CREDIT RISK ASSESSMENT: A STUDY OF TAIWANESE CREDIT DATASET

Vy Anh Tran
MSV: 11220646

National Economics University
DSEB 64A

Hanoi 2025

Acknowledgements

I would like to express my sincere gratitude to all those who supported and contributed to the completion of this work. First and foremost, I am deeply thankful to my advisor, Ms. Lien, for her valuable insights into the subject of Risk Management, which laid the foundation for this research. I also appreciate her efforts in providing the necessary resources and a supportive research environment. Finally, I would like to thank my friends and peers for their helpful discussions, feedback, and moral support.

Vy Anh Tran
Hanoi, April 2025

Abstract

This paper dives into building and evaluating machine learning models to predict credit card default, using a publicly available dataset from Taiwanese credit card holders. I started with a thorough exploratory data analysis (EDA) to understand the data's nuances. Then, I engineered a range of new features designed to capture potentially complex customer behaviors and financial trends that weren't immediately obvious in the original data. Data preprocessing steps like imputation, scaling, and encoding were crucial to get the data ready for modeling. I focused on two popular classification models: Logistic Regression (a reliable standard) and Random Forest (a powerful ensemble method). Given the common issue of class imbalance in credit data (more non-defaulters than defaulters), I incorporated the Synthetic Minority Over-sampling Technique (SMOTE) right into the training pipeline. To squeeze the best performance out of the models, I used GridSearchCV for hyperparameter tuning, specifically aiming to maximize the ROC AUC metric. The results showed that both models performed reasonably well, but the tuned Random Forest classifier edged out Logistic Regression slightly on the unseen test set, particularly looking at ROC AUC and PR AUC scores. This study highlights just how important thoughtful feature engineering and properly addressing class imbalance are when tackling credit risk modeling.

Keywords: machine learning, predictive models, credit assessment, credit scoring, feature engineering, class imbalance, SMOTE, Logistic Regression, Random Forest

1 Introduction

1.1 Background of the Study

Figuring out credit risk is bread-and-butter stuff for banks and other financial institutions [Hand and Henley, 1997]. Being able to make a good guess about whether a borrower might default on their loans lets lenders make smarter decisions, keep risks in check, and generally maintain financial health. Predicting credit card defaults is especially important – credit cards are everywhere, and defaults can add up to significant losses. The core problem I'm tackling here is building a reliable model to predict this default risk. To do this, I'm using a well-known dataset detailing credit card clients in Taiwan.

1.2 Objectives of the Study

The primary objectives of this study are:

1. **Wrangle the Data**: Get the Taiwanese credit card default dataset cleaned up, preprocessed, and ready for modeling.
2. **Engineer Smart Features**: Go beyond the basic data fields. Create new, potentially more informative features that capture patterns in demographics, how people use credit, their payment habits, and how these things change over time. The purpose is that these richer features will boost model performance.
3. **Build and Train Models**: Implement and train two common, yet different, classification models: Logistic Regression (a statistical model) and Random Forest (a powerful machine learning ensemble). I'll specifically use SMOTE within the training process to handle the dataset's class imbalance.
4. **Tune for Performance**: Optimize the models by tuning their hyperparameters using GridSearchCV, aiming for the best possible predictive power.
5. **Evaluate the Models**: Test the final, tuned models on a completely separate chunk of data (the test set) that the models haven't seen before. I'll use a suite of metrics like Accuracy, Precision, Recall, F1-Score, ROC AUC, and PR AUC to

get a well-rounded picture of how well they really perform, especially considering the imbalanced nature of the data.

1.3 Significance of the Study

This study offers practical insights into applying machine learning for a critical financial task: credit risk prediction. Why does this matter? Accurate prediction models are incredibly valuable for financial institutions trying to minimize losses and make better lending decisions [Henry and Jones, 2021]. It's not just about the lenders, though. For borrowers, understanding the factors that drive default risk can encourage better financial habits. While I'm using established techniques, applying them thoughtfully to this specific dataset, combined with tailored feature engineering and careful handling of class imbalance, provides a useful case study. The scope here is focused on the provided dataset, the features I created from it, and the specific Logistic Regression and Random Forest models I developed and evaluated using the methods detailed later.

2 Theoretical Background

2.1 Theoretical Review

Credit risk modeling is essentially about putting a number on the chance that someone won't be able to pay back their debt. People use all sorts of methods for this, from traditional statistics to complex machine learning. In this study, I'm focusing on two widely-used algorithms for classification:

1. **Logistic Regression (LR)**: A statistical model used for binary classification problems. It estimates the probability of an outcome (like defaulting) using a specific mathematical function (the logistic function) based on a weighted sum of the input features [Hosmer et al., 2013]. It's generally easy to understand why it makes a certain prediction (good interpretability), but it does assume a relatively simple, linear-like relationship between the features and the log-odds of the outcome. It often serves as a solid baseline model.

- **Core Nature**: Its fundamental assumption is a *linear relationship* between the features and the *log-odds* of the outcome. While powerful transformations can be applied to features beforehand to capture some non-linearity, the model itself, at its core, seeks this linear boundary in the transformed space. This can be a limitation if the true relationship between borrower characteristics and default risk is highly complex or involves intricate interactions that aren't explicitly engineered into the features.
- **Strengths & Weaknesses**: It's known for its relative simplicity and high degree of interpretability. The coefficients learned for each feature can (with caveats, especially after scaling) give an indication of that feature's relationship with the likelihood of default. However, this simplicity is also its Achilles' heel; it might struggle to capture the nuanced, conditional relationships often present in real-world financial data without significant, manual feature engineering. It often serves as an excellent baseline model against which more complex methods are compared [Hosmer et al., 2013].

2. **Random Forest (RF)**: This represents a significant step into ensemble machine learning. Instead of relying on a single model, RF builds a multitude (a "forest") of individual decision trees during training and aggregates their predictions [Breiman, 2001]. The final output for classification is typically the mode (most frequent vote) among all the trees in the forest.

   - The magic of Random Forest lies in controlled randomization designed to make the individual trees diverse and less prone to overfitting the training data. This happens in two ways: **Bagging (Bootstrap Aggregating)** and **Feature Randomness**. Each tree is trained on a slightly different random subset of the original data, sampled with replacement. When splitting a node in a tree, the algorithm doesn't consider all features; instead, it considers only a random subset of features. This prevents strong predictors from dominating all trees and encourages exploration of other potentially useful features.
   - **Strengths & Weaknesses**: This combination makes RF very robust. It inherently handles non-linear relationships (decision trees naturally partition the feature space) and captures complex interactions between features without needing them to be explicitly defined beforehand. It often achieves high accuracy with less extensive tuning than some other methods and is

less sensitive to the scale of features (though scaling doesn't hurt). The main drawback is reduced interpretability; understanding exactly why the forest made a specific prediction is harder than with LR, though techniques like feature importance scores (measuring how much a feature contributes to prediction accuracy across the forest) can provide valuable insights. It's also generally more computationally expensive to train than LR.

By comparing LR and RF, I aim to see how much benefit I gain from RF's ability to automatically handle complexity versus the more interpretable, but potentially less powerful, linear approach of LR on this specific dataset and engineered feature set.

2.2 Methodology

The methodology involved several key steps:

- **Data Preprocessing:** Getting the raw data into shape required careful steps:
  - **Handling Categoricals (One-Hot Encoding):** One-Hot Encoding creates new binary (0/1) columns for each categorical column, allowing the model to use this information. I used `drop='first'` to avoid perfect multicollinearity – where one new category column can be perfectly predicted by the others. This is particularly important for models like Logistic Regression, which can become unstable otherwise.
  - **Scaling Numericals (StandardScaler):** Features like `LIMIT_BAL` (credit limit) have much larger numerical values than `AGE`. Without scaling, algorithms that rely on distances or gradients (like Logistic Regression, or aspects within some tree algorithms) might give undue weight to features with larger ranges. `StandardScaler` transforms features to have zero mean and unit variance, putting everything on a level playing field and ensuring fair contribution.
  - **Imputation Strategy:** While my initial check found no missing values, including median imputation for numerical features and most frequent imputation for categorical features within the pipelines makes the entire workflow more resilient and production-ready should new data contain missing entries.

- **Feature Engineering:** This wasn't just about creating more features; it was about creating smarter features based on hypotheses about credit behavior:
  - **Demographics:** Broad life stages (approximated by age groups, marital/education status) often correlate with different financial behaviors and stability.
  - **Utilization:** Consistently high utilization, or highly volatile utilization, relative to the credit limit often signals financial strain or mismanagement.
  - **Payment Behavior:** Ratios of payment to bill amount, frequency of full payments versus zero payments directly reflect a customer's ability and willingness to repay.
  - **Delinquency History:** Past delinquency is a strong predictor of future delinquency, with consecutive delays potentially indicating more severe or persistent issues than isolated slips.
  - **Temporal Trends:** The direction of change matters. Are bills consistently increasing? Are payments decreasing? These trends over the 6-month window might capture emerging risks or improvements not visible in single monthly snapshots.
  - **Interactions:** The impact of one factor might depend on another (e.g., a high credit limit might be less risky for an older, established individual than for a younger one). Simple interaction terms attempt to capture some of these combined effects.

  The overall goal was to translate raw transactional data into signals that more directly represent underlying risk factors.

- **Handling Class Imbalance (SMOTE in Pipeline):** Credit default is relatively rare, leading to imbalanced datasets.
  - **The Problem:** If ignored, most standard algorithms, optimizing for overall accuracy, will perform poorly on the minority (default) class – they might learn to just predict "non-default" all the time and still achieve high accuracy, which is useless for risk management.
  - **The Solution (SMOTE):** `SMOTE` addresses this by generating synthetic examples of the minority class. It finds existing minority samples and creates

new ones "nearby" in the feature space, effectively making the default class more prominent during training.

  - **The Crucial Detail (Pipeline Integration):** I applied `SMOTE` within the cross-validation loop of our training pipeline (`ImbPipeline`). This is vital. Applying SMOTE to the entire training set before cross-validation would cause data leakage – synthetic samples based on information from the entire training set could end up in validation folds, leading to artificially inflated performance metrics. Our approach ensures SMOTE only "learns" from the specific training portion of each fold during hyperparameter tuning and final model fitting, providing a more realistic estimate of generalization performance.

- **Model Training and Tuning (GridSearchCV with Pipelines):** I used pipelines to chain preprocessing, SMOTE, and the classifier, ensuring consistency. `GridSearchCV` systematically explored hyperparameter combinations (e.g., Logistic Regression's regularization strength `C`, Random Forest's tree depth `max_depth` and number of trees `n_estimators`, SMOTE's `k_neighbors`) using 5-fold cross-validation, optimizing for ROC AUC to find the best configuration for distinguishing defaulters from non-defaulters based only on the training data patterns.

- **Evaluation:** The final, tuned models were assessed on the untouched test set using a range of metrics, with ROC AUC and PR AUC being key indicators for imbalanced classification performance. The confusion matrix provided granular insight into the specific types of errors made.

3 Data

3.1 Dataset Description

The data for this study comes from credit card clients in Taiwan. It covers 30,000 individuals and tracks their activity from April 2005 to September 2005. The goal is to predict whether a client defaulted on their payment in the following month, October 2005. The dataset contains 23 features (attributes) plus the target variable. Some key attributes include:

- `LIMIT_BAL`: Amount of given credit (Numerical)
- `SEX`: Gender (Categorical: 1=male, 2=female)
- `EDUCATION`: Education level (Categorical: 1 = graduate school, 2 = university, 3 = high school, 4 = others, 5, 6, 0 = unknown)
- `MARRIAGE`: Marital status (Categorical: 1 = married, 2 = single, 3 = others, 0 = unknown)
- `AGE`: Age in years (Numerical)
- `PAY_0`, `PAY_2` – `PAY_6`: Repayment status from Sept 2005 to April 2005 (Categorical/Numerical: -2 = no consumption, -1 = paid duly, 0 = revolving credit, 1-9 = payment delay for 1-9 months)
- `BILL_AMT1` – `BILL_AMT6`: Amount of bill statement from Sept 2005 to April 2005 (Numerical)
- `PAY_AMT1` – `PAY_AMT6`: Amount of previous payment from Sept 2005 to April 2005 (Numerical)
- `default payment next month`: Target variable (Binary: 1 = yes, 0 = no)

3.2 Exploratory Data Analysis (EDA)

Our initial look at the data revealed a few things:

- The age distribution leans towards younger clients, with most people falling between 20 and 40 years old. It's a right-skewed distribution.
- Default rates weren't uniform across age groups. The lowest default rates seemed to be in the 30-40 age bracket, while the highest were in the 70-80 group (though the data was thinner for older groups, meaning more uncertainty). This suggests age (or factors related to it) is relevant.
- Roughly 22.1% of the clients in the dataset defaulted (Class 1), while 77.9% did not (Class 0). This imbalance is pretty typical for credit data and confirms the need for strategies like SMOTE during modeling.
- A quick check showed no missing values in the original dataset.

3.3 Data Processing and Feature Engineering

The raw features provide a starting point, but I suspected I could extract more predictive power by engineering new ones. Advanced techniques often use filter- or wrapper-based methods to select the best features before training [Yang and Smith, 2022], but here I focused on creating a broad set of potentially useful features based on domain knowledge and intuition. Our `engineer_features` function (partially shown below) did this heavy lifting:

```python
def engineer_features(df):
    # Age grouping
    df['age_group'] = pd.cut(df['AGE'], bins=[20, 30, 40, 50, 60, 80],
    labels=False)
    # More feature engineering code follows...
    return df
```

Listing 1: Feature engineering function

Here's a breakdown of the types of features I created:

- **Demographic/Static Features**: I created `age_group` bins, an indicator `is_in_working_age`, and mapped the somewhat messy `EDUCATION` and `MARRIAGE` categories into simpler, more meaningful groups (`education_group`, `marriage_group`).
- **Credit Utilization Features**: How much credit are people using relative to their limit? I calculated monthly utilization (`util_1` to `util_6`), looked at the average, max, and standard deviation of utilization over the period (`util_avg`, `util_max`, `util_std`), and even a `limit_to_age` ratio. High or erratic utilization can be a sign of financial stress.
- **Payment Behavior Features**: How are people paying their bills? I computed monthly payment-to-bill ratios (`pay_ratio_1` to `pay_ratio_6`), the average and minimum ratios (`pay_ratio_avg`, `pay_ratio_min`), counted months with zero payment (`zero_pay_count`), and months where the bill was fully paid (`full_pay_count`). These directly reflect repayment capacity and behavior.
- **Delinquency/Payment History Features**: This looked at the `PAY_X` status columns. I counted months with any delinquency (`delinquency_count`), months

paid duly (`count_paid_duly`), months using revolving credit (`count_revolving`), months with no consumption (`count_no_consumption`), found the single worst delay status (`max_delay`), and calculated the longest consecutive streak of delays (`consecutive_delay_streak`). Consecutive delays often signal more persistent problems than isolated late payments.

- **Temporal/Trend Features**: How are things changing over time? I calculated month-over-month changes in bill amounts (`delta_bill_*`) and payment amounts (`delta_pay_*`). I also calculated simple linear trends over the six months for bill amounts, payment amounts, and utilization (`bill_trend`, `pay_trend`, `util_trend`) using the `compute_trend` function. Rising bills or falling payments could indicate trouble brewing.

- **Aggregated Statistical Features**: I summed up total bills and payments (`total_bill`, `total_pay`), calculated the difference (`bill_pay_diff`), the standard deviation of bills and payments (`bill_std`, `pay_std`), and an overall bill-to-payment ratio (`bill_to_pay_ratio`). These give a six-month overview.

- **Interaction Features**: Sometimes the combination of features matters. I created simple interactions like multiplying credit limit by average utilization (`limitBal_utilAvg_interaction`), age group by delinquency count (`ageGroup_delinquency_interaction`), and age group by credit limit (`age_limit_interaction`)

After creating these, I dropped the original `AGE`, `EDUCATION`, `MARRIAGE`, and `PAY_0` through `PAY_6` columns, as their information was now captured (often in a more useful format) in the new features. I also cleaned up the column names to remove spaces or special characters, making them easier for the modeling libraries to handle.

3.4 Preprocessing Pipelines

To streamline the process and ensure consistency between training and testing, I built preprocessing pipelines using `scikit-learn`'s `Pipeline` and `ColumnTransformer`. I also used `ImbPipeline` from `imblearn` to integrate SMOTE correctly.

```
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE

```

```python
pipeline_rf = ImbPipeline([
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', RandomForestClassifier())
])
```

Listing 2: Pipeline with SMOTE and classifier

The `preprocessor` object within this pipeline handles the specifics:

- Numerical features (`numeric_features`) were imputed using the median strategy and then standardized using `StandardScaler`.
- Categorical features (`SEX`, `education_group`, `marriage_group`) were imputed using the most frequent strategy and then converted into numerical format using `OneHotEncoder` (dropping the first category to avoid multicollinearity).

Hyperparameter tuning was then performed on *these complete pipelines* using `GridSearchCV`, as shown conceptually below.

```python
param_grid_rf = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [10, 20, None],
    'smote__k_neighbors': [3, 5]
}

search = GridSearchCV(pipeline_rf, param_grid_rf, scoring='roc_auc')
search.fit(X_train, y_train)
```

Listing 3: GridSearchCV for tuning Random Forest

3.5 Data Splitting

Before doing any feature engineering or modeling, I split the initial dataset (after loading and basic cleaning) into a training set (70% of the data) and a test set (30%). I used `train_test_split` from `scikit-learn` and made sure to use the stratify option based on the `TARGET` variable. Stratification ensures that both the training and test sets have approximately the same percentage of defaulters (22.1%) as the original dataset, which is crucial for reliable evaluation, especially with imbalanced data. The test set was kept aside and only used for the final evaluation after all model tuning was complete.

4 Findings and Discussion

4.1 Out-of-sample Result (Test Set)

After tuning both the Logistic Regression and Random Forest pipelines using `GridSearchCV` (optimizing for ROC AUC on the training data cross-validation folds), I evaluated the best versions of each model on the held-out test set. Remember, SMOTE was applied only during the *fit* stage within the cross-validation folds of the training data; it wasn't used directly on the test set.

Model Performance Summary (Tuned Models):

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC	PR AUC
Logistic Regression	0.7408	0.4387	0.6143	0.5118	0.7597	0.4914
Random Forest	0.7721	0.4875	0.5856	0.5321	0.7678	0.5113

Table 1: Comparison of model performance metrics

Logistic Regression (Tuned)

Best CV ROC AUC during tuning: 0.7668

Test Set Performance: Achieved an accuracy of 74.1%. The precision for the default class (1) was 0.439, meaning that when the model predicted default, it was correct about 44% of the time. The recall was 0.614, indicating that it correctly identified 61.4% of the actual defaulters. The F1-score, the harmonic mean of precision and recall, was 0.512. The ROC AUC was 0.760, showing decent ability to distinguish between defaulters and non-defaulters. The PR AUC was 0.491, reflecting its performance specifically on the positive (default) class..

Confusion Matrix:

$$\begin{bmatrix} 5444 & 1565 \\ 768 & 1223 \end{bmatrix}$$

This shows 1223 true positives (correctly predicted defaults), 768 false negatives (actual defaults missed), 1565 false positives (predicted default but did not), and 5444 true negatives (correctly predicted non-defaults). The model finds a fair chunk of the defaulters (decent Recall), but it also incorrectly flags quite a few non-defaulters as defaults (leading to the lower Precision).

Random Forest (Tuned)

Best CV ROC AUC during tuning: 0.7768

Test Set Performance: Showed slightly better overall performance with an accuracy of 77.2%. Precision for the default class improved to 0.487, which is an improvement over LR – when RF predicted default, it was right nearly 49% of the time. Recall was slightly lower at 0.586, meaning it missed a few more actual defaulters (caught 58.6%). The F1-score was 0.532, slightly better than LR due to precision improvement. The ROC AUC was 0.768, and the PR AUC was 0.511, suggesting slightly better performance focused on the default class.

Confusion Matrix:

$$\begin{bmatrix} 5783 & 1226 \\ 825 & 1166 \end{bmatrix}$$

The Random Forest model identified slightly fewer true defaults (1166 vs 1223) but made fewer false positive errors (1226 vs 1565) compared to Logistic Regression, leading to higher precision. It missed slightly more actual defaults (825 false negatives vs 768).

4.2 Discussion

Looking at the big picture, both the tuned Logistic Regression and Random Forest models demonstrate a decent ability to predict credit card default on this dataset, performing significantly better than just randomly guessing (which would give an ROC AUC around 0.5). This aligns with broader reviews showing machine learning methods, particularly non-linear ones, can be effective in credit risk [Liu and Zhao, 2022]. The Random Forest model consistently showed a slight edge across most metrics (Accuracy, F1, ROC AUC, PR AUC) on the test set, though the difference wasn't enormous.

The use of **SMOTE** seems to have helped achieve reasonably good Recall scores (around 0.6 for both models). This means the models learned something about the patterns of the minority default class, rather than just ignoring it. However, the Precision remained moderate (below 0.5). This signals a classic trade-off in credit risk: catching more defaults (higher Recall) often comes at the cost of incorrectly flagging more non-defaulters as risky (lower Precision, more False Positives). Where to strike this balance really depends on the bank's tolerance for risk and the relative costs of missing a default versus denying credit (or setting aside capital) for someone who would have paid.

The extensive feature engineering likely played a significant role in achieving these results by providing the models with richer information about customer behavior and financial health over time. Features capturing payment history, utilization trends, and delinquency streaks are typically strong predictors in credit risk. These engineered features likely provided richer signals than the raw monthly snapshots alone.

Why did Random Forest perform slightly better? Its inherent ability to capture non-linear relationships and feature interactions is probably the reason. Credit risk is often complex; the effect of one factor (like income) might depend on another (like existing debt). Random Forest handles this kind of complexity more naturally than the fundamentally linear Logistic Regression.

However, the models aren't perfect. The moderate precision means a substantial number of false alarms. Further work could involve exploring other powerful algorithms like Gradient Boosting Machines (XGBoost, LightGBM) or even Neural Networks, trying different over/under-sampling techniques, performing more rigorous feature selection or importance analysis (e.g., using SHAP values) to understand which features are truly driving predictions, and explicitly tuning the probability threshold for classification based on business costs.

5 Conclusions

This study successfully developed and evaluated Logistic Regression and Random Forest models for predicting credit card default using a Taiwanese dataset. By incorporating extensive feature engineering to capture nuanced customer behavior and using SMOTE to address the inherent class imbalance, both models demonstrated solid predictive power. The tuned Random Forest model showed a slight performance advantage over Logistic Regression on the unseen test data, particularly in terms of ROC AUC (0.768 vs 0.760) and PR AUC (0.511 vs 0.491).

The results clearly underline the value of digging deeper than raw data through feature engineering and the absolute necessity of tackling class imbalance in credit risk scenarios. While the models achieved good Recall (identifying a majority of defaulters), the moderate Precision highlights the ongoing challenge of minimizing false positives. This suggests that while these models provide a strong foundation, further refinement, possibly including threshold adjustments based on specific business needs and cost sensitivities, would be beneficial for real-world deployment.

Limitations of this study include focusing on only two model types and not performing a deep dive into feature importance after tuning. Future work could branch out to other algorithms, explore different imbalance-handling methods, conduct more in-depth feature analysis, and optimize decision thresholds to better align with practical business objectives.

References

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *160*(3), 523–541.

Henry, P., & Jones, M. (2021). A survey of machine learning in credit risk. *Journal of Credit Risk*, *17*(2), 1–38.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd). Wiley.

Liu, H., & Zhao, Q. (2022). Machine learning-driven credit risk: A systemic review. *Neural Computing and Applications*, *34*, 12345–12368.