Using Machine Learning to Predict Loan Defaulters

Benjamin Hu, Hanna Yen, Kristi Zhang

1 Introduction

Defaulting on a loan occurs when a customer fails to keep up with loan repayments agreed upon for a specified period of time, indicating a pattern of risky behavior. Defaulting triggers the remainder of the loan balance due in full and significantly lowers the credit score of the client, making it harder for them to borrow in the future. In addition, there may be harsher punishments including the repossession of property and assets.

It is important to distinguish the difference between defaulting and payment delinquency. Missing a single scheduled payment which affects a borrower's credit score, but is remedied by paying any overdue amount and fees is not considered defaulting. Forecasting whether a client will default is not only key to reducing the risk of financial losses a bank may incur upon approving loans, but also building customer trust by only approving customers lenders know will be able to take on the responsibility of a loan. Thus, given a customer's application for a loan, we would like to be able to accurately predict whether or not the customer will default on the loan.

2 Exploratory Data Analysis

To help us potentially be able to predict whether or not a client will default on a loan, we utilized the Loan Defaulter dataset. This dataset includes information that is typically required on a loan application such as the client's gender, whether they own a car/house, how many children they have, etc. It also includes more subtle information such as the day of the week the application was submitted, how many times it was edited, etc.

The dataset initially has 307,511 rows, each being a unique applicant. It contains 122 features which consist of categorical and real features all relating to an applicant's loan application information. The target variable we are trying to predict is 1 if the customer defaulted and 0 if they did not default. Upon preliminary analysis of the data, we aimed to identify useful predictors to include in our models and remove unimportant ones. We discovered that many of the features had 50-70% missing entries as shown in Figure 1. In addition, we looked at the correlation between various quantitative features and the target variable. Figure 2 is a heatmap visualizing the relationship between the target and features pertaining to any available contact information a requestor provided in their

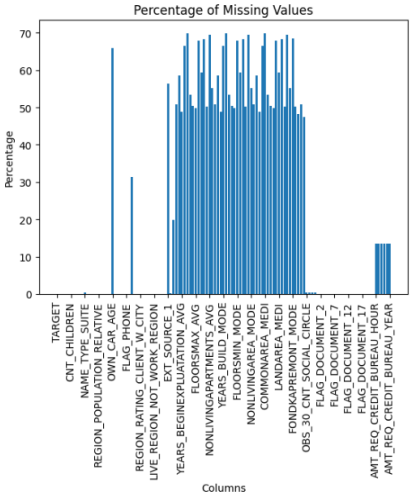

Fig 1. Null percentages of all features.

application. It appears that there does not appear to be any significant correlation. Figure 3 illustrates the class distribution of whether or not an applicant submitted a specific document

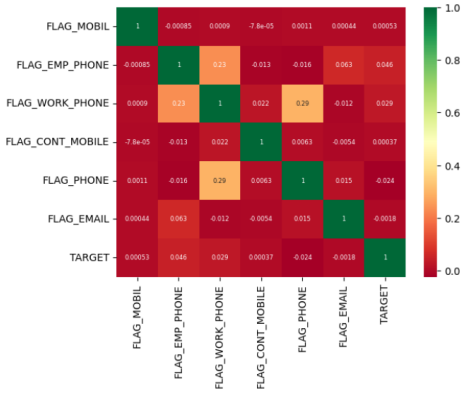

Fig 2. Heatmap between target variable and applicant contact information.

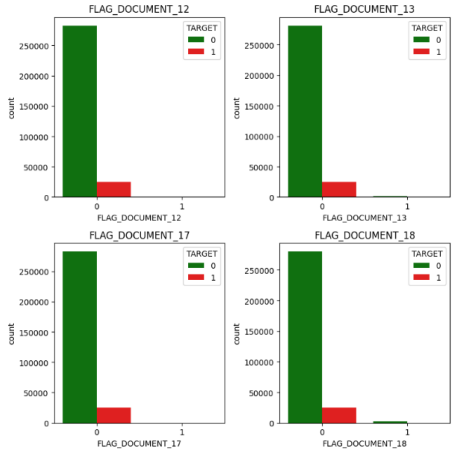

Fig 3. Target variable class distribution of applicant document 12, 13, 17, 18 submissions.

in their application. We observed that the distribution across all documents (2-22) were similar with the exception of possibly document 3. We also compared the age distribution between applicants who defaulted or did not default, and noticed that the age distribution of those who defaulted was less uniform than the distribution of those who were able to pay their loan on time illustrated by Figure 4.

Additional comparisons between the target variable and other features were made. We observed that education type and gender also had different class distributions between applicants who defaulted and applicants who did not default. With these initial observations, we were able to get a sense of which predictors were important to include in our models and which predictors were unnecessary and could potentially be left out in order to improve model performance from overfitting and improve interpretability by reducing model complexity.

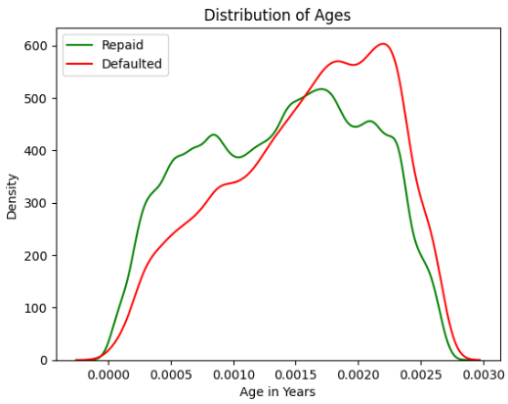

Fig. 4 Age of applicant distribution among target variable classes.

3 Data Preprocessing

After performing initial exploratory data analysis on the dataset, we used our observations to clean and transform the raw data into a format suitable for modeling. Preprocessing ensures that our data is accurate, complete, and consistent.

3.1 Class Imbalance

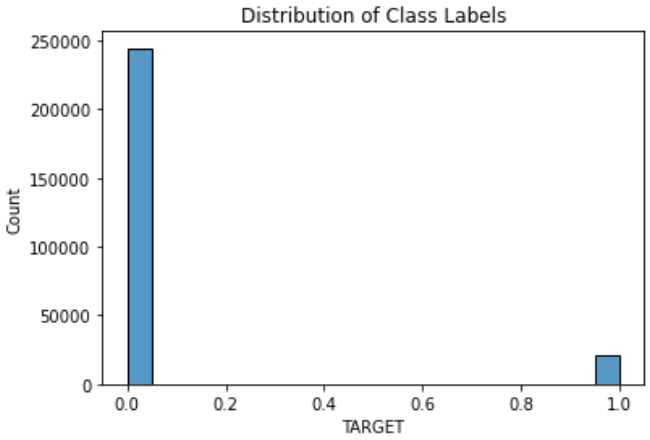

Fig 5. Histogram of distribution among classes (1 represents the client has payment difficulties, and 0 otherwise).

Figure 5 shows the imbalance between classes in the dataset. Approximately, 8% of the dataset contains examples that have a class of 1. As a result, the models could achieve very high accuracy solely predicting class 0 for every example. Class imbalance is typically remedied through oversampling or undersampling. Oversampling is the process of continuously drawing samples from the minority class until the distribution is balanced. Undersampling is the process of sampling from the majority group only up to the amount of examples in the minority class. Although these approaches are simple, they have undesirable properties, namely overfitting and data underutilization (more bias), respectively. Instead, we chose to add class weights, depending on the size of the class. This increases the penalty on misclassifications of the minority class.

3.2 Train-Test Split

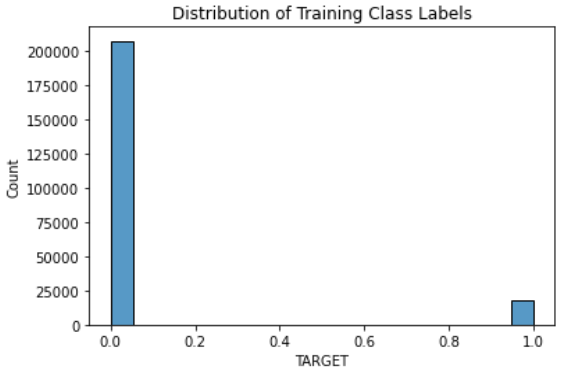

Fig 6a. Histogram of distribution among classes of the train dataset.

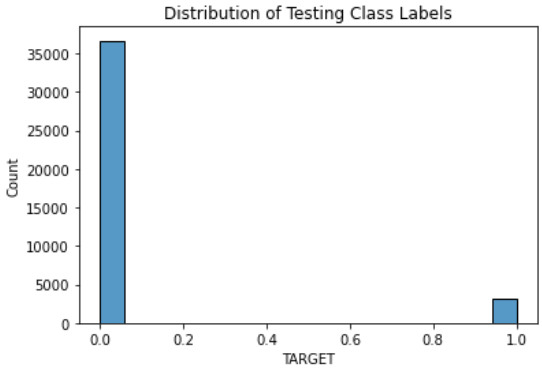

Fig 6b. Histogram of distribution among classes in the test dataset.

Due to class imbalance, we split the data while preserving the relative ratios of classes in the entire dataset. Figure 6a and 6b show the distribution between the train and test datasets after splitting. This is desirable as we expect future data will follow a similar distribution. The train set consisted of 15% of the entire dataset and the train set consisted of 85% of the entire dataset.

3.3 Feature Engineering

We filtered through all of the features that were available, prioritizing interpretability of the features when selecting features to include in our models. Our feature engineering also included transforming categorical features, dealing with null values, and normalizing.

3.3.1 Feature Selection

The dataset had the description of the 122 features, several of which did not carry meaning because they were protected information. We excluded boolean features that denoted whether or not a client supplied a certain unnamed document. We also excluded information on the client's neighborhood and apartment information because they were preprocessed (only statistics of these features were given). The dataset also contained many categorical features, such as occupation and family status. Many of these features were strings so to make them more interpretable for the model, we used one-hot encoding.

3.3.2 Missing Values

Applicants often left out fields in their application. Depending on the feature missing, we excluded that example from the dataset. For example, since we cannot reasonably impute occupation type or number of family members, we left the examples with those values missing out. On the other hand, for the feature *EXT_SOURCE_1*, a normalized measure of credit score, we instead used the mean value of that feature because the credit score is an important feature and the mean value is a good estimate for the general value for credit score across applicants.

3.3.3 Normalization

Since many of the models use regularization, we normalized all the real features so that their values fall between 0 and 1. Normalization is a key preprocessing step for the logistic classification model and the neural network because these gradient descent based algorithms are sensitive to the range of data points and affected by the scale of the features. It improves the performance and training stability of the models.

4 Supervised Learning Models

Given the cleaned, normalized data, we built three classifiers: a logistic regression model, an extreme gradient boosting decision tree classifier, and a simple neural network classifier. The models were all trained based on the same train-test split and evaluated through k-fold cross validation.

4.1 Logistic Regression[1]

The logistic/sigmoid function maps real values to values between 0 and 1. The interpretation of the logistic function is that the output represents a probability distribution over the likelihood of the two different classes. Logistic regression uses the sigmoid function to discriminate between the two labels, values less than 0.5 map to 0 and values greater than 0.5 map to 1.

4.2 Extreme Gradient Boosted Decision Tree[2]

Boosting is an ensemble method that builds models sequentially, each weak learner learning from the mistakes from previous weak learners. In boosting, each subsequent learner is trained by a method similar to gradient descent. The gradient of a loss function of choice is approximated and used to update the predictor. The weak learners in this case are a series of decision trees. Decision trees are a non-linear machine learning algorithm that assigns a label based on partition of the feature space. The splits among a decision tree are decided using a greedy approach that maximizes the purity function (e.g: Gini index). Extreme gradient boosted decision trees (XGBoost trees) streamlines the process by using the data distribution to reduce the search space of possible splits. In addition, it employs regularization techniques that help generalization compared to regular gradient boosted decision trees.

4.3 Neural Network[3]

Neural networks are a set of algorithms that aim to characterize patterns within data. The neurons or nodes are organized in layers, each layer getting input from the previous layer. Neural networks are typically split into three types of layers: input layer, hidden layer(s), and output layer. The input layer takes in raw data while the hidden layer(s) attempt(s) the data into something meaningful for the output layer to make a prediction. The connections between each layer are represented by weights, which can be optimized by backpropagation. Backpropagation takes advantage of the multivariable chain rule to determine the effect on the loss function of changing each weight in the network. Then, the weights are updated in a gradient descent manner. Combined with activation functions, neural networks have the ability to represent complex, non-linear data.

We used a feedforward neural network, which is a simpler type of neural network because it does not have cycles so it only pushes information forward. It only had one hidden layer in addition to the input and output layers. The hidden layer had a neuron for every feature in the clean dataset and was fully connected to the other layers.

4.4 K-fold Cross Validation

Since the problem is a supervised learning problem, *k*-fold cross validation is a robust way to evaluate the ability of a model on unseen data. In k-fold cross validation, the training set is split into *k* equal partitions. Then, for each partition, a model is trained on the remaining data and is evaluated on that partition. In the end, there will be *k* models. The average error is an estimate of the model's predictive power on unseen data. To test the predictive power of the models above, we employed *k*-fold cross validation with k=5.

5 Model Prediction Results

Figure 7 illustrates the performance of each prediction model. Accuracy is not a fitting metric for this classification task because if the model predicts the majority label, then the model will automatically achieve a very high accuracy. Instead, we used the F1 score and the area under the ROC curve because they are more responsive to class imbalance.

Model	Logistic Regression		XGBoost Decision Tree		Neural Network	
Training/Test	Training	Test	Training	Test	Training	Test
Accuracy	69.4%	68.54%	77.31%	**73.54%**	72.93%	72.63%
F1 Score	0.257	0.246	0.359	**0.260**	0.287	0.257
ROC AUC Score	0.753	**0.747**	0.874	0.745	0.783	0.743
False Negative Rate	0.319	0.305	0.1808	**0.3826**	0.3230	0.3736
False Positive Rate	0.3120	**0.314**	0.2308	0.2350	0.2555	0.2656

Fig. 7 Table of Logistic Regression, XGBoost Decision Tree, and Neural Network performance metrics. For each performance metric, the greatest number is bolded.

The recall is the amount of positive class samples (defaulters) that were correctly identified by the model. Precision is the amount of positive predictions that are right. The F1 score is the

harmonic mean, meaning it tries to maximize both at the same time; thus, preventing the problem we had with using accuracy. The ROC curve compares the true positive rate with the false positive rate across all classification thresholds and is a measure of discriminatory power. Like F1 Score, it solves the problems we see with accuracy.

5.1 Logistic Regression Results

For logistic regression, we observe a lower training and test accuracy across all three models. Most notably, the F1 test set score is lowest among all three models. Thus, we believe that this may not be the best model for predicting defaulters. The upside to the logistic regression model is that it has the lowest false negative rate. This could be important in a business setting because by making a wrong prediction on a defaulter, the company can lose money.

5.2 Extreme Gradient Boosted Decision Tree Results

The Extreme Gradient Boosted Decision Tree has the best F1 score. However, it is currently overfitting which can be seen from the discrepancy between the ROC AUC scores, the F1 scores, and the false negative rates. Even so, we believe this to be the best model because it has the most potential to improve, and because there is room to generalize further.

5.3 Neural Networks Results

Overall, the neural network overfits less compared to the trees, but it still slightly overfits. In addition, it has a high false negative rate, but this can be adjusted depending on whether a lender wants to prioritize the false negative rate or the false positive rate.

5.4 Interpreting Results

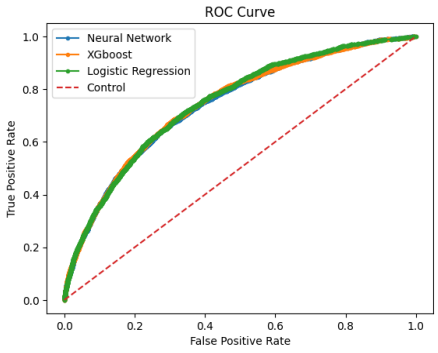

Fig 8. ROC curve for three models.

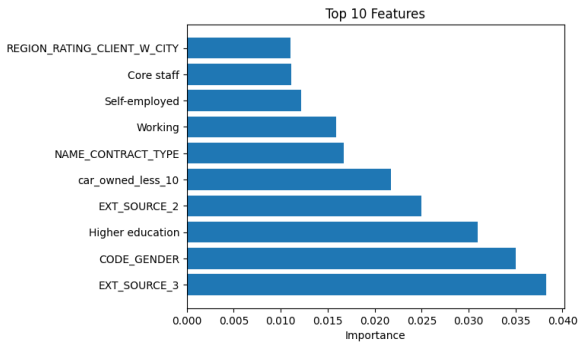

Fig 9. XGBoosted Decision Tree top ten important features.

Figure 8 illustrates the ROC curve comparison between all three models. Analyzing the results from Figure 7 and 8, we believe that the Boosted Decision Tree is the best model that predicts loan defaulters. Though overfitting may be present, the model has the highest F1 score, highest training AUC score, lowest false positive rate, lowest training false negative rate, lowest training false negative rate. It also has the marginally second highest test AUC score. In Figure 9, we can observe that the most important features are an applicant's previous score, gender, and education type. Other notable features that may predict whether or not an applicant will default on a loan include residence rating, occupation, income source, loan type, and car age.

6 Conclusion

Accurately predicting loan defaults is a very difficult task. While our model may not be sufficient as a standalone decision maker, it has enough predictive power to be an effective consulting tool for our bank's employees. We recommend integrating the model into a bank's decision making pipeline in order to better protect both the bank and its customers regarding the distribution of loans. Additionally, by incorporating our model, bank employees can gain valuable insights into the applicant factors that contribute to loan defaults.

6.1 Future Work

Future work could consist of utilizing previous loan application information of applicants. It could be interesting to compare information across applications to identify trends like change in income, frequency of loan requests, etc. In order to improve the boosted decision tree model itself, there are a couple of aspects that can be considered. If the model is overfitting, the depth of the tree and number of features should be reduced. If false positives are less costly than false negatives, the prediction threshold should be adjusted accordingly (i.e. from .5 to .4) so that defaults are predicted more aggressively.

6.2 Fairness

In the industry of credit and banking, the Equal Credit Opportunity Act makes discrimination unlawful in credit applications based on demographics such as race, color, religion, national origin, sex, marital status, age, etc. Thus, when creating any supervised learning model we must be cognizant about the predictors included. In the case of our Boosted Decision Tree model, for future models we should not be allowing classifiers to use any non-protected attributes. When including features on which discrimination is prohibited, we need to ensure that the prediction is independent of the protected attribute given other covariates. In other words, the classifier must be unaware of the protected attribute.

References

[1] https://realpython.com/logistic-regression-python/
[2] https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5
[3] https://wiki.pathmind.com/neural-network

Note: To view any project code please refer to our repository ipynb notebooks.