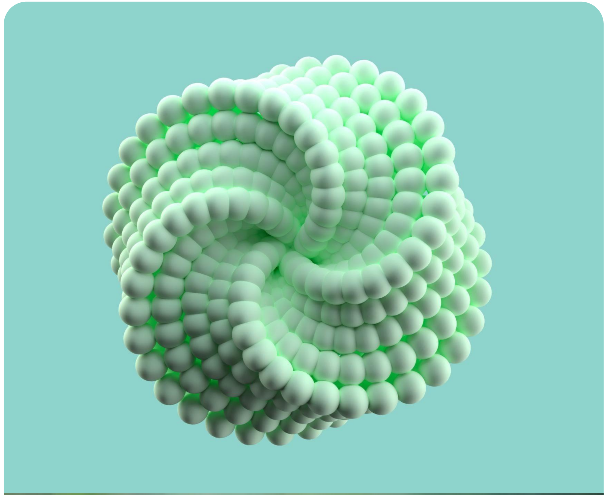


University of North Carolina – Chapel Hill

Multi-Model Financial Fraud Detection

Contributor

Ehsan Bagheri

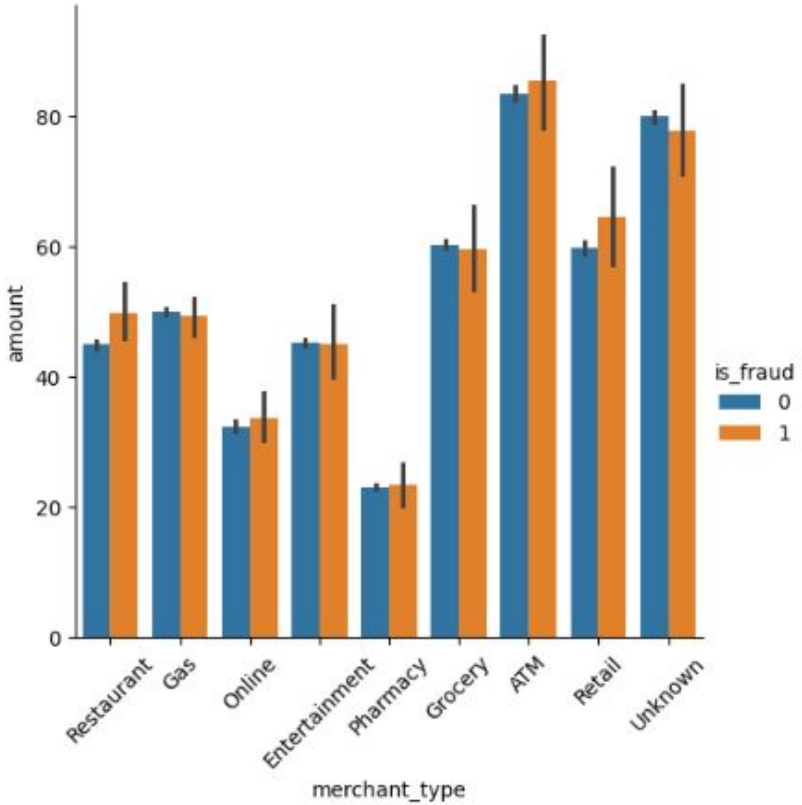
Contributor

Zachary Moran

Contributor

Quinton Wilson

Synthetic Data Creation



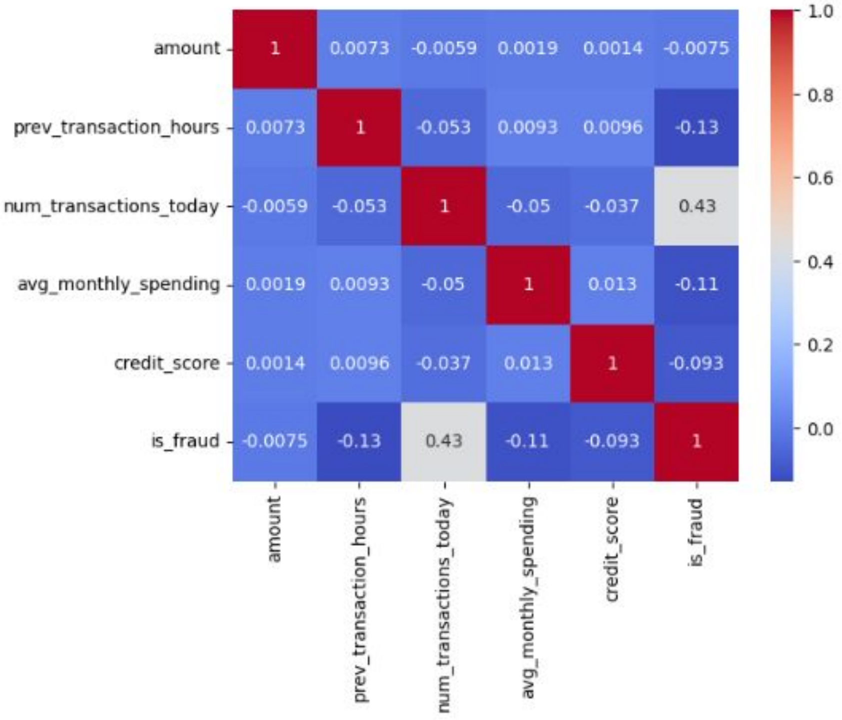
- Custom function created 100k synthetic transactions
- Fraud rate set at 2% for balanced modeling
- Realistic feature simulation: merchant type, location, account, credit score
- Distributions tailored by transaction type and fraud probability

Exploratory Data Analysis

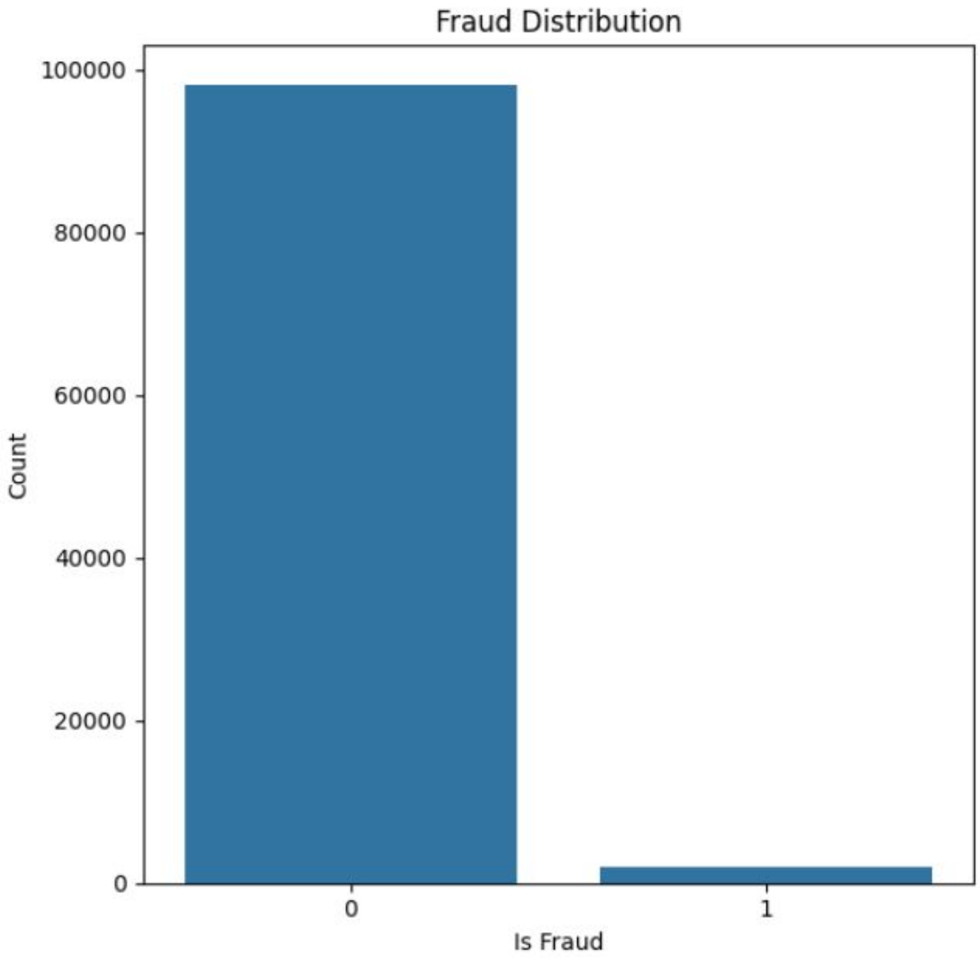
EDA - Insights and Patterns

- Confirmed class imbalance: 98% non-fraud, 2% fraud

- Correlation heatmap showcases key variables include amount, credit score, and transaction timing



Feature Engineering



Preprocessing

Timestamp extraction

Encoding days of the week

Removing unnecessary columns

Train - Test - Validation Split

Train: 80%

Test: 10%

Validation: 10%

Data Normalization(RobustScaler)

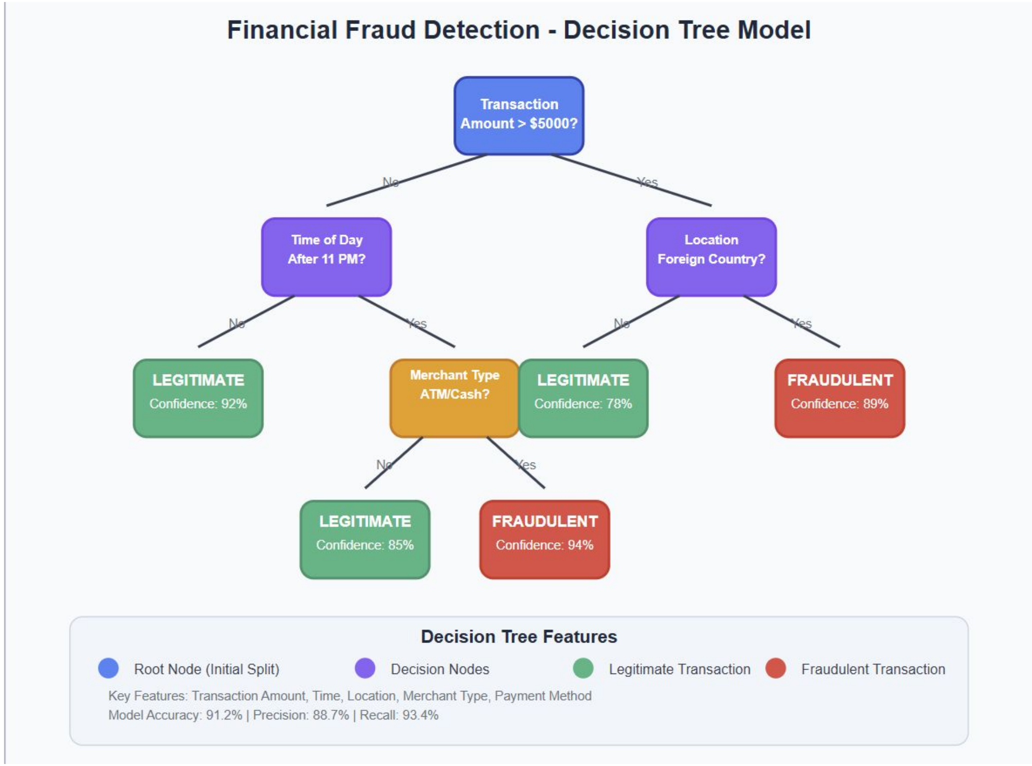
Fraud Distribution

Transaction Amounts

Ensemble Models

Random Forest

XGBoost



Random Forest

300 trees, max depth = 30

Feature sampling: log2

Strengths: Robust to noise, low tuning complexity

Method: Bagging (parallel tree growth)

XGBoost

200 estimators, max depth = 5

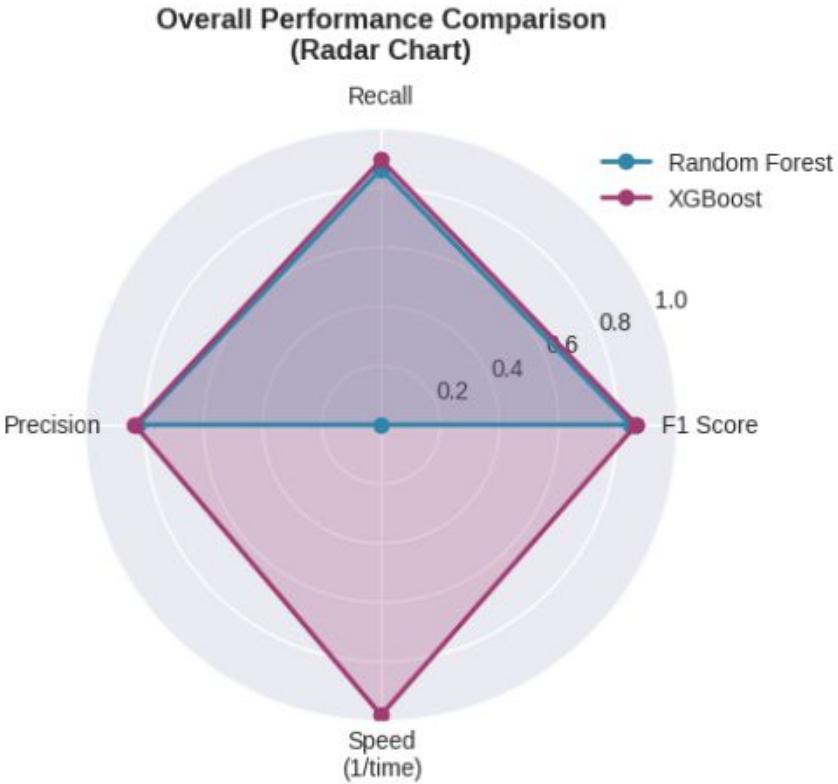
Learning rate = 0.1, early stopping = 50 rounds

Feature sampling: 80% per tree

Strengths: High accuracy, effective on tabular data

Method: Boosting (sequential correction)

Visual Comparison

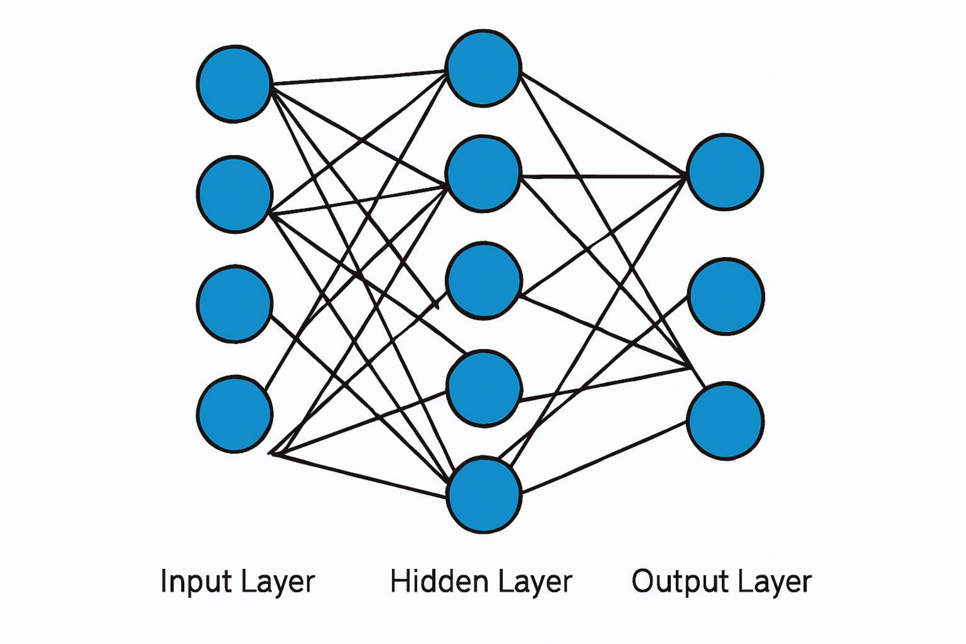


XGBoost outperformed Random Forest in recall, while training nearly 50x faster

Slight trade-off in precision, but overall higher F1 shows better balance

Ideal for real-time or high-volume fraud detection systems

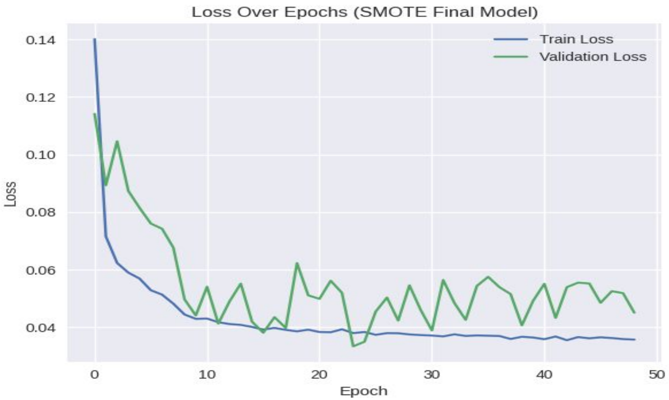
Neural Network

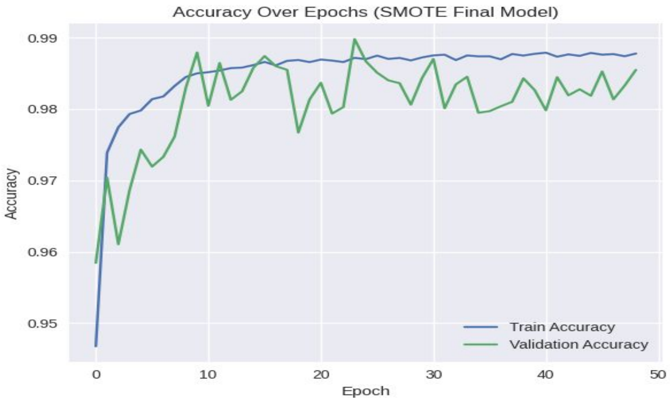


Neural Network Results

Key Highlights:

- **High Recall (0.97)** on test set — captures nearly all fraud cases
- **AUC:** 0.9986 — near-perfect class separation
- **Cross-Validated Recall:** 0.93 ± 0.01
- **F1 Score (Test):** 0.78
- **Precision (Test):** 0.66
- **Accuracy:** 98.93%
- **Training Stability:** Low loss & consistent accuracy over epochs
- **Balanced Generalization:** Minimal overfitting observed



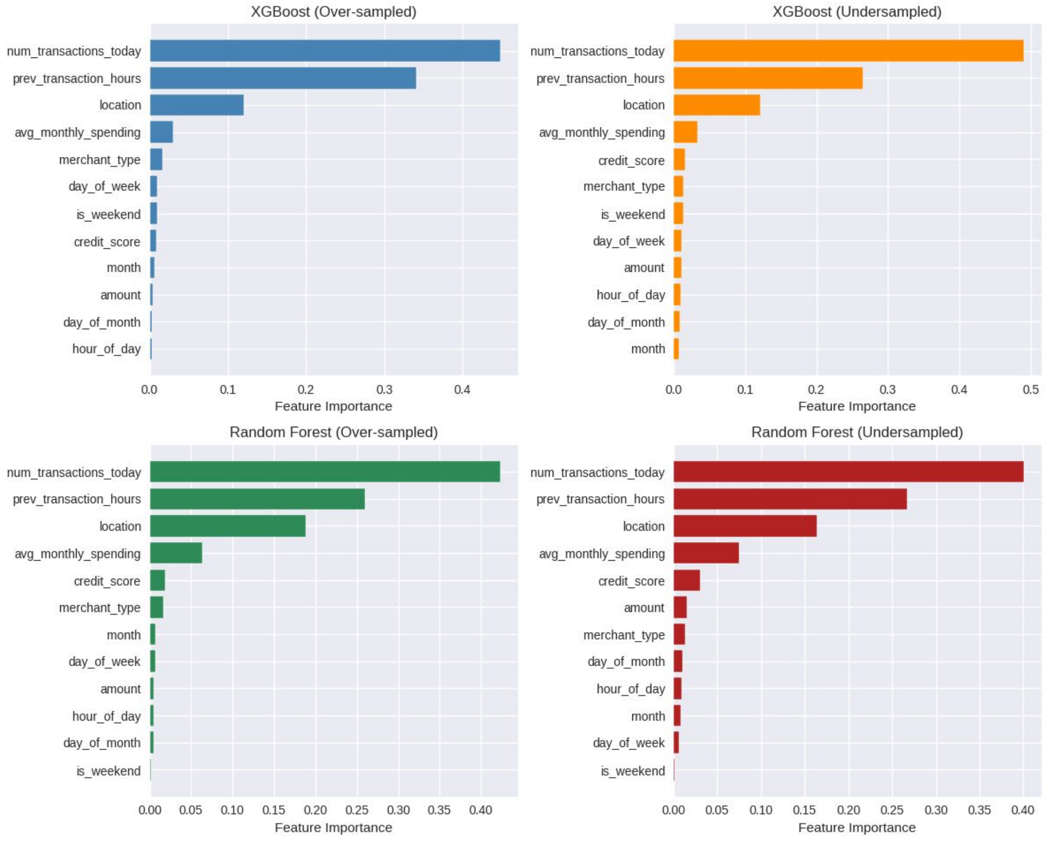


Imbalanced Data Solutions: Sampling & Threshold Tuning

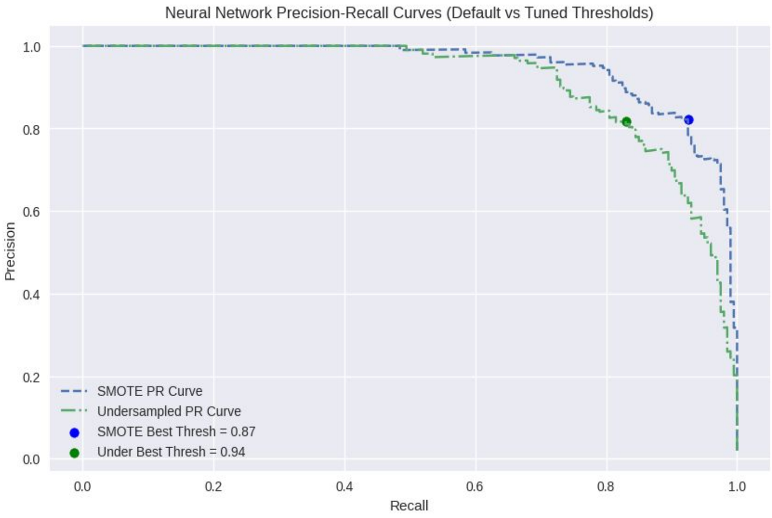
Oversampling vs. Undersampling Feature Importance

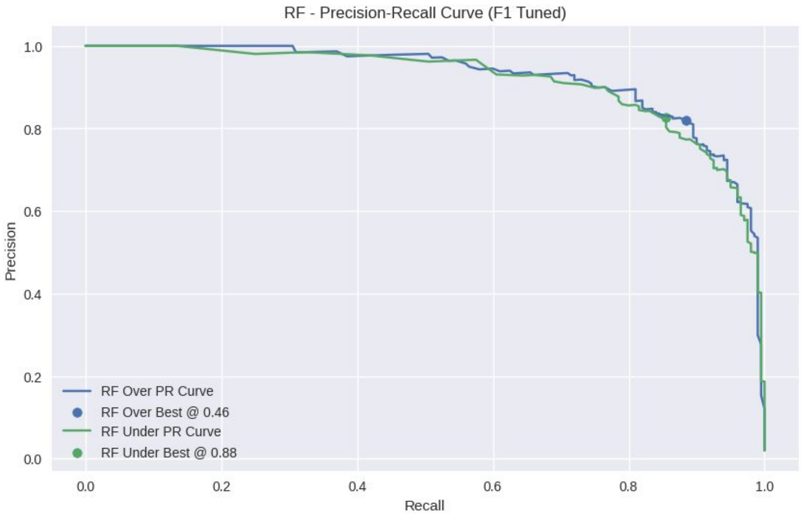
Feature Importance Stability

- Top features (num_transactions_today, prev_transaction_hours, location) remained consistent
- Sampling method (SMOTE vs. undersampling) had minimal effect
- Suggests robust, model-independent signal in feature ranking



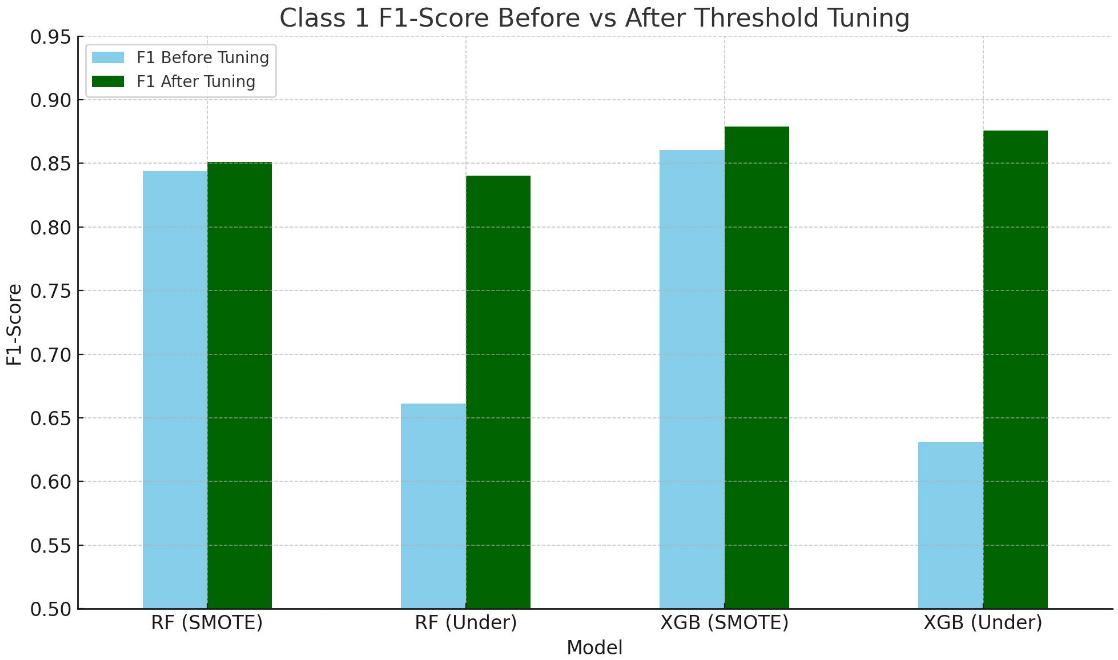
PR Curves for Threshold Tuning: RF, XGB, and NN





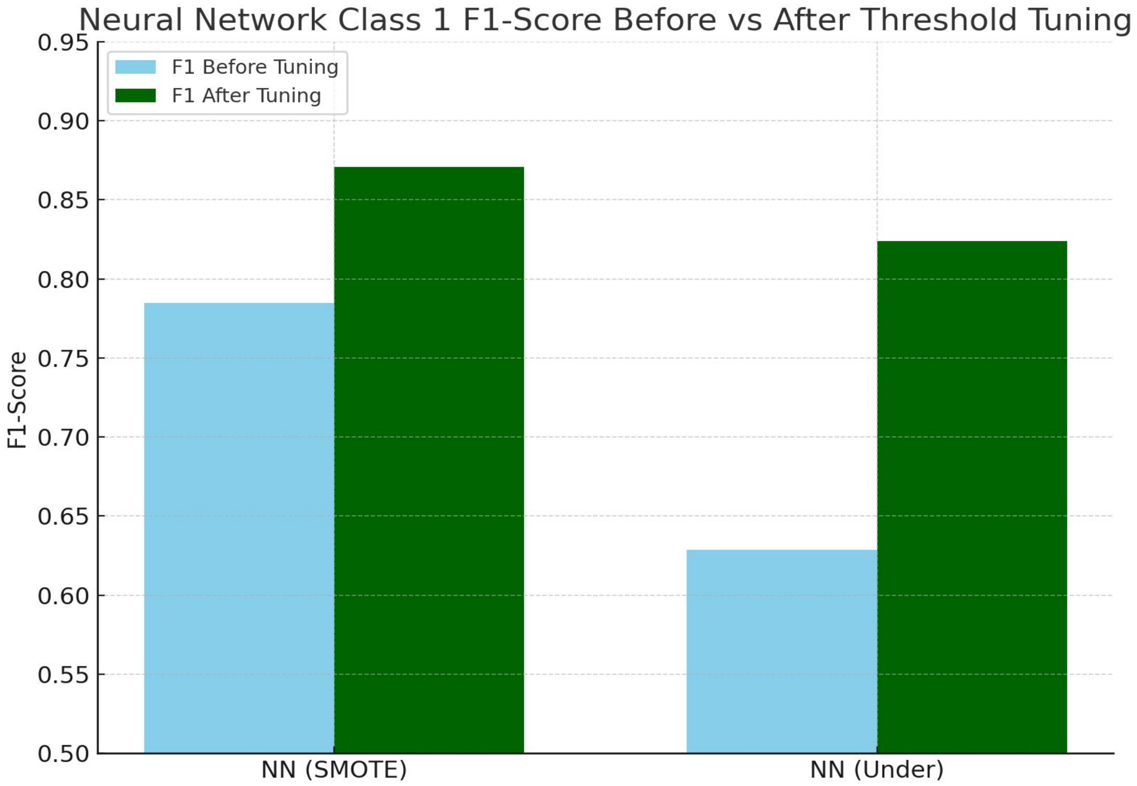
Improved RF and XGB Class 1 Detection

- **XGBoost consistently outperforms Random Forest** across both sampling strategies
- **Threshold tuning greatly improves F1-score** for undersampled models (RF: +18 pts, XGB: +25 pts)
- **SMOTE yields better baseline**, but tuned undersampling closes the gap or surpasses it



Improved Neural Network Class 1 Detection

- **Threshold tuning boosts Class 1 F1-score** (SMOTE: +9 pts, Under: +19 pts)
- **Undersampling sees biggest gain**, narrowing gap with SMOTE-based training



Model Comparison: Sampling & Threshold Tuning

- **XGBoost consistently outperformed RF and NN** in Class 1 F1-score across both SMOTE and undersampling.
- **Threshold tuning significantly improved Class 1 detection**, especially for undersampled models (e.g., +25 pts for XGB Under).
- **SMOTE models had stronger baseline performance**, but **undersampled models caught up or surpassed** after tuning.
- **Neural Networks improved with tuning**, but **still underperformed compared to XGBoost**, particularly on undersampled data.
- **Model calibration mattered** — optimal thresholds varied widely (e.g., 0.46 for RF Over vs. 0.97 for XGB Under), showing tuning is essential for imbalanced datasets.

Hyperparameter Tuning

XGBoost Optimization via Grid Search

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.2, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=9,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=300,
              n_jobs=None, num_parallel_tree=None, random_state=None, ...)
```

- **Objective:** Improve performance for the XGBoost classification

- Used Grid Search Cross Validation to tune:
  - N_estimators (number of trees)
  - Max_depth (tree depth)
  - learning_rate (step size shrinkage)
  - subsample (row sampling)
  - Colsample_bytree (feature sampling)

- Manual tuning provided similar results than Grid Search

Optuna Neural Network Tuning

```
AUC Scores: ['0.9947', '0.9956', '0.9960', '0.9954', '0.9959']
Mean AUC: 0.9955
Std AUC: 0.0005
```

```
Optuna Best Hyperparameters:
batch_size: 128
patience: 18
n_units1: 26
n_units2: 98
n_units3: 30
dropout: 0.12042960989760738
lr: 0.0004461073453710296

Best ROC AUC: 0.9972
```

- **Objective:** Improve performance for the Neural Network
- Utilized Optuna to create 5 trials examining:
    - Layer Sizes
    - Dropout rate (to prevent overfitting)
    - Activation functions
    - Learning rate and batch sizes
- Selected parameters from best of 5 trails
- Final model showed strong performance with stable convergence

```
Cross-validation ROC AUC scores: ['0.9972', '0.9976', '0.9957', '0.9961', '0.9976']
Mean Cross-validation ROC AUC: 0.9968
Standard Deviation of Cross-validation ROC AUC: 0.0008
```