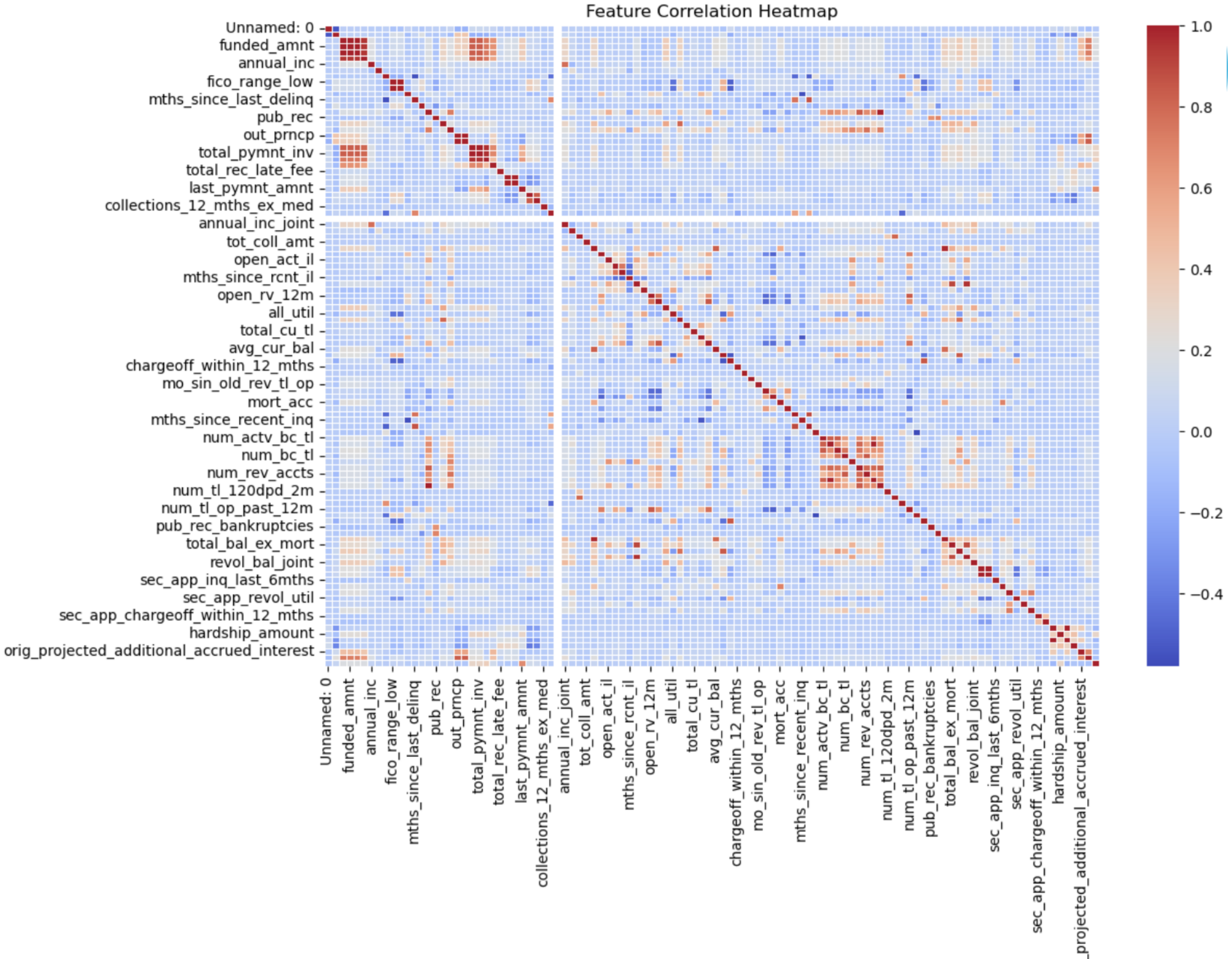
# Predicting loan default with Machine Learning

Victor Jansen

# Correlation heatmap

I use a correlation heatmap to identify and remove highly correlated features. This helps prevent multicollinearity, ensuring the model learns from diverse, independent factors rather than redundant information



# Feature engeneering

**By carefully inspecting the dataset and column descriptions, I identified and removed additional irrelevant or redundant features. This improved model efficiency, reduced noise, and enhanced predictive performance**

| | |
|---|---|
| **Prone to Data Leakage:** | Features that reveal future payment outcomes<br>loan status updates<br>hardship records |
| **Irrelevant to Analysis:** | Unnamed: 0<br>ID |
| **Features with only one unique value** | pymnt_plan |
| **Specific to Joint Loans:** | Verification_status_joint<br>Sec_app_mort_acc |

# Filtering & Data Preperation

- Filter
  - individual loans -['application_type'] == 'Individual’]
  - Status -["Fully Paid", "Charged Off”]
- Fill missing values
- Convert date-time
- Convert categorical to numerical values

```
# Define grade ranking
grade_mapping = {'A': 7, 'B': 6, 'C': 5, 'D': 4, 'E': 3, 'F': 2, 'G': 1}
```

```
# Convert '< 1 year' to 0.5 years and '10+ years' to 11, handle 'Unknown' as NaN
df_cleaned['emp_length'] = df_cleaned['emp_length'].replace({'10+ years': '11', '< 1 year': '0.5', 'Unknown': np.nan})

# Extract numbers and convert to float
df_cleaned['emp_length'] = df_cleaned['emp_length'].str.extract('(\d+)').astype(float)
```

# Measuring predictive value of features

- Define X and y features for Mutual Info Classifier
- Run MI Classifier and set a threshold
- Threshold set to 0.002
- Final column count = 51

| | | |
|---|---|---|
| 13 | total_acc | 0.000199 |
| 46 | num_il_tl | 0.000188 |
| 53 | tax_liens | 0.000059 |
| 48 | num_tl_120dpd_2m | 0.000047 |
| 49 | num_tl_90g_dpd_24m | 0.000000 |
| 16 | collections_12_mths_ex_med | 0.000000 |
| 30 | total_cu_tl | 0.000000 |
| 33 | chargeoff_within_12_mths | 0.000000 |
| 12 | revol_bal | 0.000000 |
| 7 | delinq_2yrs | 0.000000 |

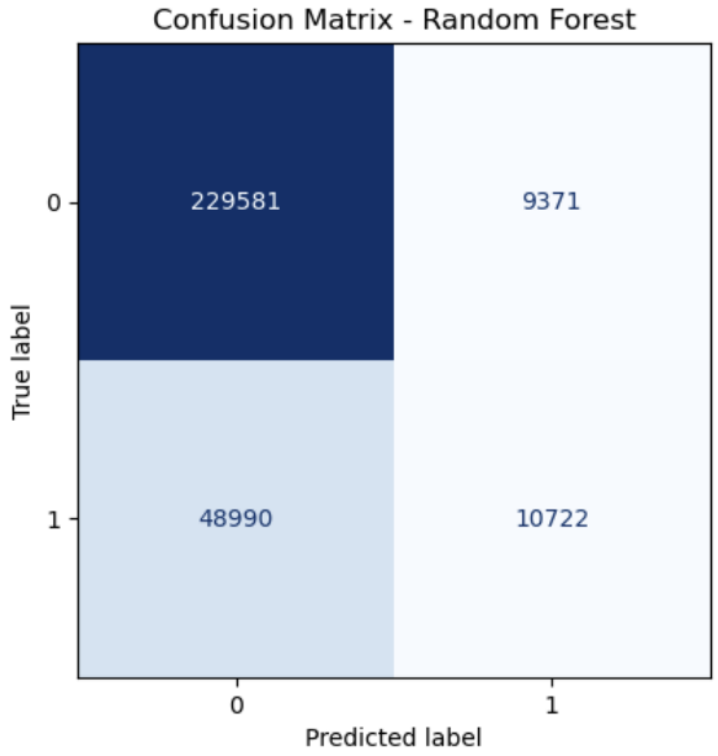
```python
# Define MI threshold
THRESHOLD = 0.002  #increase later if needed

# Select numerical features with MI Score above threshold
important_num_features = mi_scores_df[mi_scores_df['MI Score'] > THRESHOLD]['Feature'].tolist()

# Identify dropped features (features that were in num_cols but not in important_num_features)
dropped_features = list(set(num_cols) - set(important_num_features))
```

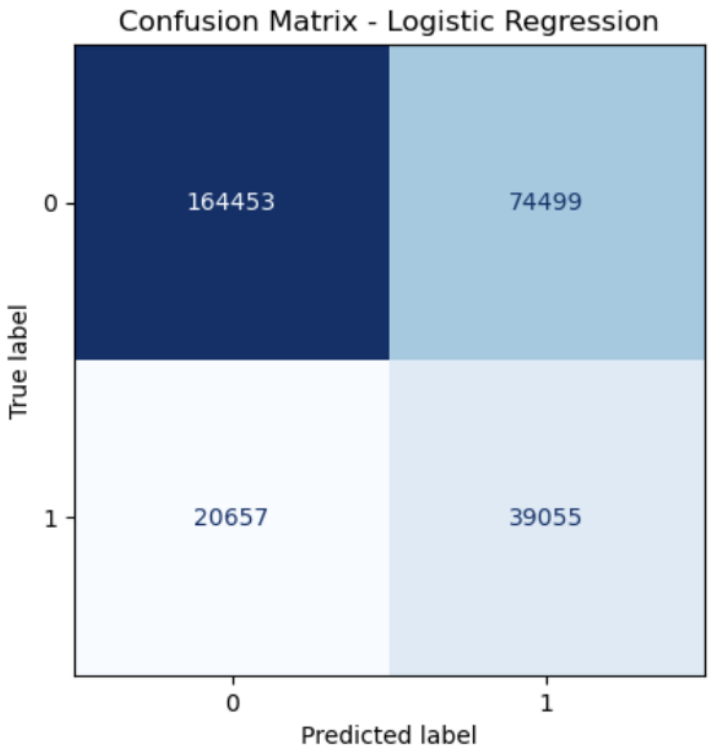
# Pipeline

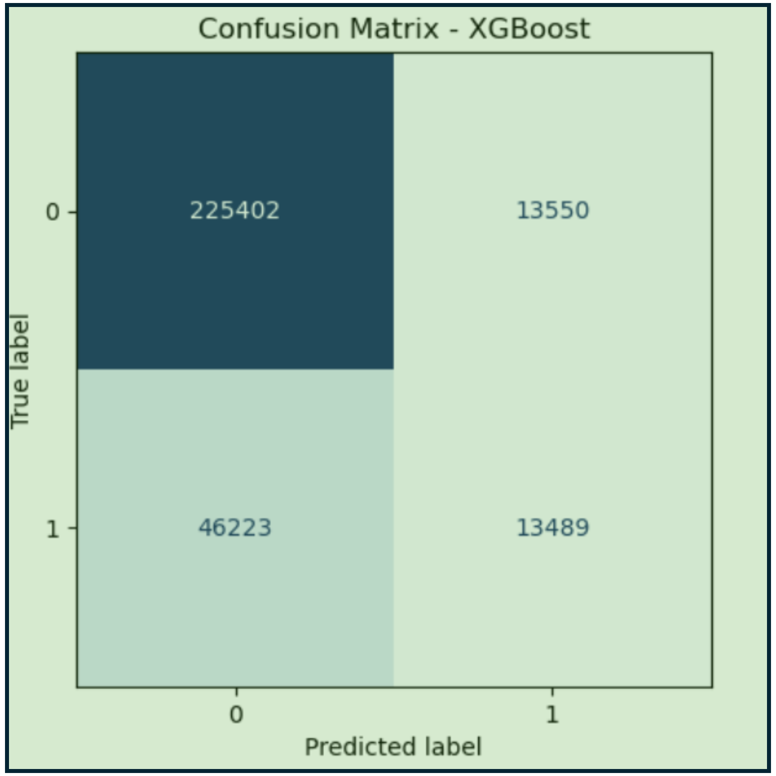
- Traintest spilt using SMOTE balancing
- Transform columns
- Test different models models: Regression, Random Forrest, Xgboost
- Hyper parameter tuning

| | Model | Accuracy | Precision | Recall | F1-score | ROC-AUC |
|---|---|---|---|---|---|---|
| 0 | Logistic Regression | 0.684032 | 0.778486 | 0.684032 | 0.712714 | 0.73206 |
| 1 | Random Forest | 0.801441 | 0.762059 | 0.801441 | 0.763946 | 0.71837 |
| 2 | XGBoost | 0.800625 | 0.764041 | 0.800625 | 0.768428 | 0.72975 |

# Confusion matrixes



# Model discussion

Baseline accuracy is 80% (238,952 / 298,664) if we assume all loans will be paid, matching the model's performance.

So the model does not outperform accepting each loan

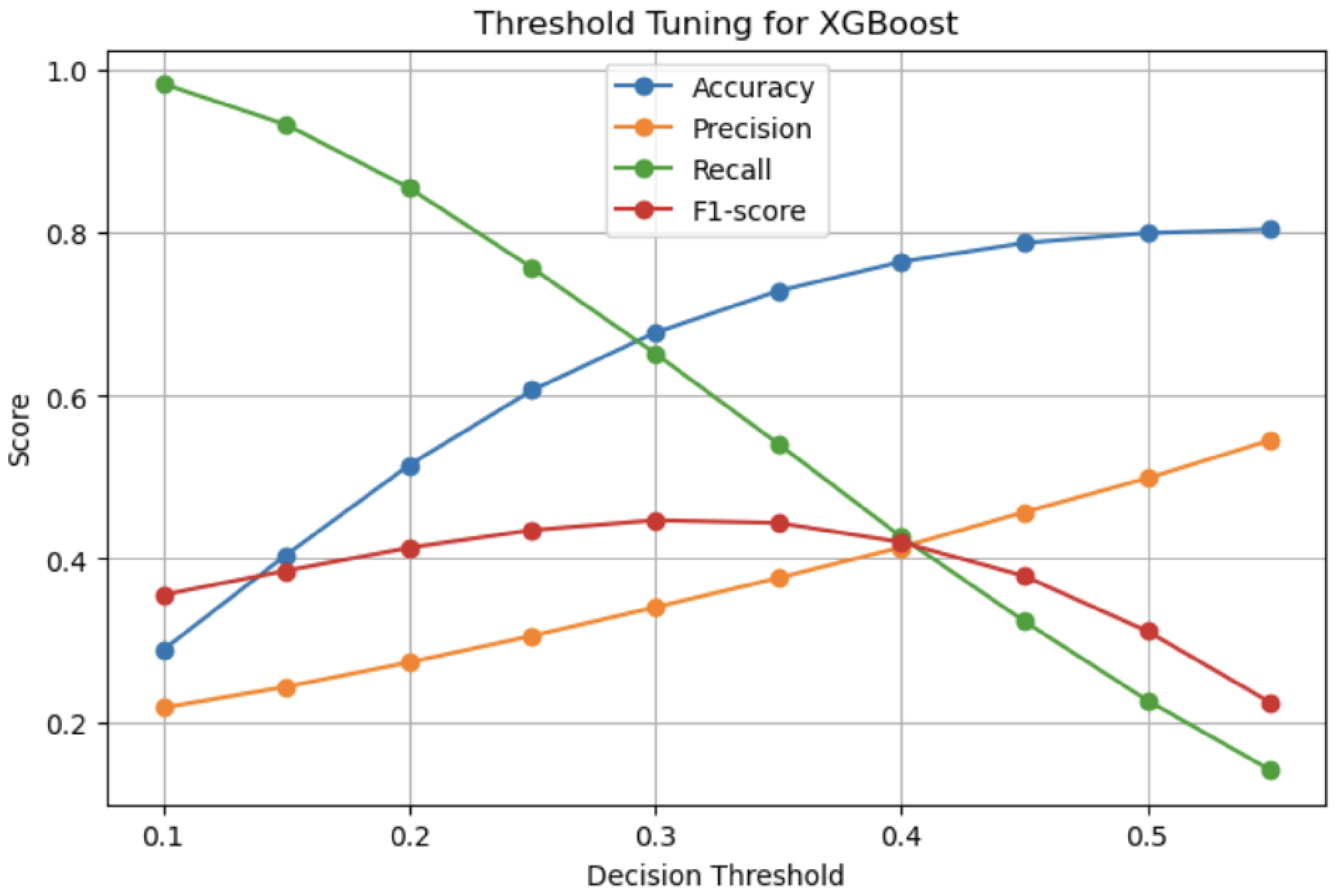
However the cost of approving a defaulting loan is much higher than rejecting a fully paid one.

The model identifies high-risk loans, allowing lenders to mitigate financial risk through higher interest rates, further verification, or rejection.

Even though not perfect, the model provides valuable insights that can reduce losses and improve decision-making.

We can further improve the model by choosing the right threshold

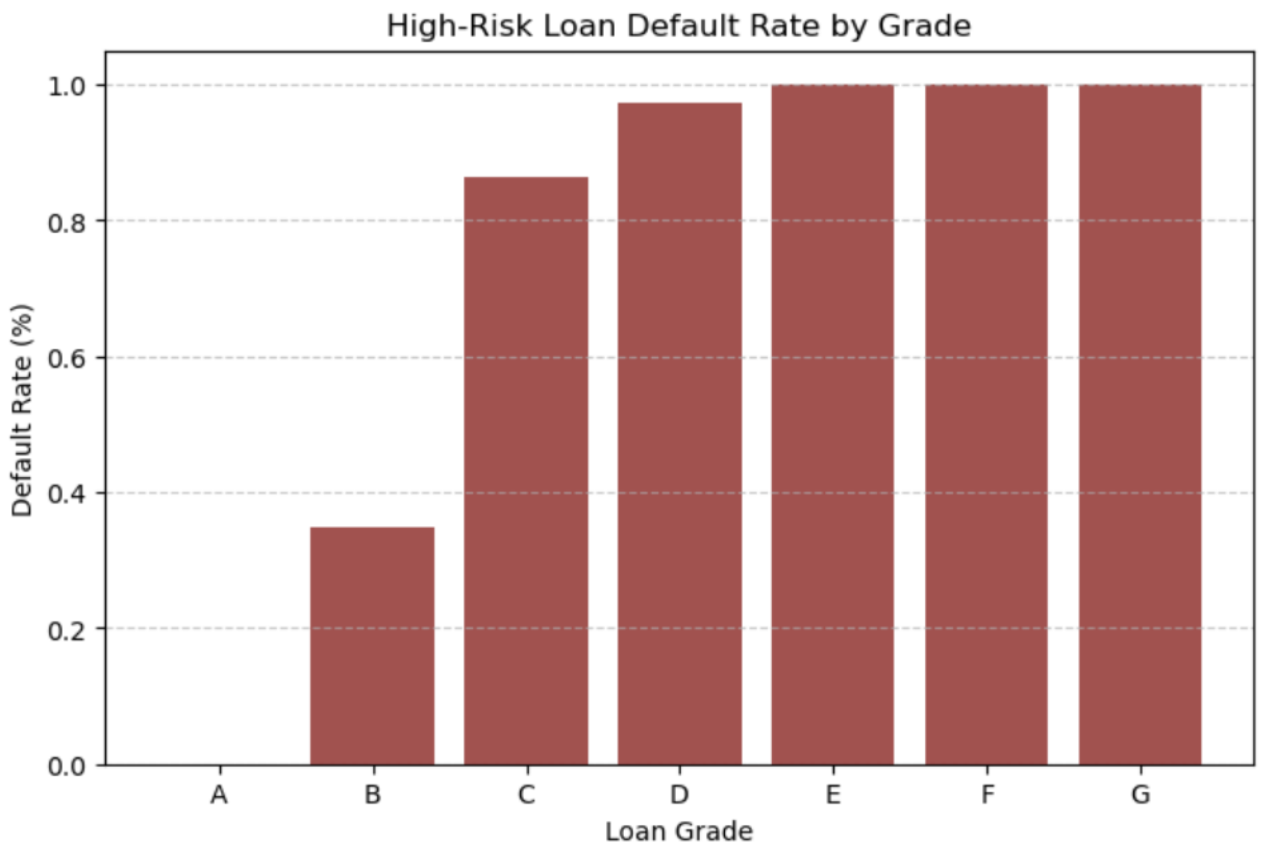
# Threshold tuning



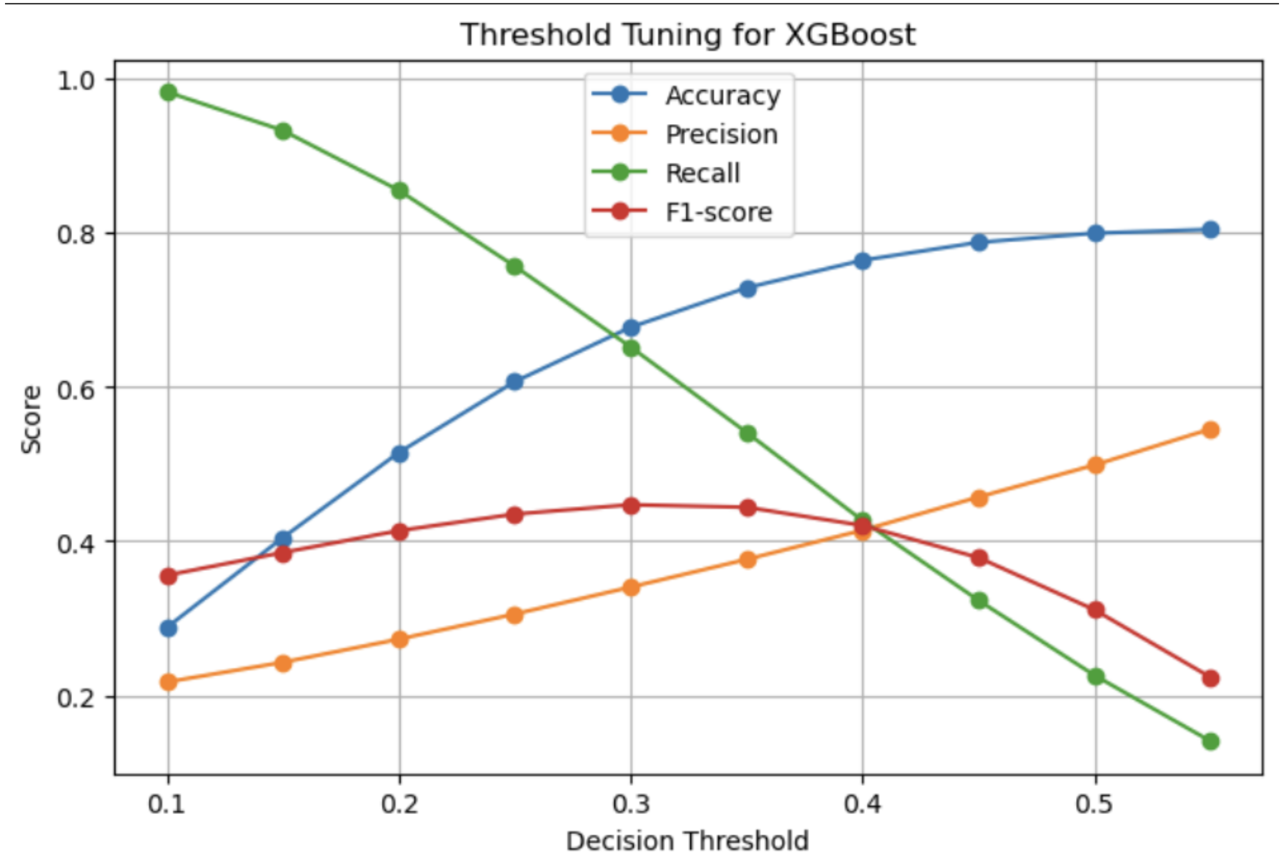

First I chose a 0.25 threshold for the best recall-accuracy trade-off. With **75.7% recall**, we catch most defaults while maintaining **60.7% accuracy** to limit false positives. This is a more risk averse strategy but will lead to limited unpaid loans.

# Advise (.25 threshold)

At a 0.25 threshold loans graded E, F, and G are almost always predicted to default, making them the riskiest. Using this threshold I would suggest only in less risky a grade loans.

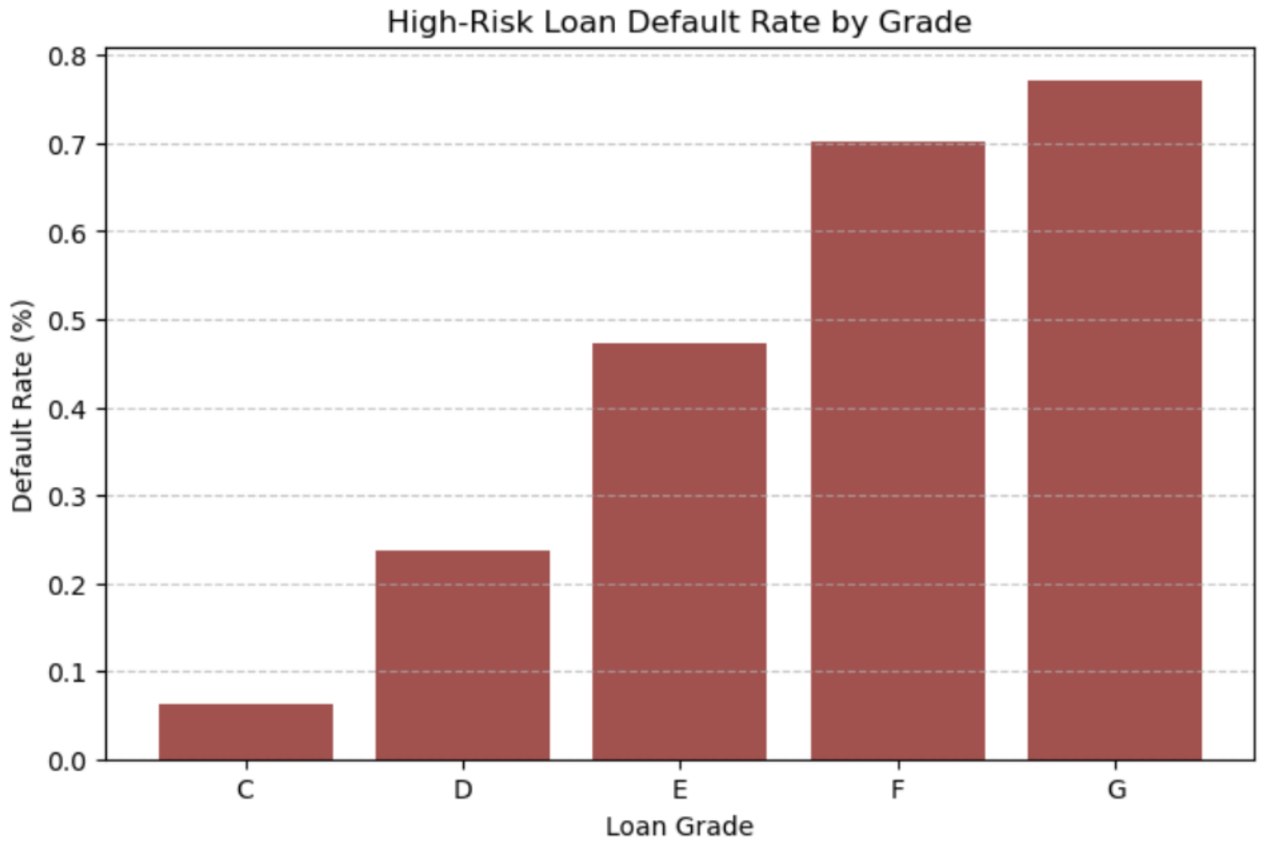


If we prioritize higher accuracy (0.8), we increase the threshold to 0.5. This reduces false positives but will increase the chance of a loan defaulting.



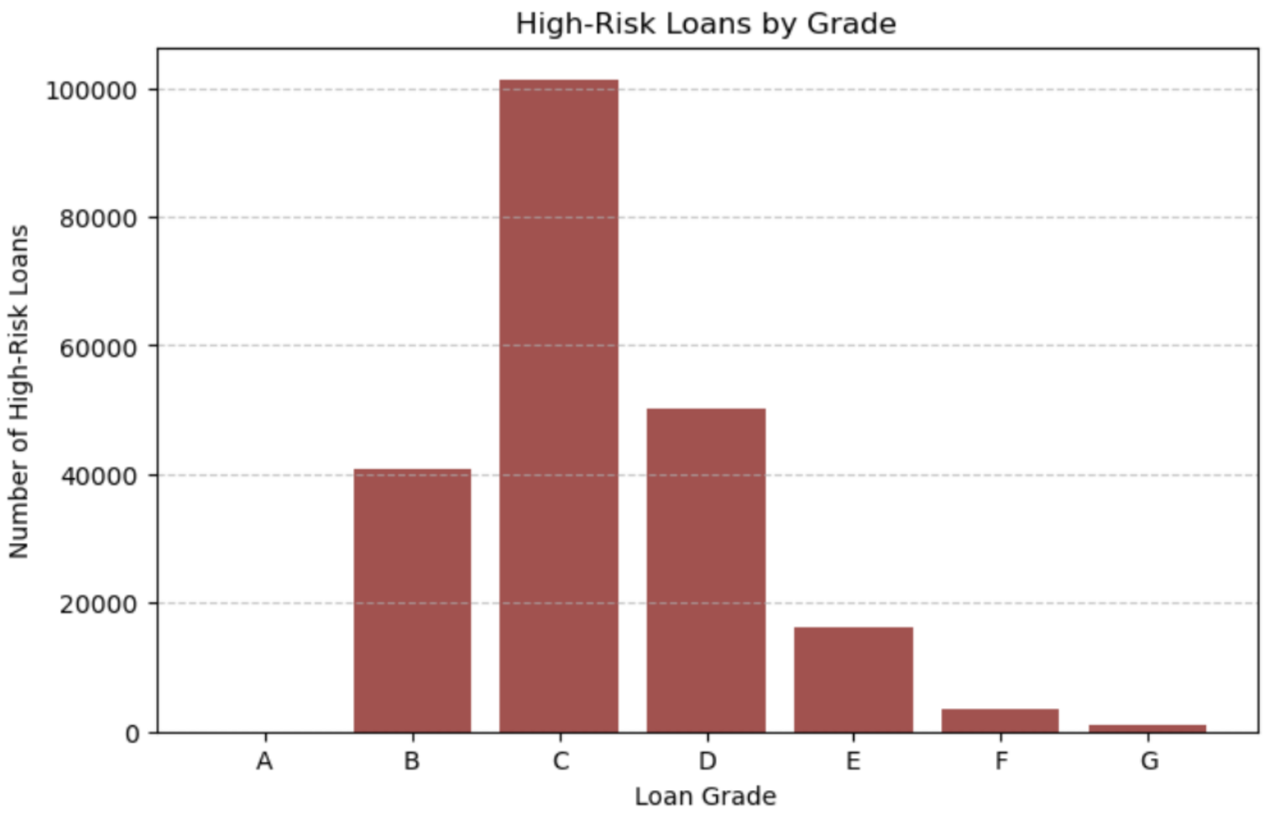
# Advise (.5 threshold)

At a 0.5 threshold, more loans are approved, but default rates for E, F, and G remain high. With a more risky approach the best option is still to mainly focus on A, B, and C grades to manage risk and returns.



# Future suggestions

Grade C has the highest number of high-risk loans, suggesting a wide variance in borrower quality. A future model could refine predictions by segmenting Grade C further, identifying lower-risk subgroups within it.



# Calculate risk return

- I made so calculations at the end of my jupyter notebook file, unfortunately I could not fix the errors in time