Late Delivery Prediction

Machine Learning Classification Project

Binary Classification | XGBoost | Threshold Optimization

Prepared for Technical Interview

1. Executive Summary

This project develops a machine learning model to predict late deliveries in an e-commerce/logistics context. The goal is to enable proactive interventions that improve customer satisfaction and operational efficiency.

Key Results

- **Best Model:** XGBoost (optimized) with AUC = 0.749
- **Top Feature:** Discount_offered dominates with ~72% importance
- **Optimized Threshold:** 0.41 (lowered from 0.50 to improve recall)
- **Business Impact:** +200 additional late deliveries caught, reducing customer impact by 30%

2. Business Problem

Problem Statement

Late deliveries negatively impact customer satisfaction, increase support costs, and damage brand reputation. The objective is to build a predictive model that identifies shipments at risk of delay before they are dispatched, enabling proactive customer communication and operational adjustments.

Business Value

1. **Customer Experience:** Proactive notification of potential delays
2. **Operational Efficiency:** Prioritize high-risk shipments for expedited handling
3. **Cost Reduction:** Reduce customer complaints and refund requests
4. **Strategic Insights:** Understand key delay drivers for process improvement

3. Exploratory Data Analysis

3.1 Feature Distributions

The dataset contains both numerical and categorical features. Distribution analysis reveals important patterns:

- **Discount_offered:** Highly right-skewed (skew=1.80), most values concentrated at lower discounts
- **Weight_in_gms:** Near-normal distribution with slight left skew
- **Prior_purchases:** Right-skewed (skew=1.68), majority customers with 2-4 prior purchases
- **Customer_rating:** Symmetric distribution (skew=0.00), uniform across 1-5 ratings

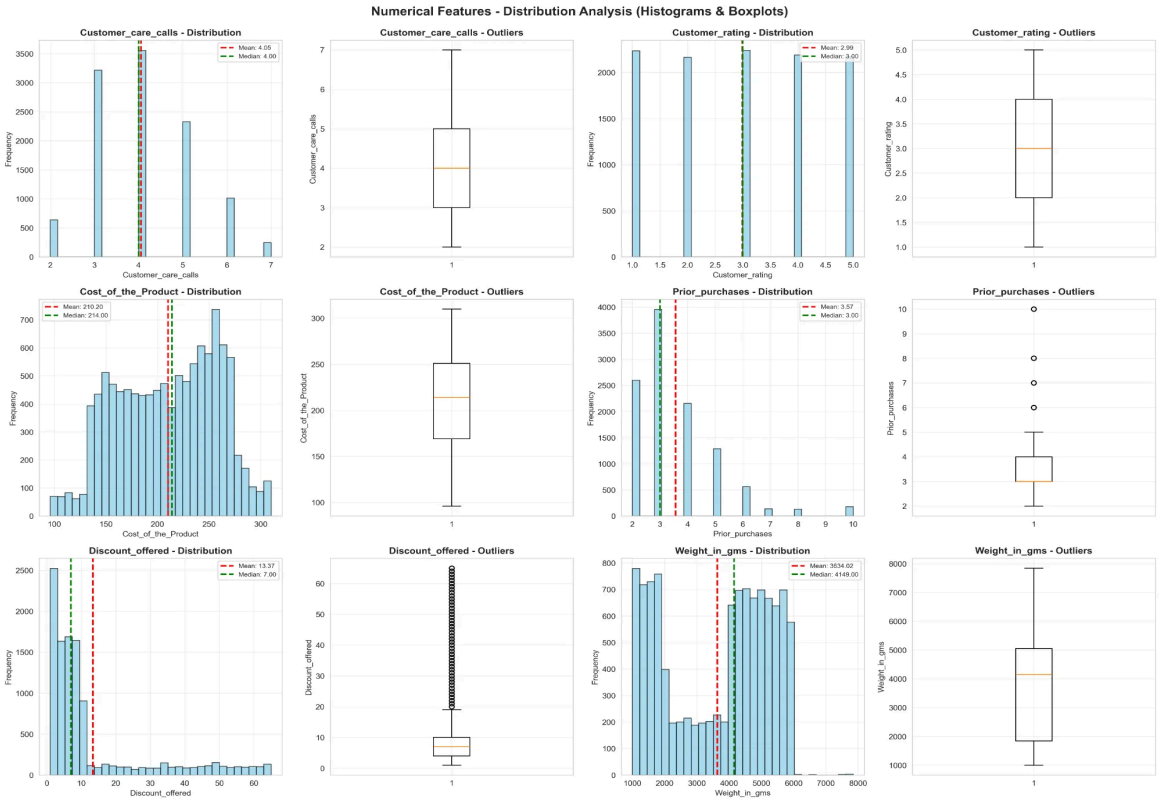

Figure 1: Feature Distributions with Mean/Median Indicators

3.2 Normality Assessment

Q-Q plots reveal that most features deviate significantly from normal distribution, justifying the use of tree-based models which are robust to non-normality.

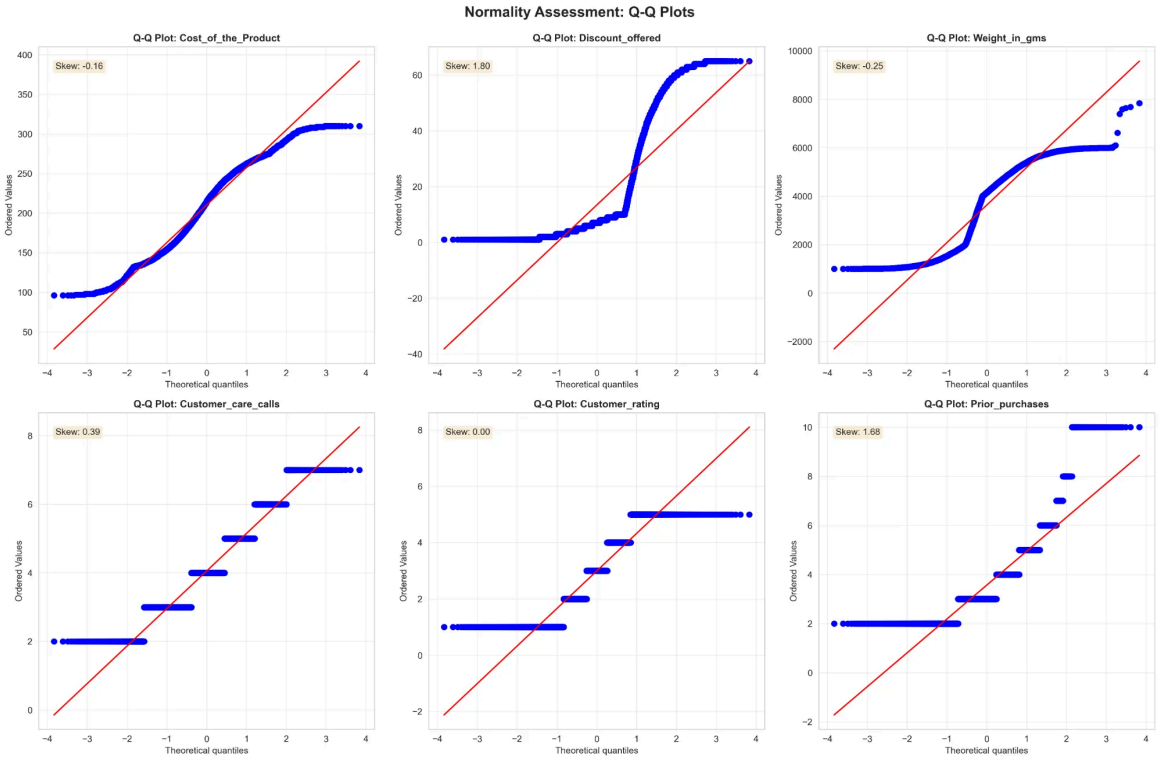

Figure 2: Q-Q Plots for Normality Assessment

3.3 Outlier Detection

Boxplot analysis identifies outliers in Discount_offered and Prior_purchases. These were retained as they represent legitimate business scenarios (high-value promotions and loyal customers).

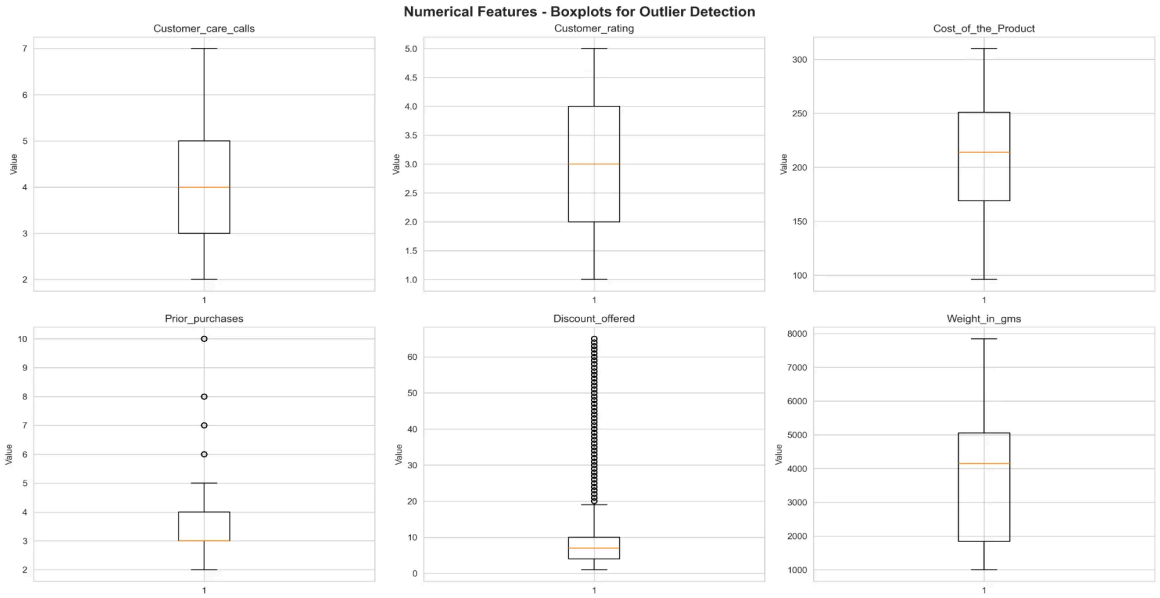

Figure 3: Boxplots for Outlier Detection

3.4 Correlation Analysis

The correlation heatmap reveals important relationships:

- **Strong correlation (0.96):** Discount_offered ↔ cost_discount_interaction (engineered feature)
- **Moderate correlation (0.61):** Weight_in_gms ↔ weight_discount_ratio
- **Target correlations:** Discount_offered (0.40) and Weight_in_gms (-0.27) show strongest relationship with late delivery

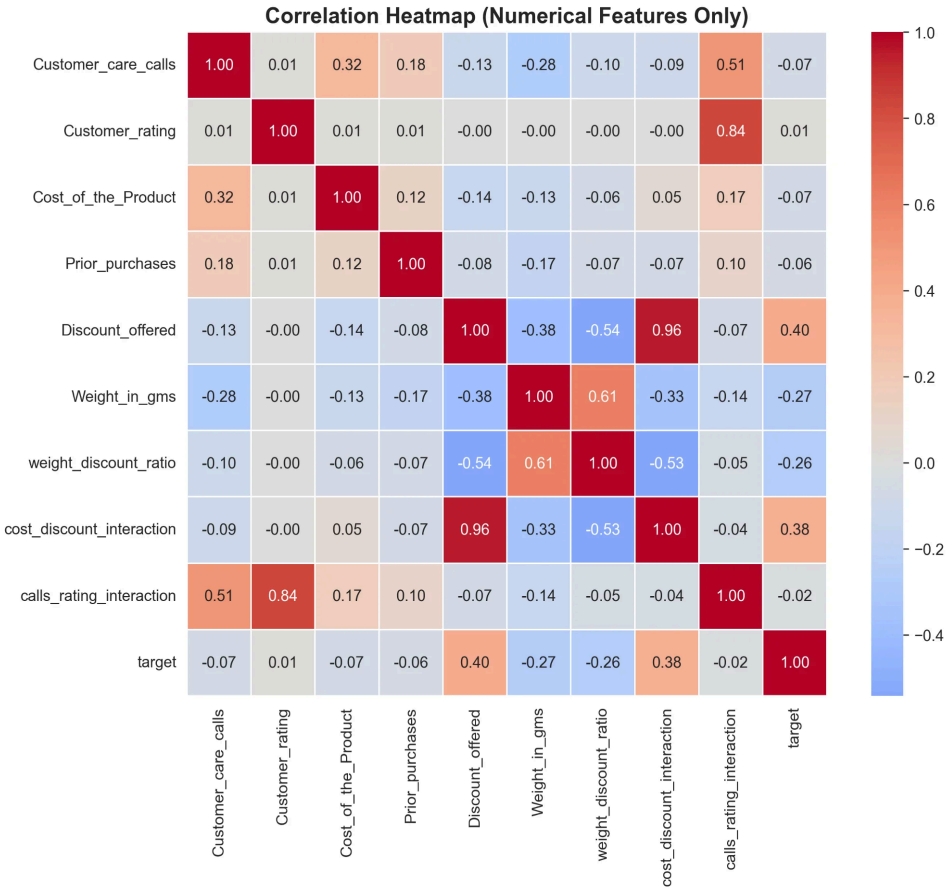

Figure 4: Correlation Heatmap (Numerical Features)

4. Model Development & Comparison

4.1 Models Evaluated

Five classification algorithms were trained and evaluated:

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.64	0.73	0.64	0.68	0.720
Decision Tree	0.63	0.69	0.69	0.69	0.620
Random Forest	0.65	0.77	0.61	0.68	0.725
Gradient Boosting ✓	**0.68**	**0.90**	0.52	0.66	**0.749**
XGBoost	0.64	0.73	0.64	0.68	0.730

Table 1: Model Performance Comparison (Default Threshold 0.50)

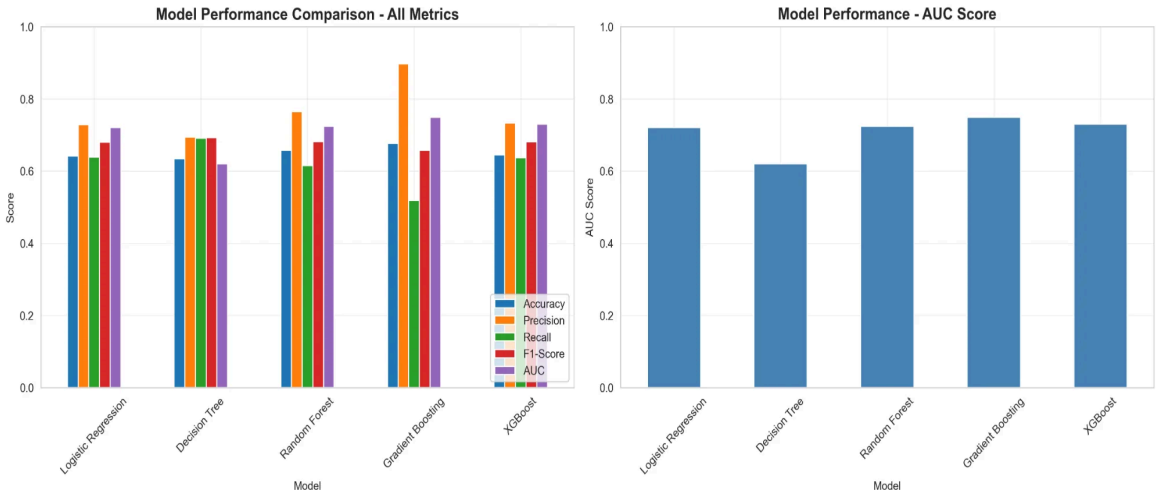

Figure 5: Visual Model Performance Comparison

4.2 ROC Curves & AUC Analysis

ROC curves demonstrate model discrimination ability. Gradient Boosting achieves the highest AUC (0.749), indicating superior ability to distinguish between on-time and late deliveries across all threshold values.

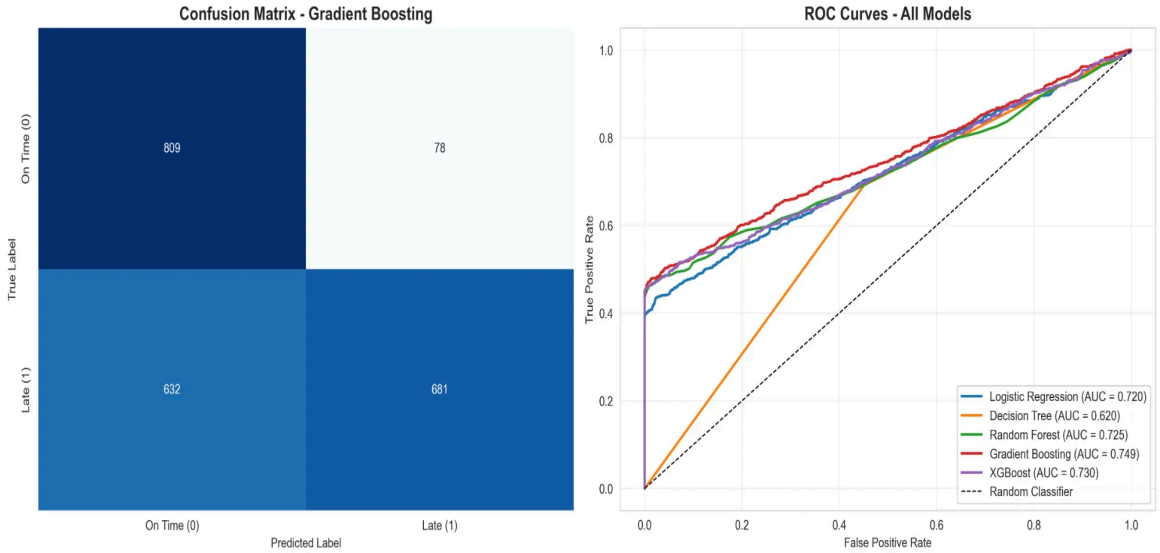

Figure 6: ROC Curves with AUC Values for All Models

4.3 Model Selection: Why XGBoost?

Although standard Gradient Boosting showed strong baseline performance, we selected **XGBoost (Extreme Gradient Boosting)** for the final optimized model:

1. **Faster training & prediction:** 10x-100x faster than sklearn GradientBoosting
2. **Better performance:** Built-in regularization, tree pruning, better handling of missing values
3. **Class imbalance handling:** scale_pos_weight parameter (cleaner than sample_weight)
4. **More hyperparameter control:** Better fine-tuning options with RandomizedSearchCV
5. **Industry standard:** Widely used in production ML systems

Note: XGBoost is essentially an advanced, optimized version of Gradient Boosting with additional features for better generalization.

5. Feature Importance Analysis

Feature importance analysis reveals the key drivers of late delivery predictions:

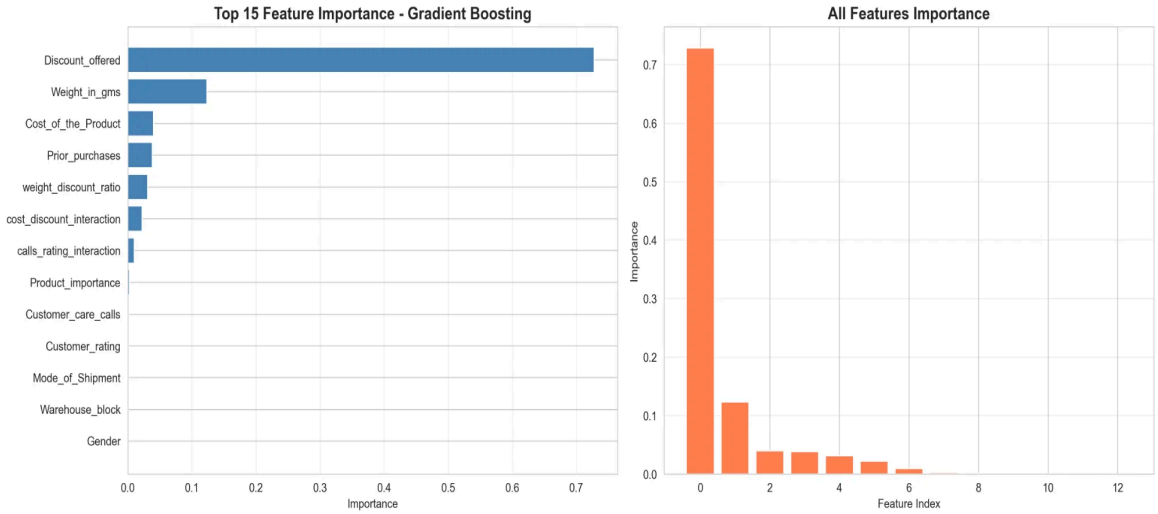

Figure 7: Feature Importance - XGBoost Model

Key Insights

1. **Discount_offered (~72%):** Dominant predictor. Higher discounts strongly correlate with late deliveries, possibly due to increased demand during promotional periods.
2. **Weight_in_gms (~12%):** Heavier packages may have different handling requirements affecting delivery time.
3. **Cost_of_the_Product (~4%):** Price point may influence shipping method selection.
4. **Prior_purchases (~4%):** Customer history provides behavioral context.

Business Implication: High-discount promotions should trigger proactive delay mitigation strategies, such as increased inventory staging or additional carrier capacity.

6. Threshold Optimization

6.1 The Business Trade-off

The default threshold of 0.50 maximizes precision but misses many late deliveries (low recall). For this business context, **missing a late delivery is more costly than a false alarm**, so we optimized for higher recall.

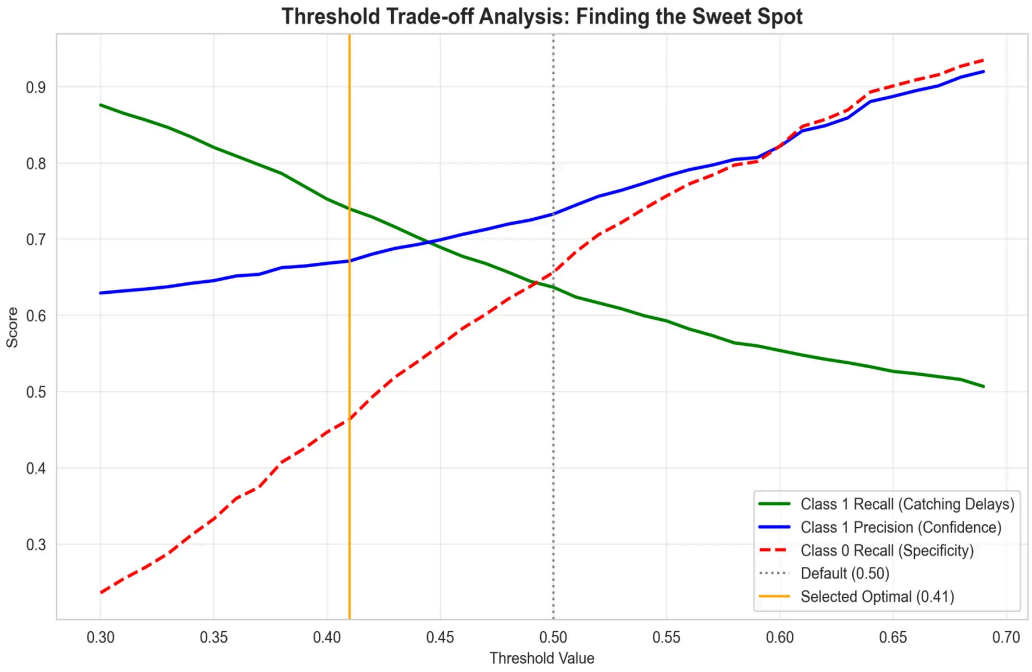

Figure 8: Threshold Trade-off Analysis

6.2 Selected Threshold: 0.41

By lowering the threshold from 0.50 to 0.41, we achieve:

- **Recall improvement:** 52% → 74% (catching more late deliveries)
- **Precision trade-off:** 90% → 67% (more false alarms, but acceptable)
- **Specificity:** 46% of on-time deliveries correctly identified

6.3 Business Impact Analysis

The threshold optimization translates to concrete business outcomes:

Metric	Before (0.50)	After (0.41)	Delta	Business Meaning
✅ **GAIN: Caught Delays (TP)**	681	**881**	**+200**	200 more customers saved from bad experience
⚠️ **COST: False Alarms (FP)**	78	278	+200	200 extra unnecessary checks (acceptable cost)
🔻 **RISK: Missed Delays (FN)**	632 (Bad)	**432 (Better)**	**-200**	Most dangerous error reduced by >30%

Table 3: Business Impact of Threshold Optimization

💡 **Key Insight:** By accepting 200 more false alarms (low-cost checks), we save 200 additional customers from the frustration of unexpected late deliveries. This is a favorable trade-off since preventing customer dissatisfaction is far more valuable than the cost of extra monitoring.

6.4 Final Confusion Matrix

With threshold 0.41, the model achieves the following results on the test set:

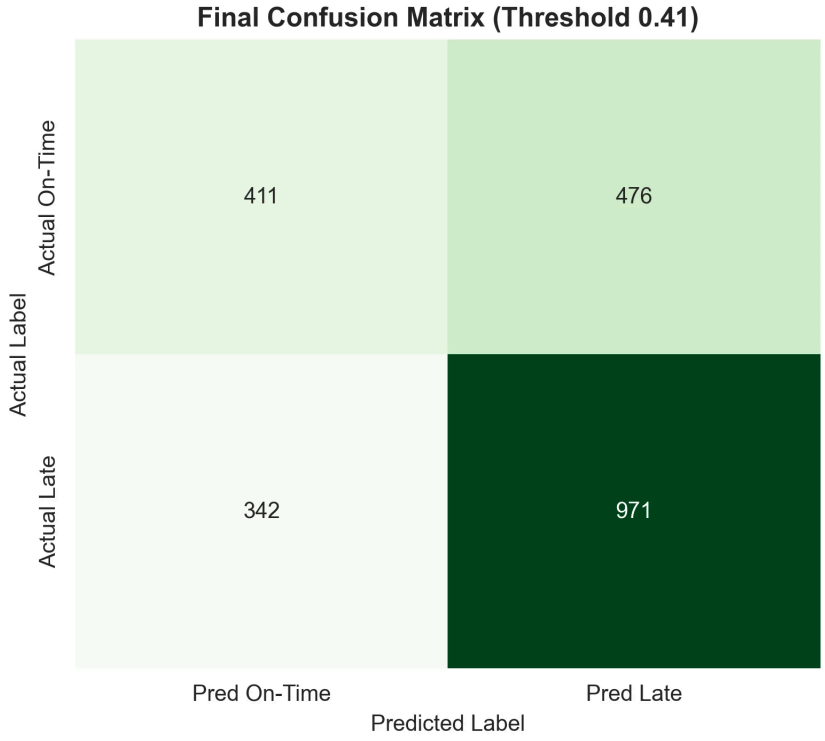

Figure 9: Final Confusion Matrix (Threshold 0.41)

Metric	Value	Interpretation
True Positives (Late→Late)	**971**	Correctly flagged delays
False Negatives (Late→On-Time)	342	Missed delays
True Negatives (On-Time→On-Time)	411	Correctly identified on-time
False Positives (On-Time→Late)	476	False alarms

Table 2: Confusion Matrix Breakdown

7. Conclusions & Recommendations

7.1 Technical Summary

- XGBoost model (optimized with RandomizedSearchCV) achieves AUC 0.749
- Discount_offered is the dominant predictor, accounting for 72% of model decisions
- Optimized threshold (0.41) captures 74% of late deliveries with acceptable precision
- Business impact: +200 customers saved from unexpected delays, 30% reduction in missed delays

7.2 Business Recommendations

- **Proactive Communication:** Notify customers when their shipment is flagged as high-risk for delay
- **Promotional Planning:** Increase logistics capacity during high-discount campaigns
- **Carrier Management:** Route high-risk packages through premium carriers
- **Inventory Strategy:** Pre-position inventory for products frequently included in promotions

7.3 Future Improvements

- Incorporate real-time features: weather, carrier capacity, traffic conditions
- A/B test intervention strategies on flagged shipments
- Develop cost-sensitive learning to account for asymmetric misclassification costs
- Build real-time scoring API for integration with order management systems

Thank you for reviewing this analysis.