Good day, everyone.

For this presentation, I’d like to show the results of a simplified model development process for loan default classification using supervised learning models.

(next)

The main motivation of this project is to develop a simple model to help optimize default and non-default classification. In financial institutions like banks, loan default is one of the most common problems being addressed and optimized. Loan default is basically when a loan goes unpaid after a certain period (most common is 3 months or 90 days past the due date) or the borrower goes through other difficulties like bankruptcy, repossession, or litigation. Higher loan defaults may lead to liquidity problems and reserves depletion for the financial institution so banks would employ underwriting rules or a set of rules to guide which borrowers will be granted loans. However, loan application and underwriting usually takes time, and even longer if no application models are in place. Hence, banks would develop models to expedite the application process.

For this project, we will utilize Kaggle’s open dataset as our data source. In this dataset, however, no data dictionary was provided so some assumptions were made.

(next)

The flow and process followed for this project is simplified given time constraints. As usual, we begin with exploratory data analysis before we have our final dataset and develop our models. We then decide on the final model to be used based on model performance.

For exploratory data analysis, we can investigate our features and dataset, check for null values, do imputation, merge bins, do label encoding or one-hot encoding, normalize numeric variables, and investigate correlation and muticollinearity. Once we have addressed our concerns during EDA, we arrive at the final dataset after dropping redundant variables and transforming necessary features. Then, we do a train-test split. For this project, we use a 70-30 train-test split, which is a standard split ratio, though we can also utilize other split ratios depending on data availability. For model development and performance, we explore 4 models – logistic, decision tree, random forest, and XGB. These four are some of the most common methodologies used specifically for loan default classificaition. Hence, they were explored for this project.

(next)

Our dataset is comprised of 148,670 observations with 34 columns. Our target variable is Status with a value of 1 referring to default and 0 otherwise. We have 2 identifiers – the borrower ID and the year (possibly application year). Out of the remaining 31 columns, 21 are categorical, while 10 are numeric. On the columns list, we also notice that some variables have null or missing values. Hence we can check and drop variables with more than 5% null or missing and impute the rest.

(next)

This is what we’ve done in the next step where we drop 7 variables, most of which are numeric. Then we proceed with imputation for the remaining variables still with missing or null values. To simplify, for categorical variables, I decided to use mode, while for the numeric variable, I used mean. It is also observed that some bins have small samples only compared to other bins. In this case, we can merge some bins so that the total % of the population is at least 5%.

(next)

Now, some machine learning models may be able to handle categorical variables but this isn’t always the case. Hence, to ensure all the models we will employ can handle these data automatically, we can do label-encoding and one-hot encoding. Label encoding creates one new column that assigns a number to each value of the column. Usually, this is applicable when the variable is ordinal or there is ranking. On the other hand, one-hot encoding creates n new columns depending on the number of unique values in the variable. Example: loan\_type has 3 values so 3 new columns with binary values 0 and 1 will be created. However, when employing one-hot encoding, it is very possible to fall for the dummy variable trap, which may lead to columns having perfect multicollinearity. In this case, we can remove one of the columns depending on what value we want to keep for analysis.

(next)

We can also normalize numeric values. This is helpful when numeric variables easily sway the impact of the model, essentially making the model volatile to changes to the numeric feature or when features are on different scales, example 20-99 for age, and 0 to 1 million USD for loan amount.

(next)

Once we have completed the necessary steps, we can check for pairwise correlation and drop features with a correlation of 0.7 or higher.

(next)

We also check for Multicollinearity through the calculation of VIF. We start by dropping variables with VIF > 10 and rerun once again to see if there are more variables with high VIF. If we want to be more conservative, we can set the benchmark even lower, but this depends on the availability of data.

(next)

After dropping redundant features, imputing, and transforming necessary features, we are left with still 148,670 observations, but only 18 columns – one target variable and 17 features. The data is then split into train and test datasets with a split ratio of 70-30.

(next)

Finally, we run the four models. Overall, XGB perform best in most metrics, except for precision where it registered the lowest value. In terms of precision, decision tree and random forest performed best with all actual positives properly classified as defaults. We also note that the performance of XGB is very close to logistic regression, which, interestingly, is a much simpler model. What is oddly surprising is that random forest actually performed worse than a simple decision tree, which shows that simpler models may actually perform better in certain scenarios.

(next)

Let’s take a closer look at the logistic model. The model started with 17 features, but one feature (‘Region2\_NE/Central’) did not have a significant p-value so it was eventually dropped. No further exclusion was made since all remaining features have significant p-values. The variable with the highest absolute coefficient is loan amount with a positive impact. This means that a higher loan amount may lead to a higher risk of defaulting. Although, had there been more information it may be better to look at debt-to-income ratio instead, since higher loan amount does not necessarily mean a borrower has a higher chance of defaulting. This will have to be affected by his capacity to pay, in the form of income and other investments. Next, credit\_type variables have the highest absolute impact second to loan\_amount. Again, since no data dictionary was provided, I have only assumed that credit\_type refers to the credit bureaus where credit scores were gathered from. These bureaus are assumed to be EXP or Experian, CRIF, and CIB. This may mean that borrowers who have credit information from reputable credit bureaus have lower probabilities of default. After them, we have submission of application to the institution, which means that the borrower goes directly to the bank instead of a middleman. In this case, the impact is positive, which means that eliminating the middle man may pose a higher risk of defaulting, possibly due to removing an added layer of risk assessment. Over all, the logistic regression model is a sound choice for implementation in this scenario.

(next)

The decision tree model was developed by tuning max\_depth, min\_samples\_leaf, and min\_impurity\_decrease hyperparameters using grid search. We note that credit\_type variables contributed greatly in order to reduce the leaves’ gini impurity to 0 (click mouse), which means actual positives are all properly classified as positive. This is shown by the perfect precision of 100 for the decision tree model. However, if we look at the nodes and leaves shaded in yellow (click mouse), we notice that this branch doesn’t really provide better classification of the borrowers as they are all classified as good. So even though the performance metrics look good, business-wise, it may not be the best choice.

(next)

The random forest model was also developed by tuning max\_depth, min\_samples\_leaf, and min\_impurity\_decrease hyperparameters using grid search for n\_estimators = 100. Though not in the slides, without tuning the hyperparameters, overfitting is observed with the train dataset getting accuracy of 92.6% while the test dataset only gets accuracy of 82%. Hence, hyperparameter tuning was necessary to control for overfitting. Better accuracy for both datasets may be achieved by increasing n\_estimators. However, this takes sufficient time and resource, which is why it was no longer included in this project. As noted, with random forest, we are seeing the same credit\_type variable topping feature importance, underscoring the importance of credit bureaus. However, as far as coefficient signs are concerned, random forest does not show the direction of the feature with regard to loan defaults. So similar to decision trees, random forest may not be the best choice for implementation.

(next)

Lastly, XGB is one of the most used models for competitions due to the model’s high performance. Previously, the usage of such models is quite difficult for banks due to transparency concerns. However, tools such as Shapley enable us to see which variables affect the result as shown here. Similar to the other 3 models, credit\_type offers the highest impact and the best delineation among good and bad borrowers. Points in red mean higher values and blue mean lower values. Since credit\_type is binary, this means, red corresponds to 1 (or that the borrower has credit bureau information in either Experian, CIB, or CRIF) while blue means credit bureau information is absent. Again, this underscores that a borrower with credit bureau information from reputable credit bureaus has a lower probability of default. The same analyses as the ones for logistic regression model are observed for application submission to the institution and negative amortization. Over all, XGB is another candidate that the financial institution may look into for implementation.

(next)

In conclusion, the choice of the final model will depend on the objective of the financial institution. Decision tree and random forest models have the highest precision, which is beneficial if FIs are looking at accepting more applicants and predicting bads only if the certainty of default is mostly certain. However, both models have issues in terms of the business sense of some variables and overall implementation. Hence, they may not be the best choice. On the other hand, Logistic and XGB perform well and the performance values are mostly similar. However, XGB, in this case is more conservative as it classified more borrowers as default. Ultimately, the choice of the model to be implemented is dependent on the FI’s risk appetite and objective. However, in this case, since the difference is not sufficient, I may look into using the logistic model instead since it’s simpler and I won’t be losing more good borrowers that were falsely tagged as bad.

(next)