Predicting Opportunity of Subrogation with Real Claim Data

Model Citizens, 2025 Travelers UMC

Wenjie Gong, Cecilia Liu, Simeng Wu, Carol Zhou, Franklin Zhou

Department of Statistical Science, Duke University

Introduction

Business Context

Subrogation is a critical part of the claim lifecycle. When a third party is liable, recovery reduces net incurred loss, improves loss ratios, and enhances reserving accuracy—making subrogation a key financial and loss-mitigation lever.

The Core Challenge

Current subrogation identification relies heavily on adjuster judgment and manual file review. This process is slow, inconsistent, and often results in missed recovery opportunities across thousands of claims.

Subrogation Modeling Framework

Our Mission

Build a predictive model using 2020–2021 first-party physical damage claims to flag potential subrogation opportunities, identify key indicators, and provide recommendations for operational use.

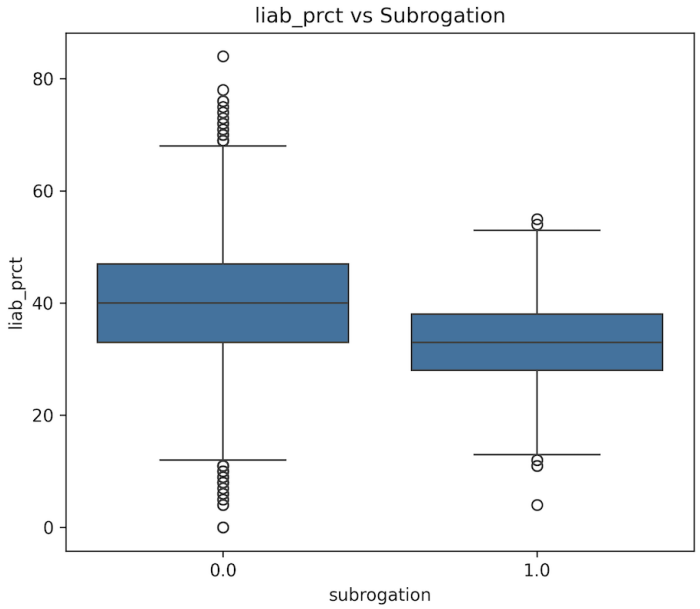
Modeling Objective

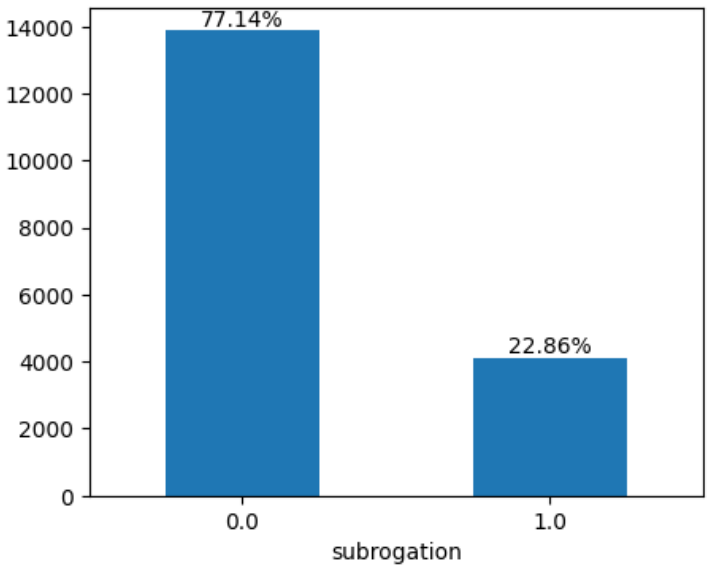
Predict a binary outcome *Subrogation Opportunity*
(1 = likely recovery, 0 = not likely).
Evaluation Metric: F1 score (balances precision and recall due to asymmetric business costs).

Business Value

- Improve recovery rates and reduce net incurred losses
- Help subrogation specialists prioritize high-value cases
- Reduce time spent on low-likelihood opportunities
- Support data-driven decision-making in the claims process

Data Overview





Dataset Summary

- Training set: **18,000** rows with binary subrogation indicator
- Test set: **12,000** rows without the indicator
- Features describe policyholder, driver, vehicle, accident context, and estimated payout

Data Quality Steps

- Removed **2** training rows with missing subrogation indicator
- Test set contains **no missing values**
- Dropped `vehicle_made_year` (inconsistent with claim date) and `age_of_vehicle` (unreliable)

General Patterns

- Several numeric features exhibit skewness
- Subrogation indicator is **imbalanced** (few positives)
- `liab_prct` is strongly associated with the target

Why We Used Multiple Preprocessing Pipelines?

Each model family has its own optimized pipeline, in accordance with their algorithm characteristics.

Model	**TabM**	**Linear Model**	**CatBoost**	**LightGBM**	**XGBoost**
Numerical	Normalize (quantile or z-score)	Normalize (z-score)	Works well with raw data	Works well with raw data	Works well with raw data
Categorical	One-Hot Encoding	One-Hot Encoding	Native handling	Native handling	One-Hot Encoding

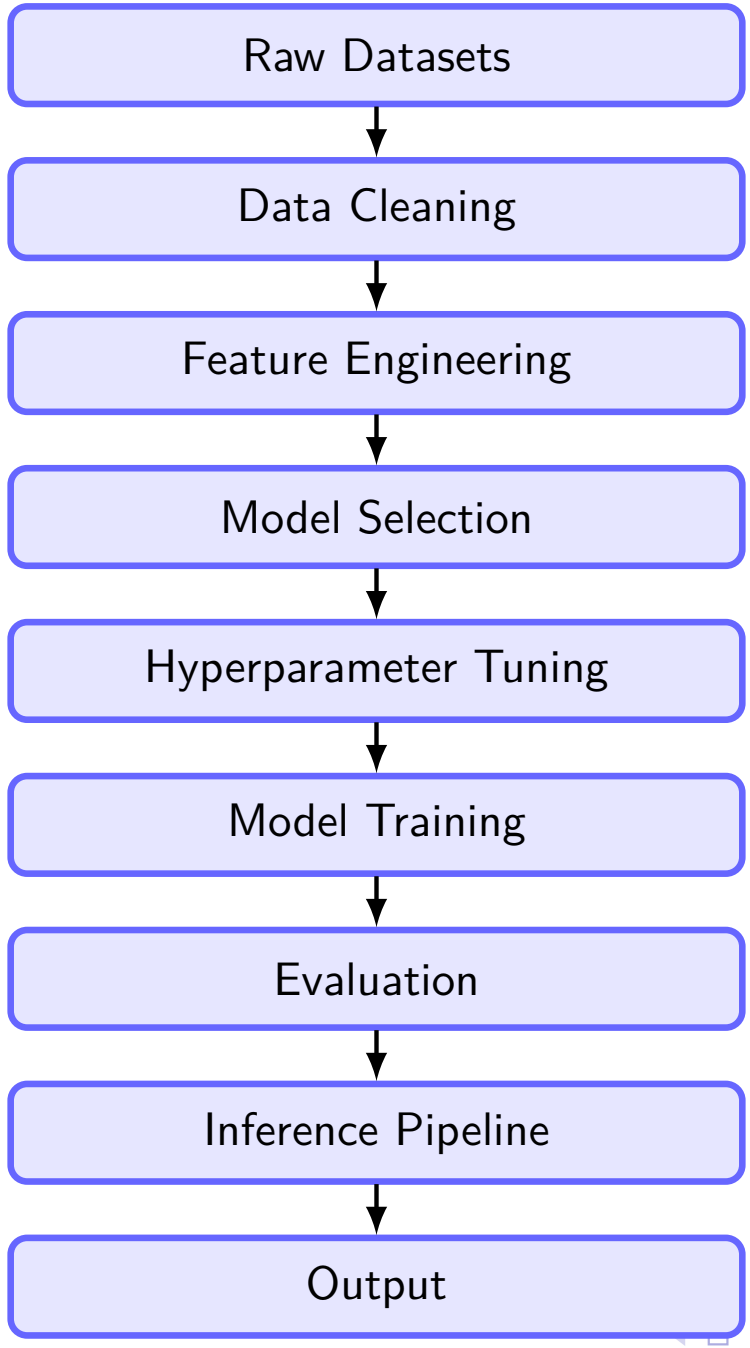
We Leverages Multiple Model Families

- **Linear Model**: Logistic Regression model with LASSO
- **Tree-based Model**: XGBoost, CatBoost, LightGBM
- **MLP**: TabM

Model	Private Score	Public Score
TabM	0.58325	0.60298
CatBoost	0.58215	0.60180
XGBoost	0.57952	0.60078
LogReg + Lasso	0.57698	0.59407
LightGBM	0.57467	0.59093

Additional models considered: LogitBoost, Random Forest

Sample Workflow



Motivation for Majority Voting: Wisdom of the Crowd

- While we obtained decent individual models, aggregating opinions from multiple models is a good way to optimize bias and variance

Theorem (Condorcet's Jury Theorem)

If each invidual model outperforms random guessing ($p > 0.5$) and is independent, the probability of majority voting being correct approaches 1 as such models are added.

- According to Condorcet's Jury Theorem, the majority voting is likely to produce a better prediction than individual models
- The systematic error on specific feature subspaces of each individual model is now overridden by the consensus of others, leading to a smaller model bias
- Overfitting is mitigated as the a single model might obsess over a tiny, unimportant detail in the data, while the group ignores these details and focuses on the main trends

Majority Voting Ensemble

Model	Public Score
XGBoost	0.60600
CatBoost	0.60491
XGBoost	0.60352
XGBoost	0.60213
CatBoost	0.60319
TabM	0.60298
LightGBM	0.59682

Why We Choose Majority Voting Ensemble

Majority Voting is better then stacking, because

- We trained each individual model with a tailor-made preprocessing pipeline, to optimize the individual model performance
- Stacking requires a unified input feature matrix
- If we use one unified preprocessing pipeline to retrain individual models, their performance will greatly degrade

Majority Voting is better than weighted majority, because

- For validation set performance, XGBoost $>$ CatBoost $>$ TabM
- For test set performance (per private score): TabM $>$ CatBoost $>$ XGBoost
- Majority voting algorithm tends to give XGBoost the highest weight
- The resulting ensemble is an "expert" with validation set but a "failure" with test set

Conclusion

- Placeholder

Comparison of Preprocessing Pipelines for GBMs

We designed three classes of preprocessing pipelines for XGBoost, CatBoost, and LightGBM.

LightGBM

- Has most features (most extensive transformation)
- Creates binary flags for categorical variables
- Applies Z-scores, Log transforms, and bucketing for numerical variables
- Applies aggressive feature engineering, with 2-way/3-way interactions, polynomials, domain-specific flags

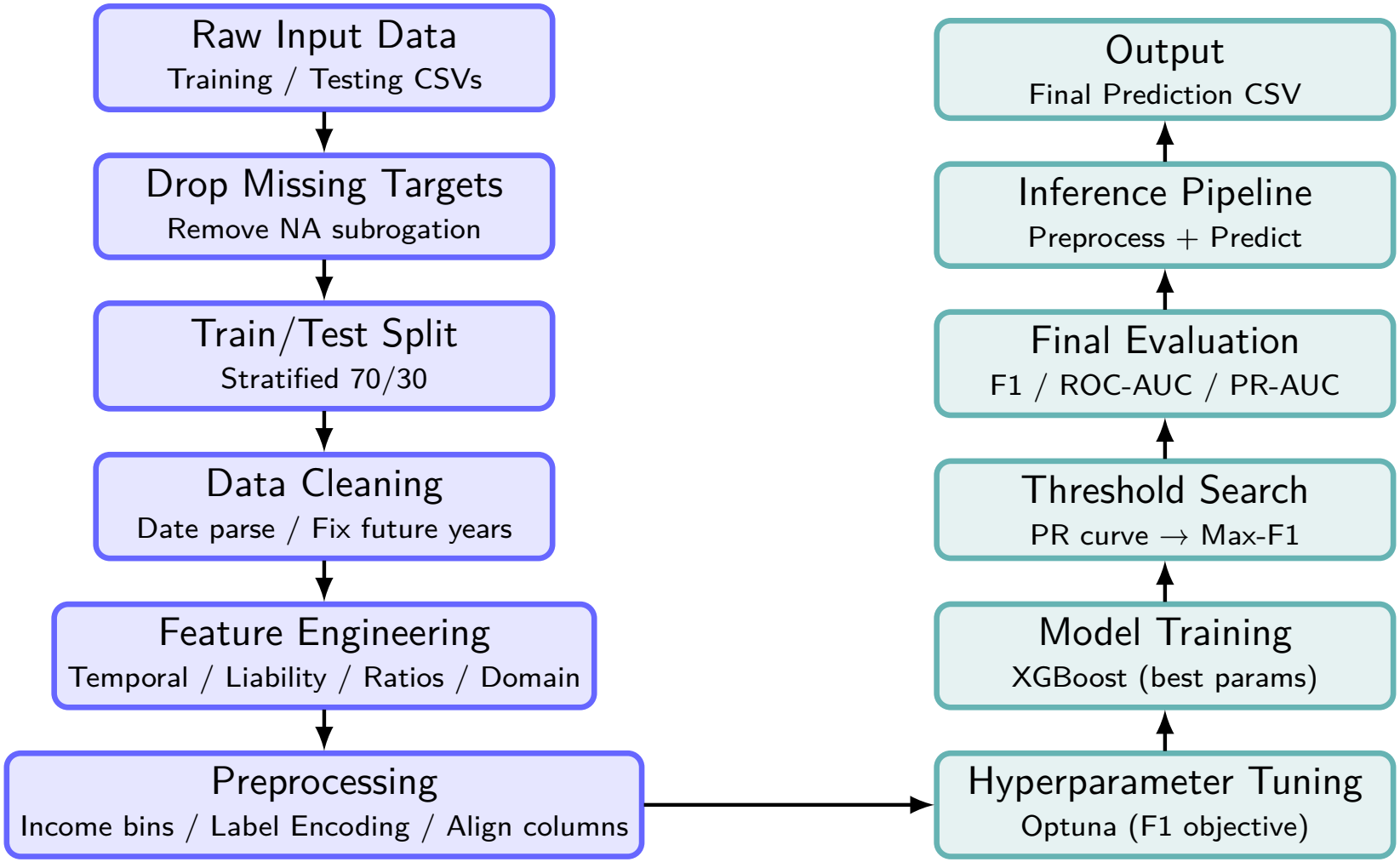
CatBoost

- Has least features (minimal transformation)
- Casts categorical values into strings
- Keeps numerical data *as is*
- Approaches feature engineering very conservatively, with only a few ratio features

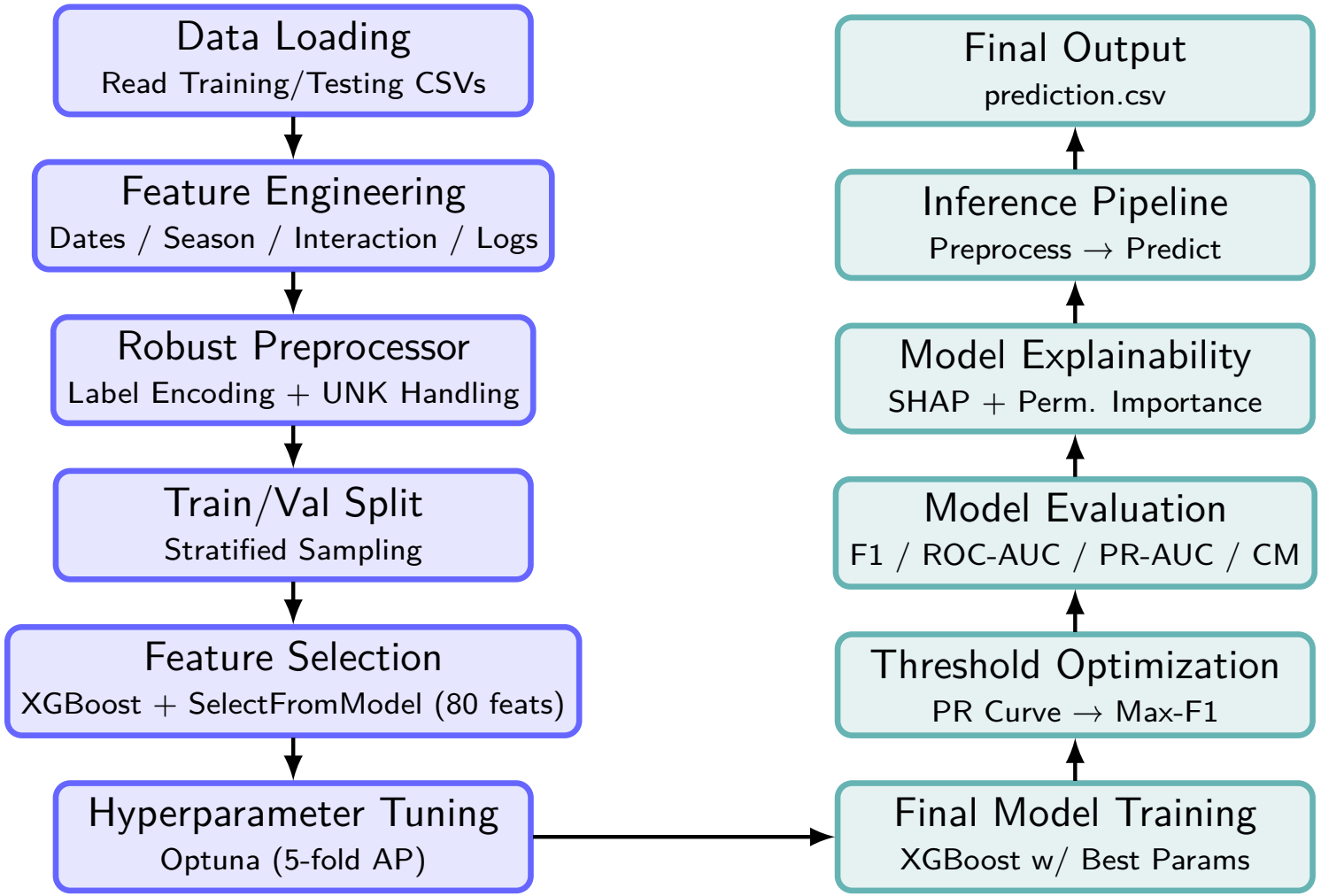
XGBoost

- Has moderately many features (a balanced approach)
- Casts categorical values into integers
- Applies scaling and some Log transforms for numerical variables
- Applies moderate level of feature engineering with temporal features, interactions, and differences

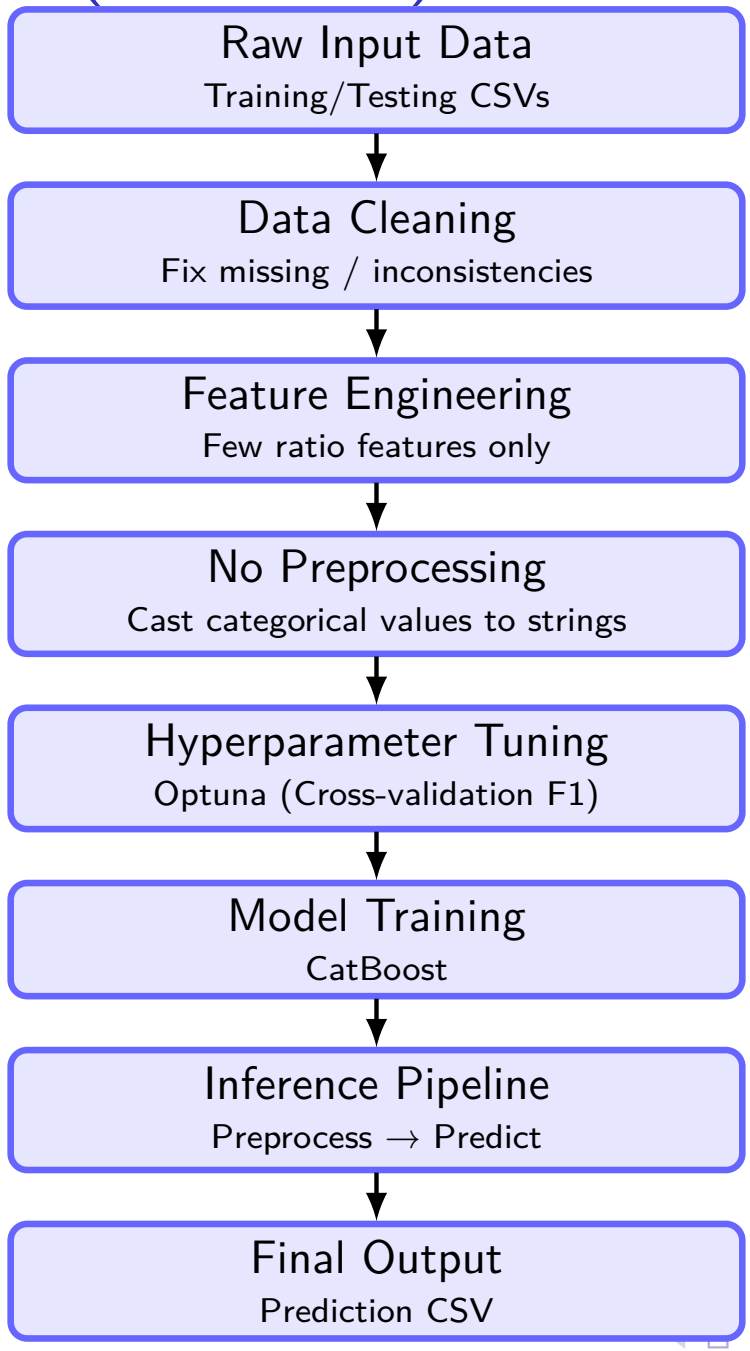
Sample Workflow 1 (XGBoost 1)



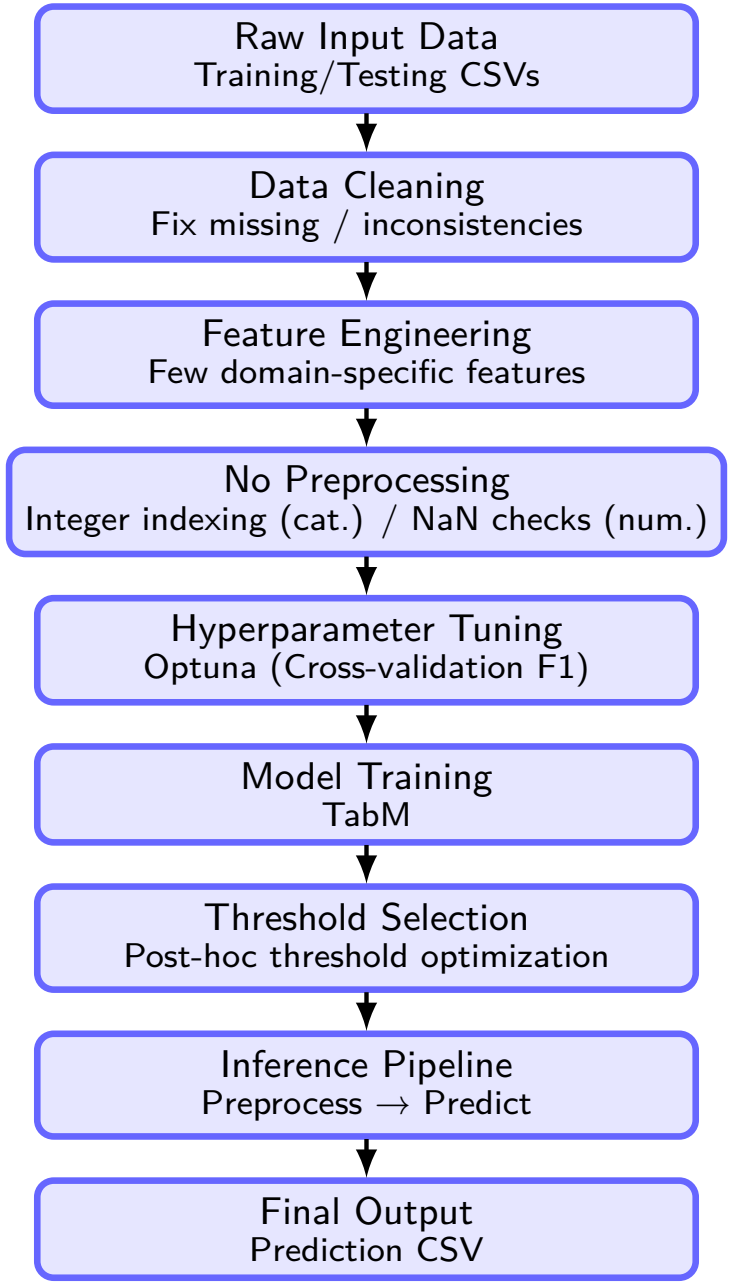
Sample Workflow 2 (XGBoost 2)



Sample Workflow 3 (CatBoost)



Sample Workflow 4 (TabM)



Sample Workflow 5 (LightGBM)