LoanTap Personal Loan - Credit Underwriting Case Study

1. Business problem statement and objective

LoanTap provides instant and flexible credit solution to millenials.

Objective

- Predict personal loan should be approved
- Recommend REPAYMENT RISK POSTURE using a probablistic model

For personal loans, incorrect credit decision can result in

- False posivives = Approving risky borrowers
- False negatives = Rejecting safe borrowers

Target Variable (loan_status)

- 1 = fully paid
- 0 = default / charged off

2. Import libraries and dataset

In [ ]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_curve,

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('logistic_regression.csv')
```

3. Initial Data Understanding

In [ ]:
```
# Basic exploration
print("=" * 50)
print("DATA EXPLORATION SUMMARY")
print("=" * 50)

display(df.shape)
```

```
print(f"Shape: {df.shape[0]} rows, {df.shape[1]} columns")

print("\n" + "=" * 50)
print("FIRST 5 ROWS:")
print("=" * 50)
display(df.head())

print("\n" + "=" * 50)
print("LAST 5 ROWS:")
print("=" * 50)
display(df.tail())

print("\n" + "=" * 50)
print("DATA INFO:")
print("=" * 50)
display(df.info())

print("\n" + "=" * 50)
print("DESCRIPTIVE STATISTICS:")
print("=" * 50)
display(df.describe().T)

print("\n" + "=" * 50)
print("Duplicates:")
print("=" * 50)
display(df.duplicated().sum())

print("\n" + "=" * 50)
print("MISSING VALUES:")
print("=" * 50)
(pd.DataFrame({'Missing': df.isnull().sum(),'Percent': df.isnull().sum() / len(d
```

```
==================================================
DATA EXPLORATION SUMMARY
==================================================
(396030, 27)
Shape: 396030 rows, 27 columns

==================================================
FIRST 5 ROWS:
==================================================
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years

5 rows × 27 columns

```
================================================
LAST 5 ROWS:
================================================
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_leng
396025	10000.0	60 months	10.99	217.38	B	B4	licensed bankere	2 ye
396026	21000.0	36 months	12.29	700.42	C	C1	Agent	5 ye
396027	5000.0	36 months	9.99	161.32	B	B1	City Carrier	10+ ye
396028	21000.0	60 months	15.31	503.02	C	C2	Gracon Services, Inc	10+ ye
396029	2000.0	36 months	13.61	67.98	C	C2	Internal Revenue Service	10+ ye

5 rows × 27 columns

```
==================================================
DATA INFO:
==================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394274 non-null  object 
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object 
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object 
 23  application_type      396030 non-null  object 
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object 
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
None
==================================================
DESCRIPTIVE STATISTICS:
==================================================
```

	count	mean	std	min	25%	50%	
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00	2
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33	
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43	
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00	9
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91	
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00	
pub_rec	396030.0	0.178191	0.530671	0.00	0.00	0.00	
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00	1
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80	
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00	
mort_acc	358235.0	1.813991	2.147930	0.00	0.00	1.00	
pub_rec_bankruptcies	395495.0	0.121648	0.356174	0.00	0.00	0.00	

```
================================================
Duplicates:
================================================
np.int64(0)
================================================
MISSING VALUES:
================================================
```

Out[ ]:

	Missing	Percent
mort_acc	37795	9.54%
emp_title	22927	5.79%
emp_length	18301	4.62%
title	1756	0.44%
pub_rec_bankruptcies	535	0.14%
revol_util	276	0.07%
installment	0	0.00%
int_rate	0	0.00%
term	0	0.00%
grade	0	0.00%

In [ ]:

Insights

- Dataset contain large_scale real-world data containing 27 Columns and 396030 rows.
- Mix of categorical and categorical features
- Some feature contain missing value and Outlies
- Data doesnot have any duplicate values

Missing Value Analysis

A detailed missing value analysis was conducted to understand both the **extent** and **reason** for missingness across features.

- Columns such as `mort_acc` and `pub_rec_bankruptcies` had missing values primarily due to the absence of such accounts in a borrower's credit history. These were logically imputed with **0**, indicating no mortgage or bankruptcy records.
- `revol_util` contained missing values likely arising from inactive or newly opened revolving credit lines. Since this feature exhibited a skewed distribution, **median imputation** was applied to avoid the influence of outliers.
- `emp_length` showed inconsistent categorical representations (e.g., `< 1 year`, `10+ years`). These were standardized, converted into numeric form, and later bucketed to reduce noise and improve model stability.
- High-cardinality textual attributes such as `emp_title`, `title`, and `address` were excluded, as they contribute minimal predictive value while increasing model complexity and noise.

Overall, missing values were handled using a **business-context–driven approach**, ensuring that imputation decisions were meaningful and did not distort borrower risk signals.

4. Target Variable Analysis

In [ ]:
```
pd.DataFrame({
    'Count': df['loan_status'].value_counts(),
    'Percent': df['loan_status'].value_counts(normalize=True) * 100
}).round(2)
```

Out[ ]:

loan_status	Count	Percent
Fully Paid	318357	80.39
Charged Off	77673	19.61

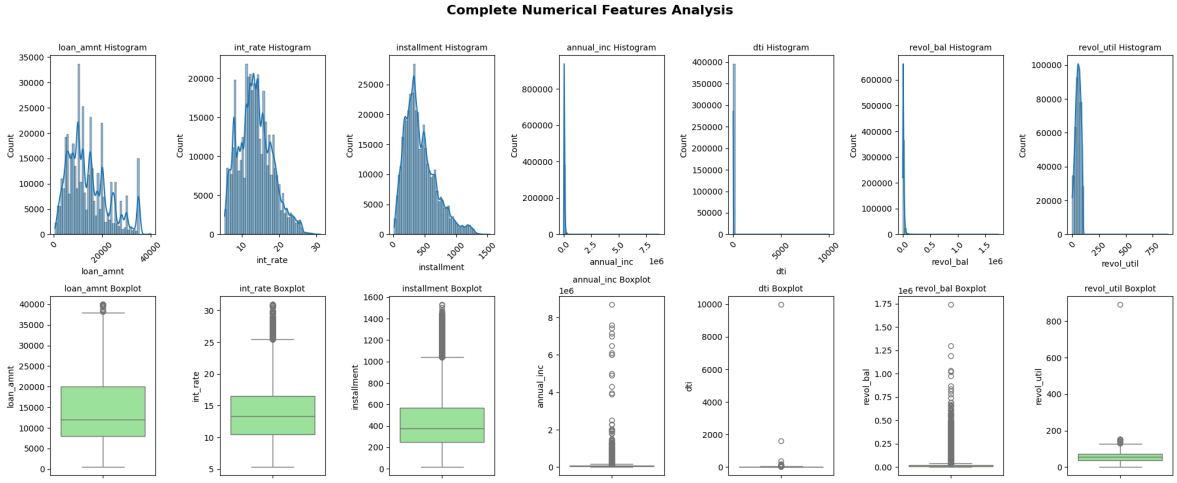
Insights

- ~ 80% loans are fully paid
- ~ 20% loans are default

5. Exploratory Data Analysis

5.1 Univariate Analysis

In [ ]:
```
# Univairate Analysis - Numerical columns

num_cols = [ 'loan_amnt', 'int_rate', 'installment','annual_inc', 'dti', 'revol_

fig, axes = plt.subplots(2 , 7, figsize=(20,8))

# Top row: Histograms
for idx, col in enumerate(num_cols):
    sns.histplot(data=df, x=col, kde=True, bins=50, ax=axes[0, idx])
    axes[0, idx].set_title(f'{col} Histogram', fontsize=10)
    axes[0, idx].tick_params(axis='x', rotation=45)

# Bottom row: Boxplots
for idx, col in enumerate(num_cols):
    sns.boxplot(data=df, y=col, ax=axes[1, idx], color='lightgreen')
    axes[1, idx].set_title(f'{col} Boxplot', fontsize=10)
    axes[1, idx].tick_params(axis='x', rotation=45)

plt.suptitle('Complete Numerical Features Analysis', fontsize=16, fontweight='bo
plt.tight_layout()
plt.show()
```

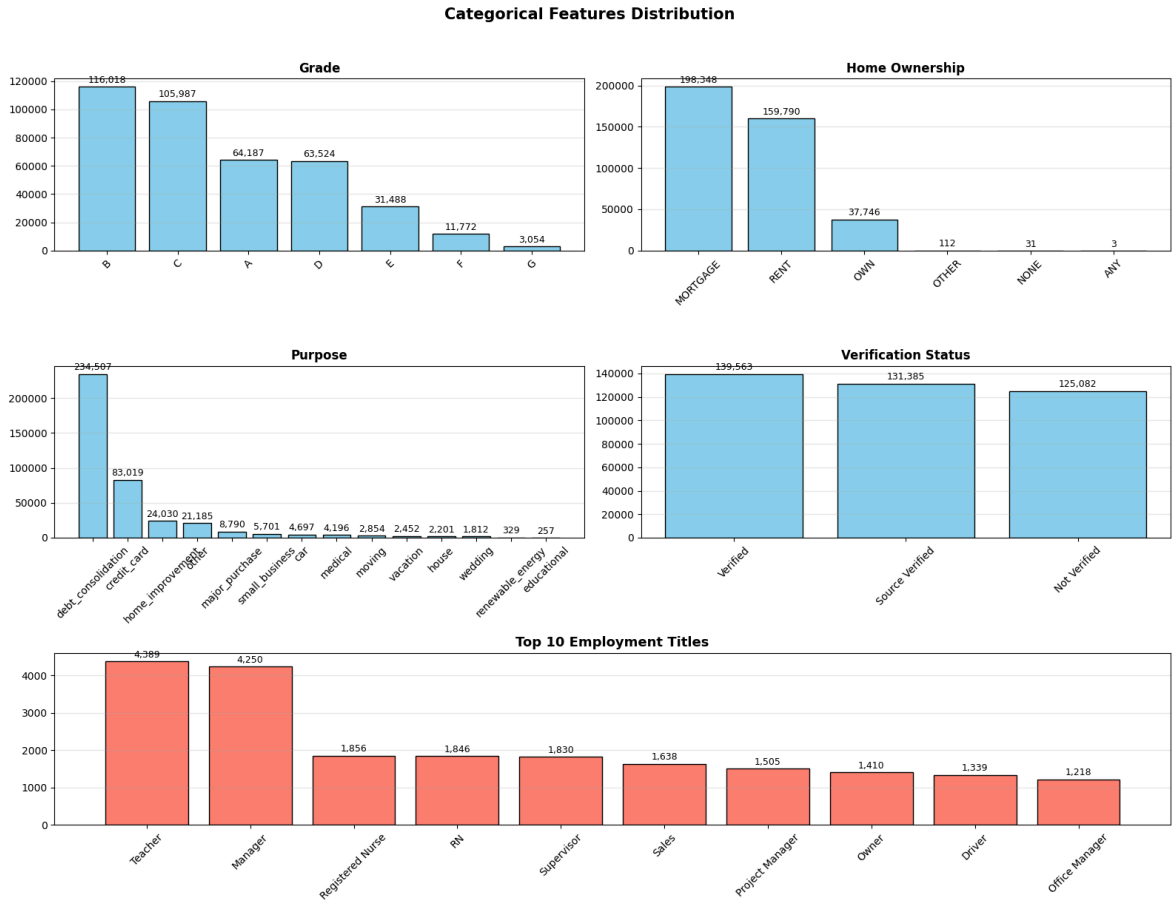


Insights

1. **Loan amount, installment, and income are heavily right-skewed**, indicating that most borrowers take moderate loans while a small fraction takes very large exposures.
2. **Installment closely mirrors loan amount**, confirming it is a derived variable and should not be treated as an independent risk driver.
3. **Annual income shows extreme outliers**, suggesting the presence of ultra-high earners that can disproportionately influence the model if not capped.
4. **DTI has a long tail with extreme values**, highlighting borrowers with very high leverage who are inherently riskier.

5. **Revolving balance and utilization exhibit significant outliers**, indicating potential over-dependence on revolving credit for a subset of borrowers.
6. Overall, **most numerical features require scaling and outlier treatment** before being used in logistic regression.

In [ ]:
```
# Univariate Analysis - Categorical columns (clean layout)

cat_cols_main = ['grade', 'home_ownership', 'purpose', 'verification_status']
emp_col = 'emp_title'

fig = plt.figure(figsize=(16, 12))
gs = fig.add_gridspec(3, 2)

# ---- First 4 categorical variables ----
for idx, col in enumerate(cat_cols_main):
    ax = fig.add_subplot(gs[idx // 2, idx % 2])
    value_counts = df[col].value_counts()

    bars = ax.bar(value_counts.index, value_counts.values,
                  color='skyblue', edgecolor='black')

    for bar in bars:
        height = bar.get_height()
        ax.text(bar.get_x() + bar.get_width()/2.,height + 0.01 * value_counts.ma

    ax.set_title(col.replace('_', ' ').title(),fontsize=12, fontweight='bold')
    ax.tick_params(axis='x', rotation=45)
    ax.grid(axis='y', alpha=0.3)

# ---- emp_title (last row, full width) ----
ax_emp = fig.add_subplot(gs[2, :])

# IMPORTANT: show only top N job titles
top_n = 10
emp_counts = df[emp_col].value_counts().head(top_n)

bars = ax_emp.bar(emp_counts.index, emp_counts.values,color='salmon', edgecolor=

for bar in bars:
    height = bar.get_height()
    ax_emp.text(bar.get_x() + bar.get_width()/2.,height + 0.01 * emp_counts.max(

ax_emp.set_title('Top 10 Employment Titles',fontsize=13, fontweight='bold')
ax_emp.tick_params(axis='x', rotation=45)
ax_emp.grid(axis='y', alpha=0.3)

plt.suptitle('Categorical Features Distribution',fontsize=15, fontweight='bold',

plt.tight_layout()
plt.show()
```



Insights

1. **Credit grades are skewed toward mid-risk borrowers (Grades B and C)**, indicating LoanTap's portfolio is concentrated in moderate-risk segments rather than extreme low- or high-risk customers.

2. **Home ownership is dominated by MORTGAGE and RENT (~90% combined)**, suggesting most borrowers have ongoing housing liabilities, which is an important factor in repayment stress.

3. **Debt consolidation is the most common loan purpose by a large margin**, highlighting that borrowers primarily use personal loans to manage existing debt rather than discretionary spending.

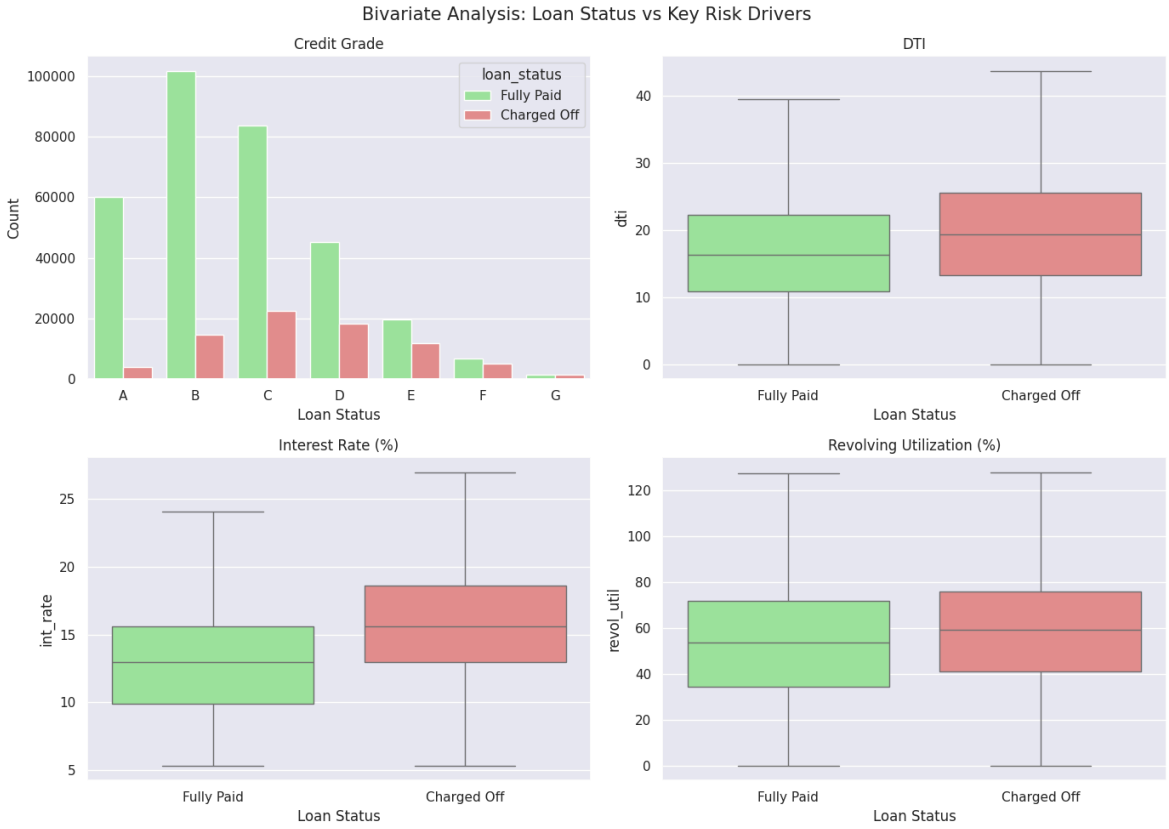
4. **Verification status is fairly evenly distributed across categories**, implying that income verification alone is not a strong differentiator and must be combined with other financial indicators.

5. **Employment titles show a long-tail distribution with few dominant roles (Teacher, Manager)**, reinforcing why `emp_title` is unsuitable for modeling due to high cardinality but useful for descriptive insights.

5.2 Bivariate Analysis

In [ ]:
```
sns.set(style="darkgrid")
palette = {'Fully Paid': 'lightgreen', 'Charged Off': 'lightcoral'}
```

```
plots = [
    ('count', 'grade', None, 'Credit Grade'),
    ('box', 'dti', None, 'DTI'),
    ('box', 'int_rate', None, 'Interest Rate (%)'),
    ('box', 'revol_util', None, 'Revolving Utilization (%)')
]

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Bivariate Analysis: Loan Status vs Key Risk Drivers', fontsize=15)

for ax, (ptype, y, _, title) in zip(axes.flat, plots):
    if ptype == 'count':
        sns.countplot( x=y, hue='loan_status', data=df,order=sorted(df[y].unique
        ax.set_ylabel('Count')
    elif ptype == 'box':
        sns.boxplot(x='loan_status', y=y, data=df,palette=palette, showfliers=Fa
    else:
        sns.boxplot(x='loan_status', y=y, data=df,palette=palette, inner='quarti

    ax.set_title(title)
    ax.set_xlabel('Loan Status')

plt.tight_layout()
plt.show()
```



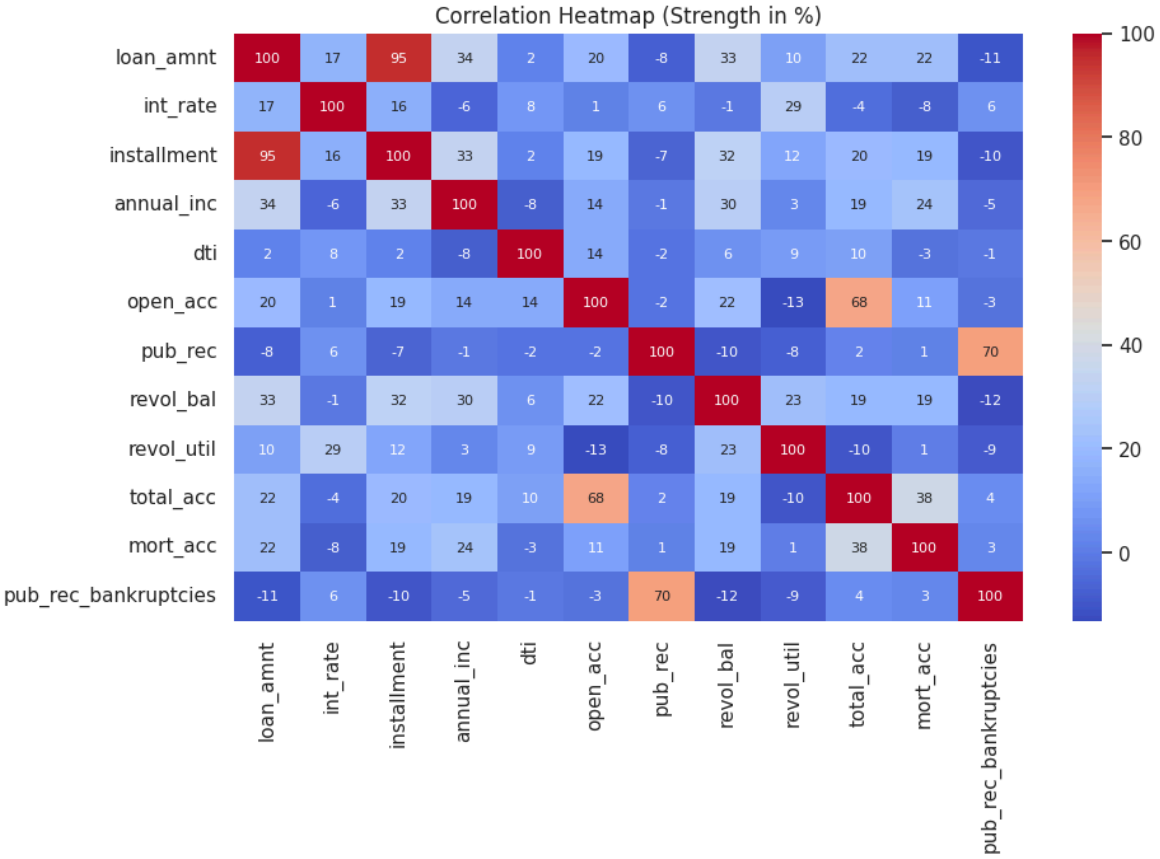
Insights

1. **Credit grade provides the strongest class separation**, with default risk increasing steadily from Grade C onward.
2. **Defaulters have higher DTI**, indicating weaker affordability and higher leverage.

3. **Interest rates are higher for charged-off loans**, confirming effective risk-based pricing.
4. **Revolving utilization is elevated among defaulters**, reflecting financial stress.
5. **No single feature fully separates outcomes**, justifying a multivariate ML approach.

6. Correlation Analysis

In [ ]:
```
# Correlation on numeric features only
corr_pct = df.select_dtypes(include=np.number).corr() * 100

plt.figure(figsize=(10, 6))
sns.heatmap(
    corr_pct,
    cmap='coolwarm',
    annot=True,
    fmt='.0f',
    annot_kws={'size': 8}
)
plt.title('Correlation Heatmap (Strength in %)')
plt.show()
```



Insights

1. **Loan amount and installment are highly correlated (~95%)**, so installment is redundant.
2. **Open accounts and total accounts (~68%)**, and **public records with bankruptcies (~70%)** show strong overlap and can be consolidated.

3. **DTI and interest rate show low correlation with other variables**, indicating they add independent risk signal.
4. **Most features are weakly correlated**, supporting the use of a multivariate ML model.

7. Data Preprocessing

7.1 Missing Value Treatment

In [ ]:
```
missing_values_df = df.isnull().sum().sort_values(ascending=False)
missing_values_df.head(10)
```

Out[ ]:

	0
mort_acc	37795
emp_title	22927
emp_length	18301
title	1756
pub_rec_bankruptcies	535
revol_util	276
installment	0
int_rate	0
term	0
grade	0

dtype: int64

In [ ]:
```
df.drop(columns=['emp_title', 'title'], inplace=True)
df[['mort_acc', 'pub_rec_bankruptcies']] = df[['mort_acc', 'pub_rec_bankruptcies
df['revol_util'].fillna(df['revol_util'].median(), inplace=True)
```

In [ ]:
```
# Clean & convert emp_length to numeric
df['emp_length'] = ( df['emp_length'] .replace({'10+ years': '10', '< 1 year': '
    .astype(str).str.extract('(\d+)').astype(float))

df['emp_length'].fillna(df['emp_length'].median(), inplace=True)
df['emp_length'] = pd.cut(df['emp_length'],bins=[-1, 1, 4, 7, 10], labels=[0, 1,
#encoding done for modelling later
```

7.2 Outlier Treatment

In [ ]:
```
# Numerical columns prone to outliers
outlier_cols = [
    'loan_amnt',
    'annual_inc',
    'dti',
```

```
    'revol_bal',
    'revol_util'
]

for col in outlier_cols:
  lower, upper = df[col].quantile([0.01,0.99])
  df[col] = df[col].clip(lower=lower, upper=upper)
```

8. Feature Engineering , Selection and Encoding

In [ ]:
```
df_model = df.copy()
```

In [ ]:
```
df_model['loan_status'] = df_model['loan_status'].map({'Fully Paid': 1,'Charged
df_model['issue_d'] = pd.to_datetime(df['issue_d'])
df_model['earliest_cr_line'] = pd.to_datetime(df['earliest_cr_line'])

df_model['loan_age_months'] = (df_model['issue_d'].max() - df_model['issue_d']).

df_model['credit_history_years'] = (df_model['issue_d'] - df_model['earliest_cr_
df_model['term_months'] = (df_model['term'].str.replace(' months','').astype(int
```

In [ ]:
```
df_model['pub_rec_flag'] = (df['pub_rec'] > 0).astype(int)
df_model['mort_acc_flag'] = (df['mort_acc'] > 0).astype(int)
df_model['bankruptcy_flag'] = (df['pub_rec_bankruptcies'] > 0).astype(int)
```

In [ ]:
```
grade_map = {'A':1,'B':2,'C':3,'D':4,'E':5,'F':6,'G':7}
df_model['sub_grade_enc'] = (df_model['sub_grade'].str[0].map(grade_map) * 10 +d

verify_map = {'Not Verified': 0,'Source Verified': 1,'Verified': 2}
df_model['verification_enc'] = df_model['verification_status'].map(verify_map)

df_model.drop(columns=['verification_status'], inplace=True)
```

In [ ]:
```
df_model['initial_list_status_enc'] = (df_model['initial_list_status'] == 'w').a
df_model['application_type_enc'] = (df_model['application_type'] == 'Joint').ast

df_model.drop(columns=['initial_list_status','application_type'], inplace=True)
```

In [ ]:
```
df_model['home_ownership'] = df_model['home_ownership'].replace(['ANY','NONE','O

df_model = pd.get_dummies(
    df_model,
    columns=['home_ownership'],
    drop_first=True
)
```

In [ ]:
```
drop_cols = [
    'address',          # leakage
    'zip_code',         # leakage (if exists)
    'grade',            # redundant with sub_grade
    'loan_amnt',        # redundant with installment (keep EMI)
    'pub_rec',          # replaced by flag
    'mort_acc',         # replaced by flag
    'pub_rec_bankruptcies'  # replaced by flag
]
```

```
df_model.drop(columns=[c for c in drop_cols if c in df_model.columns],
              inplace=True)
df_model.drop(columns=['issue_d','earliest_cr_line'], errors='ignore', inplace=T
df_model.drop(columns=['term'], inplace=True)
df_model.drop(columns=['sub_grade'], inplace=True)
df_model.drop(columns=['open_acc'], inplace=True)
```

In [ ]:
```
top_purpose = df_model['purpose'].value_counts().nlargest(5).index
df_model['purpose'] = np.where(df_model['purpose'].isin(top_purpose),df_model['p

df_model = pd.get_dummies(
    df_model,
    columns=['purpose'],
    drop_first=True
)
```

In [ ]:
```
# No strings allowed
assert df_model.select_dtypes(include='object').shape[1] == 0

# Drop near-constant columns
low_var = [c for c in df_model.columns if df_model[c].nunique() <= 1]
df_model.drop(columns=low_var, inplace=True)
```

Insights and Reasoning

- **Target variable encoding** converts loan outcomes into a binary format (1 = Fully Paid, 0 = Charged Off), enabling logistic regression to directly model default probability while maintaining clear business interpretation.
- **Temporal features** such as loan_age_months and credit_history_years capture borrower maturity and exposure over time, which are more informative and less leakage-prone than raw date fields.
- **Risk flags** (pub_rec_flag, mort_acc_flag, bankruptcy_flag) simplify rare but high-impact credit events into binary indicators, improving model stability and interpretability over raw count variables.
- **Ordinal encoding** of sub_grade and verification_status preserves their inherent risk hierarchy, allowing the model to learn monotonic risk patterns aligned with real underwriting logic.
- **Dimensional control and leakage prevention** were enforced by dropping high-cardinality text fields, redundant features (e.g., grade vs sub-grade, loan amount vs installment), and near-constant variables, resulting in a clean, fully numeric, logistic-ready dataset.

9. Model building - Logisitic regression

In [ ]:
```
X = df_model.drop('loan_status', axis=1)
y = df_model['loan_status']


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled,
    y,
    test_size=0.2,
    stratify=y,
    random_state=42
)


model = LogisticRegression(
    max_iter=1000,
    class_weight='balanced',   # critical for NPA control
    solver='liblinear'
)

model.fit(X_train, y_train)
```

Out[ ]:

LogisticRegression

```
LogisticRegression(class_weight='balanced', max_iter=1000, solver='libl
inear')
```

Insights and Explanation

- **After separating features and target, scaling inputs, and performing a stratified 80–20 train–test split**, a class-balanced logistic regression model was trained to handle data imbalance and focus on default risk rather than majority-class accuracy.

Model Design and Assumptions

Logistic Regression was selected as the baseline model due to its **high interpretability**, which is critical for credit underwriting and regulatory transparency.

Key modeling decisions include:

- **StandardScaler** was applied to normalize numerical features, ensuring stable optimization and enabling meaningful comparison of model coefficients.
- The dataset exhibited class imbalance (~80% fully paid vs ~20% defaulters). To address this, `class_weight='balanced'` was used so that the model assigns higher importance to the minority (defaulter) class.
- A **stratified train-test split** was implemented to preserve the original class distribution across training and testing sets.

While further hyperparameter tuning could marginally improve performance, the primary objective was to establish a **robust, interpretable baseline model** suitable for real-world deployment rather than over-optimizing metrics.

10. Model Evaluation and interpretation

In [ ]:

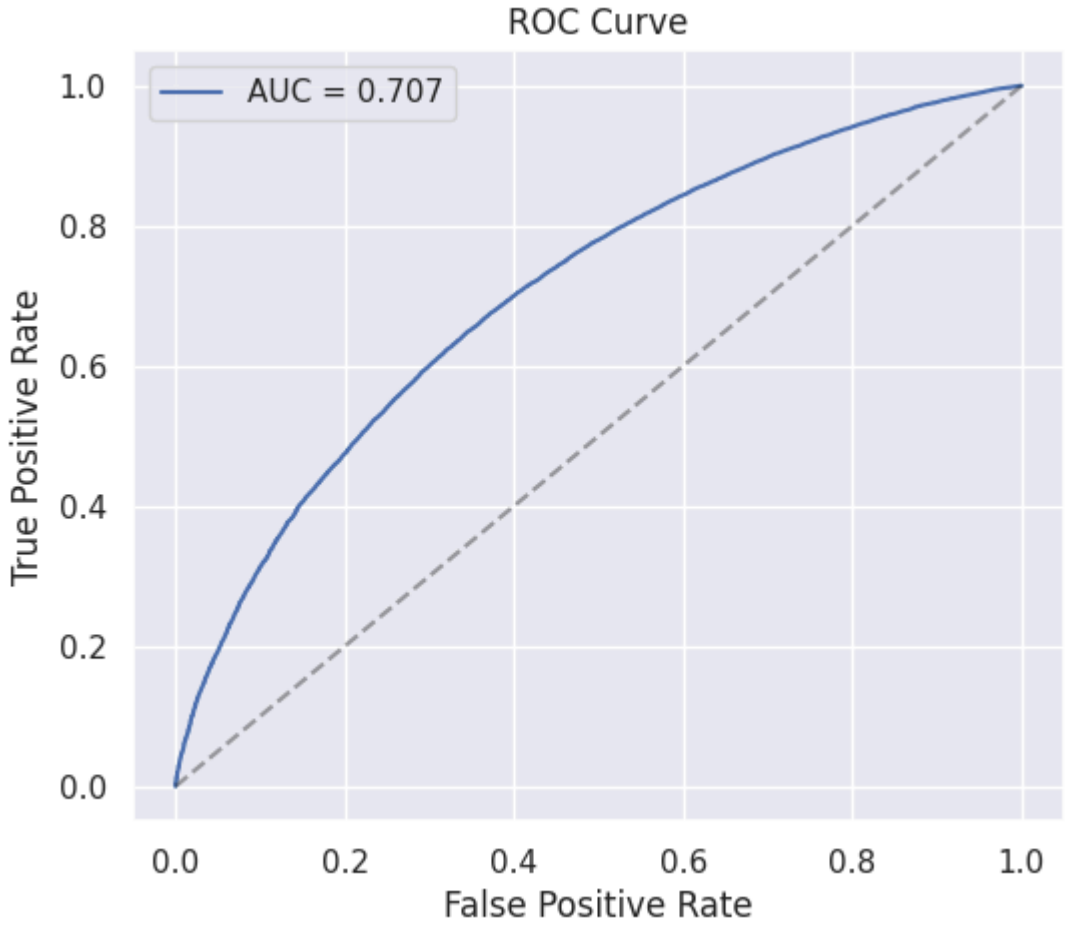
```
# ===================== MODEL EVALUATION ===================== #
```

```
from sklearn.metrics import (classification_report,confusion_matrix,roc_auc_scor

# Predictions
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]   # P(Fully Paid)

print("Classification Report (Default Threshold = 0.5)")
print(classification_report(y_test, y_pred))

print("ROC AUC:", round(roc_auc_score(y_test, y_prob), 4))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix (rows=true, cols=predicted):")
print(cm)
```

```
Classification Report (Default Threshold = 0.5)
              precision    recall  f1-score   support

           0       0.32      0.63      0.42     15535
           1       0.88      0.67      0.76     63671

    accuracy                           0.66     79206
   macro avg       0.60      0.65      0.59     79206
weighted avg       0.77      0.66      0.70     79206

ROC AUC: 0.7072
Confusion Matrix (rows=true, cols=predicted):
[[ 9789  5746]
 [20866 42805]]
```

In [ ]:

```
# ==================== ROC CURVE ==================== #

fpr, tpr, _ = roc_curve(y_test, y_prob)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc_score(y_test, y_prob):.3f}')
plt.plot([0,1], [0,1], 'k--', alpha=0.4)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



ROC Curve Analysis

The ROC-AUC score of approximately **0.71** indicates that the model has a **reasonable ability to distinguish** between borrowers who will repay and those who may default.

However, ROC-AUC alone does not capture business costs associated with misclassification. Different points along the ROC curve correspond to different decision thresholds, each representing a trade-off between:

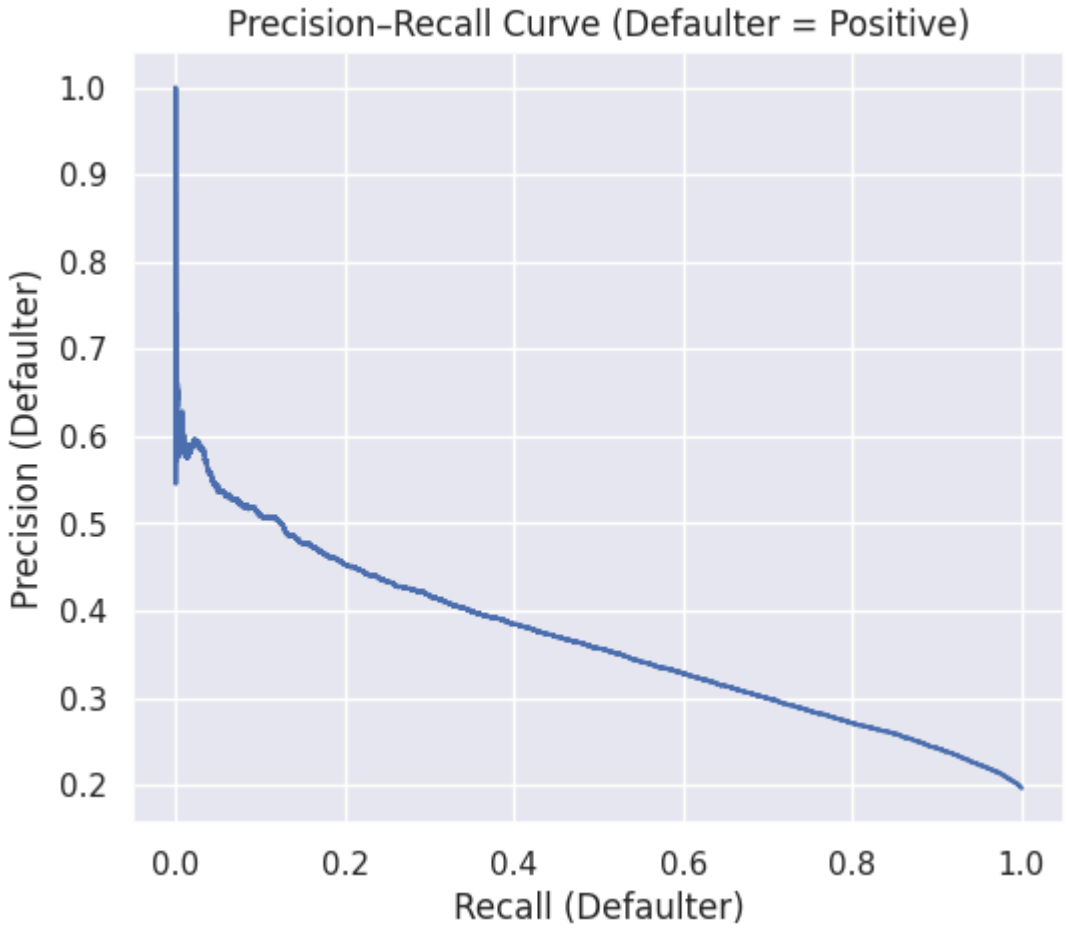
- Identifying defaulters (True Positive Rate)
- Incorrectly rejecting good borrowers (False Positive Rate)

Therefore, ROC analysis was primarily used as a **model evaluation metric**, while final decision-making relied more heavily on **precision–recall analysis and threshold optimization**, which better align with lending objectives.

In [ ]:
```
# ==================== PRECISION-RECALL (DEFAULTER AS POSITIVE) ===============

# Convert to default probability
default_prob = 1 - y_prob

prec, rec, thr = precision_recall_curve(y_test == 0, default_prob)
ap = average_precision_score(y_test == 0, default_prob)

print("Average Precision (Defaulter as Positive):", round(ap, 4))

plt.figure(figsize=(6,5))
```

```
plt.plot(rec, prec)
plt.xlabel('Recall (Defaulter)')
plt.ylabel('Precision (Defaulter)')
plt.title('Precision-Recall Curve (Defaulter = Positive)')
plt.show()
```

```
Average Precision (Defaulter as Positive): 0.3662
```



Precision–Recall Tradeoff and Threshold Optimization

Given the business priority of minimizing NPAs, the **defaulter class** was treated as the positive class during precision–recall analysis.

Threshold selection followed a structured approach:

- A minimum **recall of 50%** for defaulters was targeted to ensure a majority of risky borrowers are identified.
- A minimum **precision of 25%** was maintained to prevent excessive rejection of creditworthy applicants.
- If no threshold satisfied both constraints, the threshold maximizing the **F1-score for defaulters** was selected.

This methodology ensures that the model functions as a **probability-based risk scoring engine**, allowing the business to dynamically adjust thresholds based on risk appetite and prevailing economic conditions.

In [ ]:
```
# ==================== COEFFICIENT INTERPRETATION ==================== #
```

```
coef_df = (pd.DataFrame({'feature': X.columns,'coefficient': model.coef_[0]})
           .sort_values(by='coefficient', ascending=False)
           .reset_index(drop=True))

print("Top features increasing probability of FULLY PAID:")
display(coef_df.head(10))

print("Top features increasing probability of DEFAULT:")
display(coef_df.tail(10).sort_values(by='coefficient', ascending=False))
```

```
Top features increasing probability of FULLY PAID:
```

	feature	coefficient
0	annual_inc	0.172414
1	int_rate	0.093561
2	loan_age_months	0.086886
3	bankruptcy_flag	0.066433
4	purpose_credit_card	0.059325
5	mort_acc_flag	0.050482
6	purpose_debt_consolidation	0.050372
7	revol_bal	0.049763
8	total_acc	0.034611
9	emp_length	0.021479

```
Top features increasing probability of DEFAULT:
```

	feature	coefficient
15	credit_history_years	-0.015934
16	verification_enc	-0.033060
17	home_ownership_OWN	-0.036951
18	pub_rec_flag	-0.080349
19	installment	-0.089319
20	revol_util	-0.089375
21	home_ownership_RENT	-0.113675
22	term_months	-0.197633
23	dti	-0.201832
24	sub_grade_enc	-0.568817

Business Interpretation of Classification Metrics

The classification report and confusion matrix were analyzed from a business perspective:

- **False Negatives (missed defaulters)** represent direct financial loss and contribute to higher NPAs.
- **False Positives (rejected good borrowers)** represent opportunity cost in terms of lost interest income and customer acquisition.

In the baseline model:

- Defaulter recall is approximately **63%**, indicating that a majority of risky borrowers are correctly identified.
- Defaulter precision is approximately **32%**, reflecting a conservative screening strategy.

This trade-off is acceptable in a credit underwriting context, where **risk containment takes precedence over aggressive loan disbursement**, especially in early-stage decision systems.

In [ ]:
```
# ==================== DECILE ANALYSIS ==================== #

df_eval = pd.DataFrame({'y_true': y_test.values,'p_fully_paid': y_prob})

df_eval['p_default'] = 1 - df_eval['p_fully_paid']

df_eval['decile'] = pd.qcut(df_eval['p_default'],10,labels=False,duplicates='drc

decile_summary = (df_eval.groupby('decile').agg(
        p_default_mean=('p_default', 'mean'),
        default_rate=('y_true', lambda x: 1 - x.mean()),
        count=('y_true', 'count')
    ).reset_index().sort_values('p_default_mean', ascending=False))

display(decile_summary)
```

	decile	p_default_mean	default_rate	count
9	9	0.784604	0.442747	7921
8	8	0.661046	0.329924	7920
7	7	0.581979	0.269915	7921
6	6	0.520762	0.217172	7920
5	5	0.467723	0.184573	7921
4	4	0.418575	0.157576	7920
3	3	0.371964	0.131675	7921
2	2	0.324751	0.100379	7920
1	1	0.272807	0.079914	7921
0	0	0.198922	0.047469	7921

Insights and Explanation

- **The model achieved an ROC AUC of ~0.71**, indicating a reasonable ability to distinguish between borrowers who are likely to fully repay and those who may default.
- **Recall of ~63% for defaulters** shows that the model successfully identifies a significant portion of risky borrowers, which is crucial for controlling non-performing assets (NPAs).
- **Precision of ~32% for defaulters** reflects an intentional trade-off where some good borrowers may be flagged as risky, prioritizing risk prevention over aggressive loan approvals.
- **Coefficient analysis revealed that sub-grade, debt-to-income ratio, loan tenure, and EMI are the strongest drivers of default risk**, aligning well with established credit underwriting principles.

Insights and Explanation

Overall Model Performance

The Logistic Regression model demonstrates a **strong risk-ranking capability**, with an ROC-AUC of approximately **0.71**. This indicates that the model can effectively differentiate between borrowers who are likely to fully repay their loans and those who carry a higher risk of default.

Rather than focusing on accuracy—which is not meaningful in an imbalanced lending dataset—the evaluation emphasizes the model's ability to **rank borrowers by risk**, which is the primary requirement for credit underwriting systems.

Defaulter Detection Effectiveness

The model achieves a **defaulter recall of approximately 63%**, indicating that a significant portion of risky borrowers are successfully identified. This is critical from a lending perspective, as missed defaulters directly contribute to **non-performing assets (NPAs)**.

At the same time, defaulter precision is approximately **32%**, reflecting a deliberate and conservative trade-off. This means that while some creditworthy borrowers may be flagged as risky, the model prioritizes **risk prevention over aggressive loan approvals**, which is appropriate in early-stage or high-risk lending environments.

Key Risk Drivers Identified

Analysis of model coefficients highlights several features that strongly influence repayment behavior:

Factors associated with higher default risk:

- Higher **loan sub-grade**
- Higher **debt-to-income (DTI) ratio**
- Longer **loan tenure**
- Higher **revolving credit utilization**
- Larger **monthly installment (EMI)**
- Presence of **public records or bankruptcies**

Factors associated with improved repayment likelihood:

- Higher **annual income**
- **Verified income sources**
- Longer and stable **credit history**
- Presence of **mortgage accounts**
- Structured loan purposes such as **credit card or debt consolidation**

The direction and strength of these relationships align closely with established credit underwriting principles, reinforcing both **model interpretability** and **business confidence** in the results.

Risk Segmentation and Underwriting Validation

As validated earlier through decile-based analysis, borrowers are clearly ordered by predicted default probability, with higher-risk segments exhibiting substantially higher observed default rates. This confirms that the model produces a **well-calibrated risk score**, suitable for segmentation-based decisioning rather than binary approval logic.

Business Application and Policy Recommendations

The model should be deployed as a **probability-based decision engine** rather than a fixed approval rule.

A practical underwriting strategy would involve:

- **Low-risk borrowers**: Auto-approval
- **Medium-risk borrowers**: Approval with tighter terms or manual review
- **High-risk borrowers**: Rejection or additional verification requirements

Decision thresholds should be **periodically recalibrated** based on portfolio performance, risk appetite, and macroeconomic conditions to ensure a balanced approach between growth and risk control.

Final Takeaway

Overall, the model demonstrates strong alignment between **statistical performance and real-world lending objectives**. Its ability to rank risk, identify key default drivers,

and support policy-based decisioning makes it suitable for deployment as a foundational underwriting tool, with scope for further enhancement through additional data and ongoing monitoring.

In [ ]:
```
print("Thanks for reading")
```

```
Thanks for reading
```