library. By reducing the number of data samples, we achieved a slightly lower accuracy of 0.68 but maintained a high F1 score of 0.712, indicating that the model generalizes well even with fewer samples.

Similarly, reducing the number of features to just 9 resulted in an accuracy of 0.679 and an F1 score of 0.698, showing that feature selection can significantly impact model performance and computational efficiency.

5.3. Random Forest



	Dataset	Accuracy (%) (Implemented)	F1 Score (Implemented)	Accuracy (%) (External Lib)	F1 Score (External Lib)
Random Forest – Unbalanced	Training	87.63	0.0	87.63	0.0
	Testing	87.78	0.0	87.78	0.0
Random Forest – Balanced	Training	54.47	0.466	54.32	0.438
	Testing	54.32	0.399	54.19	0.445
Random Forest - Outliers Removed	Unbalanced Training	93.11	0.0	93.11	0.0
	Unbalanced Testing	93.02	0.0	93.02	0.0
	Balanced Training	68.61	0.703	68.46	0.700
	Balanced Testing	69.38	0.709	69.75	0.712

5.3.1. Insights from obtained results:

Unbalanced Dataset vs. Balanced Dataset: Random Forest models perform significantly better on unbalanced datasets in terms of accuracy but fail to generalize (F1 = 0.0), as they predict only the majority class. Balancing the dataset improves F1 score, indicating better precision and recall balance despite lower accuracy.

Effect of Outlier Removal (Based on Income): Outlier removal significantly improves model performance on both balanced and unbalanced datasets, as evidenced by increased accuracy and F1 score. After outlier removal, the models demonstrate improved generalization and better balance between precision and recall.

Comparison with External Library Implementation: The performance of the external library implementation (using 100 trees) is comparable to the scratch implementation (using 20 trees) in terms of accuracy and F1 score. Both implementations show similar trends in performance on balanced and unbalanced datasets before and after outlier removal.

Overall, the results highlight the importance of dataset balancing and outlier removal for improving the effectiveness and robustness of Random Forest models, especially in scenarios involving imbalanced data. The F1 score serves as a critical metric to assess model performance, especially when dealing with class imbalance. Further tuning and optimization of model parameters may lead to enhanced performance and better generalization capabilities.

6. Future Scope

i. Feature Engineering:
Advanced Feature Selection: Explore more sophisticated feature selection techniques such as recursive feature elimination (RFE), principal component analysis (PCA), or L1 regularization to identify the most predictive features for model training. Feature Transformation: Experiment with feature transformation methods like scaling, normalization, or binning to enhance the model's ability to capture complex relationships within the data.

ii. Model Enhacement:
Hyperparameter Tuning: Conduct systematic hyperparameter tuning using techniques like grid search or random search to optimize parameters such as the number of trees, tree depth, and minimum samples per leaf. Ensemble Variants: Implement ensemble techniques such as gradient boosting or stacking to leverage diverse models and potentially boost overall performance.

iii. Data Preprocessing:
Handling Imbalanced Data: Explore advanced strategies for handling class imbalance, such as synthetic data generation (SMOTE), ensemble techniques like EasyEnsemble or BalanceCascade, or cost-sensitive learning approaches. Outlier Detection and Removal: Continue refining outlier detection methods tailored to specific features or domains to improve data quality and model robustness.

iv. Model Evaluation and Interpretability:
Model Interpretability: Utilize model interpretation tools like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to gain insights into feature importance and decision-making processes. Cross-Validation Techniques: Implement robust cross-validation methods (e.g., k-fold cross-validation) to obtain more reliable estimates of model performance and generalize better to unseen data.

References

1. Our Git hub repo Link :- ML-project-CS9
2. Kaggle dataset and paper:
   - Data set and paper: Loan Prediction Based on Customer Behavior.
3. References for models:
   - Logistic regression: Logistic Regression on Spiceworks.
   - Naive Bayes: Naive Bayes Classifiers on GeeksforGeeks.
   - Random Forest: Random Forest Algorithm on Medium.
4. Additional reference:
   - Categorical Data Encoding Techniques: Medium Article.
5. Research paper:
   - Research paper reference: Loan Default Prediction Model.