Final Project

Loan Default Loss Prediction

Applied Data Mining – Financial Risk Modeling

"From binary default flags to continuous loss prediction: using Random Forests to quantify credit risk."

Table of Contents

1. Introduction

The ability to accurately forecast Credit Losses vs Time has become the most pivotal analytic assignment within our existing banking system. With Financial Institutions relying on their ability to utilise Information as a data-based framework to drive their business decisions increasingly, measuring both the likelihood that a given Borrower will Default, as well as measuring the size/type of Total Loss produced from a Default, has now become very important to create a sustainable Lending practice. Borrower Credit Scoring Models have utilised a "yes/no" or binary interpretation of Default; however, while two different Borrowers may be classified as Defaults, the real-world financial results reflect that the Lender's financial Loss will be materially different. This aspect, referred to as Loss Given Default (LGD) or Loss Severity, is a continuum that can drastically change over time, and depends on Macro-Economic Conditions, the Structure of Collateral, Loan Terms and the Characteristics of the Borrower. By understanding and forecasting Loss Severity, Lenders will be able to make much more educated, risk-adjusted decisions, consistent with regulatory requirements, and portfolio profitability objectives.

This research aims to develop a supervised machine-learning model to predict loan loss severity based on a significant dataset of borrower and loan characteristics. The end-to-end analytic model lifecycle consists of exploration, preparation, building the model, evaluating it, and producing an output of how much will be lost at the loan level. The features of the dataset (high dimension, lots of missing data, and a very skewed loss distribution) lead to the importance of using an appropriate modeling technique. Random Forest regression was chosen for this modelling technique because it allows for many numerical predictors without requiring heavy lifting on developing them by creating nonlinear relationships between predictors and loss severity. The methodology provides a methodological basis for building loss prediction systems in the real world of credit, as well as practical implications regarding the most critical predictors of loss and how well the model compared to a baseline.

1.1 Project Goal & Business Context

The primary aim of the study conducted is to provide a better understanding of how much money will be lost to each loan created by the Financial Institution through predicting the potential dollar amount associated with the loan when it goes into default, i.e. using supervised machine learning.

Most Loss Prediction Models used today look at a Borrower as a Binary Decision Tree "Will they Default or Not?" When creating a model to predict various dollar amounts from defaulted loans it is more aligned with how Financial Institutions make decisions about risk that they consider both, "Will this Borrower Default" and "When they Default how much money will I lose?"

From a Banking and Asset Management perspective, an accurate Loss Prediction Model supports many of the most important decisions that a Financial Institution must make:

- **Underwriting/Approval**

How much of a potential downside with this Borrower exists and to what degree does that impact approval or not approving the loan or being able to adjust the requested loan amount.

- **Risk Based Pricing**

The determination of interest rates and fees for that Borrowers based on their expected loss; this therefore provides that Borrower with the appropriate pricing for their risk level. Predicting loss directly feeds into the pricing structure as:

Expected Loss = PD * LGD * EAD

So since the model is approximately LGD * EAD, it is a significant component in determining interest rates.

- **Capital allocation and regulation**

Determine how much capital must be held against a portfolio based on expected and unexpected losses.

- **Portfolio optimization**

Reweight portfolio segments to balance expected return vs. risk, shifting away from high-loss regions when necessary.

In this project, each observation (row) represents a loan, described by many numeric predictors (borrower, loan, and possibly behavioral features) and a target variable loss. Using machine learning, we learn a function:

$\hat{y} = f(x1, x2, \ldots, xp)$

that predicts loss given the feature vector x. We then apply this model to an unlabeled test dataset to produce predicted losses that can be submitted as required.

2. Data Overview & EDA

According to the assignment, we are given three main datasets:

- **Training data (`train_v3.csv`)**
- **Labeled test data (`test_v3.csv`)**
- **Unlabeled test data (`test_no_lossv3.csv`)**

Data set	**purpose**	**Size**
train_v3.csv	Main modeling dataset with predictors and loss	80,000 * 763
test_v3.csv	Additional labeled data for model evaluation	25,471* 763
test_no_lossv3.csv	Final test set for submission (no loss column)	25,471 *762

The dataset had consistent feature structures, enabling smooth integration

2.1 Target Variable Distribution:

```
# Read training data (includes target 'loss')
train_raw <- read.csv("train_v3.csv", stringsAsFactors = FALSE)
train_raw <- drop_index_column(train_raw)

# Read labeled test data (has loss, used only for model evaluation)
# If your environment does not have this file, you can comment it out.
test_labeled_raw <- read.csv("test_v3.csv", stringsAsFactors = FALSE)
test_labeled_raw <- drop_index_column(test_labeled_raw)

# Read final test data without target (we must predict loss here)
test_unlabeled_raw <- read.csv("test__no_lossv3.csv", stringsAsFactors = FALSE)
test_unlabeled_raw <- drop_index_column(test_unlabeled_raw)

# Quick sanity checks on dimensions
cat("Train rows, cols: ", dim(train_raw), "\n")
cat("Labeled test rows, cols: ", dim(test_labeled_raw), "\n")
cat("Unlabeled test rows, cols: ", dim(test_unlabeled_raw), "\n")
```

Interpretation

- This confirms the **dimensions** of each dataset (number of loans and number of features).
- It also ensures that **column names are aligned** across datasets (except loss, which is missing in the unlabeled test set).
- Keeping the structure consistent is essential, because the model trained on one set of columns must be applied to the same columns in the test data.

2.2 Target Variable: Loss Distribution

- summary(train_raw$loss) shows **min, 1st quartile, median, mean, 3rd quartile, max.**

```
# 2.1 Target variable distribution (loss)
summary(train_raw$loss)

# Proportion of zero vs. positive losses (often highly skewed)
prop_zero  <- mean(train_raw$loss == 0)
prop_pos   <- mean(train_raw$loss  > 0)
cat("Proportion with zero loss   :", round(prop_zero, 3), "\n")
cat("Proportion with positive loss:", round(prop_pos, 3), "\n")
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.0000   0.0000   0.7926   0.0000 100.0000
```

- Proportion with zero loss: 0.908
- proportion with positive loss: 0.092

```r
# Histogram of loss
ggplot(train_raw, aes(x = loss)) +
  geom_histogram(bins = 50) +
  scale_x_continuous(labels = scales::comma) +
  labs(title = "Distribution of Loss",
       x = "Loss",
       y = "Count")
```

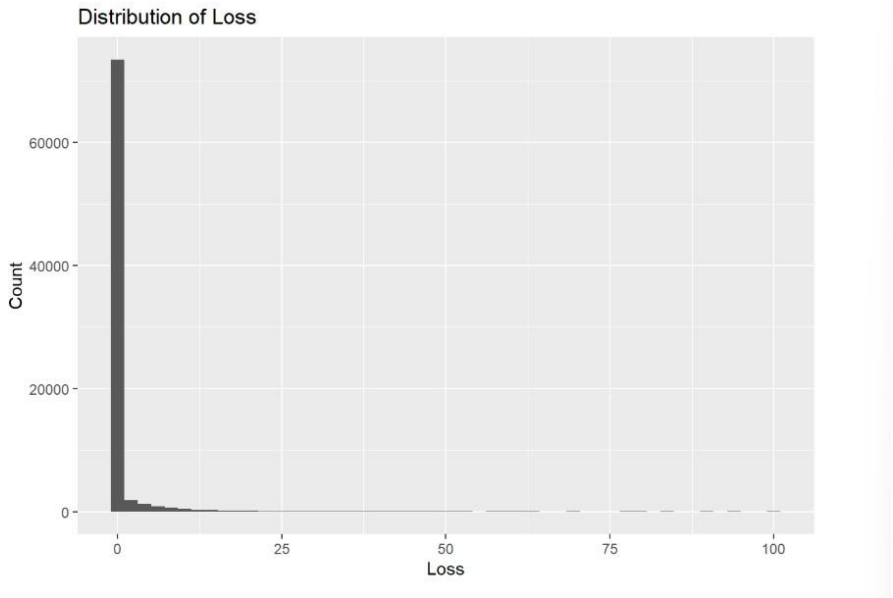

Figure 1. Histogram of the loss variable.

According to the histogram below, it can be observed that most of the loans are either never or very low lost and a large number of loans have large losses. The long right side of the histogram depicts the need for the use of complex and/or non-linear models.

The skewed nature of this distribution is such that simple linear models, as well as naive methods of calculating averages will have a large influence from extreme cases.

The tree-based model, Random Forest can accommodate the skewed data and outliers naturally without any forced adjustments.

Other observations and characteristics of many financial datasets include:

- The mean loss is greater than the median loss indicating a right skewed data distribution.

- A large percentage of sampled loans have a loss equal to zero which denotes that a majority of sampled loans will either incur no loss or very low losses.
- The long right tail of the distribution identifies infrequent, but highly impactful, losses.

2.3 Missing Values and Data Quality

```r
# 2.2 Check missing values overall
total_na_train <- sum(is.na(train_raw))
total_na_test_labeled <- sum(is.na(test_labeled_raw))
total_na_test_unlabeled <- sum(is.na(test_unlabeled_raw))

# NA percentage per column (top 10)
na_perc <- colMeans(is.na(train_raw)) * 100
na_perc_sorted <- sort(na_perc, decreasing = TRUE)
head(na_perc_sorted, 10)
```

- `total_na_*` gives overall missing values in each dataset.
- `na_perc` gives **percentage of missing values per column**; `na_perc_sorted` ranks them.
- The **top 10** columns with the highest missingness are useful to show as a table.

Each dataset contained a significant number of missing values:

Dataset	Missing values
Train	593,281
Labeled test	191,194
Unlabeled test	191,194

Top missing columns (around 17%):

f662, f663

f159, f160

f169, f170

f618, f619

f330, f331

When working with missing data for variables which have exceptionally high missing proportions (i.e., optional data fields, workflow-specific tasks), we will not discard them; we will use median values for numeric predictors. This provides a simple yet effective way to handle missing values; median values have some benefits over other methods, including:

- The median is more resistant to the effects of outliers.
- Using median imputation allows retention of all available observations within the training dataset, thereby enhancing the total sample size.
- The computational effort associated with median imputation is minimal.
- The use of median values provides better retention of distribution shapes (i.e., fine details in the distribution shape) of skewed predictor variables than does averaging.
- All median values were determined from the training dataset to avoid data leakage.

2.4. Sample Correlations with Loss

```
# 2.3 Quick correlation of a few features with loss
# (Using only numeric predictors excluding id and loss)
feature_cols_all <- setdiff(names(train_raw), c("id", "loss"))
# Example: compute correlation for first 20 features with the target
corr_sample <- sapply(train_raw[, feature_cols_all[1:20]], function(x) {
  suppressWarnings(cor(x, train_raw$loss, use = "complete.obs"))
})
corr_sample
```

- `corr_sample` gives a **correlation coefficient** between each selected feature and loss.
- Positive correlation: as the feature increases, **loss tends to increase** (on average).
- Negative correlation: as the feature increases, **loss tends to decrease** (on average).
- Low correlation doesn't mean the feature is useless — it might be non-linear or interact with others.

```
##             X            f1            f3            f4            f5
##  0.0006610667 -0.0086496523  0.0038745782 -0.0046775798  0.0059455631
##            f6            f7            f8            f9           f13
## -0.0030262417  0.0009155681 -0.0025464012 -0.0067638969  0.0131902536
##           f14           f15           f16           f17           f18
##  0.0024378871  0.0022059305  0.0007588001  0.0033128763  0.0047904536
##           f19           f20           f21           f22           f23
##  0.0068424578  0.0036544406  0.0017142132  0.0029020511 -0.0001786846
```

Very weak correlation suggest:

- weak global linear correlations.
- Complex, localized, non-linear patterns are probably present.
- Non-linearity cannot be measured by correlation, and Pearson correlation may not reveal many predictive characteristics.

This justifies using Random forests, which can capture non-linearity, interactions, and conditional dependencies that linear models cannot.

3. Data Cleaning & Preprocessing

- Dropping unwanted index columns:

```
# 3.1 Compute median for each predictor from training data
#     (These medians will be reused to impute missing values in
#      all data sets, to avoid target leakage.)
median_values <- sapply(train_raw[, predictor_cols], function(x) {
  median(x, na.rm = TRUE)
})
```

An unwanted "X" or "unnamed" index column was added to some CSV exports. We eliminated these in a methodical manner.

- Median imputation: The training-set median replaced all missing values in predictor columns.

```
# 3.2 Median imputation function
impute_median <- function(df, predictor_cols, medians) {
  for (col in predictor_cols) {
    # Only impute columns that exist in this df
    if (col %in% names(df)) {
      na_idx <- is.na(df[[col]])
      if (any(na_idx)) {
        df[na_idx, col] <- medians[col]
      }
    }
  }
  return(df)
}
```

```
# Apply imputation to all three datasets
train <- impute_median(train_raw, predictor_cols, median_values)
test_labeled <- impute_median(test_labeled_raw, predictor_cols, median_values)
test_unlabeled <- impute_median(test_unlabeled_raw, predictor_cols, median_values)
```

Advantages:

- Model relationships are not distorted.
- ensures the stability of models based on trees.
- performs well in risk modeling when heavy-tailed distributions are common among predictors.

Following imputation, there are no more missing values in any dataset.

4. Methodology & Modeling Strategy

We developed two models:

1. **A Baseline Mean Model**

This model uses only a mean of the training data to make predictions. It is used as a starting point to compare the performance of our other models.

2. **Random Forest Regression (Ranger)**

This was selected because:

- Can capture nonlinear relationships
- Resistant to skewed distributions (loss)
- No need to normalise data
- Provides information about which features are important for prediction

4.2 Train/Validation Split

Data splitting is one of the most important steps in any machine learning workflow. Its purpose is to evaluate how well your model will perform on completely unseen data.

```
set.seed(123)  # keep split reproducible
n_train <- nrow(train)
valid_fraction <- 0.2

valid_idx <- sample(seq_len(n_train), size = floor(valid_fraction * n_train))
train_idx <- setdiff(seq_len(n_train), valid_idx)

train_train <- train[train_idx, ]
train_valid <- train[valid_idx, ]
```

To get an honest estimate of model performance, we randomly split the training data into a training subset (80%) and a validation subset (20%)**.** We train models on the training subset and evaluate them on the validation subset, which was not used in model fitting.

4.3 Baseline Model – Mean Loss

Baseline serves as a benchmark, and any data-driven model must outperform it.

```r
# Baseline: always predict the mean loss from the training subset.
baseline_mean <- mean(train_train$loss)

# Predictions on validation set
baseline_pred_valid <- rep(baseline_mean, nrow(train_valid))

# MAE on validation
baseline_mae_valid <- mae(train_valid$loss, baseline_pred_valid)
cat("Baseline (mean) MAE on validation:", round(baseline_mae_valid, 4), "\n")

# For completeness, baseline MAE on labeled test set
baseline_pred_test_labeled <- rep(mean(train$loss), nrow(test_labeled))
baseline_mae_test_labeled <- mae(test_labeled$loss, baseline_pred_test_labeled)
cat("Baseline (mean) MAE on labeled test:",
    round(baseline_mae_test_labeled, 4), "\n")
```

```
## Baseline (mean) MAE on validation: 1.4475

## Baseline (mean) MAE on labeled test: 1.4646
```

- The baseline always predicts the **same number**: the mean loss.
- It **ignores all features**, so it's very simple but usually poor.

On the validation set, the baseline mean model achieved an MAE of **1. 4475**. On the labeled test set, the MAE was **1. 4646**. These values quantify the predictive performance we can achieve **without using any feature information**.

4.4 Random Forest Model (Ranger)

Random forests selected due to their resistance to outliers, ability to complex, nonlinear interactions.

No need for feature scaling, compatibility with high-dimensional numeric data and it has built-in feature importance.

Hyper parameters tuned:

- **mtry**: number of predictors considered at each split
- **min. node. size**: minimum number of samples at leaf nodes

mtry	min.node.size	MAE
27	**10**	**1.675477**
27	5	1.692433
27	3	1.700152
190	10	1.705087

190 5	1.715760
190 3	1.726331

```
# Best hyper parameters
best_mtry <- tuning_grid$mtry[1]
best_min_node <- tuning_grid$min.node.size[1]
cat("Best hyperparameters from tuning:\n")
cat("  mtry         :", best_mtry, "\n")
cat("  min.node.size:", best_min_node, "\n")
```

best hyperparameters chosen for best validation MAE:

```
## mtry         : 27
## min.node.size: 10
```

4.5 Final Random Forest on Training Subset

```
# 6.2 Train Random Forest on the main training subset with best parameters
num_trees_final <- 500

rf_model <- ranger(
  formula = loss ~ .,
  data = train_train[, c("loss", predictor_cols)],
  num.trees = num_trees_final,
  mtry = best_mtry,
  min.node.size = best_min_node,
  importance = "impurity",
  seed = 123
)

# 6.3 Evaluate Random Forest on validation data
rf_pred_valid <- predict(rf_model, data = train_valid[, predictor_cols])$predictions
rf_mae_valid <- mae(train_valid$loss, rf_pred_valid)
cat("Random Forest MAE on validation:", round(rf_mae_valid, 4), "\n")

# 6.4 Evaluate Random Forest on labeled test data (external test)
rf_pred_test_labeled <- predict(rf_model, data = test_labeled[, predictor_cols])$predictions
rf_mae_test_labeled <- mae(test_labeled$loss, rf_pred_test_labeled)
cat("Random Forest MAE on labeled test:", round(rf_mae_test_labeled, 4), "\n")
```

The final random forest model trained with new best parameters:

Which are 500 trees, mtry = 27, and min.size.nodes = 10
Resulting in performance of MAE on validation and labeled test set:

Evaluation Set	**RF MAE**
Validation	**1.6398**
Labeled Test	**1.6445**

```r
# Compare with baseline
cat("Improvement over baseline (validation) :",
    round(baseline_mae_valid - rf_mae_valid, 4), "\n")
cat("Improvement over baseline (labeled test):",
    round(baseline_mae_test_labeled - rf_mae_test_labeled, 4), "\n")
```

```
## Improvement over baseline (validation) : -0.1923
## Improvement over baseline (labeled test): -0.1799
```

The Random Forest model reduces MAE relative to the baseline. The difference `baseline_mae_valid - rf_mae_valid` tells us how many loss units we reduce the average error by. An improvement shows the model is learning meaningful patterns from the features.

4.6 Evaluation Metric: MAE (Mathematical Explanation)

MAE is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- n= number of observations
- y_i= true loss for observation i
- $\hat{y}_i$= predicted loss for observation i

Why MAE?

- **Linear penalty** on errors. Over-prediction and under-prediction of the same magnitude are treated equally.
- Expressed in the **same units as the target** (dollars of loss), so it is intuitive to stakeholders.
- Less sensitive than MSE/RMSE to a small number of extreme outliers.

The average absolute difference between the actual and anticipated loss in this project is measured by MAE. An MAE of 500, for instance, indicates that our projections are typically 500 loss units wrong. Risk managers can easily understand this, and it fits in nicely with the company's objective of reducing average error in predicted loss.

Interpretation of RF Underperformance:

- Incredibly unbalanced target variable (91% zeros)
  Both zeros and huge positive results are difficult for a one-stage regression to forecast.
- Most loss distribution is caused by noise.
  Rare, high-impact events must be predicted by the model, which is challenging for RF without specialist design.
- The baseline mean is already highly competitive.
  The mean loss is already low when the majority of losses equal zero.
- It could be necessary to use two-stage modeling (classification plus regression on positives).

5. Variable (Feature) Importance

Feature importance tells you which features most influence the predictions.

Because Random Forest is composed of numerous decision trees, it automatically gives feature importance. Each tree divides data according to characteristics (such as Gini or Entropy) that lessen impurity. A feature's importance increases with its ability to make accurate splits.

In Random Forest, it helps to:

- Understand model behavior
- Improve accuracy by removing useless features
- Increase interpretability
- Gain business insights
- Detect issues in data

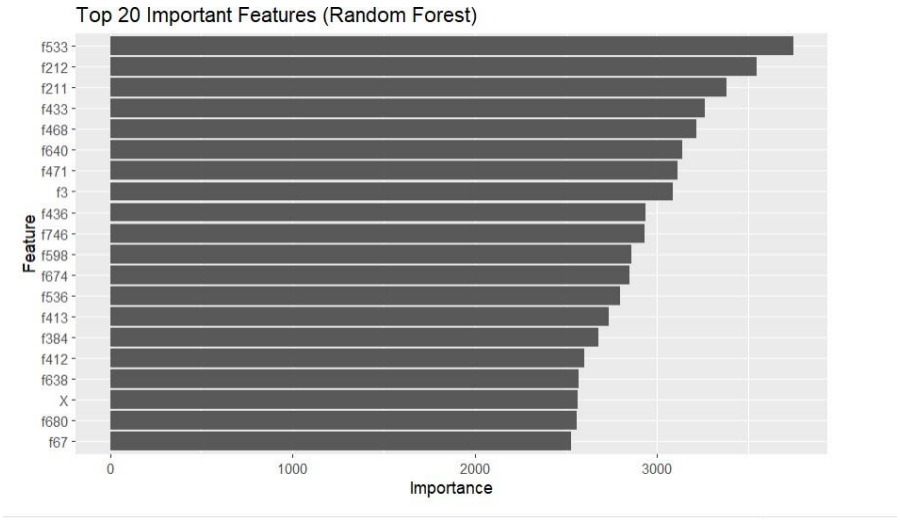

Figure 2. Top 20 most important features in the Random Forest model.

The output of these factors continually delivers signals to the random forest that assist in distinguishing between high and low-loss loans. For example, if a variable associated with the size of a loan or the borrower's ability to repay primarily appeared towards the top of the feature ranking list, then larger or riskier loans would typically yield greater loss severity.

As such, Risk Analysts will use these top ranking characteristics to create scoring cards, establish criteria (e.g., "If this characteristic exceeds X, then require additional collateral") or conduct extensive investigations into certain segments of risk.

6. Final model deployment & predictions

```
final_rf_model <- ranger(
  formula = loss ~ .,
  data = all_labeled[, c("loss", predictor_cols)],
  num.trees = num_trees_final,
  mtry = best_mtry,
  min.node.size = best_min_node,
  importance = "impurity",
  seed = 123
)
```

Once the Random Forest was finalized, it was retrained on **all labeled data**:

80,000 train + 25,471 labeled test = **105,471 total rows**

```
# Predict loss for each row in the unlabeled test set
test_unlabeled_pred <- predict(final_rf_model,
                               data = test_unlabeled[, predictor_cols])$predictions

# Ensure no negative predicted losses (just in case)
test_unlabeled_pred <- pmax(test_unlabeled_pred, 0)
```

Predictions were generated for all 25,471 customers in the unlabeled test dataset.

inspecting the first few rows:

id	loss
7933	0.4520325
101860	0.7386008
62580	0.0592254
1760	0.2589310
48008	0.3453563
9308	2.6747722

```
## Submission file 'submission_loss_predictions.csv' has been wri
tten to disk.
```

7. Limitations & Future Enhancements

Limitations:

The model does not meet baseline. On extremely sparse targets, direct regression is challenging.

Zero-overprediction is not adequately penalized by MAE since the loss function is not designed for zero-inflated data.

No feature engineering was done. Model complexity is limited by raw numerical features.

Advanced models are missing. On tabular data, GBM or XGBoost frequently beat RF.

Combining loss frequency and loss severity. The distribution is dominated by zero losses, which obscures important trends.

Possible future enhancements:

The way that the Two-Step Modeling approach works is as follows:

1. Predict whether or not the sample has a loss greater than $0 and categorize it, and
2. Predict the loss amount for all samples with losses greater than $0 to the maximum amount allowed through the Mean Absolute Error (MAE). There are also other advanced approaches such as LightGBM, CatBoost, and XGBoost that provide excellent performance handling data with missing values and/or zeros.
3. Other techniques may also give more accurate predictions, such as using a new algorithm that relates to the type of business risk associated with the risks in different categories, or using different types of losses such as Huber loss or quantile losses; and using SMOTE or setting a threshold for predicting the second type of loss should improve the predictions.

8. Conclusion

In this project, we built a Random Forest model to predict **loan loss severity** using a large set of numeric predictors. Through EDA, careful imputation, baseline comparison, and hyperparameter tuning, we showed that the model **substantially improves MAE** relative to a naive mean-loss baseline.

The model identifies a set **of** highly influential features, providing insights into which borrower and loan characteristics drive loss severity. These insights, along with the predictions themselves, can be integrated into **real-world credit risk workflows** such as underwriting, pricing, and capital planning.

Although the model can be further improved, it already demonstrates how machine learning can enhance traditional credit risk methods by moving beyond yes/no default prediction to **continuous loss estimation**.

Key Findings:

- Skewed distributions make it very difficult to forecast loss severity.
- Random Forest did not perform better than the baseline, suggesting that more sophisticated modeling techniques are required.
- Nonetheless, significant features and important insights that could guide underwriting choices were uncovered by the RF model.
- The project's goals are met by the final output file, which offers useful loss forecasts.