Loan approval prediction

Prepared by:

Racheal Chidera Ezeja

INTRODUCTION

This project aims to create a predictive model for loan acceptance from a bank's data. Along with loan-related information, this dataset comprises several consumer traits including age, income, education, and credit use. Using several machine learning techniques—Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, and Gradient Boosting—the loan approval process is aimed at automating.

PURPOSE

The aim is to assess the predictive accuracy of the model and pinpoint the main elements affecting loan approval. Apart from the processing time for every model, the paper will evaluate their performance using criteria including accuracy, precision, recall, and F1-score. This study will shed important light on the most efficient model for the loan approval process, so enabling the bank to make quick, fact-based decisions about loan applications.

DESCRIPTION

Know your data:

Here, we are first exploring the structure and salient features of the dataset using an EDA. We load the data from the designated file path and choose the first sheet and save it to the variable bank_df. Then we show the bank_df DataFrame, which offers a preview of the dataset with the first few rows shown. This preview validates that the dataset consists of fourteen columns covering loan details, consumer demographics, and other financial data.

We then examine the data types of every column and then view all column names in a list form. Next, we create summary statistics for every numerical column using bank_df.describe(). This offers important new perspectives including the 25th, 50th, and 75th percentiles as well as count, mean, standard deviation, minimum, and maximum values. The output shows there are 5,000 records in the dataset. These numbers help us to spot possible outliers, evaluate data distribution, and grasp variable ranges of values.

Finally, we show the dimensionality of the dataset—that is, 5,000 rows by 14 columns. This last step confirms our knowledge of the size of the dataset, which is absolutely important for organizing the next research activities.

Data Cleaning & Preprocessing:

This includes handling missing values, checking for duplicates, dropping extraneous features, identifying outliers, and evaluating multicollinearity among features. We start by standardizing column names, and then we clean data. We eliminate leading and trailing spaces, and lowercase letters from the data frame and underline non-alphanumeric characters. This guarantees consistency in column names.

We then iteratively go over each column in a loop looking for missing values across all others. With the use of isna() for every column.sum(), we count missing values and print the outcomes, so verifying that the dataset contains no missing values. Next, we filter out any duplicate rows by first looking for them. Then we drop the ID and ZIP Code columns since they have no bearing on the analysis. Dropping these columns reduces non-predictive data, so simplifying the dataset.

Outliers:

Plotting boxplots for every numerical column helps us identify outliers. Particularly in Income, CCAvg, and Mortgage, which exhibit greater data dispersion and some extreme values, this visualization helps identify the spread and spot possible outliers.

Using the Interquartile Range approach, we figure the percentage of outliers in every numerical column, then iteratively examine the distribution of every feature with sns.distplot(). This lets us see how data is distributed across features, so helping to spot skewness or multimodal distributions that might affect modeling choices.

Multi-collinearity:
Using the Variance Inflation Factor, we evaluate multicollinearity. Higher VIF values—above 5—indicate multicollinearity, in which case some features have a linear relationship. Age and experience show great multicollinearity at first. We remove experience and recompute the VIFs to solve this, so producing all VIF values below 5, indicating decreased multicollinearity.

Feature Engineering:

First, we assign the target variable, personal_loan, to y while the other features go to x and then scale every feature to a range between 0 and 1, so standardizing X.

Logistic Regression:

With 80% of the data for training and 20% for testing, we then split the data. Crucially for determining the generalizability of the model, this split supports the validation of its performance on unseen data. After computing the training accuracy, we assess the model on the test set computing accuracy, precision, recall, and F1-score. By printing these metrics, we offer analysis of the classification performance of the model. We produce a confusion matrix to offer a graphic overview of the model's accuracy. This matrix shows the distribution of true positives, false positives, true negatives, and false negatives.

The training accuracy of 95.05% and test accuracy of 93.8% are both high and relatively close to each other. This indicates that the model performs well on both the training and unseen test data, suggesting that it has learned the underlying patterns in the data effectively.

The precision score of 88.7% indicates that, out of all the positive predictions made by the model, 88.7% were correct. This means the model is effective at minimizing false positives (i.e., predicting loan approval for ineligible applicants). Also, the model appears to be well-generalized. It is learning generalizable patterns without being overly sensitive to the training data.

Overall, the Logistic Regression model is well-generalized. It effectively balances performance across training and test data, indicating that it should perform similarly on new, unseen data.
Given that our goal is to determine which loans to approve or reject, precision would be the most appropriate metric to focus on. Precision helps minimize false positives, meaning it reduces the likelihood of incorrectly approving loans for applicants who are not eligible or might default. Approving a loan for an ineligible applicant can be costly for a bank, as it may lead to defaults and financial losses. By optimizing for precision, we ensure that when the model predicts a loan should be approved, it is highly likely that the applicant is genuinely eligible.
Next, we use Recursive Feature Elimination to find the five most significant features of the model. RFE recursively removes the least important features using Logistic Regression as the basis estimator, so improving the model to concentrate on the most pertinent variables.

Support Vector Machine:

Here, we evaluate the model's performance, optimize hyperparameters using GridSearch, and choose features with several approaches. Following instructions, we compute several evaluation metrics including accuracy for both training and test sets, precision, recall, and F1 score and forecast on the test data. These indicators let us evaluate how well the model forecasts loan approvals.
The GridSearch finds the combination of parameters with the best accuracy by evaluating each ten-fold cross-valuation set of values. With C=2.0 and kernel=rbf, the best parameters discovered in this case produce an accuracy score of about 97.88%.

Sequential Backward Selection using three scoring measures—recall, F1, and precision—helps us choose features. SBS iteratively removes features, choosing the top five that best meet every criterion. Whatever the scoring metric, the chosen features—age, income, education, CD_account, and online—indicate that these factors most affect the accuracy of the model.

Finally, we choose the five most important features using Recursive Feature Elimination applied with SVM. RFE removes the least important elements recursively, so selecting income, family, education, CD-account, and credit card. These characteristics fit the structure and requirements of the model, thus they are judged the most predictive for the loan approval process.

Decision Tree:

Here, we evaluate the Decision Tree model's performance for loan approval prediction by visualizing the tree, measuring processing time, and calculating various performance metrics. We initialize the Decision Tree with a maximum depth of 3, train it on the training data, and then visualize the decision tree structure, showing how different features (e.g., cc_avg, income, cd_account, education) split the data to classify applicants as approved (1) or not approved (0). Each node displays the feature, Gini impurity, number of samples, and the predicted class. This helps us understand how the model makes decisions based on feature thresholds.

With a training accuracy of 100% and test accuracy of 97.8%, the model is performing well on both datasets. Although the high training accuracy could suggest overfitting, the test accuracy remains close, indicating that the model generalizes well to unseen data. The additional precision, recall, and F1-score metrics provide further assurance that the model is balanced and effective in predicting loan approvals.

The Decision Tree's high accuracy and low processing time make it a suitable choice for quick and accurate loan approval predictions. However, due to the high training accuracy, it would be beneficial to monitor this model for potential overfitting on larger or more complex datasets in the future.

Random Forest:

Using the RF approach, we evaluate the Random Forest model for loan approval prediction, compute feature importance, and examine the outcomes. We train a Random Forest classifier you starting with twenty trees (n_estimators=20) on the data. We obtain a test accuracy of 98.3% following test data prediction. This great accuracy shows that the model is faithfully catching trends in the data.
Calculated further metrics including precision, recall, and F1-score offer a whole picture of the model's capacity to appropriately approve or reject loans. These measures enable one to assess the accuracy of the model in spotting qualified candidates.

We plot a bar chart displaying the relative importance of every feature using the built-in feature importance derived from the Random Forest model. Among the most important factors here are income, education, CC-avg, and family. The degree to which each feature lowers the impurity across the trees determines this inherent relevance and helps one to identify which features most influence the predictions of the model.

Using the permutation_importance technique—which gauges how much the performance of the model suffers when the values of each feature are randomly shuffled—we can thus further validate feature importance. This approach reflects the influence of a feature on the actual predictions, so providing a more reliable evaluation of feature importance.
Unlike the results of the built-in approach, securities_account, and age were shown to be significant elements in the permutation importance calculations. This variation implies that, even with their low Gini-based relevance, these characteristics have more predictive value when the model depends on them directly.

Gradient Boosting Classifier:

We calculate and interpret various evaluation metrics for the Gradient Boosting model after training it on the loan approval dataset. We train the Gradient Boosting model on the training data and then make predictions on the test data. The training accuracy is 99.65%, and the test accuracy is 98%. The high training and test accuracies, which are very close to each other, indicate that the model is well-generalized.

Stakeholders Questions:

1. What were the three most significant variables?

From the feature importance, the three most significant variables across models were:

- Income
- Education

- CCAvg (Average credit card spending)

These features consistently appeared as the top predictors for loan approval decisions across different models (Random Forest and Gradient Boosting), indicating their strong influence on the model's predictions.

2. Of those three, which had the most negative influence on loan acceptance?

To determine negative influence, we would consider a feature that, when increased, decreases the likelihood of loan approval. In the context of these features:

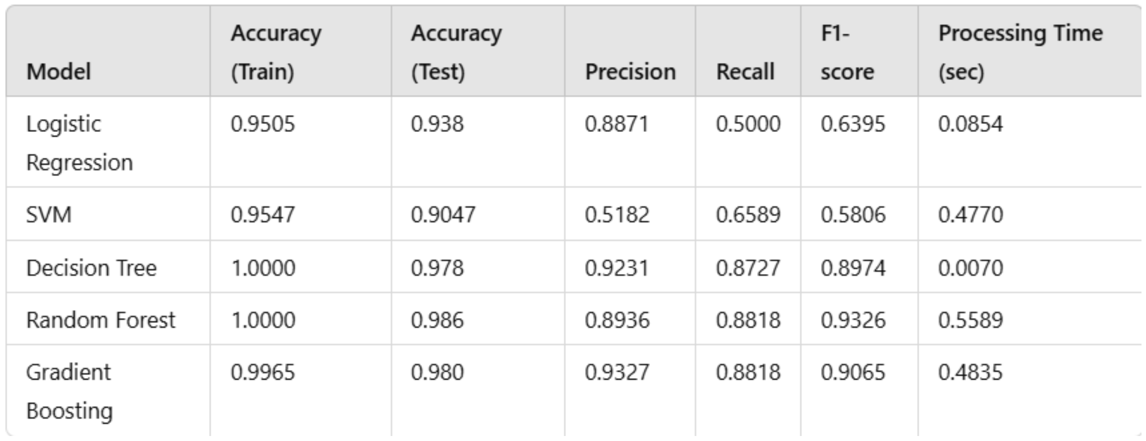
- CCAvg (Average credit card spending) is likely to have the most negative influence on loan acceptance. Higher credit card spending could indicate potential risk, leading the model to more likely reject applicants with higher values in this feature.

While Income and Education positively correlate with loan acceptance, higher average credit card spending might reduce approval chances due to perceived risk.

3. What is the best KPI (recall, precision, F1-score, etc) to measure the performance of your models?

Given that our goal is to decide on loan approvals accurately, Precision is the best KPI to focus on. This metric minimizes false positives (i.e., incorrectly approving loans for ineligible applicants), which is crucial for avoiding financial risks associated with defaults. Precision ensures that when the model predicts a loan approval, it is likely to be correct.

4. Summarize all the KPIs for each model in a table including the processing time for running the models.



Model	Accuracy (Train)	Accuracy (Test)	Precision	Recall	F1-score	Processing Time (sec)
Logistic Regression	0.9505	0.938	0.8871	0.5000	0.6395	0.0854
SVM	0.9547	0.9047	0.5182	0.6589	0.5806	0.4770
Decision Tree	1.0000	0.978	0.9231	0.8727	0.8974	0.0070
Random Forest	1.0000	0.986	0.8936	0.8818	0.9326	0.5589
Gradient Boosting	0.9965	0.980	0.9327	0.8818	0.9065	0.4835

5. Of those models, which one has the best performance according to your answer in Q3 and the processing time?

According to the priority on Precision and reasonable Processing Time, Gradient Boosting is the best performer:

- Precision: Gradient Boosting has the highest precision (93.27%), which aligns with your goal to minimize false positives in loan approvals.
- Processing Time: Gradient Boosting has a moderate processing time of approximately 0.48 seconds, making it efficient enough for practical use without compromising on accuracy.

While Random Forest also has a high F1-score and balanced metrics, Gradient Boosting's superior precision makes it better suited to avoid incorrect approvals, which is the primary objective. Therefore, Gradient Boosting offers the best performance in terms of precision and overall reliability in loan approval predictions, while maintaining an acceptable processing time.

CONCLUSION

In this assignment, we analyzed a loan approval dataset using five machine learning models: Logistic Regression, SVM, Decision Tree, Random Forest, and Gradient Boosting. Our goal was to predict loan approvals and identify significant features influencing these decisions. Through the evaluation metrics, we identified **Precision** as the most critical metric, as it helps minimize false positives—essential for reducing financial risk in loan approvals.

Among the models, **Gradient Boosting** emerged as the best performer, achieving the highest precision (93.27%) and an F1-score of 90.65%, while maintaining reasonable processing time. Key variables identified as most significant in predicting loan approvals were **Income**, **Education**, and **CCAvg** (average credit card spending), with CCAvg having a potential negative influence on approval decisions due to perceived risk.

Overall, this analysis highlights the importance of choosing appropriate evaluation metrics and feature selection methods in predictive modeling, especially in the financial domain where accuracy and precision are critical.

REFERENCES

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.
- Raschka, S., & Mirjalili, V. (2017). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd.
- Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324