
Credit Risk Modeling & Loan Default Prediction

An End-to-End Learning-First Business Summary

Purpose of This Document
This document is written as a learning-oriented business and technical explanation of an end-to-end credit risk modeling project. It is intended to demonstrate not only final results, but also the reasoning, mathematics, and decision-making process behind each modeling choice.

---

1. What Is This Problem? (From Absolute Zero)

A bank gives money to people as loans.
Some people **pay the loan back fully**.
Some people **stop paying**.

When a person stops paying, the bank **loses money**.

So the bank asks one very important question **before giving a loan**:

"How risky is this person?"

This project builds a system that **estimates that risk using data**.

We are not trying to predict the future perfectly.
We are trying to **estimate probability**, so that **humans can make better decisions**.

This problem is called **credit risk modeling**.

---

2. What Does "Default" Mean?

A loan can end in two clear ways:

- **Fully Paid** → the borrower paid everything back
- **Charged Off** → the bank accepts that the money will not be recovered

We define:

- **Default = 1** → Charged Off
- **Default = 0** → Fully Paid

Loans that are still running ("Current") are **removed**, because their outcome is unknown.

This is extremely important:

We only train models on loans where the truth is already known.

---

3. Understanding the Dataset (Before Any Code)

Each **row** in the dataset = one loan application
Each **column** = information known at approval time

Examples:

- Income
- Existing debt
- Credit score
- Loan size
- Employment history

These are the **signals** a bank uses to judge risk.

---

4. Why Raw Numbers Are Not Enough

Raw numbers alone can be misleading.

Example:

- Person A earns ₹50,000 and takes ₹10,000 loan
- Person B earns ₹1,00,000 and takes ₹10,000 loan

Same loan amount, **very different risk**.

Banks therefore think in **ratios**, not raw values.

This idea drives feature engineering.

---

5. Feature Engineering: Turning Life Into Numbers

Feature engineering means:

Converting real-world financial behavior into meaningful numerical signals

Below are the final features used, explained clearly.

5.1 Loan Amount (loan_amnt)

This is the size of the loan.

By itself, it does not define risk.
A large loan can be safe if income is high.

So it is kept, but **not trusted alone**.

5.2 Loan Term (term)

The duration of the loan:

- 36 months
- 60 months

Longer loans are riskier because:

- More uncertainty
- More time for negative life events

5.3 Interest Rate (int_rate)

This is extremely important.

Banks **already charge higher interest to risky borrowers**.

So interest rate acts like:

A summary of the bank's own risk judgment

That is why it becomes the strongest predictor.

5.4 Annual Income (annual_inc)

Income tells how much money comes in.

But income alone is not enough — obligations matter.

So income is most useful when combined with ratios.

5.5 Debt-to-Income Ratio (dti)

This answers:

"How much of the income is already used to pay debts?"

Higher DTI = higher stress = higher risk.

This is a core banking feature.

5.6 Credit Utilization (revol_util)

This measures how much of available credit is already used.

High utilization often means:

- Living close to limits
- Less financial flexibility

Values are capped at **100%**, because anything above already indicates stress.

5.7 Employment Length (emp_length)

This represents job stability.

Longer employment usually means:

- More stable income
- Lower risk

5.8 Credit Score (fico_avg)

Credit score summarizes past repayment behavior.

Higher score = lower risk.

We combine low and high values into a single average to:

- Reduce redundancy
- Improve interpretability

5.9 Loan-to-Income Ratio (loan_to_income)

This answers:

“Is the loan too large for what the person earns?”

This is one of the **strongest behavioral signals**.

---

6. Why Logistic Regression Is Used (Theory + Intuition)

6.1 The Core Question

We want: **The probability that a borrower will default**

Not just yes or no.

6.2 Logistic Regression Idea

Logistic regression is used as the first model because it is interpretable, regulator-friendly, and commonly deployed in real banking systems.

First, the model computes:

[
z = b_0 + b_1x_1 + b_2x_2 + \dots
]

Then converts it into probability using the sigmoid function:

[
P(default) = \frac{1}{1 + e^{-z}}
]

This guarantees:

- Output between 0 and 1
- Smooth probability curve
- Interpretability

6.3 Interpreting Coefficients

- Positive coefficient → increases risk
- Negative coefficient → reduces risk

This transparency is why banks trust logistic regression.

---

7. Why Scaling Is Required

Features are on different scales:

- Income in thousands
- Ratios near zero

Scaling ensures:

- Fair comparison
- Stable learning
- Meaningful coefficients

---

8. Model Output

The model outputs **probabilities**, for example:

- 0.12 → low risk
- 0.65 → high risk

Probabilities are later converted into decisions using thresholds.

---

9. Cost of Mistakes (Most Important Business Idea)

False Negative (Very Dangerous)

- Model says “safe”
- Borrower defaults

- Bank loses money

False Positive (Less Dangerous)

- Model says “risky”
- Borrower would have paid
- Bank loses opportunity

Banks prefer:

Rejecting many good borrowers over approving one bad borrower

---

10. Threshold Selection (Decision Logic)

The model gives probabilities.
The bank must decide:

“Above what probability do we reject the loan?”

This is the **threshold**.

Lower threshold:

- Safer
- More rejections

Higher threshold:

- More approvals
- More losses

---

11. Metrics Explained Simply

Recall (for default)

“Out of all defaulters, how many did we catch?”

This is the **most important metric**.

Precision (for default)

“Out of those flagged risky, how many truly defaulted?”

This affects business volume.

ROC-AUC

“How well does the model rank risky vs safe borrowers?”

A value of ~0.70 indicates strong separation.

---

12. Programmatic Threshold Selection

Instead of guessing, we impose a **business rule**:

- Minimum default recall = **94%**

Among thresholds satisfying this rule, we choose the one with **highest precision**.

This leads to:

Final threshold = 0.3

This choice is **coded**, not subjective.

---

13. Final Business Outcome

At threshold = 0.3:

- ~95% of defaulters are caught
- Very few bad loans are approved
- Many good borrowers are rejected intentionally

This reflects a **conservative lending strategy**.

---

14. How Banks Use This Model

1. Model generates probability
2. Threshold applied
3. Risky cases flagged
4. Human credit officer reviews
5. Final decision made

The model **assists**, not replaces humans.

---

15. Final Conclusion

This project shows:

- How raw data becomes risk signals
- How math converts signals into probabilities
- How business logic converts probabilities into decisions

Good models are not about perfect accuracy, but correct trade-offs.

LINE-BY-LINE CODE EXPLANATION

Exploratory Data Analysis (EDA)

import pandas as pd

import numpy as np

Loads libraries for data handling and mathematics.

---

df = pd.read_csv("../data/raw/lendingclub.csv")

Reads the dataset into memory as a table.

---

df.head()

Displays first few rows to understand structure.

---

df["loan_status"].value_counts()

Counts loan outcomes to understand labels.

Feature Engineering

```
use_cols = [
    "loan_status","loan_amnt","term","int_rate",
    "annual_inc","dti","fico_range_low",
    "fico_range_high","revol_util","emp_length"
]
```

df = pd.read_csv("../data/raw/lendingclub.csv", usecols=use_cols)

Loads only required columns for efficiency and safety.

---

df = df[df["loan_status"].isin(["Fully Paid","Charged Off"])].copy()

Keeps only loans with known outcomes.

---

df["default"] = np.where(df["loan_status"]=="Charged Off",1,0)

Creates numeric target variable.

---

df["term"] = df["term"].str.extract(r"(\d+)").astype(int)

Extracts numeric loan duration from text.

---

df["fico_avg"] = (df["fico_range_low"] + df["fico_range_high"]) / 2

df = df.drop(columns=["fico_range_low","fico_range_high"])

Creates a single credit score feature.

---

df.loc[df["annual_inc"]==0,"annual_inc"] = np.nan

Prevents division by zero.

---

df["loan_to_income"] = df["loan_amnt"] / df["annual_inc"]

Creates a key behavioral ratio.

---

df["revol_util"] = df["revol_util"].clip(upper=100)

df["revol_util"] = df["revol_util"].fillna(df["revol_util"].median())

Cleans utilization values.

```
df["emp_length"] = df["emp_length"].replace({
    "10+ years":10,"< 1 year":0,
    "1 year":1,"2 years":2,"3 years":3,
    "4 years":4,"5 years":5,
    "6 years":6,"7 years":7,
    "8 years":8,"9 years":9
})
```

Converts employment length to numeric stability signal.

---

df.to_csv("../data/processed/model_data.csv",index=False)

Saves final dataset for modeling.

Modeling

X = df.drop(columns=["default"])

y = df["default"]

Separates inputs and target.

X_train,X_test,y_train,y_test = train_test_split(

X,y,test_size=0.25,stratify=y,random_state=42

)

Simulates unseen future data.

---

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

Normalizes features.

---

model = LogisticRegression(

max_iter=1000,

class_weight="balanced"

)

model.fit(X_train_scaled,y_train)

Trains probability-based risk model.

---

y_test_proba = model.predict_proba(X_test_scaled)[:,1]

Produces default probabilities.

Evaluation & Thresholds

y_pred = (y_test_proba >= threshold).astype(int)

Converts probabilities into decisions.

---

confusion_matrix(y_test,y_pred)

Measures decision outcomes.

```
eligible = results[results["recall_default"] >= 0.94]
best_threshold = eligible.sort_values(
    "precision_default",ascending=False
).iloc[0]
```

Selects threshold using business rules.

```
FINAL_THRESHOLD = best_threshold["threshold"]
```

Locks production threshold.
