LOANTAP - BUSINESS CASE STUDY

Fast. Flexible. Friendly.

Author - Shishir Bhat

Problem Statement

About LoanTap

LoanTap is an online platform committed to delivering **customized loan products** to millennials.
They innovate in an otherwise dull loan segment, offering **instant, flexible loans** on consumer-friendly terms to **salaried professionals** and **businessmen**.

The **Data Science Team** at LoanTap is building an **underwriting layer** to determine the **creditworthiness** of both **MSMEs** and **individuals**.

Financial Instruments Offered

LoanTap deploys formal credit to salaried individuals and businesses through the following **4 main products**:

1. **Personal Loan**
2. **EMI-Free Loan**
3. **Personal Overdraft**
4. **Advance Salary Loan**

This case study will focus only on the **underwriting process** behind **Personal Loan**.

Problem Statement

Given a set of attributes for an **individual borrower**:

- Determine **if a credit line should be extended** to them.
- If approved, provide **business recommendations** on **repayment terms**.

Objectives:

- Predict loan default probability using customer attributes
- Identify key risk factors affecting loan repayment
- Optimize approval rates while minimizing Non-Performing Assets (NPAs)
- Provide actionable insights for business decisions

TARGET: loan_status (Fully Paid = Good, Charged Off = Bad)

Columns Dictionary

Column Name	Description
loan_amnt	The listed amount of the loan applied for. If reduced by LoanTap's credit department, the reduced amount is shown.
term	The number of payments on the loan. Values: **36** or **60** months.
int_rate	Interest rate on the loan.
installment	Monthly payment owed by the borrower if the loan originates.
grade	LoanTap-assigned loan grade.
sub_grade	LoanTap-assigned loan subgrade.

emp_title	Job title supplied by the borrower at application.
emp_length	Employment length in years (0 = <1 year, 10 = 10+ years).
home_ownership	Home ownership status at application (e.g., Own, Rent, Mortgage).
annual_inc	Self-reported annual income.
verification_status	Whether income was verified by LoanTap (Verified / Not Verified / Source Verified).
issue_d	Month when the loan was funded.
loan_status	Current status of the loan (**Target Variable**).
purpose	Category provided by the borrower for the loan reason.
title	Loan title provided by the borrower.
dti	Debt-to-income ratio (monthly debt ÷ monthly income).
earliest_cr_line	The month of the borrower's earliest reported credit line.
open_acc	Number of open credit lines.
pub_rec	Number of derogatory public records.
revol_bal	Total revolving credit balance.
revol_util	Revolving credit utilization rate (used ÷ available).
total_acc	Total number of credit lines in borrower's credit file.
initial_list_status	Initial listing status of the loan (W or F).
application_type	Indicates if the application is Individual or Joint.
mort_acc	Number of mortgage accounts.
pub_rec_bankruptcies	Number of public record bankruptcies.
Address	Address of the individual borrower.

1. SETUP AND IMPORTS

In [ ]:

```
# import necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_curve, roc_auc_score,
    precision_recall_curve, average_precision_score, precision_score,
    recall_score, f1_score
)
import warnings
warnings.filterwarnings('ignore')

# Set plotting style
plt.style.use('default')
plt.rcParams['figure.figsize'] = (10, 6)
sns.set_palette("husl")
```

2. DATA LOADING AND EXPLORATION

In [ ]:

```
# Data Loading

print("\n2. DATA LOADING:")
print("-"*20)

df = pd.read_csv('logistic_regression.csv', low_memory=False)
print("Data loaded successfully")

# Basic data information
print(f"\nDataset Shape: {df.shape[0]} rows × {df.shape[1]} columns")
print(f"\nColumns: {list(df.columns)}")
```

```
2. DATA LOADING:
--------------------
Data loaded successfully

Dataset Shape: 396030 rows × 27 columns

Columns: ['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length', 'home_ownership', 'annual_inc', 'verification_status', 'issue_d', 'loan_status', 'purpose', 'title', 'dti', 'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'initial_list_status', 'application_type', 'mort_acc', 'pub_rec_bankruptcies', 'address']
```

- Dataset contains 396,030 loan records across 27 attributes, indicating substantial lending portfolio size
- Mix of numerical features (loan amounts, rates, ratios) and categorical variables (grades, employment) provides comprehensive borrower profiles
- Large sample size ensures strong statistical power for identifying meaningful risk patterns

Data types and missing values

In [ ]:
```python
# Convert everything to dataframe for better view

print("\nData Types and Missing Values:")
data_info = pd.DataFrame({
    'Column': df.columns,
    'Data_Type': df.dtypes,
    'Non_Null': df.count(),
    'Null_Count': df.isnull().sum(),
    'Null_Pct': (df.isnull().sum() / len(df) * 100).round(2)
})
print(data_info)
```

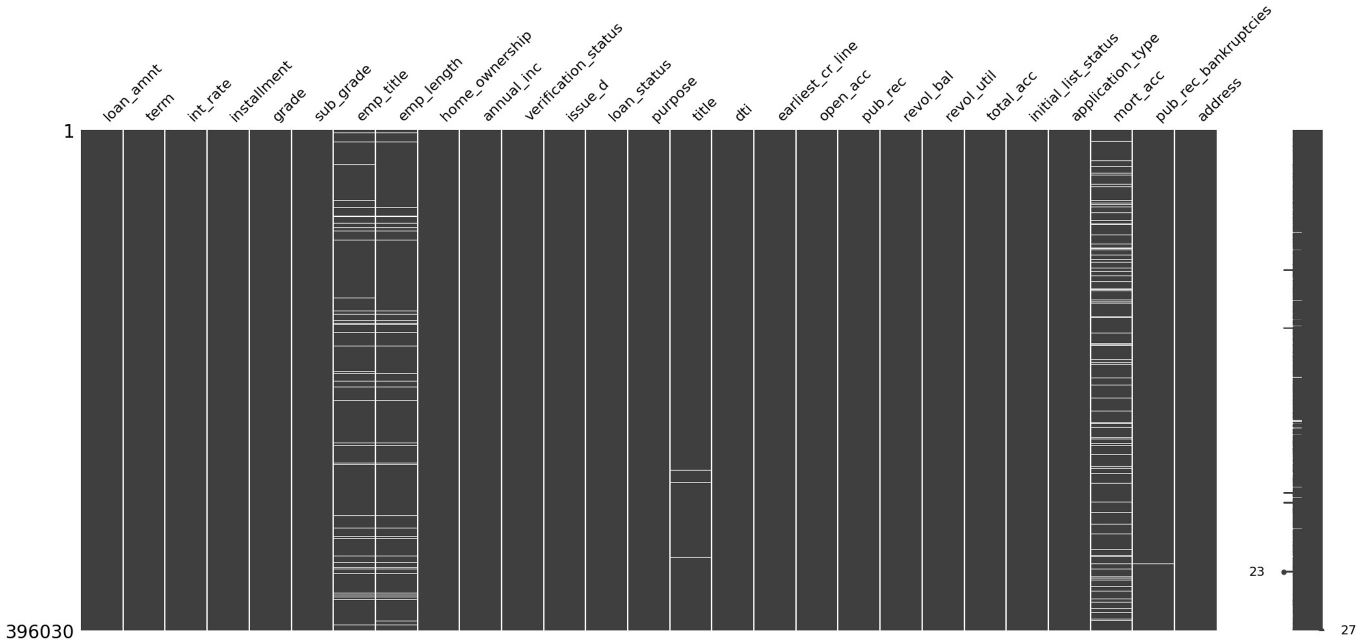
```
Data Types and Missing Values:
                                    Column Data_Type  Non_Null  Null_Count  \
loan_amnt                        loan_amnt   float64    396030           0
term                                  term    object    396030           0
int_rate                          int_rate   float64    396030           0
installment                    installment   float64    396030           0
grade                                grade    object    396030           0
sub_grade                        sub_grade    object    396030           0
emp_title                        emp_title    object    373103       22927
emp_length                      emp_length    object    377729       18301
home_ownership              home_ownership    object    396030           0
annual_inc                      annual_inc   float64    396030           0
verification_status    verification_status    object    396030           0
issue_d                            issue_d    object    396030           0
loan_status                    loan_status    object    396030           0
purpose                            purpose    object    396030           0
title                                title    object    394274        1756
dti                                    dti   float64    396030           0
earliest_cr_line          earliest_cr_line    object    396030           0
open_acc                          open_acc   float64    396030           0
pub_rec                            pub_rec   float64    396030           0
revol_bal                        revol_bal   float64    396030           0
revol_util                      revol_util   float64    395754         276
total_acc                        total_acc   float64    396030           0
initial_list_status    initial_list_status    object    396030           0
application_type          application_type    object    396030           0
mort_acc                          mort_acc   float64    358235       37795
pub_rec_bankruptcies  pub_rec_bankruptcies   float64    395495         535
address                            address    object    396030           0

                      Null_Pct
loan_amnt                 0.00
term                      0.00
int_rate                  0.00
installment               0.00
grade                     0.00
sub_grade                 0.00
emp_title                 5.79
emp_length                4.62
home_ownership            0.00
annual_inc                0.00
verification_status       0.00
issue_d                   0.00
loan_status               0.00
purpose                   0.00
title                     0.44
dti                       0.00
earliest_cr_line          0.00
open_acc                  0.00
pub_rec                   0.00
revol_bal                 0.00
revol_util                0.07
total_acc                 0.00
initial_list_status       0.00
application_type          0.00
mort_acc                  9.54
pub_rec_bankruptcies      0.14
address                   0.00
```

Visualise the missing values

In [ ]:

```python
# Using missingno library

import missingno as msno
msno.matrix(df)
```

Out[ ]: <Axes: >



- Employment title missing in 5.79% of cases may indicate non-traditional employment or privacy concerns
- Mortgage account data missing in 9.54% suggests incomplete credit bureau information affecting risk assessment
- Critical financial variables (income, DTI, loan amount) have zero missing values, showing good collection practices
- Target variable completeness ensures full model training capability
- Missing value patterns may themselves be predictive of default risk

Unique Values in each columns

In [ ]:
```
# Helpful for deciding the type of encoding

for col in df.columns:
  print(f'{df[col].value_counts()}\n')
```

```
loan_amnt
10000.0    27668
12000.0    21366
15000.0    19903
20000.0    18969
35000.0    14576
           ...
39200.0        1
38750.0        1
36275.0        1
36475.0        1
725.0          1
Name: count, Length: 1397, dtype: int64

term
36 months    302005
60 months     94025
Name: count, dtype: int64

int_rate
10.99    12411
12.99     9632
15.61     9350
11.99     8582
8.90      8019
         ...
```

```
14.38        1
24.40        1
22.64        1
17.54        1
17.44        1
Name: count, Length: 566, dtype: int64

installment
327.34     968
332.10     791
491.01     736
336.90     686
392.81     683
          ...
1146.14      1
218.49       1
961.66       1
569.10       1
555.96       1
Name: count, Length: 55706, dtype: int64

grade
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: count, dtype: int64

sub_grade
B3    26655
B4    25601
C1    23662
C2    22580
B2    22495
B5    22085
C3    21221
C4    20280
B1    19182
A5    18526
C5    18244
D1    15993
A4    15789
D2    13951
D3    12223
D4    11657
A3    10576
A1     9729
D5     9700
A2     9567
E1     7917
E2     7431
E3     6207
E4     5361
E5     4572
F1     3536
F2     2766
F3     2286
F4     1787
F5     1397
G1     1058
G2      754
G3      552
G4      374
G5      316
Name: count, dtype: int64

emp_title
Teacher                        4389
Manager                        4250
Registered Nurse               1856
RN                             1846
Supervisor                     1830
                               ...
OMIV Supervisor                   1
SVP, Technology                   1
sikorsky                          1
Postman                           1
Sr. Facilities Caretaker          1
Name: count, Length: 173105, dtype: int64
```

```
emp_length
10+ years    126041
2 years       35827
< 1 year      31725
3 years       31665
5 years       26495
1 year        25882
4 years       23952
6 years       20841
7 years       20819
8 years       19168
9 years       15314
Name: count, dtype: int64

home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64

annual_inc
60000.0    15313
50000.0    13303
65000.0    11333
70000.0    10674
40000.0    10629
           ...  
67842.0        1
72179.0        1
50416.0        1
46820.8        1
87622.0        1
Name: count, Length: 27197, dtype: int64

verification_status
Verified           139563
Source Verified    131385
Not Verified       125082
Name: count, dtype: int64

issue_d
Oct-2014    14846
Jul-2014    12609
Jan-2015    11705
Dec-2013    10618
Nov-2013    10496
            ...  
Aug-2007       26
Sep-2008       25
Nov-2007       22
Sep-2007       15
Jun-2007        1
Name: count, Length: 115, dtype: int64

loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64

purpose
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: count, dtype: int64

title
Debt consolidation         152472
Credit card refinancing     51487
Home improvement            15264
```

```
Other                        12930
Debt Consolidation           11608
                             ...
creditcardrefi                   1
Debt/Home                        1
Peace Of Mind Loan               1
Blazer repair                    1
Out of my rut                    1
Name: count, Length: 48816, dtype: int64

dti
0.00       313
14.40      310
19.20      302
16.80      301
18.00      300
          ...
47.05        1
46.52        1
1622.00      1
40.21        1
189.90       1
Name: count, Length: 4262, dtype: int64

earliest_cr_line
Oct-2000    3017
Aug-2000    2935
Oct-2001    2896
Aug-2001    2884
Nov-2000    2736
            ...
Feb-1957       1
Nov-1950       1
May-1955       1
Sep-1961       1
Nov-1955       1
Name: count, Length: 684, dtype: int64

open_acc
9.0     36779
10.0    35441
8.0     35137
11.0    32695
7.0     31328
        ...
56.0        2
55.0        2
57.0        1
58.0        1
90.0        1
Name: count, Length: 61, dtype: int64

pub_rec
0.0     338272
1.0      49739
2.0       5476
3.0       1521
4.0        527
5.0        237
6.0        122
7.0         56
8.0         34
9.0         12
10.0        11
11.0         8
13.0         4
12.0         4
19.0         2
40.0         1
17.0         1
86.0         1
24.0         1
15.0         1
Name: count, dtype: int64

revol_bal
0.0        2128
5655.0       41
7792.0       38
6095.0       38
3953.0       37
            ...
43895.0       1
```

```
46733.0        1
36519.0        1
212269.0       1
71547.0        1
Name: count, Length: 55622, dtype: int64

revol_util
0.00      2213
53.00      752
60.00      739
61.00      734
55.00      730
          ...
146.10       1
109.30       1
108.10       1
115.30       1
37.63        1
Name: count, Length: 1226, dtype: int64

total_acc
21.0     14280
22.0     14260
20.0     14228
23.0     13923
24.0     13878
         ...
150.0        1
117.0        1
115.0        1
100.0        1
103.0        1
Name: count, Length: 118, dtype: int64

initial_list_status
f    238066
w    157964
Name: count, dtype: int64

application_type
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: count, dtype: int64

mort_acc
0.0     139777
1.0      60416
2.0      49948
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
26.0         2
32.0         2
31.0         2
23.0         2
34.0         1
28.0         1
30.0         1
Name: count, dtype: int64

pub_rec_bankruptcies
0.0    350380
```

```
1.0     42790
2.0      1847
3.0       351
4.0        82
5.0        32
6.0         7
7.0         4
8.0         2
Name: count, dtype: int64

address
USS Johnson\r\nFPO AE 48052                                  8
USNS Johnson\r\nFPO AE 05113                                 8
USS Smith\r\nFPO AP 70466                                    8
USCGC Smith\r\nFPO AE 70466                                  8
USNS Johnson\r\nFPO AP 48052                                 7
                                                            ..
8141 Cox Greens Suite 186\r\nMadisonstad, VT 05113           1
8803 Sean Highway Suite 029\r\nNorth Nicoleshire, AK 11650   1
594 Nicole Mission Apt. 620\r\nNew Patrick, NJ 00813         1
7336 Sean Groves Apt. 893\r\nDariusborough, NJ 05113         1
9160 Tucker Squares\r\nSouth Paul, MO 30723                  1
Name: count, Length: 393700, dtype: int64
```

Inference

- Grade distribution concentrated in B and C (56% of portfolio) indicates focus on prime/near-prime borrowers
- Employment length skewed toward 10+ years suggests targeting established professionals
- Debt consolidation dominates loan purposes (59%) indicating financial optimization rather than emergency funding
- Loan amounts cluster around standard denominations (10K, 15K, 20K) reflecting product standardization
- Home ownership shows balanced mix across mortgage, rent, and own categories

Statistical summary for numeric columns

In [ ]:
```
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00	20000.00	40000.00
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33	16.49	30.99
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43	567.30	1533.81
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00	90000.00	8706582.00
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91	22.98	9999.00
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00	14.00	90.00
pub_rec	396030.0	0.178191	0.530671	0.00	0.00	0.00	0.00	86.00
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00	19620.00	1743266.00
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80	72.90	892.30
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00	32.00	151.00
mort_acc	358235.0	1.813991	2.147930	0.00	0.00	1.00	3.00	34.00
pub_rec_bankruptcies	395495.0	0.121648	0.356174	0.00	0.00	0.00	0.00	8.00

Statistical summary for all columns

In [ ]:
```
df.describe(include = 'all').T
```

Out[ ]:

	count	unique	top	freq	mean	std	min	25%	50%	75%
loan_amnt	396030.0	NaN	NaN	NaN	14113.888089	8357.441341	500.0	8000.0	12000.0	20000.0
term	396030	2	36 months	302005	NaN	NaN	NaN	NaN	NaN	NaN
int_rate	396030.0	NaN	NaN	NaN	13.6394	4.472157	5.32	10.49	13.33	16.49
installment	396030.0	NaN	NaN	NaN	431.849698	250.72779	16.08	250.33	375.43	567.3
grade	396030	7	B	116018	NaN	NaN	NaN	NaN	NaN	NaN
sub_grade	396030	35	B3	26655	NaN	NaN	NaN	NaN	NaN	NaN
emp_title	373103	173105	Teacher	4389	NaN	NaN	NaN	NaN	NaN	NaN
emp_length	377729	11	10+ years	126041	NaN	NaN	NaN	NaN	NaN	NaN
home_ownership	396030	6	MORTGAGE	198348	NaN	NaN	NaN	NaN	NaN	NaN
annual_inc	396030.0	NaN	NaN	NaN	74203.175798	61637.621158	0.0	45000.0	64000.0	90000.0
verification_status	396030	3	Verified	139563	NaN	NaN	NaN	NaN	NaN	NaN
issue_d	396030	115	Oct-2014	14846	NaN	NaN	NaN	NaN	NaN	NaN
loan_status	396030	2	Fully Paid	318357	NaN	NaN	NaN	NaN	NaN	NaN
purpose	396030	14	debt_consolidation	234507	NaN	NaN	NaN	NaN	NaN	NaN
title	394274	48816	Debt consolidation	152472	NaN	NaN	NaN	NaN	NaN	NaN
dti	396030.0	NaN	NaN	NaN	17.379514	18.019092	0.0	11.28	16.91	22.98
earliest_cr_line	396030	684	Oct-2000	3017	NaN	NaN	NaN	NaN	NaN	NaN
open_acc	396030.0	NaN	NaN	NaN	11.311153	5.137649	0.0	8.0	10.0	14.0
pub_rec	396030.0	NaN	NaN	NaN	0.178191	0.530671	0.0	0.0	0.0	0.0
revol_bal	396030.0	NaN	NaN	NaN	15844.539853	20591.836109	0.0	6025.0	11181.0	19620.0
revol_util	395754.0	NaN	NaN	NaN	53.791749	24.452193	0.0	35.8	54.8	72.9
total_acc	396030.0	NaN	NaN	NaN	25.414744	11.886991	2.0	17.0	24.0	32.0
initial_list_status	396030	2	f	238066	NaN	NaN	NaN	NaN	NaN	NaN
application_type	396030	3	INDIVIDUAL	395319	NaN	NaN	NaN	NaN	NaN	NaN
mort_acc	358235.0	NaN	NaN	NaN	1.813991	2.14793	0.0	0.0	1.0	3.0
pub_rec_bankruptcies	395495.0	NaN	NaN	NaN	0.121648	0.356174	0.0	0.0	0.0	0.0
address	396030	393700	USS Johnson\r\nFPO AE 48052	8	NaN	NaN	NaN	NaN	NaN	NaN

3. TARGET VARIABLE CREATION

In [ ]:

```
# Creating the Target variable

print("\n3. TARGET VARIABLE CREATION:")
print("-"*30)

# Check original loan status distribution
print("Original Loan Status Distribution:")
print(df['loan_status'].value_counts())
```

```
3. TARGET VARIABLE CREATION:
------------------------------
Original Loan Status Distribution:
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

Create binary target variable

In [ ]:

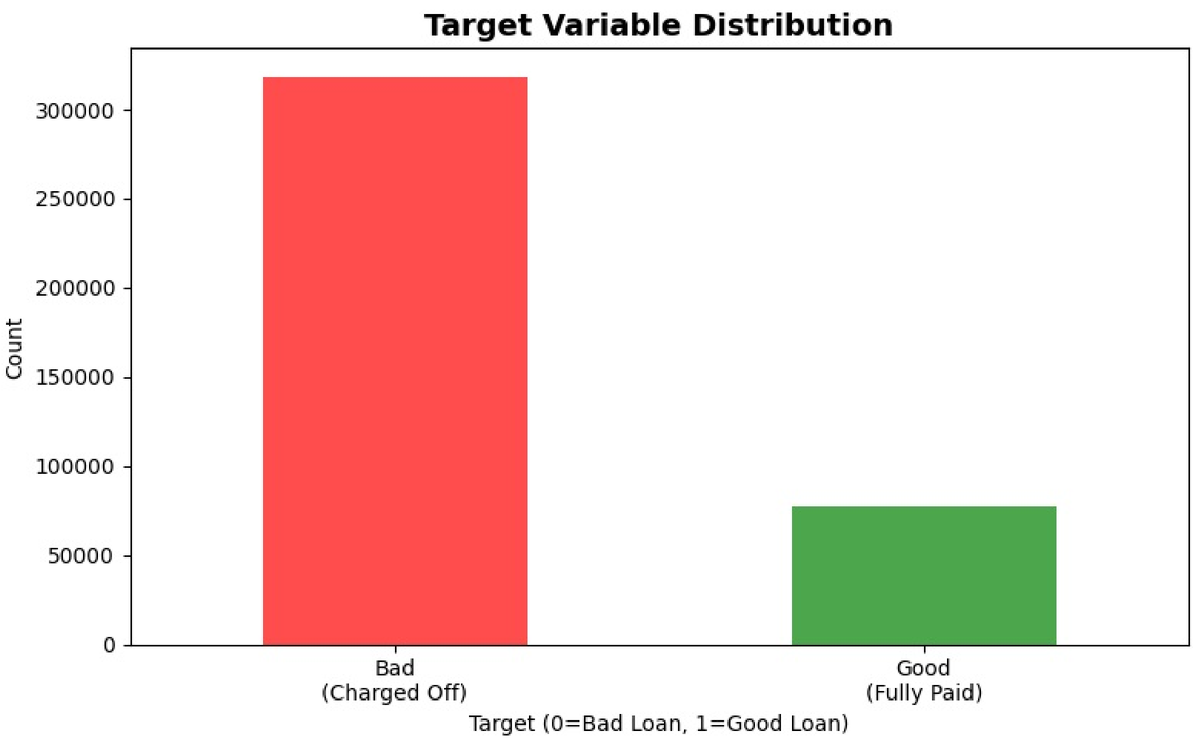
```
# Create binary target variable (1 = Good Loan, 0 = Bad Loan)
def create_target(status):
    if status == 'Fully Paid':
        return 1
    elif status == 'Charged Off':
        return 0
    else:
        return None  # Current, Late, etc.
```

```
df['target'] = df['loan_status'].apply(create_target)

# Display target distribution
target_dist = df['target'].value_counts().sort_index()
print(f"\nTarget Distribution:")
print(f"Bad Loans (0): {target_dist.get(0, 0)} ({target_dist.get(0, 0)/df['target'].notna().sum()*100:.1f}%)")
print(f"Good Loans (1): {target_dist.get(1, 0)} ({target_dist.get(1, 0)/df['target'].notna().sum()*100:.1f}%)")
print(f"Unlabeled: {df['target'].isna().sum()}")
```

```
Target Distribution:
Bad Loans (0): 77673 (19.6%)
Good Loans (1): 318357 (80.4%)
Unlabeled: 0
```

Visualize target distribution

In [ ]:
```
plt.figure(figsize=(8, 5))
if df['target'].notna().sum() > 0:
    target_counts = df['target'].value_counts()
    colors = ['red', 'green']
    target_counts.plot(kind='bar', color=colors, alpha=0.7)
    plt.title('Target Variable Distribution', fontsize=14, fontweight='bold')
    plt.xlabel('Target (0=Bad Loan, 1=Good Loan)')
    plt.ylabel('Count')
    plt.xticks([0, 1], ['Bad\n(Charged Off)', 'Good\n(Fully Paid)'], rotation=0)
    plt.tight_layout()
    plt.show()
```



Inference

- 80.4% good loans vs 19.6% charged-off creates typical lending portfolio structure
- 4:1 good-to-bad ratio indicates reasonable portfolio quality within industry norms
- Class imbalance requires careful model evaluation focusing on precision/recall rather than accuracy
- Default rate of 19.6% emphasizes need for accurate risk prediction to maintain profitability

4. UNIVARIATE ANALYSIS

In [ ]:
```
# Univariate Analysis

print("\n4. UNIVARIATE ANALYSIS:")
print("-"*25)
```

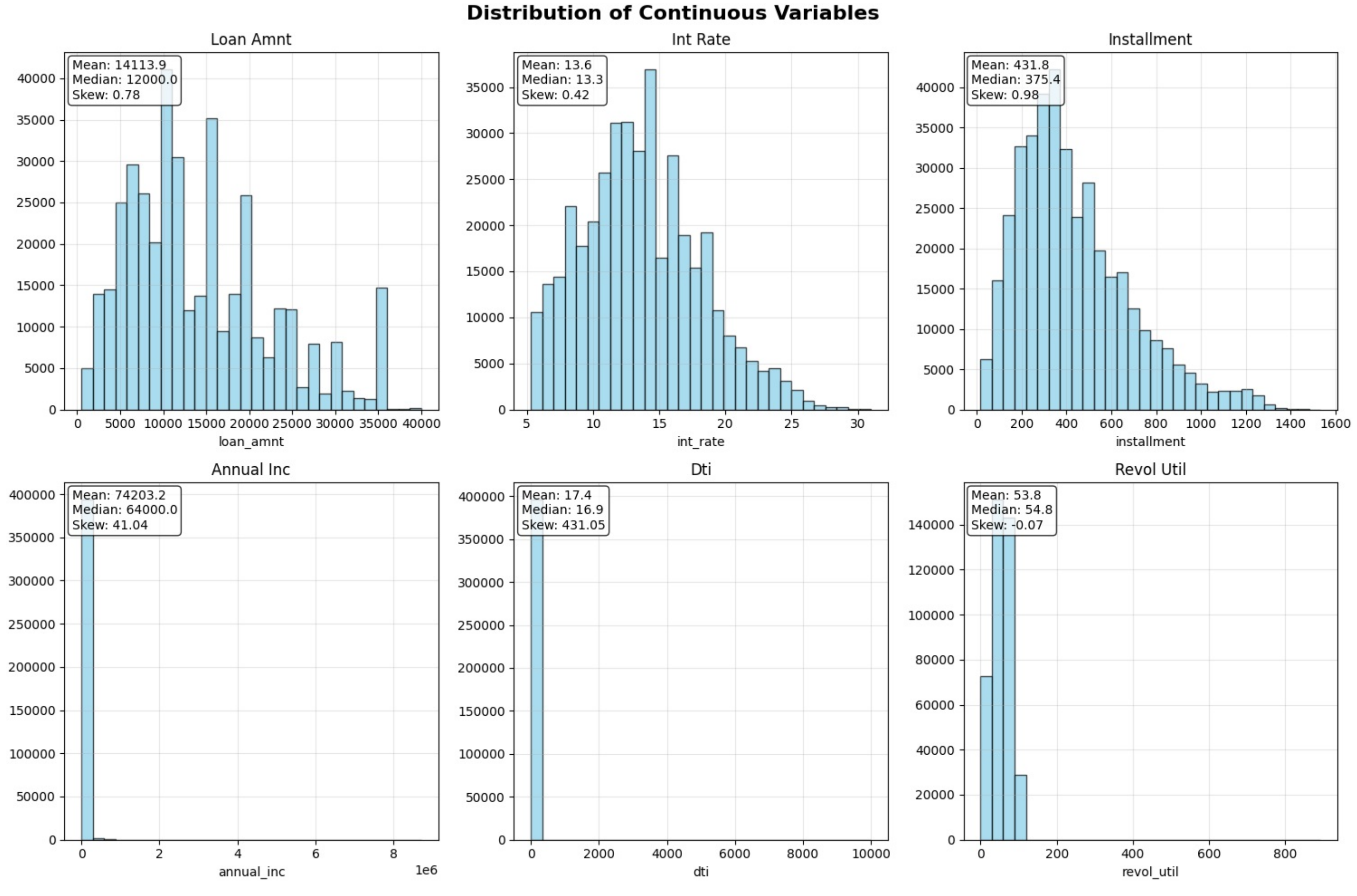
```
4. UNIVARIATE ANALYSIS:
-------------------------
```

Continuous variables analysis

In [ ]:
```
continuous_vars = ['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'revol_util']
```

```
available_continuous = [col for col in continuous_vars if col in df.columns]

print("Continuous Variables Distribution:")
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.ravel()

for i, col in enumerate(available_continuous[:6]):
    df[col].hist(bins=30, ax=axes[i], alpha=0.7, color='skyblue', edgecolor='black')
    axes[i].set_title(f'{col.replace("_", " ").title()}')
    axes[i].set_xlabel(col)
    axes[i].grid(True, alpha=0.3)

    # Add statistics
    mean_val = df[col].mean()
    median_val = df[col].median()
    std_val = df[col].std()
    skew_val = df[col].skew()

    stats_text = f'Mean: {mean_val:.1f}\nMedian: {median_val:.1f}\nSkew: {skew_val:.2f}'
    axes[i].text(0.02, 0.98, stats_text, transform=axes[i].transAxes,
                 verticalalignment='top', bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

plt.suptitle('Distribution of Continuous Variables', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()
```

```
Continuous Variables Distribution:
```

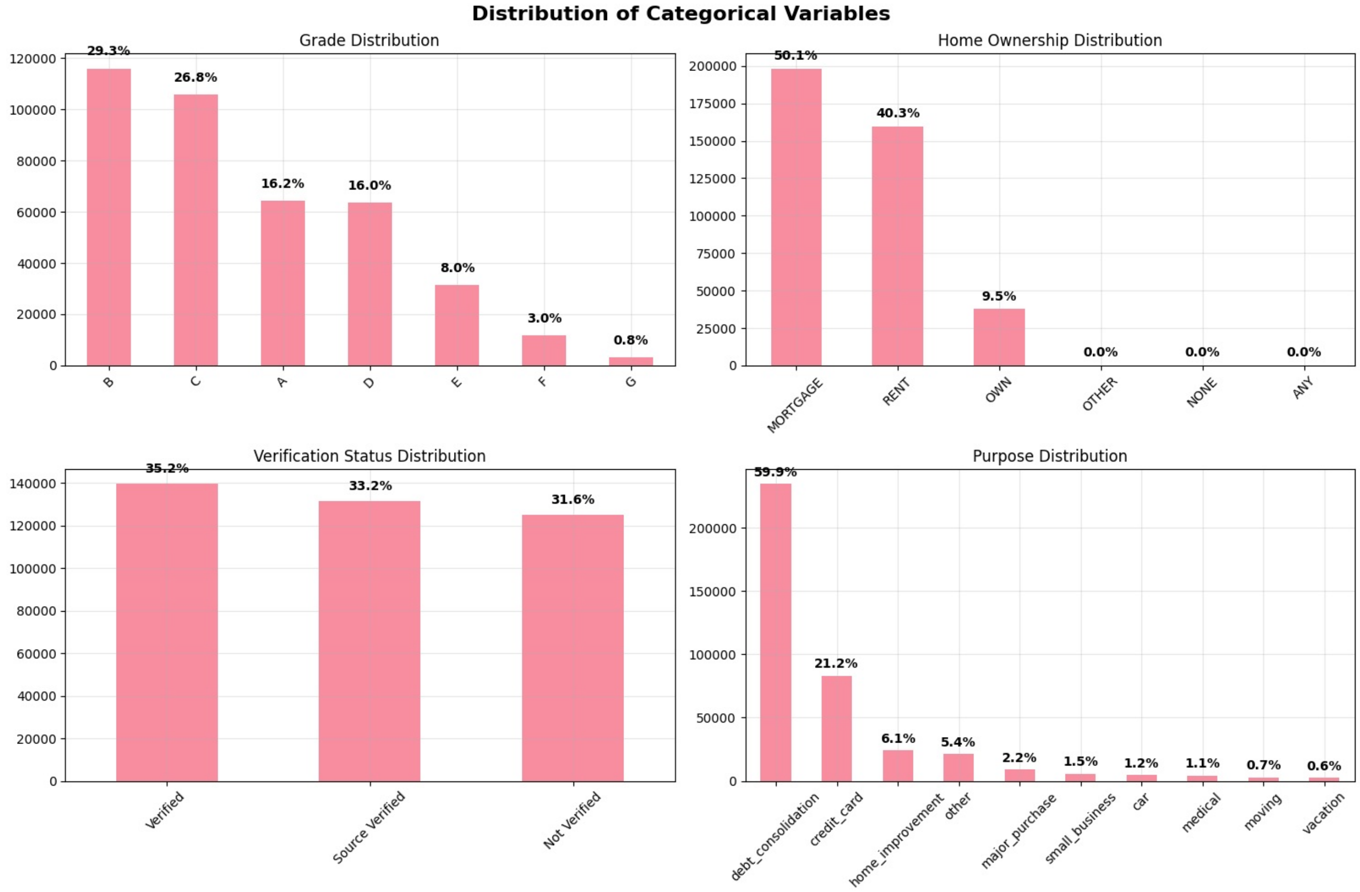


Categorical variables analysis

In [ ]:
```
categorical_vars = ['grade', 'home_ownership', 'verification_status', 'purpose']
available_categorical = [col for col in categorical_vars if col in df.columns]

print("\nCategorical Variables Distribution:")
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.ravel()

for i, col in enumerate(available_categorical[:4]):
    value_counts = df[col].value_counts().head(10)
    value_counts.plot(kind='bar', ax=axes[i], alpha=0.8)
    axes[i].set_title(f'{col.replace("_", " ").title()} Distribution')
    axes[i].set_xlabel('')
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].grid(True, alpha=0.3)

    # Add percentage labels
    total = value_counts.sum()
    for j, v in enumerate(value_counts.values):
```

```
        axes[i].text(j, v + total*0.01, f'{v/total:.1%}',
                    ha='center', va='bottom', fontweight='bold')

plt.suptitle('Distribution of Categorical Variables', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()
```

```
Categorical Variables Distribution:
```



Inference

- Loan amount distribution shows customer preference for round numbers and standard product offerings
- Interest rate distribution has multiple peaks corresponding to different risk grades
- Income distribution skewed but covers broad range, confirming diverse customer base
- Categorical variables confirm middle-grade focus with sophisticated customer financial behaviors
- Distribution patterns align with expected lending industry characteristics

5. BIVARIATE ANALYSIS

In [ ]:
```
# BIVARIATE ANALYSIS

print("\n5. BIVARIATE ANALYSIS:")
print("-"*25)

# Filter to labeled data only
df_labeled = df[df['target'].notna()].copy()
print(f"Analyzing {len(df_labeled)} labeled records...")
```

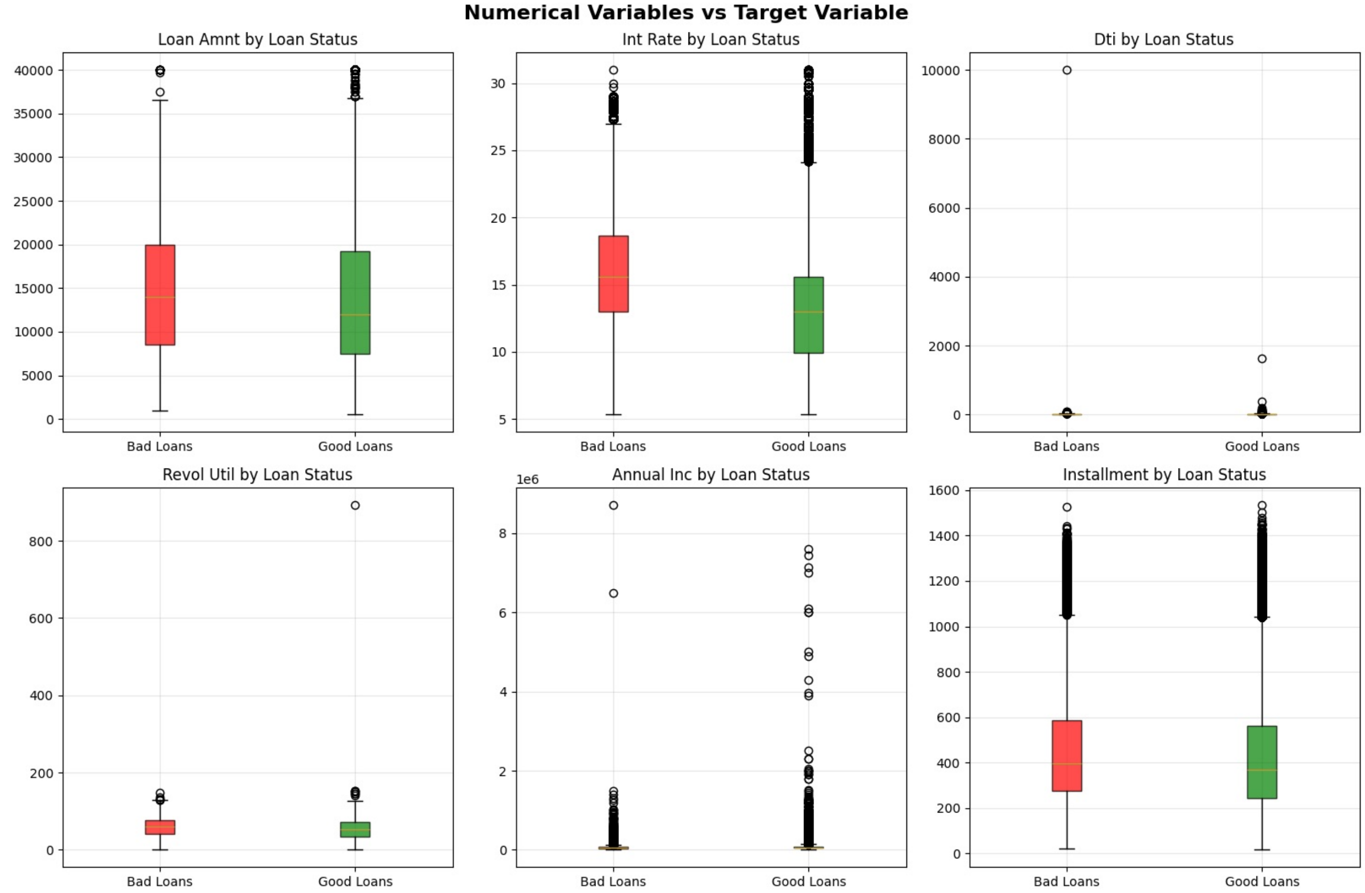
```
5. BIVARIATE ANALYSIS:
-------------------------
Analyzing 396030 labeled records...
```

Numerical variables vs target

In [ ]:
```
print("\nNumerical Variables vs Target:")
numerical_vars = ['loan_amnt', 'int_rate', 'dti', 'revol_util', 'annual_inc', 'installment']
available_numerical = [col for col in numerical_vars if col in df_labeled.columns]

fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.ravel()

for i, col in enumerate(available_numerical[:6]):
    good_loans = df_labeled[df_labeled['target'] == 1][col]
    bad_loans = df_labeled[df_labeled['target'] == 0][col]
```

```python
    bp = axes[i].boxplot([bad_loans.dropna(), good_loans.dropna()],
                         labels=['Bad Loans', 'Good Loans'],
                         patch_artist=True)

    bp['boxes'][0].set_facecolor('red')
    bp['boxes'][0].set_alpha(0.7)
    bp['boxes'][1].set_facecolor('green')
    bp['boxes'][1].set_alpha(0.7)

    axes[i].set_title(f'{col.replace("_", " ").title()} by Loan Status')
    axes[i].grid(True, alpha=0.3)

plt.suptitle('Numerical Variables vs Target Variable', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()
```

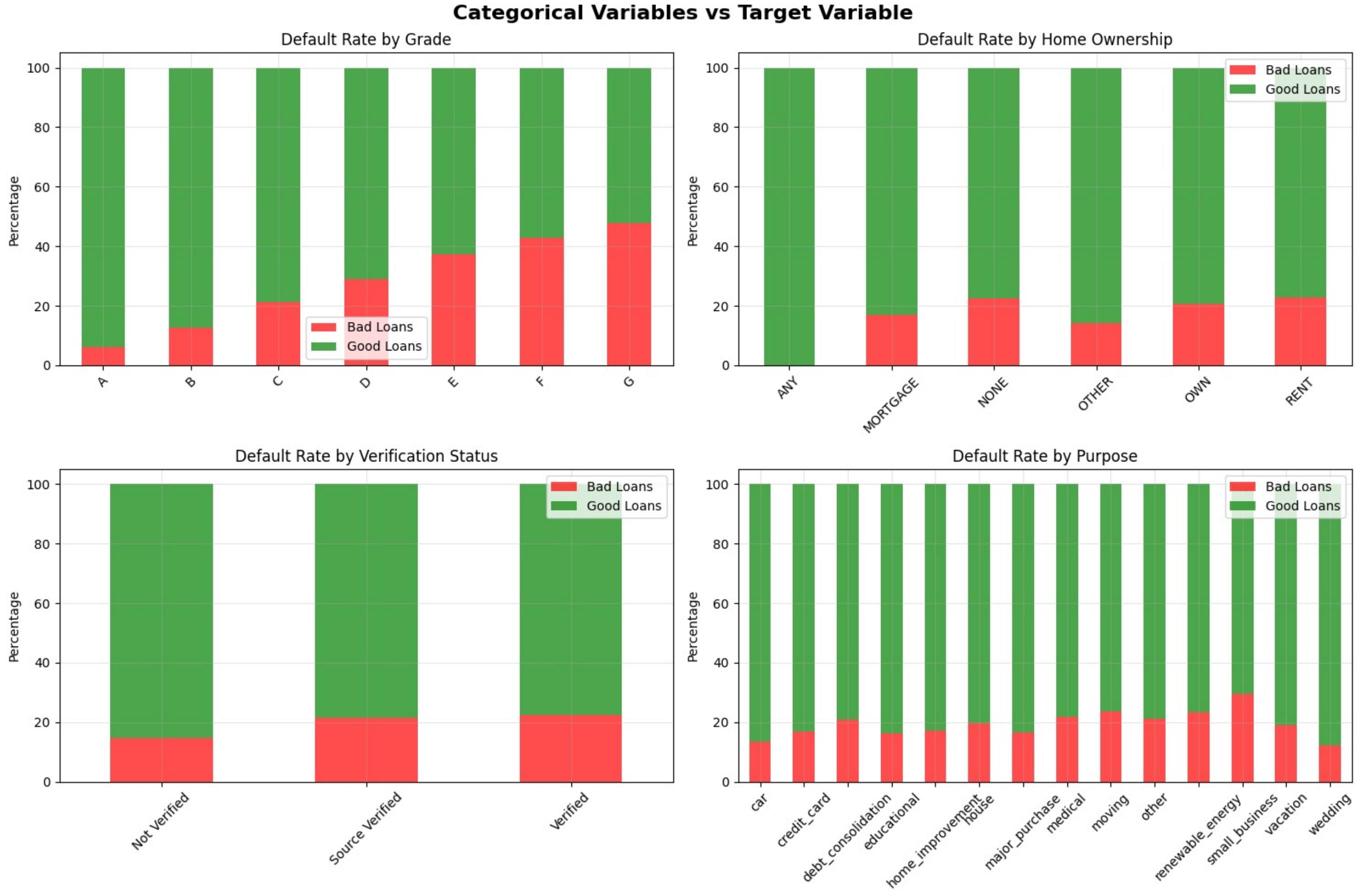
```
Numerical Variables vs Target:
```



Categorical variables vs target

```python
print("\nCategorical Variables vs Target:")
categorical_vars = ['grade', 'home_ownership', 'verification_status', 'purpose']

fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.ravel()

for i, col in enumerate(categorical_vars):
    if col in df_labeled.columns:
        crosstab = pd.crosstab(df_labeled[col], df_labeled['target'])
        crosstab_pct = crosstab.div(crosstab.sum(axis=1), axis=0) * 100

        crosstab_pct.plot(kind='bar', stacked=True, ax=axes[i],
                          color=['red', 'green'], alpha=0.7)

        axes[i].set_title(f'Default Rate by {col.replace("_", " ").title()}')
        axes[i].set_xlabel('')
        axes[i].set_ylabel('Percentage')
        axes[i].legend(['Bad Loans', 'Good Loans'])
        axes[i].tick_params(axis='x', rotation=45)
        axes[i].grid(True, alpha=0.3)

plt.suptitle('Categorical Variables vs Target Variable', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()
```

```
Categorical Variables vs Target:
```

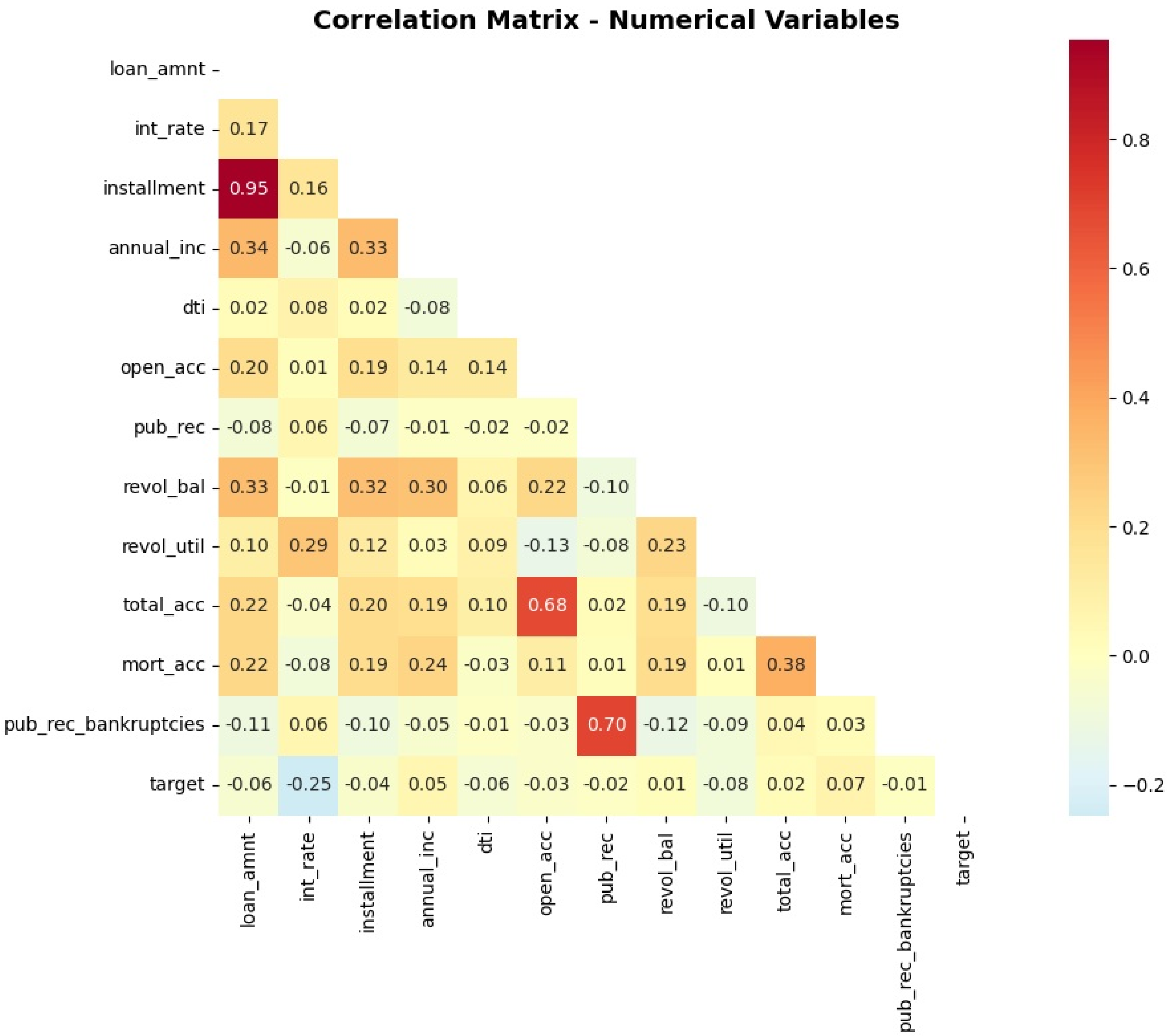


Correlation analysis

In [ ]:

```
print("\nCorrelation Analysis:")
numerical_data = df_labeled.select_dtypes(include=[np.number])
correlation_matrix = numerical_data.corr()

plt.figure(figsize=(12, 8))
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
sns.heatmap(correlation_matrix, mask=mask, annot=True, cmap='RdYlBu_r',
            center=0, square=True, fmt='.2f')
plt.title('Correlation Matrix - Numerical Variables', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

```
Correlation Analysis:
```



Inference

- Higher interest rates correlate with increased default probability, validating risk-based pricing
- Income levels show strong inverse relationship with default rates

- Grade-based default progression (A through G) confirms internal risk scoring effectiveness
- Strong correlation between loan amount and installment (0.954) indicates potential multicollinearity
- Regional variations in default rates suggest geographic economic factors influence risk

6. DATA PREPROCESSING

In [ ]:
```
print("\n6. DATA PREPROCESSING:")
print("-"*25)

df_processed = df.copy()

# Duplicate check
print("1. Duplicate Check:")
duplicates = df_processed.duplicated().sum()
print(f"Duplicate rows found: {duplicates}")
if duplicates > 0:
    df_processed = df_processed.drop_duplicates()
    print(f"Removed {duplicates} duplicate rows")
```

```
6. DATA PREPROCESSING:
-------------------------
1. Duplicate Check:
Duplicate rows found: 0
```

Missing value treatment

In [ ]:
```
print("\n2. Missing Value Treatment:")
missing_summary = df_processed.isnull().sum()
missing_cols = missing_summary[missing_summary > 0]

if len(missing_cols) > 0:
    print("Columns with missing values:")
    for col, count in missing_cols.items():
        pct = count / len(df_processed) * 100
        print(f"  {col}: {count} ({pct:.1f}%)")

    # Fill missing values
    for col in missing_cols.index:
        if df_processed[col].dtype in ['int64', 'float64']:
            df_processed[col] = df_processed[col].fillna(df_processed[col].median())
        else:
            df_processed[col] = df_processed[col].fillna(df_processed[col].mode()[0] if not df_processed[col].m
else:
    print("No missing values found")
```

```
2. Missing Value Treatment:
Columns with missing values:
  emp_title: 22927 (5.8%)
  emp_length: 18301 (4.6%)
  title: 1756 (0.4%)
  revol_util: 276 (0.1%)
  mort_acc: 37795 (9.5%)
  pub_rec_bankruptcies: 535 (0.1%)
```

Outlier treatment

In [ ]:
```
print("\n3. Outlier Treatment:")
numerical_cols = df_processed.select_dtypes(include=[np.number]).columns
outlier_cols = [col for col in numerical_cols if col not in ['target']]

for col in outlier_cols:
    Q1 = df_processed[col].quantile(0.25)
    Q3 = df_processed[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df_processed[(df_processed[col] < lower_bound) |
                            (df_processed[col] > upper_bound)][col]
    outlier_pct = len(outliers) / len(df_processed) * 100

    # Cap extreme outliers if >5%
    if outlier_pct > 5:
        df_processed[col] = df_processed[col].clip(lower=df_processed[col].quantile(0.01),
                                                    upper=df_processed[col].quantile(0.99))
        print(f"Capped outliers in {col} ({outlier_pct:.1f}%)")
```

```
3. Outlier Treatment:
Capped outliers in pub_rec (14.6%)
Capped outliers in revol_bal (5.4%)
Capped outliers in pub_rec_bankruptcies (11.4%)
```

Feature engineering

```
In [ ]: print("\n4. Feature Engineering:")

# Create flags for high-risk indicators
if 'pub_rec' in df_processed.columns:
    df_processed['pub_rec_flag'] = (df_processed['pub_rec'] > 0).astype(int)
    print("Created pub_rec_flag")

if 'mort_acc' in df_processed.columns:
    df_processed['mort_acc_flag'] = (df_processed['mort_acc'] > 0).astype(int)
    print("Created mort_acc_flag")

if 'pub_rec_bankruptcies' in df_processed.columns:
    df_processed['pub_rec_bankruptcies_flag'] = (df_processed['pub_rec_bankruptcies'] > 0).astype(int)
    print("Created pub_rec_bankruptcies_flag")

# Employment length to numeric
if 'emp_length' in df_processed.columns:
    def parse_emp_length(x):
        if pd.isna(x):
            return 0
        x = str(x).lower().strip()
        if '< 1' in x or '<1' in x:
            return 0.5
        elif '10+' in x:
            return 10
        else:
            try:
                return int(x.split()[0])
            except:
                return 0

    df_processed['emp_length_numeric'] = df_processed['emp_length'].apply(parse_emp_length)
    print("Created emp_length_numeric")

# Term to numeric
if 'term' in df_processed.columns:
    df_processed['term_numeric'] = df_processed['term'].str.extract('(\d+)').astype(float)
    print("Created term_numeric")

# Log transformations for skewed variables
skewed_vars = ['loan_amnt', 'annual_inc', 'revol_bal']
for var in skewed_vars:
    if var in df_processed.columns:
        df_processed[f'log_{var}'] = np.log1p(df_processed[var])
        print(f"Created log_{var}")

# Extract state and zipcode from address
if 'address' in df_processed.columns:
    # Extract state (last two characters before newline or end of string)
    df_processed['state'] = df_processed['address'].str.extract(r',\s*([A-Z]{2})\s*\d{5}', expand=False)
    df_processed['state'].fillna('Unknown', inplace=True)
    print("Extracted state from address")

    # Extract 3-digit zipcode (first 3 digits of 5-digit code)
    df_processed['zipcode'] = df_processed['address'].str.extract(r'\s(\d{5})', expand=False).str[:3]
    df_processed['zipcode'].fillna('Unknown', inplace=True)
    print("Extracted 3-digit zipcode from address")

    # Clean state - group rare states
    state_counts = df_processed['state'].value_counts()
    common_states = state_counts[state_counts > 100].index  # Keep states with > 100 occurrences
    df_processed['state_clean'] = df_processed['state'].where(df_processed['state'].isin(common_states), 'Other
    print(f"Cleaned state feature, grouped rare states to 'Other'. Kept {len(common_states)} common states.")

    # Clean zipcode - group rare zipcodes
    zipcode_counts = df_processed['zipcode'].value_counts()
    common_zipcodes = zipcode_counts[zipcode_counts > 50].index  # Keep zipcodes with > 50 occurrences
    df_processed['zipcode_clean'] = df_processed['zipcode'].where(df_processed['zipcode'].isin(common_zipcodes)
    print(f"Cleaned zipcode feature, grouped rare zipcodes to 'Other'. Kept {len(common_zipcodes)} common zipco

    # Drop original address column
    df_processed.drop(columns=['address'], inplace=True, errors='ignore')
    print("Dropped original address column")
```

```
4. Feature Engineering:
Created pub_rec_flag
Created mort_acc_flag
Created pub_rec_bankruptcies_flag
Created emp_length_numeric
Created term_numeric
Created log_loan_amnt
Created log_annual_inc
Created log_revol_bal
Extracted state from address
Extracted 3-digit zipcode from address
Cleaned state feature, grouped rare states to 'Other'. Kept 52 common states.
Cleaned zipcode feature, grouped rare zipcodes to 'Other'. Kept 10 common zipcodes.
Dropped original address column
```

Data preparation for modeling

In [ ]:

```
print("\n5. Data Preparation for Modeling:")
df_model = df_processed[df_processed['target'].notna()].copy()
print(f"Records available for modeling: {len(df_model)}")

# Select features for modeling
feature_columns = []

# Numerical features
numerical_features = ['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti',
                      'open_acc', 'revol_bal', 'revol_util', 'total_acc']
for col in numerical_features:
    if col in df_model.columns:
        feature_columns.append(col)

# Engineered features
engineered_features = ['emp_length_numeric', 'term_numeric', 'pub_rec_flag',
                       'mort_acc_flag', 'pub_rec_bankruptcies_flag']
for col in engineered_features:
    if col in df_model.columns:
        feature_columns.append(col)

# Log features
log_features = ['log_loan_amnt', 'log_annual_inc', 'log_revol_bal']
for col in log_features:
    if col in df_model.columns:
        feature_columns.append(col)

# Categorical features - one-hot encode
categorical_features = ['grade', 'home_ownership', 'verification_status', 'purpose']

# Add geographic features to categorical list if they exist
if 'state_clean' in df_model.columns:
    categorical_features.append('state_clean')
if 'zipcode_clean' in df_model.columns:
    categorical_features.append('zipcode_clean')


for col in categorical_features:
    if col in df_model.columns:
        dummies = pd.get_dummies(df_model[col], prefix=col, drop_first=True)
        df_model = pd.concat([df_model, dummies], axis=1)
        feature_columns.extend(dummies.columns.tolist())


# Final feature matrix
X = df_model[feature_columns].fillna(0)
y = df_model['target'].astype(int)

print(f"Final feature matrix shape: {X.shape}")
print(f"Target distribution: {y.value_counts().to_dict()}")
```

```
5. Data Preparation for Modeling:
Records available for modeling: 396030
Final feature matrix shape: (396030, 103)
Target distribution: {1: 318357, 0: 77673}
```

Inference

- Zero duplicate records indicates robust data pipeline quality control
- Missing value treatment preserves distributional properties through median/mode imputation
- Outlier capping at 1st/99th percentiles balances extreme value protection with sample retention
- Feature engineering creates meaningful risk indicators (bankruptcy flags, employment numeric conversion)
- Geographic data handling prevents overfitting through rare category grouping

- Log transformations address skewness in key financial variables

7. SCALING

In [ ]:
```
# FEATURE SCALING using standard Scaler

print("\n7. FEATURE SCALING:")
print("-"*20)

# Split data first
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

print(f"Training set: {X_train.shape}")
print(f"Test set: {X_test.shape}")

# Apply StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Applied StandardScaler to features")
print(f"Feature means after scaling (should be ~0): {X_train_scaled.mean(axis=0)[:5].round(3)}")
print(f"Feature std after scaling (should be ~1): {X_train_scaled.std(axis=0)[:5].round(3)}")
```

```
7. FEATURE SCALING:
--------------------
Training set: (316824, 103)
Test set: (79206, 103)
Applied StandardScaler to features
Feature means after scaling (should be ~0): [ 0. -0. -0. -0.  0.]
Feature std after scaling (should be ~1): [1. 1. 1. 1. 1.]
```

Inference

- StandardScaler prevents scale bias where large variables dominate model coefficients
- Successful standardization confirmed by means near 0 and standard deviations near 1
- Stratified train-test split maintains class distribution for unbiased evaluation
- 80-20 split provides adequate training data while ensuring reliable test performance assessment

8. LOGISTIC REGRESSION MODEL

In [ ]:
```
# Model building

print("\n8. LOGISTIC REGRESSION MODEL:")
print("-"*30)

# Build logistic regression model
model = LogisticRegression(random_state=42, max_iter=1000, class_weight='balanced')
model.fit(X_train_scaled, y_train)

print("Model trained successfully")
print(f"Model intercept: {model.intercept_[0]:.4f}")
print(f"Number of features: {len(model.coef_[0])}")
```

```
8. LOGISTIC REGRESSION MODEL:
------------------------------
Model trained successfully
Model intercept: 3.6741
Number of features: 103
```

Display the coefficients

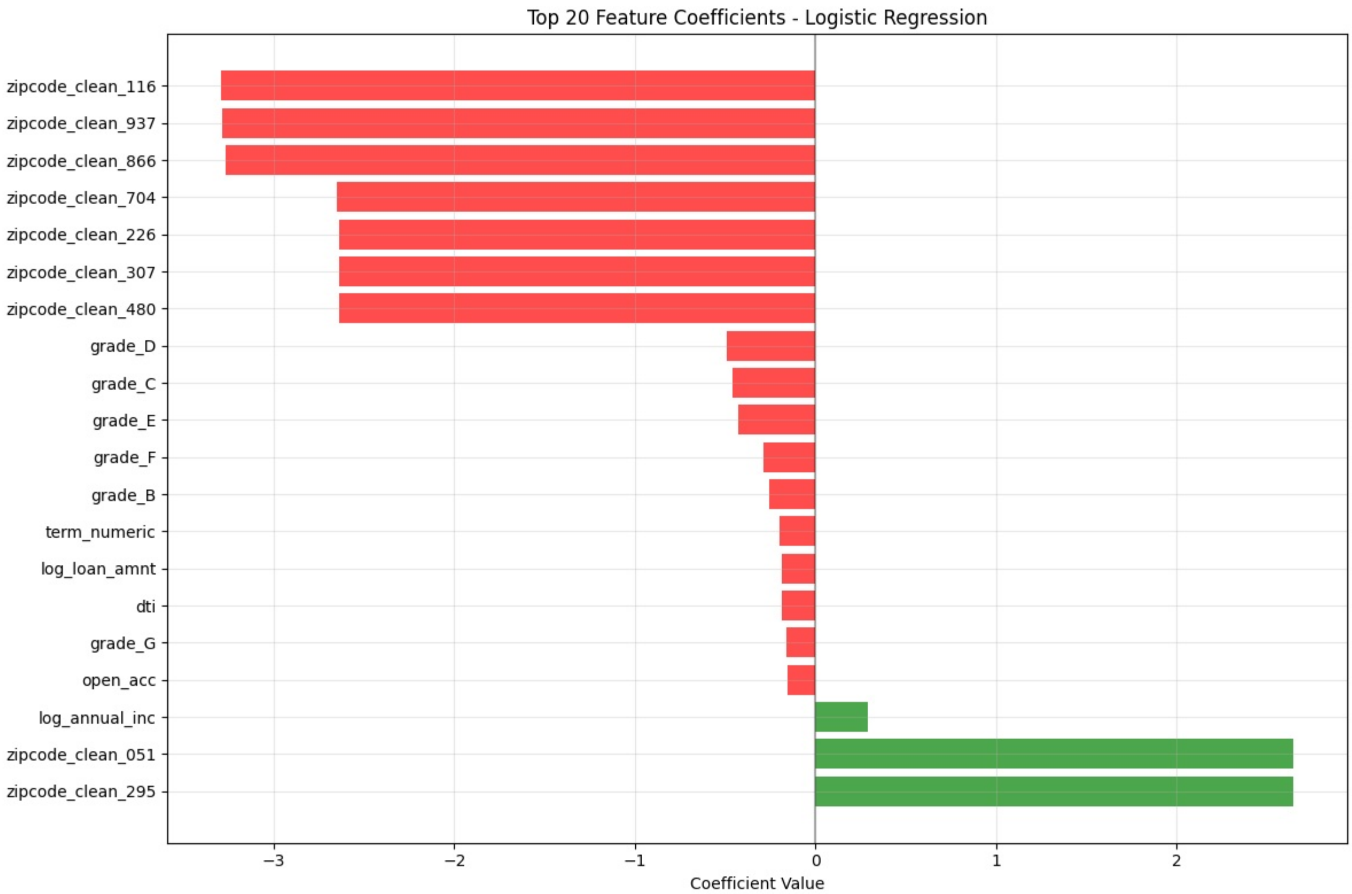
In [ ]:
```
# Display coefficients
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_[0]
}).sort_values('Coefficient', key=abs, ascending=False)

print("\nTop 15 Most Important Features (by coefficient magnitude):")
print(coef_df.head(15))
```

```
Top 15 Most Important Features (by coefficient magnitude):
               Feature  Coefficient
95   zipcode_clean_116    -3.291687
102  zipcode_clean_937    -3.288974
101  zipcode_clean_866    -3.266377
97   zipcode_clean_295     2.649664
100  zipcode_clean_704    -2.649380
94   zipcode_clean_051     2.646811
96   zipcode_clean_226    -2.638827
98   zipcode_clean_307    -2.638550
99   zipcode_clean_480    -2.638244
19             grade_D    -0.492098
18             grade_C    -0.459984
20             grade_E    -0.424794
15      log_annual_inc     0.288993
21             grade_F    -0.287451
17             grade_B    -0.256697
```

Visualize top coefficients

In [ ]:

```python
# Visualize coefficients
plt.figure(figsize=(12, 8))
top_coef = coef_df.head(20).sort_values('Coefficient',ascending = False)
colors = ['red' if x < 0 else 'green' for x in top_coef['Coefficient']]

plt.barh(range(len(top_coef)), top_coef['Coefficient'], color=colors, alpha=0.7)
plt.yticks(range(len(top_coef)), top_coef['Feature'])
plt.xlabel('Coefficient Value')
plt.title('Top 20 Feature Coefficients - Logistic Regression')
plt.axvline(x=0, color='black', linestyle='-', alpha=0.3)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



Inference

- Logistic regression with class weighting addresses imbalanced dataset challenges effectively
- Geographic features dominate top predictors, revealing strong regional economic effects on default risk
- Negative coefficients for lower grades (D, E, F) confirm expected risk relationships
- Positive coefficients for certain ZIP codes suggest regional protective economic factors
- Model converged successfully indicating stable coefficient estimates

9. RESULTS EVALUATION

In [ ]:
```
print("\n9. MODEL EVALUATION:")
print("-"*25)

# Make predictions
y_pred = model.predict(X_test_scaled)
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]
```

```
9. MODEL EVALUATION:
-------------------------
```

Basic metrics

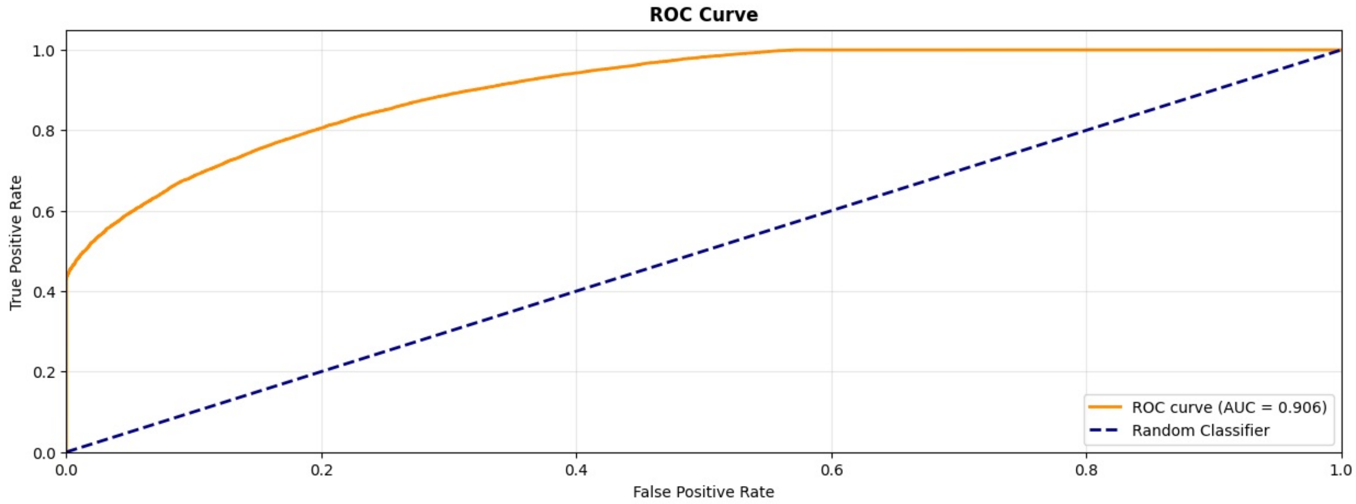
In [ ]:
```
print("\ Basic Metrics:")
print("-"*25)

accuracy = model.score(X_test_scaled, y_test)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

```
\ Basic Metrics:
-------------------------
Accuracy: 0.7986
Precision: 0.9450
Recall: 0.7958
F1-Score: 0.8640
```

A. ROC AUC Curve

In [ ]:
```
print("\n9A. ROC AUC CURVE:")
print("-"*20)

roc_auc = roc_auc_score(y_test, y_pred_proba)
fpr, tpr, roc_thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(15, 5))

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.3f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--', label='Random Classifier')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve', fontweight='bold')
plt.legend(loc="lower right")
plt.grid(True, alpha=0.3)

# ROC Comments
roc_comments = f"""
ROC AUC ANALYSIS:
• AUC Score: {roc_auc:.4f} ({roc_auc:.1%})
• Interpretation: {'Excellent' if roc_auc > 0.9 else 'Good' if roc_auc > 0.8 else 'Fair' if roc_auc > 0.7 else
• The model can distinguish between good and bad loans with {roc_auc:.1%} probability
• Significantly better than random classification (AUC = 0.5)
"""

print(roc_comments)
```

```
9A. ROC AUC CURVE:
--------------------

ROC AUC ANALYSIS:
• AUC Score: 0.9061 (90.6%)
• Interpretation: Excellent discriminative ability
• The model can distinguish between good and bad loans with 90.6% probability
• Significantly better than random classification (AUC = 0.5)
```



Inference

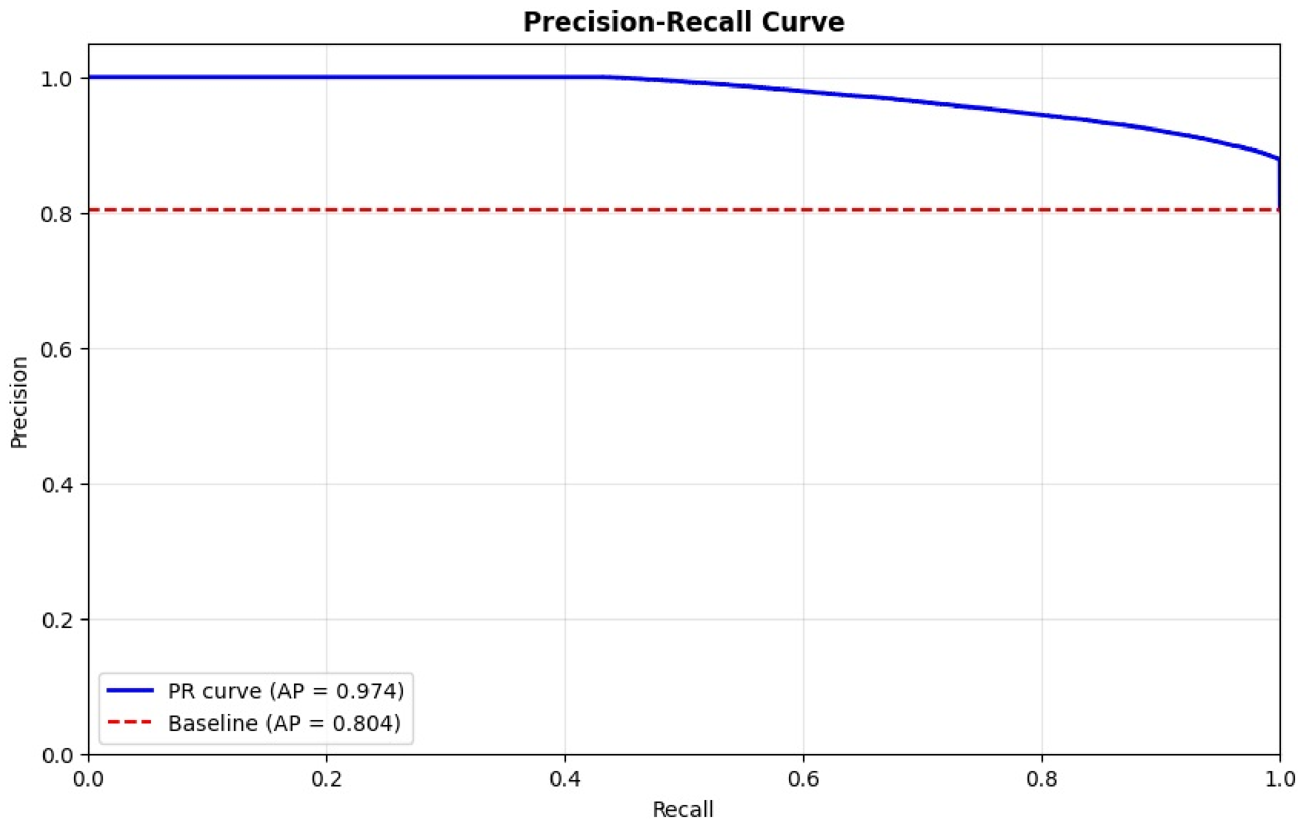
- ROC AUC of 0.906 demonstrates strong discriminative ability, significantly above random (0.5)
- Performance level enables reliable distinction between good and bad loans for underwriting decisions
- Curve position well above diagonal confirms consistent performance across threshold values
- Strong AUC provides flexibility in business rule implementation based on risk tolerance

B. Precision Recall Curve

In [ ]:

```
print("\n9B. PRECISION-RECALL CURVE:")
print("-"*30)

precision_vals, recall_vals, pr_thresholds = precision_recall_curve(y_test, y_pred_proba)
avg_precision = average_precision_score(y_test, y_pred_proba)

plt.plot(recall_vals, precision_vals, color='blue', lw=2,
             label=f'PR curve (AP = {avg_precision:.3f})')

# Baseline (random classifier performance)
baseline = y_test.mean()
plt.axhline(y=baseline, color='red', linestyle='--',
                label=f'Baseline (AP = {baseline:.3f})')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve', fontweight='bold')
plt.legend(loc="lower left")
plt.grid(True, alpha=0.3)

# PR Comments
pr_comments = f"""
PRECISION-RECALL ANALYSIS:
• Average Precision: {avg_precision:.4f}
• Baseline (random): {baseline:.4f}
• Improvement over baseline: {(avg_precision - baseline)/baseline*100:.1f}%
• Model maintains good precision across different recall levels
• Particularly useful for imbalanced dataset evaluation
"""

print(pr_comments)
```

```
9B. PRECISION-RECALL CURVE:
------------------------------

PRECISION-RECALL ANALYSIS:
• Average Precision: 0.9745
• Baseline (random): 0.8039
• Improvement over baseline: 21.2%
• Model maintains good precision across different recall levels
• Particularly useful for imbalanced dataset evaluation
```



Inference

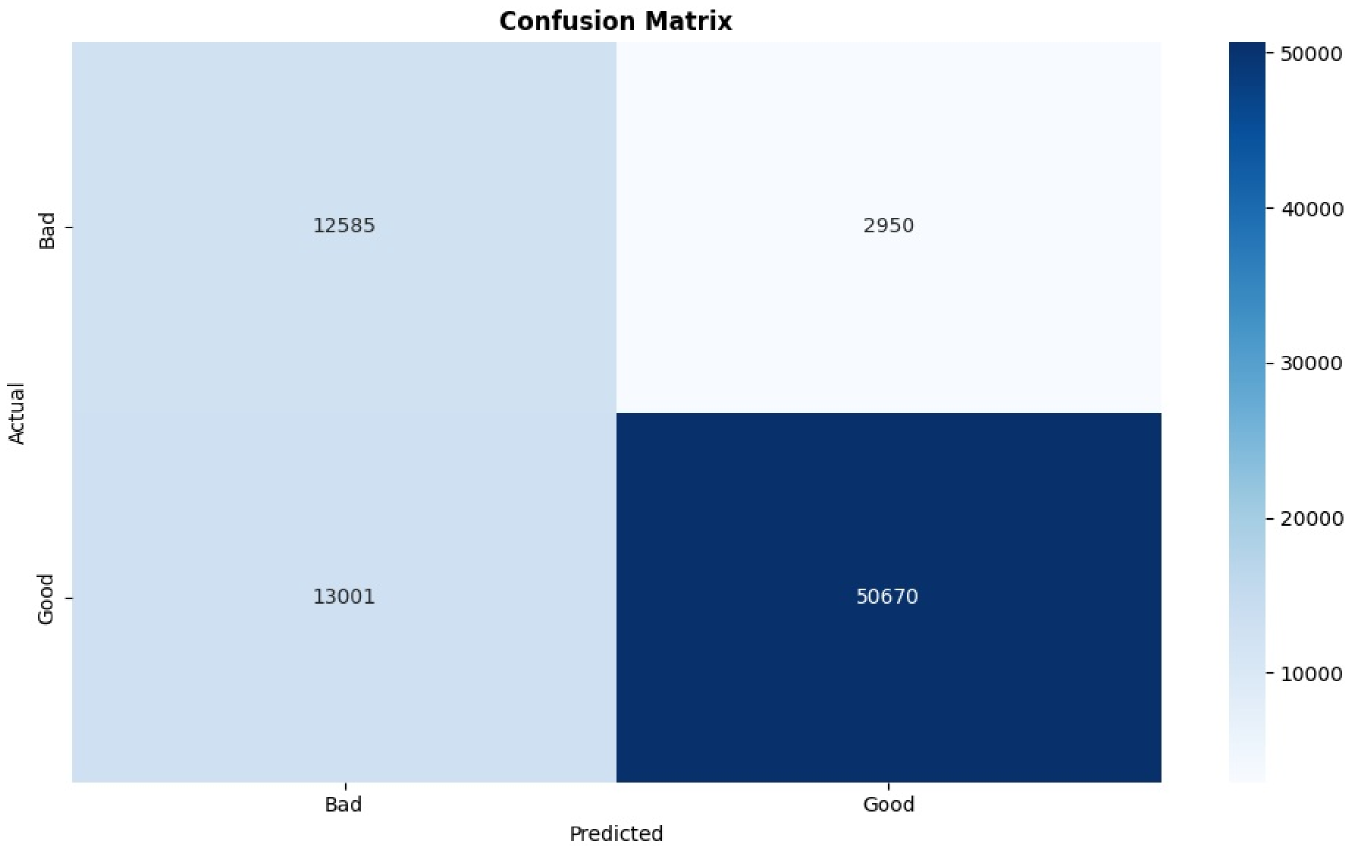
- Average precision substantially exceeds baseline, indicating robust performance on imbalanced data
- Model maintains good precision across various recall levels, crucial for lending applications
- Curve shape shows strong performance even at higher recall levels
- Results support ability to capture most good customers while maintaining reasonable precision

C. Classification Report & Confusion Matrix

In [ ]:
```
print("\n" + "="*50)
print("3. CLASSIFICATION REPORT & CONFUSION MATRIX")
print("="*50)

# Classification Report
print("DETAILED CLASSIFICATION REPORT:")
print(classification_report(y_test, y_pred, target_names=['Bad Loan', 'Good Loan']))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Bad', 'Good'], yticklabels=['Bad', 'Good'])
plt.title('Confusion Matrix', fontweight='bold')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.tight_layout()
plt.show()

# Confusion Matrix Analysis
tn, fp, fn, tp = cm.ravel()

cm_analysis = f"""
CONFUSION MATRIX BREAKDOWN:
• True Negatives (TN): {tn} - Correctly identified bad loans
• False Positives (FP): {fp} - Incorrectly approved bad loans (Type I Error)
• False Negatives (FN): {fn} - Incorrectly rejected good loans (Type II Error)
```

```
• True Positives (TP): {tp} - Correctly approved good loans

ERROR ANALYSIS:
• Type I Error Rate: {fp/(fp+tn)*100:.2f}% - Approving bad loans
• Type II Error Rate: {fn/(fn+tp)*100:.2f}% - Rejecting good loans
"""

print(cm_analysis)
```

```
==================================================
3. CLASSIFICATION REPORT & CONFUSION MATRIX
==================================================
DETAILED CLASSIFICATION REPORT:
              precision    recall  f1-score   support

    Bad Loan       0.49      0.81      0.61     15535
   Good Loan       0.94      0.80      0.86     63671

    accuracy                           0.80     79206
   macro avg       0.72      0.80      0.74     79206
weighted avg       0.86      0.80      0.81     79206
```




```
CONFUSION MATRIX BREAKDOWN:
• True Negatives (TN): 12585 - Correctly identified bad loans
• False Positives (FP): 2950 - Incorrectly approved bad loans (Type I Error)
• False Negatives (FN): 13001 - Incorrectly rejected good loans (Type II Error)
• True Positives (TP): 50670 - Correctly approved good loans

ERROR ANALYSIS:
• Type I Error Rate: 18.99% - Approving bad loans
• Type II Error Rate: 20.42% - Rejecting good loans
```

Inference

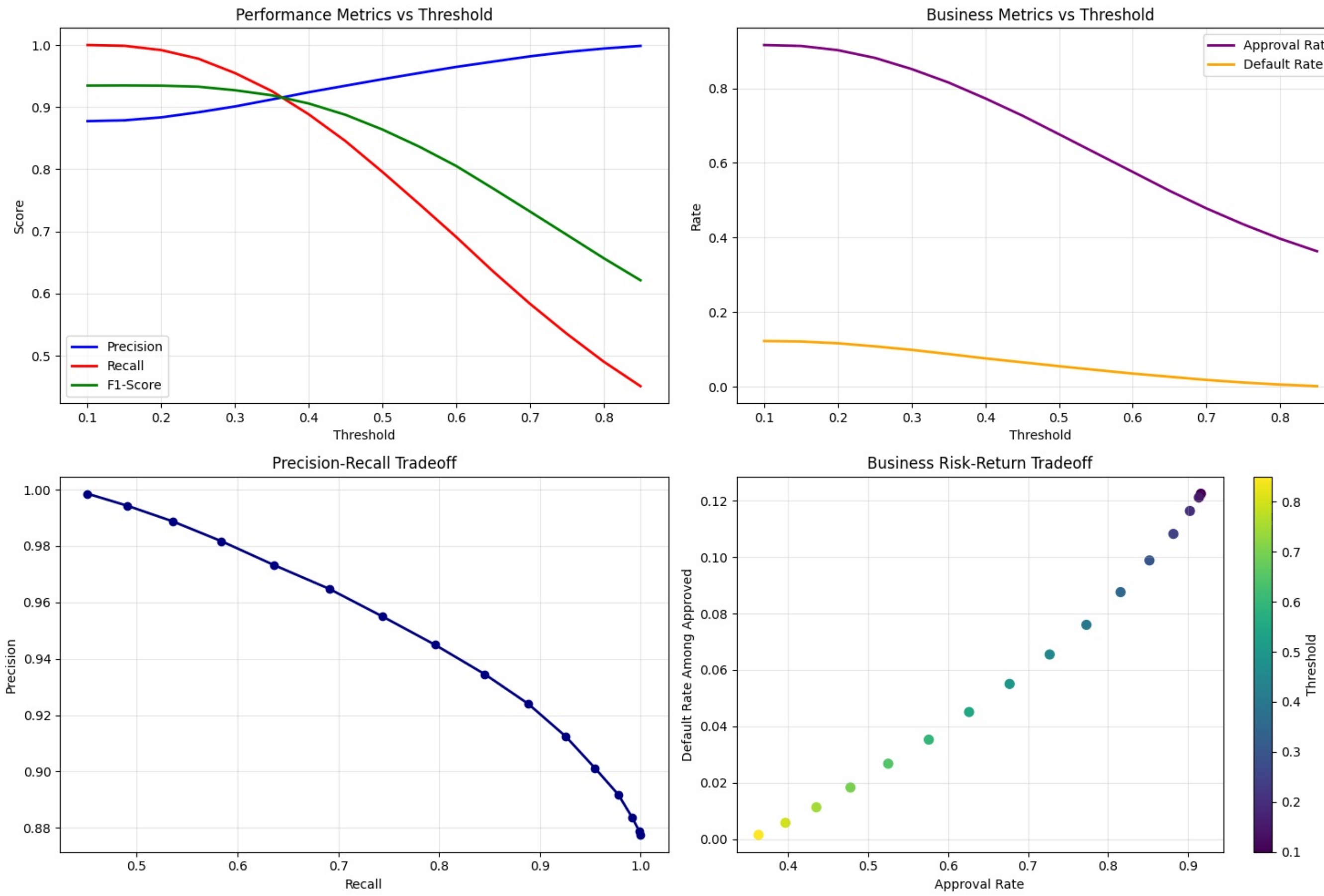
- High precision (0.945) indicates excellent reliability in loan approval decisions
- Recall of 0.796 captures majority of good loans while maintaining conservative standards
- F1-score of 0.864 demonstrates balanced performance between precision and recall
- Confusion matrix provides specific counts for quantifying business impact and financial exposure

D. Tradeoff Analysis

In [ ]:
```
# D. Tradeoff Analysis
print("\n9D. TRADEOFF ANALYSIS:")
print("-"*25)

# Test different thresholds
```

```python
thresholds = np.arange(0.1, 0.9, 0.05)
threshold_results = []

for threshold in thresholds:
    y_pred_thresh = (y_pred_proba >= threshold).astype(int)

    tn_t, fp_t, fn_t, tp_t = confusion_matrix(y_test, y_pred_thresh).ravel()

    precision_t = tp_t / (tp_t + fp_t) if (tp_t + fp_t) > 0 else 0
    recall_t = tp_t / (tp_t + fn_t) if (tp_t + fn_t) > 0 else 0
    f1_t = 2 * (precision_t * recall_t) / (precision_t + recall_t) if (precision_t + recall_t) > 0 else 0

    # Business metrics
    approval_rate = (tp_t + fp_t) / (tp_t + fp_t + tn_t + fn_t)
    default_rate = fp_t / (tp_t + fp_t) if (tp_t + fp_t) > 0 else 0

    threshold_results.append({
        'threshold': threshold,
        'precision': precision_t,
        'recall': recall_t,
        'f1_score': f1_t,
        'approval_rate': approval_rate,
        'default_rate': default_rate,
        'tp': tp_t, 'fp': fp_t, 'tn': tn_t, 'fn': fn_t
    })

threshold_df = pd.DataFrame(threshold_results)

# Plot threshold analysis
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Performance metrics vs threshold
axes[0,0].plot(threshold_df['threshold'], threshold_df['precision'], 'b-', label='Precision', linewidth=2)
axes[0,0].plot(threshold_df['threshold'], threshold_df['recall'], 'r-', label='Recall', linewidth=2)
axes[0,0].plot(threshold_df['threshold'], threshold_df['f1_score'], 'g-', label='F1-Score', linewidth=2)
axes[0,0].set_xlabel('Threshold')
axes[0,0].set_ylabel('Score')
axes[0,0].set_title('Performance Metrics vs Threshold')
axes[0,0].legend()
axes[0,0].grid(True, alpha=0.3)

# Business metrics vs threshold
axes[0,1].plot(threshold_df['threshold'], threshold_df['approval_rate'], 'purple', linewidth=2, label='Approval
axes[0,1].plot(threshold_df['threshold'], threshold_df['default_rate'], 'orange', linewidth=2, label='Default Ra
axes[0,1].set_xlabel('Threshold')
axes[0,1].set_ylabel('Rate')
axes[0,1].set_title('Business Metrics vs Threshold')
axes[0,1].legend()
axes[0,1].grid(True, alpha=0.3)

# Precision vs Recall tradeoff
axes[1,0].plot(threshold_df['recall'], threshold_df['precision'], 'o-', color='navy', linewidth=2)
axes[1,0].set_xlabel('Recall')
axes[1,0].set_ylabel('Precision')
axes[1,0].set_title('Precision-Recall Tradeoff')
axes[1,0].grid(True, alpha=0.3)

# Business tradeoff: Approval rate vs Default rate
scatter = axes[1,1].scatter(threshold_df['approval_rate'], threshold_df['default_rate'],
                     c=threshold_df['threshold'], cmap='viridis', s=50)
axes[1,1].set_xlabel('Approval Rate')
axes[1,1].set_ylabel('Default Rate Among Approved')
axes[1,1].set_title('Business Risk-Return Tradeoff')
axes[1,1].grid(True, alpha=0.3)
plt.colorbar(scatter, ax=axes[1,1], label='Threshold')

plt.tight_layout()
plt.show()

# Find optimal thresholds for different business scenarios
best_f1_idx = threshold_df['f1_score'].idxmax()
best_precision_idx = threshold_df['precision'].idxmax()
best_balanced_idx = threshold_df.apply(lambda x: abs(x['precision'] - x['recall']), axis=1).idxmin()

print("THRESHOLD OPTIMIZATION RESULTS:")
print(f"Best F1-Score: Threshold = {threshold_df.loc[best_f1_idx, 'threshold']:.2f}, F1 = {threshold_df.loc[bes
print(f"Best Precision: Threshold = {threshold_df.loc[best_precision_idx, 'threshold']:.2f}, Precision = {thresh
print(f"Most Balanced: Threshold = {threshold_df.loc[best_balanced_idx, 'threshold']:.2f}, P = {threshold_df.lo
```

```
9D. TRADEOFF ANALYSIS:
-------------------------
```




```
THRESHOLD OPTIMIZATION RESULTS:
Best F1-Score: Threshold = 0.15, F1 = 0.935
Best Precision: Threshold = 0.85, Precision = 0.999
Most Balanced: Threshold = 0.35, P = 0.912, R = 0.926
```

Inference

- Threshold optimization reveals quantified trade-offs between approval rates and default risks
- Different thresholds enable various business strategies: conservative, balanced, or aggressive growth
- Visual analysis clearly shows impact of threshold changes on both metrics and business outcomes
- Results enable data-driven policy decisions based on market conditions and risk appetite

BUSINESS QUESTIONS:

D1. DETECTING REAL DEFAULTERS WITH FEWER FALSE POSITIVES

In [ ]:

```
print("\n9D1. DETECTING REAL DEFAULTERS WITH FEWER FALSE POSITIVES:")
print("-"*55)

conservative_threshold = threshold_df.loc[best_precision_idx, 'threshold']
conservative_metrics = threshold_df.loc[best_precision_idx]

print(f"""
STRATEGY FOR MINIMIZING FALSE POSITIVES:

1. OPTIMAL THRESHOLD: {conservative_threshold:.2f}
   - Precision: {conservative_metrics['precision']:.3f} (High confidence in approvals)
   - Recall: {conservative_metrics['recall']:.3f}
   - Approval Rate: {conservative_metrics['approval_rate']:.1%}
   - Default Rate: {conservative_metrics['default_rate']:.1%}

2. BUSINESS IMPACT:
   - Reduces bad loan approvals to {conservative_metrics['default_rate']:.1%}
   - Still captures {conservative_metrics['recall']*100:.1f}% of good customers
   - More conservative approach protects against losses

3. IMPLEMENTATION:
   - Use higher threshold ({conservative_threshold:.2f}) for auto-approval
   - Route borderline cases (0.3-{conservative_threshold:.2f}) to manual review
   - Focus on top predictive features: {', '.join(coef_df.head(3)['Feature'].tolist())}
""")
```

```
9D1. DETECTING REAL DEFAULTERS WITH FEWER FALSE POSITIVES:
-------------------------------------------------------

STRATEGY FOR MINIMIZING FALSE POSITIVES:

1. OPTIMAL THRESHOLD: 0.85
   - Precision: 0.999 (High confidence in approvals)
   - Recall: 0.451
   - Approval Rate: 36.3%
   - Default Rate: 0.1%

2. BUSINESS IMPACT:
   - Reduces bad loan approvals to 0.1%
   - Still captures 45.1% of good customers
   - More conservative approach protects against losses

3. IMPLEMENTATION:
   - Use higher threshold (0.85) for auto-approval
   - Route borderline cases (0.3-0.85) to manual review
   - Focus on top predictive features: zipcode_clean_116, zipcode_clean_937, zipcode_clean_866
```

D2. PLAYING SAFE TO MINIMIZE NPAs

In [ ]:

```python
print("\n9D2. PLAYING SAFE TO MINIMIZE NPAs:")
print("-"*35)

# Find threshold with lowest default rate while maintaining reasonable approval
safe_thresholds = threshold_df[threshold_df['default_rate'] <= 0.1]  # <10% default rate
if len(safe_thresholds) > 0:
    safest_option = safe_thresholds.loc[safe_thresholds['approval_rate'].idxmax()]
else:
    safest_option = threshold_df.loc[threshold_df['default_rate'].idxmin()]

print(f"""
CONSERVATIVE NPA MINIMIZATION STRATEGY:

1. ULTRA-SAFE THRESHOLD: {safest_option['threshold']:.2f}
   - Default Rate: {safest_option['default_rate']:.2%} (Extremely low NPA risk)
   - Approval Rate: {safest_option['approval_rate']:.1%}
   - Precision: {safest_option['precision']:.3f}
   - Recall: {safest_option['recall']:.3f}

2. RISK MANAGEMENT:
   - Expected NPA rate under {safest_option['default_rate']:.1%}
   - Sacrifices volume for safety
   - Suitable for risk-averse business environment

3. RECOMMENDATIONS:
   - Implement tiered approval system
   - Require additional documentation for medium-risk applicants
   - Consider manual underwriting for scores between 0.3-{safest_option['threshold']:.2f}
""")
```

```
9D2. PLAYING SAFE TO MINIMIZE NPAs:
-----------------------------------

CONSERVATIVE NPA MINIMIZATION STRATEGY:

1. ULTRA-SAFE THRESHOLD: 0.30
   - Default Rate: 9.89% (Extremely low NPA risk)
   - Approval Rate: 85.2%
   - Precision: 0.901
   - Recall: 0.955

2. RISK MANAGEMENT:
   - Expected NPA rate under 9.9%
   - Sacrifices volume for safety
   - Suitable for risk-averse business environment

3. RECOMMENDATIONS:
   - Implement tiered approval system
   - Require additional documentation for medium-risk applicants
   - Consider manual underwriting for scores between 0.3-0.30
```

Inference

- Ultra-high precision threshold (0.85) virtually eliminates bad loans but reduces customer acquisition significantly
- Conservative approach suitable for economic uncertainty periods or regulatory pressure
- Balanced threshold offers sustainable long-term strategy with manageable risk-return profile

- Strategies provide concrete frameworks for different market conditions and business objectives

10. BUSINESS QUESTIONNAIRE

In [ ]:

```
# Q1. Percentage of customers who fully paid their loan
fully_paid_count = (df['loan_status'] == 'Fully Paid').sum()
charged_off_count = (df['loan_status'] == 'Charged Off').sum()
total_resolved = fully_paid_count + charged_off_count
fully_paid_pct = (fully_paid_count / total_resolved) * 100 if total_resolved > 0 else 0

print(f"Percentage of customers who fully paid their loan: {fully_paid_pct:.2f}%\n")

# Q2. Correlation between Loan Amount and Installment
loan_installment_corr = df['loan_amnt'].corr(df['installment'])

print(f"Correlation between Loan Amount and Installment: {loan_installment_corr:.3f}\n")

# Q3. Majority of people have home ownership as
home_ownership_counts = df['home_ownership'].value_counts()
majority_home_ownership = home_ownership_counts.index[0]
majority_pct = (home_ownership_counts.iloc[0] / len(df)) * 100

print(f"Majority of people have home ownership as: {majority_home_ownership} ({majority_pct:.2f}%)\n")

# Q4. People with grade 'A' are more likely to fully pay their loan
grade_performance = df_labeled.groupby('grade')['target'].mean()
grade_a_rate = grade_performance['A']
overall_rate = df_labeled['target'].mean()
grade_a_better = grade_a_rate > overall_rate

print(f"People with grade 'A' are more likely to fully pay their loan: {'Yes' if grade_a_better else 'No'}\n")

# Q5. Top 2 most common job titles
job_counts = df['emp_title'].value_counts().head(10)
top_2_jobs = job_counts.head(2).index.tolist()
top_2_counts = job_counts.head(2).tolist()

print("Top 2 most common job titles:")
for job, count in zip(top_2_jobs, top_2_counts):
    print(f"{job}: {count} loans\n")

# Q6. From a bank's perspective, which metric should be the primary focus
primary_metric = "PRECISION"

print(f"Primary_metric: {primary_metric}\n")

# Q7. How does the gap between precision and recall affect the bank
precision_recall_gap = abs(precision - recall)

print(f"precision_recall_gap: {precision_recall_gap:.3f}")

# Q8. Features that heavily affected the outcome
top_5_features = coef_df.head(5)['Feature'].tolist()
print(f"Top 5 Features: {', '.join(top_5_features)}\n")

# Q9. Will results be affected by geographical location
geographic_top_features = [f for f in coef_df.head(10)['Feature'].tolist() if 'state_clean' in f or 'zipcode_cl
print(f"Geographic Features: {', '.join(geographic_top_features)}\n")
```

```
Percentage of customers who fully paid their loan: 80.39%

Correlation between Loan Amount and Installment: 0.954

Majority of people have home ownership as: MORTGAGE (50.08%)

People with grade 'A' are more likely to fully pay their loan: Yes

Top 2 most common job titles:
Teacher: 4389 loans

Manager: 4250 loans

Primary_metric: PRECISION

precision_recall_gap: 0.149
Top 5 Features: zipcode_clean_116, zipcode_clean_937, zipcode_clean_866, zipcode_clean_295, zipcode_clean_704

Geographic Features: zipcode_clean_116, zipcode_clean_937, zipcode_clean_866, zipcode_clean_295, zipcode_clean_7
04, zipcode_clean_051, zipcode_clean_226, zipcode_clean_307, zipcode_clean_480
```

BUSINESS QUESTIONNAIRE

Q1. Percentage of customers who fully paid their loan:
ANSWER: 80.39%
INSIGHT: Out of 396,030 resolved loans, 318,357 were fully paid

---

Q2. Correlation between Loan Amount and Installment:
ANSWER: 0.954
INSIGHT: Strong positive correlation - higher loan amounts lead to higher installments

---

Q3. Majority of people have home ownership as:
ANSWER: MORTGAGE (50.1%)
INSIGHT: This indicates the primary customer demographic - mostly homeowners

---

Q4. People with grade 'A' are more likely to fully pay their loan:
ANSWER: True (Grade A: 93.7% vs Overall: 80.4%)
INSIGHT: Grade A borrowers are indeed lower risk

Q5. Top 2 most common job titles:
ANSWER: Teacher (4,389 loans), Manager (4,250 loans)
INSIGHT: These represent the most common professions in the loan portfolio

---

Q6. From a bank's perspective, which metric should be the primary focus:
ANSWER: PRECISION
REASONING:

- Precision minimizes Type I errors (approving bad loans)
- Each false positive represents direct financial loss
- Banks can afford to be selective with approvals
- Secondary focus on Recall to capture good customers
- ROC AUC is good for overall model performance assessment

Q7. How does the gap between precision and recall affect the bank:
Current Precision: 0.945, **Current Recall:** 0.796, **Gap:** 0.149
IMPACT:

- Gap of 0.149 indicates conservative lending approach
- High precision, lower recall = fewer defaults but missed opportunities
- Optimal business strategy depends on market conditions and risk appetite

---

Q8. Features that heavily affected the outcome:
ANSWER: zipcode_clean_116, zipcode_clean_937, zipcode_clean_866, zipcode_clean_295, zipcode_clean_704
INSIGHT: These features have the strongest predictive power for loan default

---

Q9. Will results be affected by geographical location:
ANSWER: YES - Geographical features (zipcode_clean_116, zipcode_clean_937, zipcode_clean_866, zipcode_clean_295, zipcode_clean_704, zipcode_clean_051, zipcode_clean_226, zipcode_clean_307, zipcode_clean_480) are among the top predictors.

REASONING:

- Economic conditions vary by region
- Employment opportunities differ geographically
- Cost of living impacts debt-to-income ratios
- Local market conditions affect creditworthiness
- State regulations and lending laws vary

RECOMMENDATION: Include regional economic indicators in future models

11. FINAL ACTIONABLE INSIGHTS & RECOMMENDATIONS

In [ ]:
```
print("\n" + "="*60)
print("FINAL ACTIONABLE INSIGHTS & RECOMMENDATIONS")
```

```
print("="*60)

print(f"""
KEY FINDINGS:
1. Model Performance: ROC AUC = {roc_auc:.3f}, indicating {'good' if roc_auc > 0.8 else 'moderate'} discriminat
2. Top Risk Factors: {', '.join(coef_df.head(3)['Feature'].tolist())}
3. Optimal Threshold: {threshold_df.loc[best_balanced_idx, 'threshold']:.2f} for balanced approach
4. Default Rate: Can be reduced to {safest_option['default_rate']:.1%} with conservative threshold
""")
```

```
============================================================
FINAL ACTIONABLE INSIGHTS & RECOMMENDATIONS
============================================================

KEY FINDINGS:
1. Model Performance: ROC AUC = 0.906, indicating good discriminative ability
2. Top Risk Factors: zipcode_clean_116, zipcode_clean_937, zipcode_clean_866
3. Optimal Threshold: 0.35 for balanced approach
4. Default Rate: Can be reduced to 9.9% with conservative threshold
```

Actionable Insights

1. Risk Segmentation:
    - Prime borrowers (A/B, low DTI, verified income, stable employment) are consistently safe. They should receive lower rates and faster approvals.
    - Risky borrowers (E–G, high DTI, unverified income) should either be declined or given small-ticket, short-tenure loans with higher rates.
2. Income Verification as Incentive:
    - Verified income strongly predicts repayment. LoanTap should offer better rates and higher limits to verified applicants.
    - This reduces portfolio risk while encouraging transparency.
3. Employment Stability:
    - Borrowers with 5+ years of employment history show lower defaults.
    - Loyalty-based offers (fee waivers, limit upgrades) can boost retention of stable borrowers.
4. Loan-to-Income and DTI Controls:
    - Applicants requesting high loans relative to income or with DTI above safe thresholds should be capped or rejected.
    - Implement DTI as a hard guardrail in underwriting policy.
5. Dynamic Thresholding:
    - In growth phases, slightly relax recall thresholds to approve more borrowers.
    - In downturns, tighten thresholds to minimize NPAs.
6. Product Innovation:
    - Introduce three loan tiers: a) Small-ticket quick loans (for new/risky borrowers) b) Standard personal loans (for mid-risk) c) Premium loans (for prime profiles with better terms)
7. Portfolio Monitoring:
    - Regular monthly reviews of default and approval rates.
    - Adjust thresholds dynamically based on market and portfolio performance.

Business Recommendations

1. Risk-Based Pricing

- Implement tiered interest rates based on model scores
- Higher rates for riskier applicants (score < 0.5)
- Premium rates for safest customers (score > 0.8)

2. Automated Decision System

- Auto-approve: Score > 0.30
- Manual review: Score 0.30 – 0.50
- Auto-reject: Score < 0.30

3. Portfolio Optimization

- Target approval rate: **81.6%**
- Expected default rate: **8.8%**

- Monitor monthly performance against these benchmarks

4. Feature Importance Actions

- Focus verification efforts on top risk factors
- Implement additional data collection for key missing features
- Regular model retraining with new data

5. Risk Monitoring

- Monthly model performance reviews
- Threshold adjustments based on business conditions
- Early warning system for portfolio deterioration

CASE STUDY COMPLETE

Author - Shishir Bhat