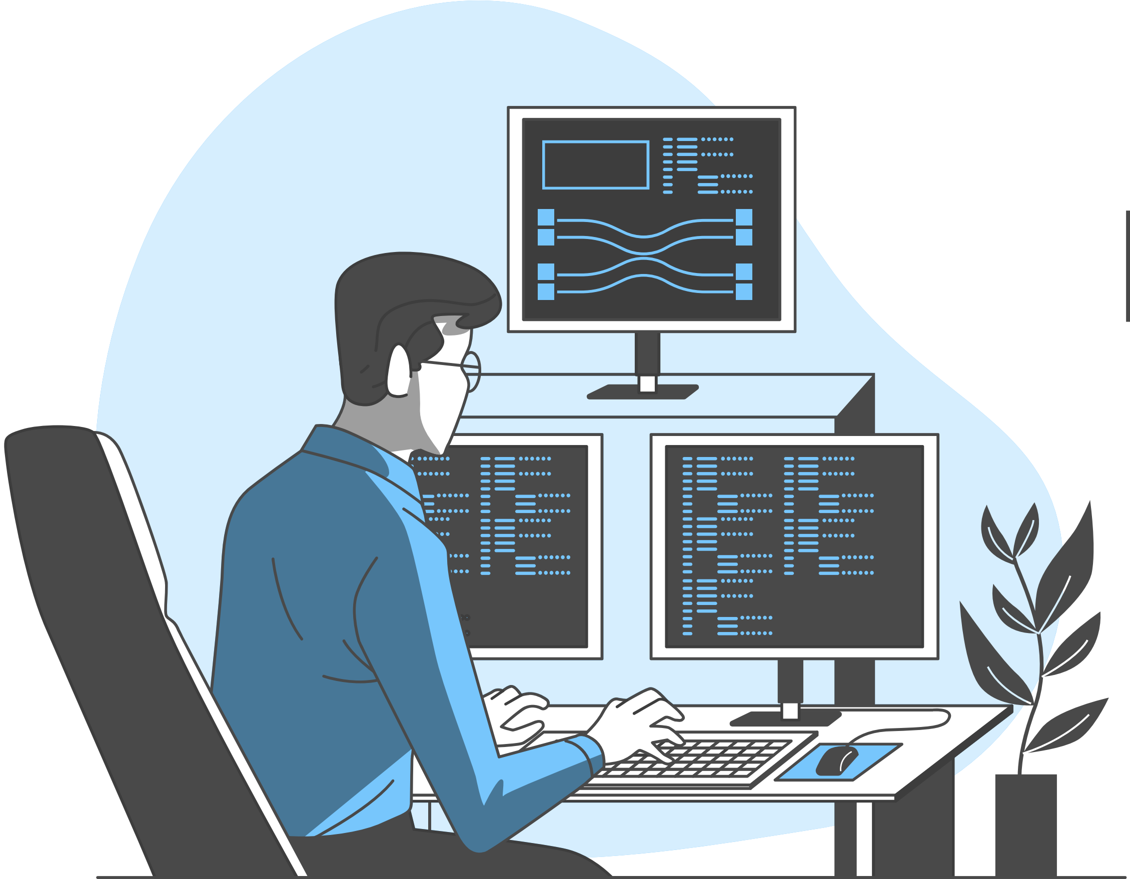


ML PROJECT

Loan Default Prediction

Nod 2025

Jonathan Avigdor

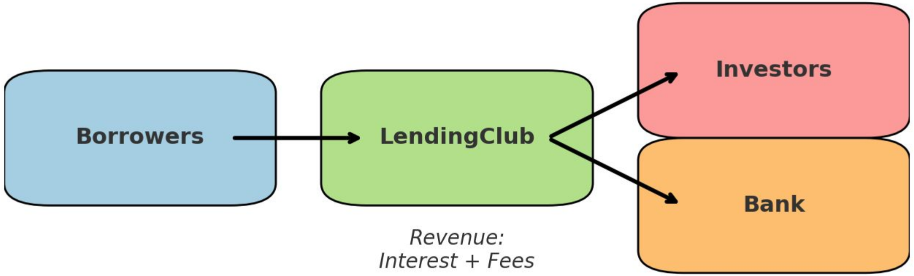
Intro – Lending Club

LendingClub

- U.S. fintech, founded 2006
- Provides personal loans
- Works as a digital bank & marketplace

Business model

- Borrowers take loans
- Bank or investors fund them
- Revenue = **interest + fees**

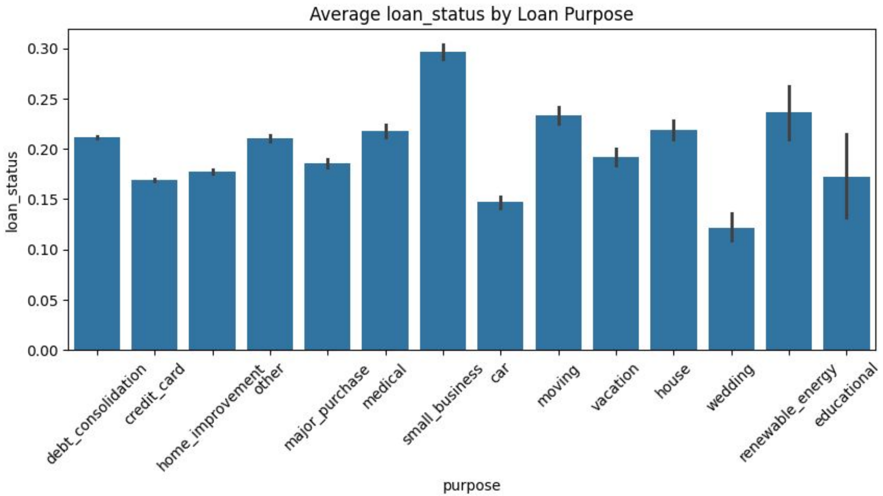


Data Set

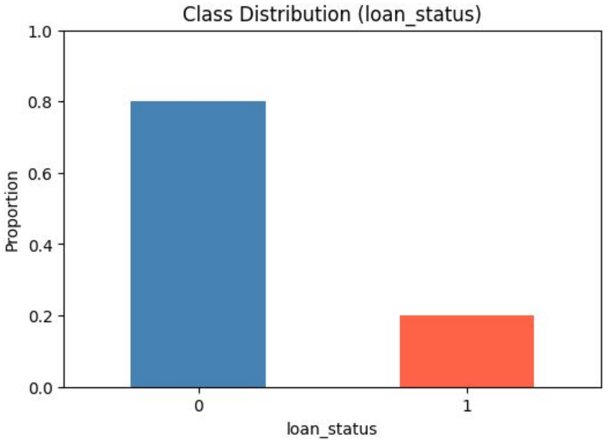
The dataset covers **2007–2020**, with **2.2M loans and 151 features**.
It includes:

- **Borrower information**: employment length, credit history length, annual income, loan grade
- **Loan terms**: loan period, loan amount, interest rate, monthly installment
- **Outcomes**: loan status (Fully Paid / Default), total payment, recoveries, last payment date, outstanding balance

EDA



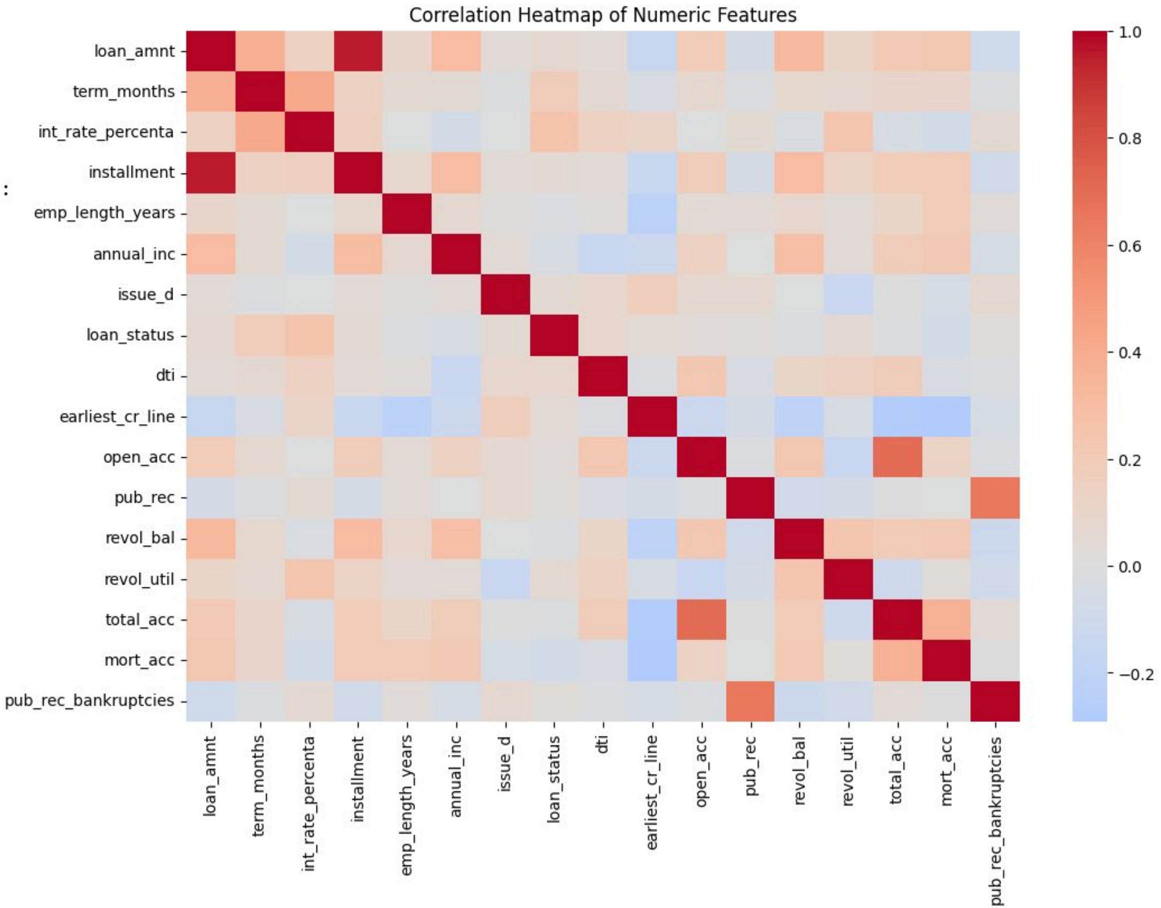
Average loan_status by Loan Purpose
loan_status
0.00
0.05
0.10
0.15
0.20
0.25
0.30
debt_consolidation
credit_card
home_improvement
other
major_purchase
medical
small_business
car
moving
vacation
house
wedding
renewable_energy
educational
purpose



Class Distribution (loan_status)
Proportion
0.0
0.2
0.4
0.6
0.8
1.0
0
1
loan_status

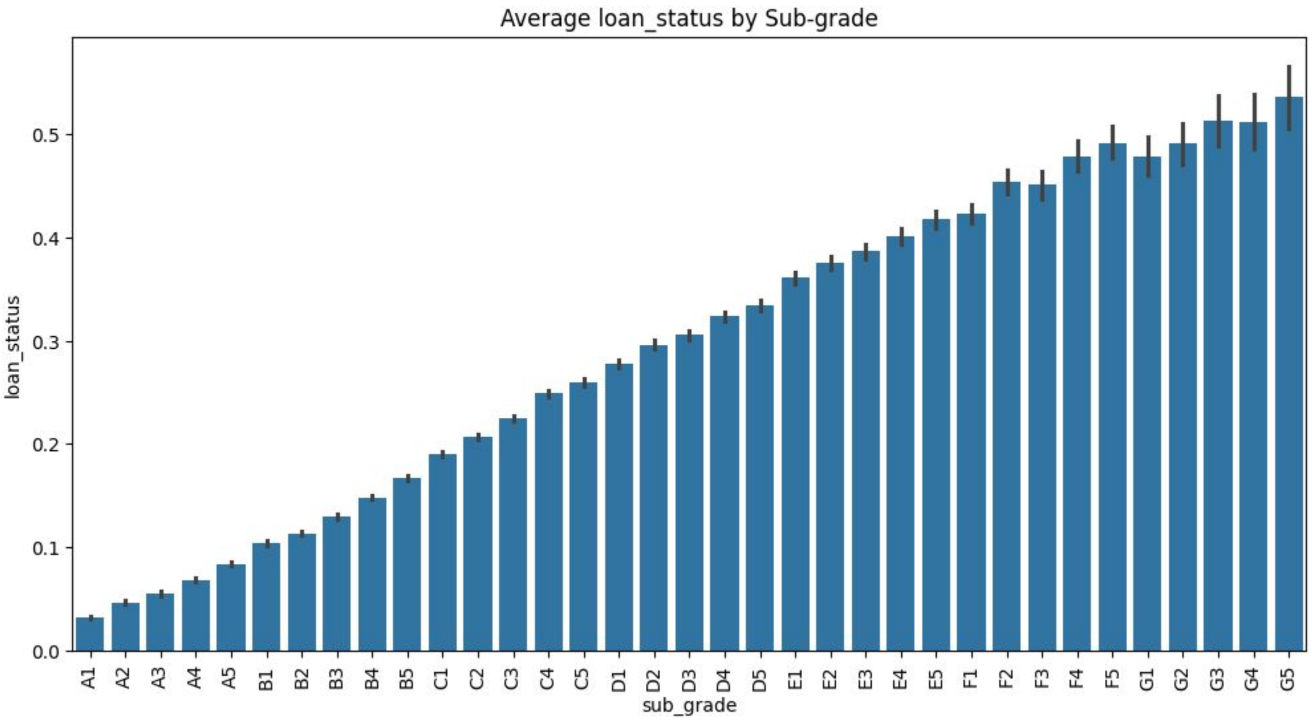
EDA – Numeric Feature Correlation

```
Correlation of numeric features with loan_status:
loan_status              1.000000
int_rate_percenta        0.258792
term_months              0.176096
dti                      0.084510
loan_amnt                0.065604
revol_util               0.060048
installment              0.051701
issue_d                  0.051381
earliest_cr_line         0.044054
open_acc                 0.028078
pub_rec                  0.026194
pub_rec_bankruptcies     0.025308
total_acc               -0.011300
emp_length_years        -0.014235
revol_bal               -0.020010
annual_inc              -0.041759
mort_acc                -0.075294
Name: loan_status, dtype: float64
```



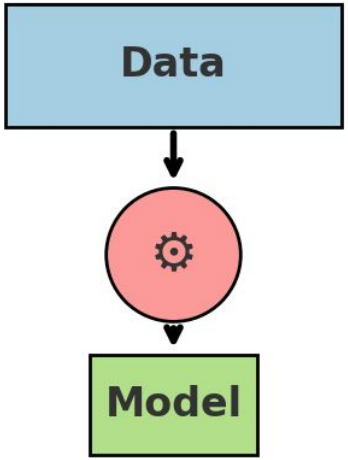
EDA

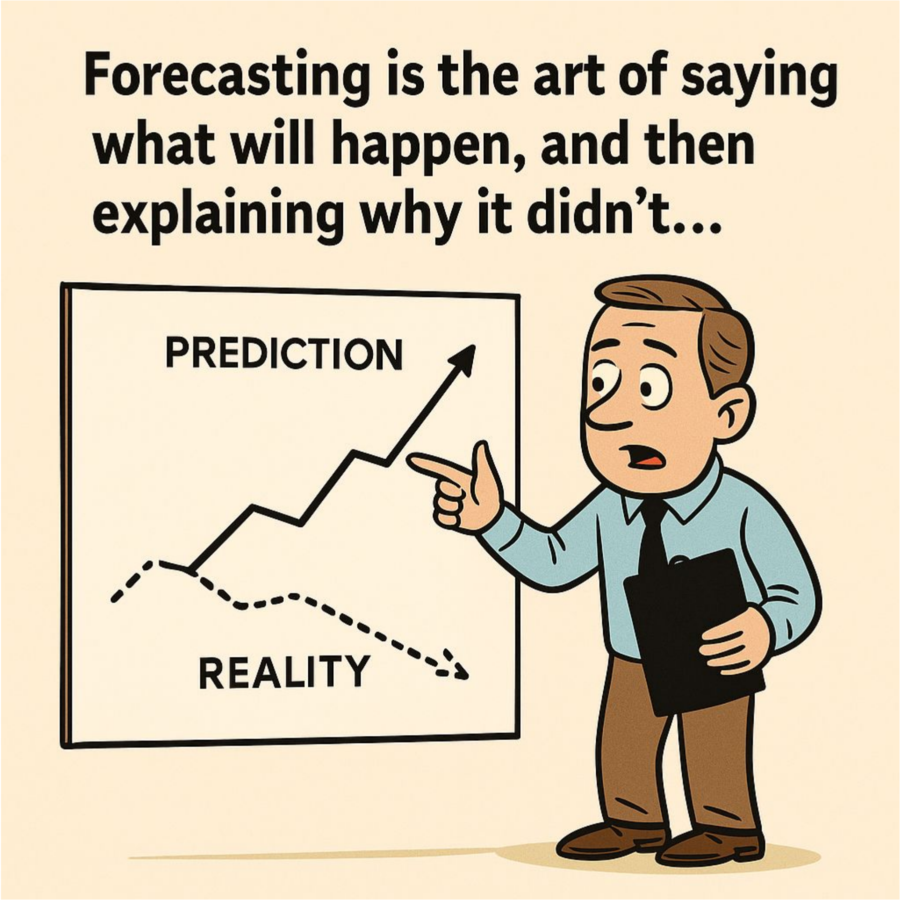
```
Sub-grade vs loan_status (mean):
sub_grade
G5    0.536036
G3    0.513631
G4    0.511719
F5    0.491758
G2    0.491319
F4    0.478905
G1    0.478478
F2    0.453459
F3    0.450723
F1    0.422969
E5    0.417701
E4    0.401018
E3    0.387121
E2    0.375450
E1    0.360647
D5    0.334270
D4    0.323680
D3    0.305415
D2    0.295866
D1    0.278054
C5    0.260050
C4    0.249204
C3    0.225233
C2    0.207188
C1    0.189838
B5    0.167014
B4    0.148271
B3    0.129835
B2    0.113598
B1    0.104212
A5    0.084043
A4    0.068670
A3    0.055085
A2    0.046640
A1    0.032236
Name: loan_status, dtype: float64
```



Data Cleaning and Preprocessing

- Drop columns with >40% missing values
- Remove rows with missing values
- Remove unnecessary columns (IDs, URLs)
- Drop leakage features (e.g., payments after issuance)
- Handle outliers (cap at 1%–99%)
- Encode categorical features (dummies)
- Define target column → **loan_status**
- Scale numeric features (for Logistic Regression)
- Split into train & test sets
- Build pipelines → automate preprocessing + model training
- Train models on training set
- Predict & evaluate on test set




Forecasting is the art of saying what will happen, and then explaining why it didn't...
PREDICTION
REALITY

First Run Results

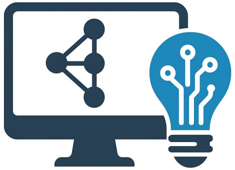
```
=== LogReg ===
Average Precision (Val): 0.3649
Recall≥0.80 -> thr=0.1450 | Test precision=0.274, recall=0.800
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[34145  8556]
 [90490 83948]]
Recall≥0.85 -> thr=0.1294 | Test precision=0.260, recall=0.851
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[ 36320   6381]
 [103583  70855]]
Recall≥0.90 -> thr=0.1117 | Test precision=0.243, recall=0.902
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[ 38534   4167]
 [119815  54623]]

=== RandomForest ===
Average Precision (Val): 0.3895
Recall≥0.80 -> thr=0.1601 | Test precision=0.280, recall=0.792
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[33827  8874]
 [86814 87624]]
Recall≥0.85 -> thr=0.1405 | Test precision=0.265, recall=0.842
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[35955  6746]
 [99745 74693]]
Recall≥0.90 -> thr=0.1188 | Test precision=0.248, recall=0.895
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[ 38204   4497]
 [115555  58883]]
```

```
=== XGBoost ===
Average Precision (Val): 0.3835
Recall≥0.80 -> thr=0.4232 | Test precision=0.282, recall=0.796
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[33970  8731]
 [86540 87898]]
Recall≥0.85 -> thr=0.3822 | Test precision=0.266, recall=0.846
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[36108  6593]
 [99579 74859]]
Recall≥0.90 -> thr=0.3312 | Test precision=0.249, recall=0.896
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[ 38254   4447]
 [115483  58955]]
```

Feature Engineering

- Merge low loan purpose - other
- New column - sub_grade_X_loan_amount
- New column - Installment_to_income column
- New column - emp_length_sq



Secund Run Results

```
=== LogReg ===
Average Precision (Val): 0.3644
Recall≥0.80 -> thr=0.4050 | Test precision=0.274, recall=0.800
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[25614  6411]
 [67841 62988]]
Recall≥0.85 -> thr=0.3716 | Test precision=0.260, recall=0.851
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[27251  4774]
 [77517 53312]]
Recall≥0.90 -> thr=0.3320 | Test precision=0.244, recall=0.904
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[28946  3079]
 [89471 41358]]

=== RandomForest ===
Average Precision (Val): 0.3712
Recall≥0.80 -> thr=0.4157 | Test precision=0.277, recall=0.802
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[25682  6343]
 [66960 63869]]
Recall≥0.85 -> thr=0.3788 | Test precision=0.263, recall=0.852
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[27272  4753]
 [76526 54303]]
Recall≥0.90 -> thr=0.3313 | Test precision=0.246, recall=0.903
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[28934  3091]
 [88742 42087]]
```

```
=== XGB ===
Average Precision (Val): 0.3767
Recall≥0.80 -> thr=0.4150 | Test precision=0.279, recall=0.802
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[25693  6332]
 [66388 64441]]
Recall≥0.85 -> thr=0.3737 | Test precision=0.264, recall=0.852
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[27290  4735]
 [75989 54840]]
Recall≥0.90 -> thr=0.3219 | Test precision=0.248, recall=0.903
Confusion matrix [rows=true 1/0, cols=pred 1/0]:
 [[28920  3105]
 [87923 42906]]
```

Data Reinspection

Feature engineering didn’t improve results → revisited raw data to check if important columns were dropped during cleaning. Ensured no useful signals were lost.

Third Run Results

```
=== LogReg ===  AP(Val)=0.7942
  recall≥0.80: thr=0.6817 | Val P=0.740, R=0.800 | Test precision=0.746, recall=0.804
  recall≥0.85: thr=0.5911 | Val P=0.708, R=0.850 | Test precision=0.713, recall=0.850
  recall≥0.90: thr=0.4755 | Val P=0.659, R=0.900 | Test precision=0.664, recall=0.897

=== XGBoost ===  AP(Val)=0.8461
  recall≥0.80: thr=0.7617 | Val P=0.758, R=0.800 | Test precision=0.763, recall=0.800
  recall≥0.85: thr=0.6817 | Val P=0.719, R=0.850 | Test precision=0.724, recall=0.848
  recall≥0.90: thr=0.5441 | Val P=0.668, R=0.900 | Test precision=0.672, recall=0.899

Summary (precision at target recalls on TEST):
LogReg | R≥0.80: P=0.746 | R≥0.85: P=0.713 | R≥0.90: P=0.664
XGBoost | R≥0.80: P=0.763 | R≥0.85: P=0.724 | R≥0.90: P=0.672
```

BAM!!!!!!!!!!

🚨 Leakage Alert !

```
Top 10 LogReg features (by absolute coefficient):
last_fico_range_high            -1.847535
last_fico_range_low             -0.980509
emp_title_infrequent_sklearn    -0.300951
term                             0.259335
id_infrequent_sklearn           -0.237214
emp_title_Teacher                0.168807
dti                              0.165097
int_rate                         0.144501
mo_sin_old_rev_tl_op             0.128806
home_ownership_MORTGAGE         -0.125083
dtype: float32

Top 10 XGBoost features (by importance):
last_fico_range_high             0.189705
last_fico_range_low              [illegible]
term                             [illegible]
emp_title_Teacher                [illegible]
title_infrequent_sklearn         [illegible]
funded_amnt                      [illegible]
emp_title_infrequent_sklearn     0.006299
application_type_Joint App       0.005438
issue_d_infrequent_sklearn       0.005434
loan_amnt                        0.005155
dtype: float32
```

`last_fico_range_high` and `last_fico_range_low` are leakage features because they're updated after the loan is issued, revealing future borrower performance. Using them gives the model unrealistically good results that wouldn't work in real life.

NOT A BAM...

Fourth run results

```
=== LogReg ===  AP(Val)=0.3961
  recall≥0.80: thr=0.4245 | Val P=0.289, R=0.800 | Test precision=0.288, recall=0.802
  recall≥0.85: thr=0.3858 | Val P=0.272, R=0.850 | Test precision=0.271, recall=0.852
  recall≥0.90: thr=0.3398 | Val P=0.254, R=0.900 | Test precision=0.254, recall=0.902

=== DecisionTree ===  AP(Val)=0.3527
  recall≥0.80: thr=0.3837 | Val P=0.264, R=0.800 | Test precision=0.263, recall=0.805
  recall≥0.85: thr=0.3498 | Val P=0.251, R=0.851 | Test precision=0.250, recall=0.854
  recall≥0.90: thr=0.2812 | Val P=0.237, R=0.900 | Test precision=0.237, recall=0.901

=== RandomForest ===  AP(Val)=0.3946
  recall≥0.80: thr=0.4300 | Val P=0.284, R=0.800 | Test precision=0.285, recall=0.807
  recall≥0.85: thr=0.4010 | Val P=0.270, R=0.850 | Test precision=0.269, recall=0.854
  recall≥0.90: thr=0.3645 | Val P=0.254, R=0.900 | Test precision=0.253, recall=0.903

=== XGBoost ===  AP(Val)=0.4152
  recall≥0.80: thr=0.4164 | Val P=0.298, R=0.800 | Test precision=0.294, recall=0.799
  recall≥0.85: thr=0.3760 | Val P=0.281, R=0.850 | Test precision=0.278, recall=0.850
  recall≥0.90: thr=0.3250 | Val P=0.261, R=0.900 | Test precision=0.259, recall=0.902
```

Conclusion

The final models achieved **high recall (>0.9)** but at the cost of **very low precision (~0.25)**. This means the models can catch most defaults but also incorrectly classify many good borrowers as risky.

Multiple rounds of **feature engineering** did not significantly improve performance, showing that the predictive signal in pre-loan data is limited.

The key insight is that **loan default is inherently difficult to predict using only pre-origination data** (income, employment, etc.). Borrower defaults often depend on **future life events** such as job loss, medical expenses, economic downturns, or changes in credit behavior—factors that are unknown at the time of application.

As a result, models built only on pre-loan features may face a ceiling in performance: they can identify *patterns of higher risk*, but they cannot perfectly foresee future borrower behavior.

Thanks for listening!