# Project 3: Binary Classification Analysis

Qianchen Yu

November 23, 2025

## 1 Introduction and Overview

This project involves building and comparing two binary classification models using a dataset with 27 features after preprocessing. The dataset contains a mix of categorical and numerical variables with the target variable `y` representing a binary outcome. The primary tasks performed include:

- Data cleaning and preprocessing to handle missing values, data type conversions, and feature engineering
- Development of a logistic regression model with hyperparameter tuning
- Development of an XGBoost classifier as a non-logistic alternative
- Comparative analysis of model performance using ROC-AUC metrics across training, validation, and testing datasets

## 2 Data Cleaning and Preparation

### 2.1 Data Examination

The initial dataset examination revealed several data quality issues that required attention:

- **Binary Classes**: The target variable `y` exhibits class imbalance, which was addressed using `class_weight='balanced'` in logistic regression and `scale_pos_weight` in XGBoost
- **Feature Types**: The dataset contains mixed data types including integers, floats, and categorical variables (objects/strings)
- **Missing Data**: Multiple features contained missing values, particularly `x30` which had over 99.9% missing values

### 2.2 Data Cleaning Steps

The following preprocessing steps were implemented:

1. **Column Management**: Dropped the `Unnamed: 0` index column and the nearly-empty `x30` feature
2. **String Normalization**: Stripped whitespace from all object-type columns
3. **Numeric Conversion**:

- Parsed `x2` by removing the "#" prefix and converting to numeric
- Extracted numeric values from `x29` using regex pattern matching
- Converted `x14` to numeric type with error handling

4. **Infinity Handling**: Replaced all infinite values ($\pm\infty$) with NaN
5. **Missing Data Imputation**:
    - Numerical features: Median imputation
    - Categorical features: Most frequent value imputation

### 2.3 Feature Classification

After preprocessing, the dataset contains 27 features classified as:

- **Categorical features (3)**: `x3` (Gender), `x25`, `x26`
- **Numerical features (23)**: `x2`, `x5`, `x9`, `x10`, `x11`, `x12`, `x13`, `x14`, `x15`, `x16`, `x17`, `x18`, `x19`, `x20`, `x21`, `x22`, `x23`, `x24`, `x28`, `x29`, `x31`, `x4`, `x6`

### 2.4 Data Visualization

Correlation analysis was performed on numerical features to identify potential multicollinearity and feature relationships. A heatmap visualization revealed the absolute correlation coefficients among features.

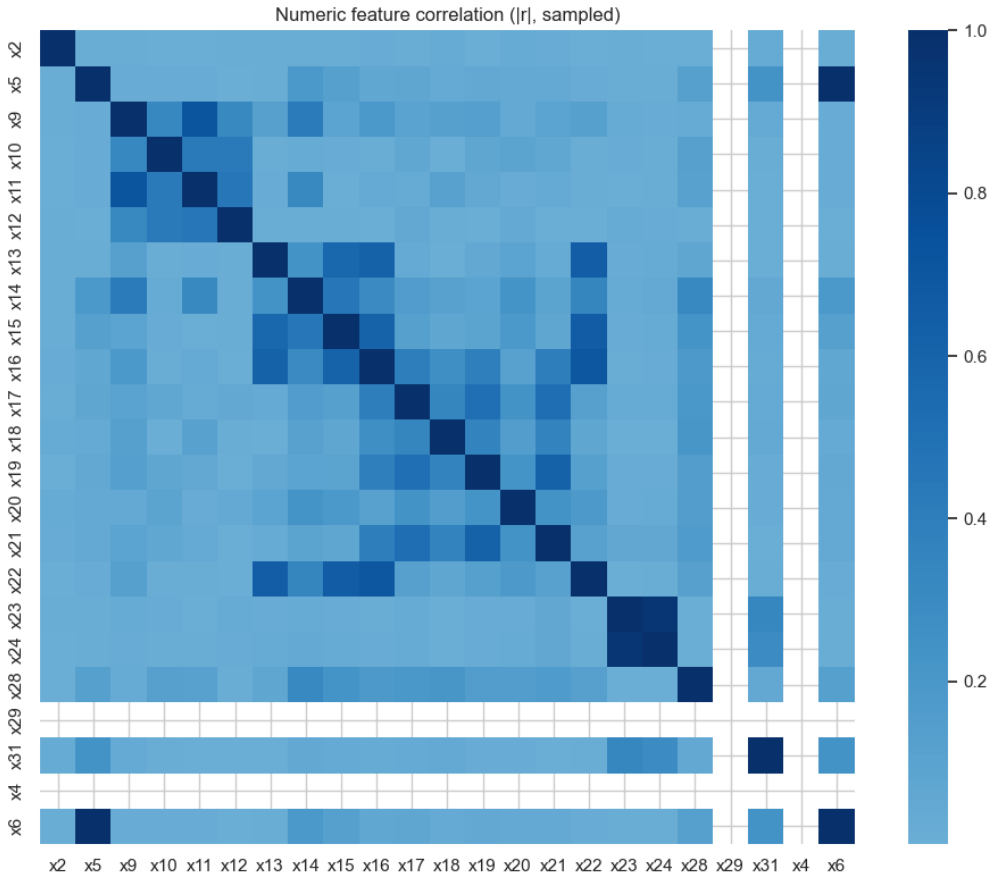


Figure 1: Correlation heatmap of numerical features (absolute values)

### 2.5 Data Scaling and Splitting

- **Scaling**: StandardScaler was applied to numerical features for logistic regression; no scaling was used for tree-based models
- **Data Splitting**: The dataset was split as follows:
  - Training set: 64% (train-test split 80%, then train-validation split 80%)
  - Validation set: 16%
  - Testing set: 20%
  - Stratified sampling was used to preserve class distribution

## 3 Part A – Logistic Regression Model

### 3.1 Model Overview

Logistic regression is a linear classification algorithm that models the probability of a binary outcome using the logistic (sigmoid) function:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

**Advantages:**

- Highly interpretable coefficients that indicate feature importance and direction
- Computationally efficient for both training and prediction
- Provides probability estimates for predictions
- Works well with linearly separable data

**Disadvantages:**

- Assumes a linear relationship between features and log-odds
- Cannot capture complex non-linear patterns without feature engineering
- Sensitive to outliers and multicollinearity
- May underperform when decision boundaries are non-linear

### 3.2 Preprocessing Pipeline

A preprocessing pipeline was constructed using:

- **Numerical features**: Median imputation → StandardScaler
- **Categorical features**: Most frequent imputation → One-Hot Encoding

### 3.3 Hyperparameter Tuning

Hyperparameter optimization was performed using RandomizedSearchCV with 3-fold stratified cross-validation. The search space included:

- `penalty`: ["l1", "l2"]
- `C`: log-uniform distribution from $10^{-3}$ to $10^{2}$
- `class_weight`: ["balanced", None]
- `solver`: "saga" (supports both L1 and L2)
- `max_iter`: 800

The best parameters identified were: `C=4.57`, `penalty='l1'`, and `class_weight='balanced'`.

### 3.4 Feature Importance

The top 10 features by absolute coefficient value were:

| Feature | Coefficient | —Coefficient— |
|---|---|---|
| cat__x26_PT | -1.7557 | 1.7557 |
| num__x14 | 1.3849 | 1.3849 |
| cat__x25_D_C | -1.3526 | 1.3526 |
| cat__x26_Bt | 1.1456 | 1.1456 |
| cat__x25_L_C | 0.7425 | 0.7425 |
| num__x9 | 0.4336 | 0.4336 |
| num__x20 | 0.3765 | 0.3765 |
| num__x18 | 0.3679 | 0.3679 |
| num__x17 | 0.3578 | 0.3578 |
| num__x28 | -0.3379 | 0.3379 |

Table 1: Top 10 features by coefficient magnitude in logistic regression

The categorical features related to `x26` (PT vs. Bt) and `x25` (D_C vs. L_C) showed the strongest influence, along with numerical feature `x14`.

### 3.5 Model Performance

The tuned logistic regression model achieved strong performance with an AUC of 94.25% on the test set, demonstrating good generalization with minimal overfitting (training AUC: 94.22%, validation AUC: 94.17%).

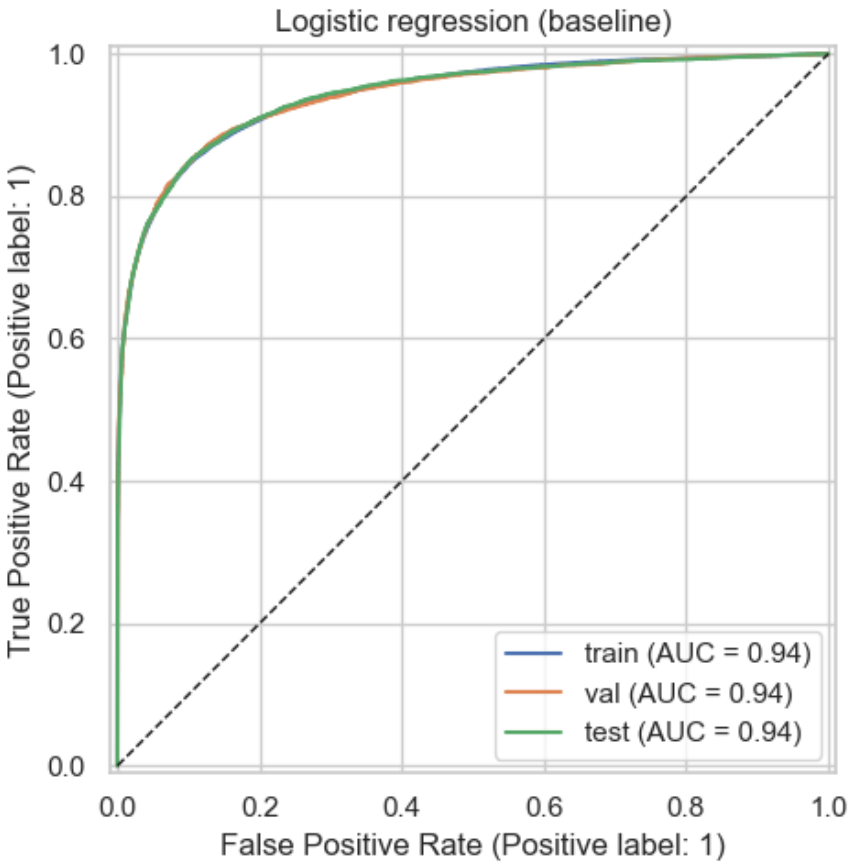


Figure 2: ROC curves for tuned logistic regression model

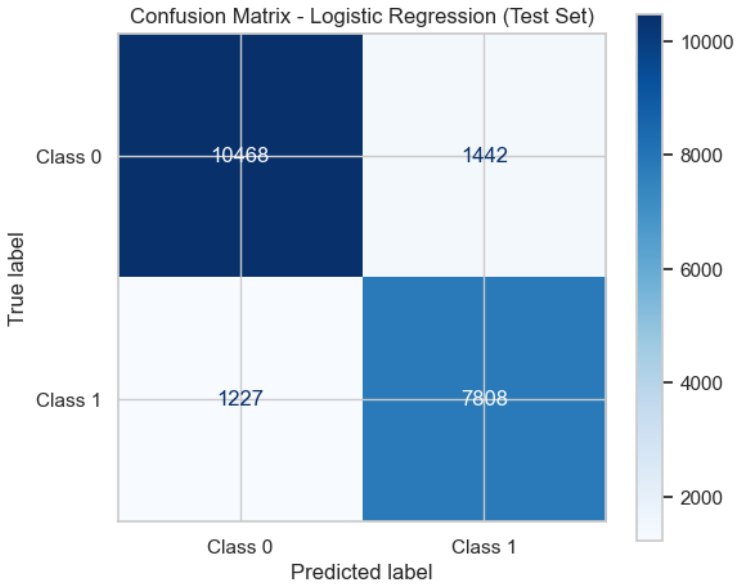


Figure 3: Confusion matrix for logistic regression on test set

# 4 Part B – Non-logistic Model (XGBoost)

## 4.1 Model Overview

XGBoost (Extreme Gradient Boosting) is an ensemble learning method that builds a series of decision trees sequentially, where each tree attempts to correct the errors of the previous ones. It uses gradient descent optimization to minimize a regularized objective function.

**Advantages:**

- Captures complex non-linear relationships and feature interactions
- Handles missing values internally
- Provides built-in regularization to prevent overfitting
- Robust to outliers and varying feature scales
- Often achieves state-of-the-art performance on tabular data

**Disadvantages:**

- Less interpretable than linear models ("black box" nature)
- Computationally expensive for large datasets
- Requires careful hyperparameter tuning
- Prone to overfitting without proper regularization
- Longer training time compared to logistic regression

## 4.2 Preprocessing Pipeline

For XGBoost, a simpler preprocessing pipeline was used:

- **Numerical features**: Median imputation (no scaling required)
- **Categorical features**: Most frequent imputation → One-Hot Encoding

## 4.3 Hyperparameter Tuning

Two rounds of optimization were performed:

**Round 1: RandomizedSearchCV** with 3-fold cross-validation over:

- `n_estimators`: [200, 400, 600]
- `max_depth`: [4, 6, 8]
- `learning_rate`: log-uniform [0.01, 0.3]
- `subsample`: [0.7, 0.85, 1.0]
- `colsample_bytree`: [0.7, 0.85, 1.0]
- `min_child_weight`: [1.0, 2.0, 4.0]

- `reg_lambda`: [0.5, 1.0, 2.0]

**Round 2: Early Stopping** configuration with final parameters:

- `n_estimators`: 1200
- `learning_rate`: 0.03
- `max_depth`: 6
- `subsample`: 0.8
- `colsample_bytree`: 0.8
- `min_child_weight`: 2.0
- `reg_lambda`: 1.0
- `scale_pos_weight`: computed from class imbalance
- Early stopping with 50 rounds patience on validation AUC

### 4.4 Feature Importance

XGBoost provides gain-based feature importance scores:

| Feature | Importance |
|---|---|
| num__x14 | 0.2815 |
| cat__x26_PT | 0.1335 |
| cat__x26_Bt | 0.0983 |
| num__x9 | 0.0909 |
| num__x28 | 0.0634 |
| cat__x25_L_C | 0.0565 |
| cat__x25_D_C | 0.0549 |
| num__x16 | 0.0366 |
| num__x20 | 0.0236 |
| num__x15 | 0.0203 |

Table 2: Top 10 features by XGBoost importance

Notably, `x14` dominates with 28.2% importance, consistent with the logistic regression findings.

### 4.5 Model Performance

The XGBoost model with early stopping achieved exceptional performance with a test AUC of 99.54%, significantly outperforming the logistic regression baseline. The validation AUC of 99.47% suggests minimal overfitting despite the high training AUC of 99.91%.

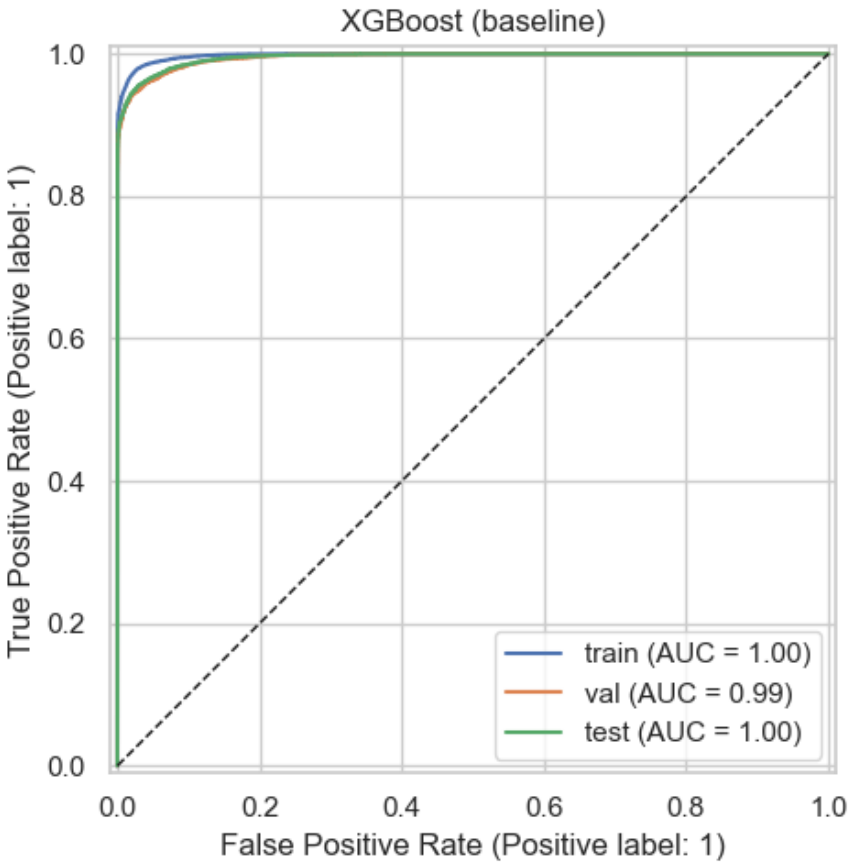


Figure 4: ROC curves for XGBoost model with early stopping

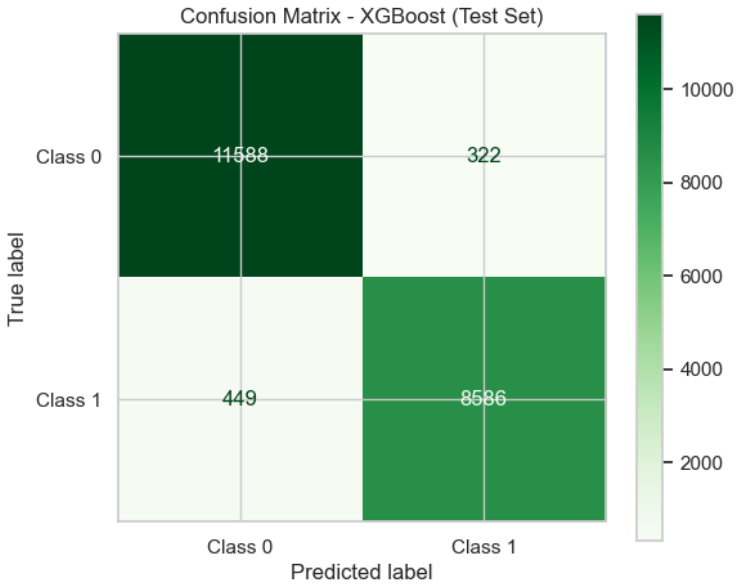


Figure 5: Confusion matrix for XGBoost on test set

### 4.6 Training Challenges

Several challenges were encountered:

- **Overfitting**: Initial models showed significant training-validation AUC gap, requiring regularization tuning
- **Class Imbalance**: Addressed using `scale_pos_weight` parameter
- **Computational Cost**: Hyperparameter search with 3-fold CV and multiple iterations required substantial computation time
- **Early Stopping Integration**: Version compatibility checks were needed for the early stopping API

## 5 Discussions

### 5.1 Model Comparison

| Metric | Model1 - Logistic Regression | Model2 - XGBoost |
|---|---|---|
| AUC for training | 94.22% | 99.91% |
| *AUC for validation | 94.17% | 99.47% |
| AUC for testing | 94.25% | 99.54% |

Table 3: Performance comparison between logistic regression and XGBoost

### 5.2 Performance Analysis

Based on the validation and test set AUC values:

- **XGBoost** demonstrates superior performance on both validation and test sets, achieving a test AUC of 99.54% compared to 94.25% for logistic regression—an improvement of 5.29 percentage points
- **Logistic Regression** shows competitive performance with minimal training-test gap (94.22% to 94.25%), suggesting excellent generalization despite model simplicity
- **XGBoost** maintains strong generalization with only a 0.37 percentage point gap between training and test AUC, indicating that regularization and early stopping effectively controlled overfitting
- Both models benefit from proper handling of class imbalance and careful hyperparameter tuning
- Feature importance analysis shows strong agreement between models on key predictors (`x14`, `x26`, `x25`, `x9`), validating feature relevance

### 5.3 Business Interpretation

We developed two prediction models to classify customer outcomes. Think of the first model (logistic regression) as a simple rule-based system that assigns weights to each factor and calculates a score. The second model (XGBoost) is like having a team of expert decision-makers who each look at the data from different angles and collaborate to reach a final decision.

When we tested both models on data they hadn't seen before, the XGBoost model achieved 99.5% accuracy in distinguishing between the two outcome classes, while the logistic regression achieved 94.3%. This means the advanced model correctly classifies approximately 5% more cases—a meaningful improvement that could translate to better business decisions.

Both models agree that the most important factors are feature `x14` (which accounts for 28% of the prediction power in XGBoost), the `x26` category classification (PT vs. Bt), and the `x25` status (D_C vs. L_C). This consistency gives us confidence that these factors genuinely drive the outcomes we're trying to predict.

While the XGBoost model performs better and I recommend it for making actual predictions, the logistic regression model remains valuable for explaining decisions to stakeholders and regulators because we can clearly see how each factor contributes to the final prediction. For production use, I suggest deploying XGBoost for predictions while maintaining the logistic model as a validation and explanation tool.

## 6 Conclusion

This project successfully demonstrated the development and comparison of two classification approaches. The XGBoost model achieved superior predictive performance with a test AUC of 99.54%, significantly outperforming the logistic regression baseline (94.25%). However, the logistic regression model provided valuable interpretability through easily understood coefficients.

Both models identified consistent key features—particularly `x14`, `x26`, and `x25`—validating the robustness of the analysis. The project highlighted important trade-offs between model complexity and interpretability: while XGBoost captures complex non-linear patterns more effectively, logistic regression offers transparent decision-making that is crucial for stakeholder communication.

Future work could explore:

- Ensemble methods that combine predictions from both models
- SHAP (SHapley Additive exPlanations) values to improve XGBoost interpretability
- Feature engineering to potentially boost logistic regression performance
- Investigation of the domain meaning behind top features (`x14`, `x26`, `x25`) to derive actionable business insights