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

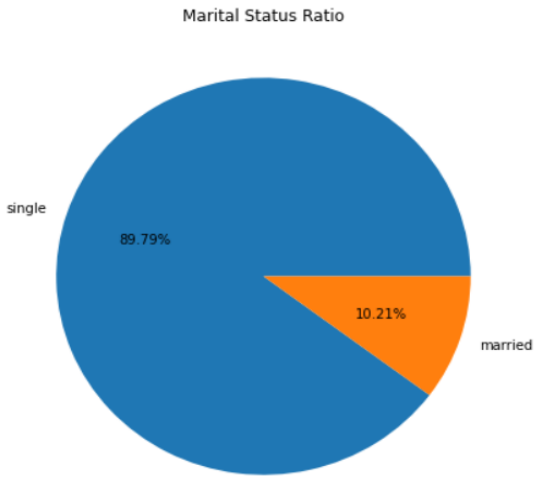
This study explores loan approval prediction using machine learning on a Kaggle dataset featuring parameters like income, age, and profession. Preprocessing methods including encoding and dataset balancing were applied to address data imbalance. Logistic Regression, Naive Bayes, and Random Forest algorithms were tested, showing varying performance levels. Logistic Regression exhibited mixed accuracy using both library-based and manual approaches. Naive Bayes demonstrated moderate performance but faced challenges in capturing complex data relationships. Random Forest showed promising results, particularly after addressing class imbalances. 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

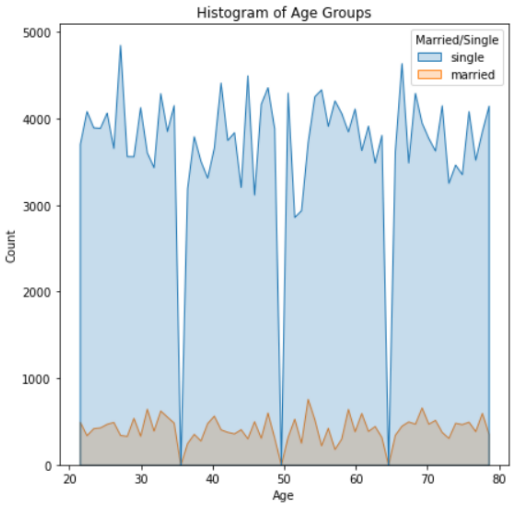
We have taken the data set from the following link: Kaggle link

We have to predict whether a person will get loan or not depending on certain parameters like income, age, experience, etc. The dataset which we have taken from kaggle has the following columns :- Id, Income, Age, Experience, Married/Single, House_Ownership, Car_Ownership, Profession, CITY, STATE, CURRENT_JOB_YRS, CURRENT_HOUSE_YRS and Risk_Flag. The dataset has 252k entries and the Risk_Flag values which is our output has 221,004 values where loan is not given and 30,996 values where the loan is given. Thus our dataset is imbalanced and we will make the dataset balanced by under-sampling the abundant class and over-sampling the minority class. .

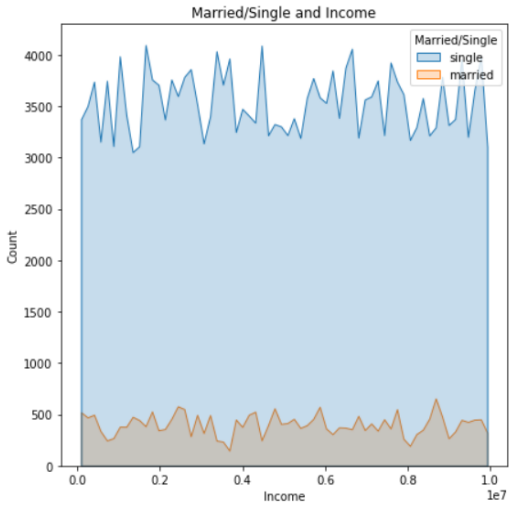
2. Feature Explanation



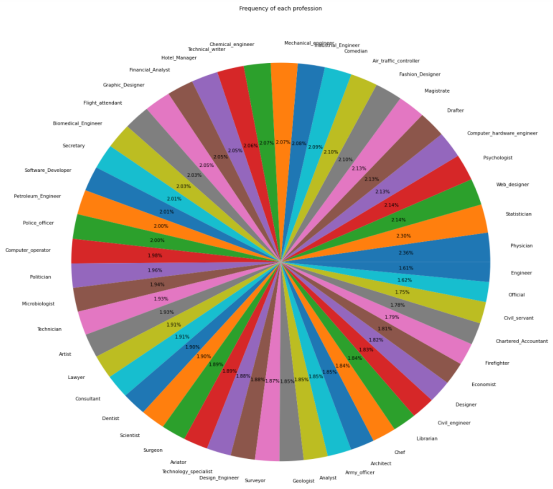
From the above pie chart, its clear that 90 percent of the people are single and rest 10 percent is married. From this we figure out that data is imbalanced with respect to this feature with most of the people belonging to one class.



The above histogram depicts equal distribution of people among various age groups. Also the ratio of married to single people in each age group is similar.



The histogram shows the distribution of count of people v/s income. It also shows details about marital status of people. We can see that we have people from various income groups and more or less they are equally distributed.



The pie chart shows the frequency of each profession with 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%.

3. Methods

3.1. Preprocessing

3.1.1. 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 (0 and 1).
ii. Equal representation of classes minimized bias and improved the model's ability to generalize across diverse scenarios.

3.1.2. Naïve Bayes:

a. Exploratory Data Analysis (EDA):
i. Graphical Analysis: Plotted graphs for all features to visualize their distributions and identify any patterns
ii. Decided on feature selection and dropped irrelevant or redundant columns

b. Data Encoding:
i. Discretized the 'Income' feature into 5 income brackets for better representation and analysis.
ii. Encoded variables including 'Married/Single','House Ownership', and 'Car Ownership' using mapping.
iii. Applied one-hot encoding for 'Profession' and 'State' columns to convert categorical variables into numerical representations.
c. Data Preparation:
i. To simplify the dataset for model training, dropped unnecessary columns like 'City', 'ID' that were not relevant to the prediction task.
ii. The features were separated from the target variable 'Risk_Flag' ensuring that the model is trained and tested using the appropriate input features

3.1.3. 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.

3.2. Implementation

3.2.1. Logistic Regression

Logistic Regression is a widely used statistical model for binary classification tasks, estimating the probability of an input belonging to a specific class using a logistic function.

Logistic regression uses the logistic function, also known as the sigmoid function, to calculate the probability of a specific class label based on input features.

3.2.2. 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.

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.

3.3. Random Forest

3.3.1. Decision Tree Implementation

The `Node` class defines the structure of a decision tree node, including attributes such as feature index, threshold for splitting, child nodes (left and right), information gain, and leaf value. The `DecisionTreeClassifier` class constructs the decision tree recursively using methods like `build_tree`, `get_best_split`, `split`, and `calculate_leaf_value`. - Splitting criteria are based on information gain, computed using entropy or Gini index. - Then we fit the decision tree on the provided dataset, and the `predict` method makes predictions for new data points.

3.3.2. Random Forest Implementation

Overview: Multiple trees together form a forest, which is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce overfitting.

Functionality: Implemented a loop to create multiple decision trees, trained on a bootstrap sample, and selected random features for each tree, combining predictions through majority voting for final prediction.

Output: The random forest generates predictions for the test data by aggregating predictions from all individual decision trees, thereby providing improved accuracy and robustness compared to a single decision tree model.

4. Progress:

a. Logistic

Implementation Details:

Model Application: The logistic regression model was implemented using Python libraries like scikit-learn, using one-hot encoding to avoid dimensionality explosion by excluding a unique categorical feature.

Results: Upon applying logistic regression with one-hot encoding and dropping the city column using external libraries, the accuracy of the model was approximately 87%.

Manual Implementation of One-Hot Encoding: To further explore data preprocessing techniques, one-hot encoding was manually implemented from scratch. Despite this manual approach, the accuracy obtained on the test data was approximately 80%.

Implementation from Scratch: Despite attempting to replicate the model's functionality, the initial accuracy achieved in logistic regression was only 53% after implementing it from scratch..

b. Naïve Bayes

TRAINING AND TESTING SPLIT:

i.Split the given dataset into training and testing sets using an 80-20 split ratio.
ii. Achieved an accuracy of approximately 64% on the test.

TESTING ON DIFFERENT DATASETS:

i.Tested the Naive Bayes model on a separate dataset to evaluate its performance.
ii. Observed an accuracy of approximately 62% on the test dataset.

HANDLING UNBALANCED DATA:

i.Addressed the issue of unbalanced data by downsampling the dataset to balance the classes.
ii. Achieved an accuracy of around 51% after balancing the dataset.

c. Random Forest

EXTERNAL LIBRARY IMPLEMENTATION - RANDOMFORESTCLASSIFIER

- The RandomForestClassifier model achieved 87.74% accuracy on training on 80-20 spilt of data, but low F1 score suggests an imbalance between precision and recall. The confusion matrix correctly classified the dominant class (class 0), but it revealed a few true positives and false positives.
- When tested on unseen data (test set), the model maintained a similar accuracy rate of around 87.60%. The F1 score remained low, indicating potential issues with generalization and model robustness.
- Undersampling class 0 reduced model accuracy to 55.99%, but improved confusion matrix and F1 score, indicating balanced classification with increased true positives and true negatives, and improved performance on balanced data with better balance between precision and recall.
- The balanced test data showed a decrease in accuracy to 55.18%, but improved classification performance in true positives and negatives and an improved F1 score.

SCRATCH IMPLEMENTATION (20 TREES) - RANDOMFORESTCLASSIFIER

- The RandomForestClassifier model, trained with 20 trees, achieved 87.73% accuracy, similar to external library implementation, but had poor performance due to dominant class classification.
- Testing on unseen data revealed similar results, with all predictions being classified as the dominant class, resulting in an F1 score of 0.0.
- The model's performance improved slightly after undersampling the dominant class, with accuracy increasing to 55.43% on training and 54.00% on balanced test data with an improved F1 Score = 0.46273. .

3.4 Conclusion

A. PROBLEM:

- Logistic Regression is a versatile algorithm used for binary classification tasks. While initial attempts involved manual preprocessing and implementation, the accuracy of from-scratch models fell short compared to established libraries.
- A significant issue was discovered where the model incorrectly predicted positive risk (1) when the actual value was negative (0) in many instances. This suggests a potential limitation of the Naive Bayes model in handling the complexities present in the dataset.
- The provided data is imbalanced, with class 0 being predominant. Both implementations of RandomForestClassifier exhibited similar performance trends. Undersampling the dominant class enhanced model performance, particularly in F1 score, emphasizing the need to address class imbalance.

B. IMPROVEMENTS:

- The focus now shifts to optimizing the logistic regression model using hyperparameter tuning techniques such as grid search or random search. This involves fine-tuning parameters like learning rate, regularization strength, and the number of iterations to enhance model performance.
- The proposed methods aim to enhance model performance: by refining feature representation through introducing new features or transforming existing ones; optimizing Naive Bayes model performance via tuning hyperparameters like smoothing parameters; and leveraging ensemble methods such as bagging or boosting to amalgamate multiple models, ultimately striving for improved prediction accuracy.
- To boost model performance, strategies such as improving feature relevance through techniques like feature selection or creating new features, addressing class imbalance using methods like SMOTE, optimizing model parameters like max_depth and n_estimators, ensuring robustness with cross-validation, considering algorithms like XGBoost or LightGBM, and exploring outlier detection techniques are essential.