COMP 4948 Assignment 2

Predicting Loan Defaults

Table of Contents

[Problem Definition 1](#_Toc131768339)

[Exploratory Data Analysis 1](#_Toc131768340)

[Data Treatment 1](#_Toc131768341)

[Model Development, Tuning, and Comparison 1](#_Toc131768342)

# Problem Definition

For this assignment, I used a loan default dataset to build several predictive models. The goal was to help predict if loan applicants were likely to default on their loans. The application of this knowledge could help assist banks and those involved in the loan/mortgage process to identify those who are more likely to struggle with the loan they are applying for.

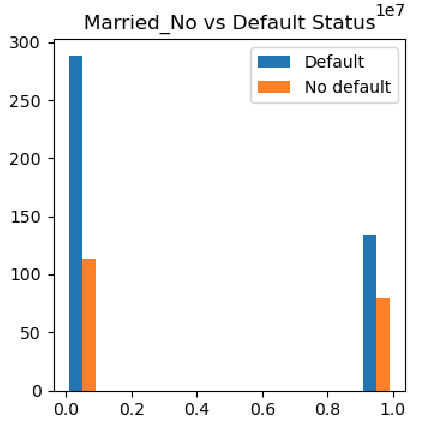
# Exploratory Data Analysis

The loan default dataset features 11 available predictors:

|  |  |
| --- | --- |
| Predictor | Possible values |
| Gender | M or F |
| Married | Yes or No |
| Dependents | 0, 1, 2, or 3+ |
| Education | Graduate or not graduate |
| Self employed | Yes or No |
| Applicant income | Integer ranging from 15,000 to 8,100,000 |
| Coapplicant income | Float ranging from 0 to 4,166,700 |
| Loan amount | Integer ranging from 0 to 70,000,000 |
| Term | Float ranging from 12 to 480 months |
| Credit history | Float ranging from 0 to 1 (binary) |
| Area | Rural, Semiurban, Urban |

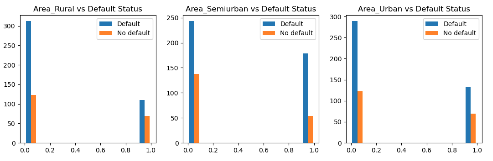
We can examine the relationships between each of these predictors and our target, ‘Status’ (referring to default/non-default status of each loan). Some of the strongest relationships identified are between loan defaults and marriage status, loan amount, credit history, and area (rural, semiurban, or urban).

This distribution shows that those who are married are more likely to default on a loan:



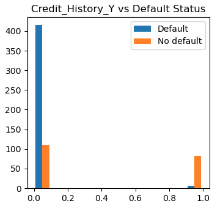
While the proportions look similar between married and not married, there is a slight skew towards non-married loan holders being more likely to default. This may be due to many married households having more than one income, so events like loss of a job are less likely to be as detrimental to loan payments.

Here we can see that those in semiurban areas are more likely to default on their loans.



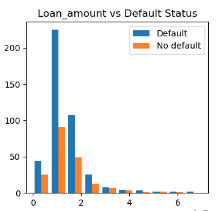
The proportion of defaults vs non defaults is much higher for semiurban areas. Possible explanations for this may include that loan holders may be taking out mortgages or large loans for larger homes relative to their income when compared to urban or rural areas.

Loan holders who did not have previous credit history are very strongly correlated with more loan defaults.



This correlation may speak to an unfamiliarity with loans and financial management; those who don’t have a previous understanding of loans may be more likely to make errors when it comes to managing payments.

The final predictor we will examine here is loan amount; loans of approximately $150,000-$200,000 seem to be the most likely to default.



This might relate to the idea that loan holders who buy homes in this price range might have low-mid range incomes; they are more likely to be impacted by financial events such as job loss and need to resort to defaulting on loans while they sort things out.

# Data Treatment

For this assignment I used several methods to treat the data. One of these methods was dummy variables, replacing categorical (non-numeric) column values with new columns that represent whether or not that data row belongs in which category (for example, if a loan holder is ‘Male’, they would have a 1 under the new ‘Gender\_Male’ column and a 0 under the ‘Gender\_Female’ column). Another method was used on the target column; rather than using these placeholder columns to represent ‘Y’ and ‘N’, I directly replaced those values in the ‘Status’ column with their equivalents (1 for ‘Y’ and 0 for ‘N’).

For missing values, I used KNNImputer from the scikit-learn library. This looks at other rows that have similar values, and uses them to infer what the missing value might be.

I used MinMaxScaler for my data; I felt it was appropriate to reduce the scale of my data as there were columns with very large values (like income and loan amount) and a lot of binary columns. Condensing the scale brought all of this data down to similar scale.

# Model Development, Tuning, and Comparison

My first model was a simple logistic regression model. I used recursive feature elimination to identify important features, and cross-referenced these features with my own observations from examining feature graphs. Beyond feature selection, this was a very simple model.

My second model was an artificial neural network (ANN). I used a Sequential model from the Tensorflow Keras library. I used several forms of grid searching: RandomizedSearchCV, manual full grid search, and a faster manual search testing only one parameter variation at a time. The latter was the easiest to find results with; the other grid searching methods took a very long time and I did not see optimal results. I optimized the following parameters:

|  |  |
| --- | --- |
| Parameter | Tested values |
| Activation function | Relu, sigmoid, softmax, softplus |
| Number of neurons | 10, 25, 50, 100 |
| Number of hidden layers | 3, 4, 5 |
| Initializer | Normal, uniform, zero, he\_normal |
| Learning rate | 0.0001, 0.0005, 0.001, 0.005 |

From this grid search I found the best results with:

* Activation function: softplus
* Number of neurons: 50
* Number of hidden layers: 4
* Initializer: he\_normal
* Learning rate: 0.0001

When processing the output of the ANN, I needed to assign each prediction a 0 or 1 value based on the float output; if the model output a number >0.5, it was read as a 1, and the opposite for 0.

I also used a stacked model, which combined the output from my logistic and ANN models to create its own predictions based on its assessment of its two base models. I used a logistic regression model, and put predictions from the two base models into a new dataframe to pass into the stacked model.

The three models performed similarly during cross-fold validation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Average Accuracy | Accuracy Std Dev | Average Precision | Precision Std Dev |
| Logistic | 0.76 | 0.029 | 0.77 | 0.023 |
| ANN | 0.75 | 0.021 | 0.76 | 0.017 |
| Stacked | 0.75 | 0.019 | 0.76 | 0.015 |

Predictions are consistent across models; performance is not great but the models work as expected.