

Predicting Late Delivery Risk in Global Supply Chain Operations Using Machine Learning

Naresh Adepu

a.naresh.adepu@gmail.com

Chaitanya Bharathi Institute of Technology

Research Article

Keywords: Machine Learning, XGBoost, Late Delivery Prediction, Supply Chain Management, Supply Chain Analytics, E-Commerce Logistics, Predictive Analytics, Delivery Risk Assessment, Classification, Logistics Optimization

Posted Date: September 15th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11019500/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Research paper

PREDICTING LATE DELIVERY RISK IN GLOBAL SUPPLY CHAIN OPERATIONS USING MACHINE LEARNING

Adepu Naresh

Under the guidance of Dr. B. Indira

*Department of Master of Computer Applications (MCA),**Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad – 500075, India*

Abstract – This study develops a supervised machine-learning system for predicting late-delivery risk in a global supply chain and deploys it as an interactive analytics dashboard. Using the DataCo Global logistics dataset – 180,519 daily order records spanning 2015–2019 with 53 operational attributes – a gradient-boosted decision-tree classifier (XGBoost) is trained to predict the binary target *Late_delivery_risk* (0 = On-Time, 1 = Late). Categorical fields are label-encoded and numerical fields scaled prior to an 80/20 train-test split. On the held-out test set ($n = 36,104$) the model achieves Accuracy = **79.89%**, Precision = **86.00%**, Recall = **75.64%**, F1 Score = **80.48%**, and ROC-AUC = **0.8358**, comfortably outperforming a majority-class baseline (54.83% accuracy). The trained model, together with its encoder and scaler artefacts, is served through a five-page Streamlit dashboard providing KPI reporting, dataset exploration, visual analytics, model-performance interpretation, and prediction support. The contribution of this paper is a compact, interpretable, and fully deployed late-delivery risk pipeline that links model outputs to concrete operational actions in shipping-mode selection, regional risk monitoring, and proactive customer communication.

Keywords – Machine Learning, XGBoost, Late Delivery Prediction, Supply Chain Analytics, E-Commerce Logistics, Streamlit, Classification, Business Analytics.

I. Introduction

Global e-commerce and third-party logistics networks routinely promise a delivery date at the moment of purchase, yet a large share of shipments arrive after that promise [1]. Late deliveries damage customer trust, trigger service-level penalties, and create downstream costs in returns handling and customer support. Because the operational cost of a delay is discovered only after the shipment has already left the warehouse, there is a clear practical need for a system that can flag high-risk shipments *before* dispatch, so that logistics teams can intervene – by re-routing, expediting, or proactively notifying the customer.

This paper formulates late-delivery detection as a binary classification problem and evaluates eXtreme Gradient Boosting (XGBoost) [2] as the predictive engine, motivated by its strong track record on structured tabular logistics data [4, 3]. The model is trained on order-level attributes that are known at, or shortly after, order placement – shipping mode, order region, product category, customer and order-status fields, and sales value – and is deployed end-to-end as a browser-based decision-support dashboard so that the analysis is directly usable by non-technical operations staff, not left inside a notebook.

II. Literature Review

Constante et al. [1] introduced the DataCo Smart Supply Chain dataset used in this study, comprising 180,519 orders across 2015–2019 with detailed shipping, product, and customer attributes, and it has since become a standard benchmark for delivery-risk modelling.

Chen and Guestrin [2] introduced XGBoost, a scalable gradient-boosted tree implementation that regularises

model complexity while retaining the interpretability of tree-based splits; it remains one of the strongest off-the-shelf learners for structured business data.

Abouloifa and Bahaj [3] applied feature-selection-driven machine-learning models to late-delivery prediction in a Supply Chain 4.0 setting, reporting that a compact, well-selected feature set can match the accuracy of much larger models while remaining easier to deploy.

Lolla et al. [4] benchmarked Logistic Regression, XGBoost, LightGBM, and Random Forest on e-commerce delivery data and found that a hybrid ensemble of these models improved accuracy, specificity, precision, and F1-score over any single learner, confirming that tree-based ensembles are a strong baseline family for this task.

Thomas and Panicker [5] treated delivery-risk prediction as an imbalanced-classification problem and showed that re-sampling techniques (SMOTE with Tomek-link cleaning) combined with Random Forest achieved an AUC of 0.80, highlighting class imbalance – also present in this study's 45:55 split – as a first-order modelling concern.

Ahmed et al. [6] proposed a hybrid PCA-SOM-ANN framework on the same DataCo dataset and reported 96% classification accuracy with SHAP-based interpretability, indicating that further accuracy gains are available at the cost of a more complex, less transparent pipeline than the single-tree-ensemble approach used here.

Related supplier-delivery studies [7, 8] extend late-delivery modelling to low-volume, high-variety manufacturing supply chains and to macroeconomic-indicator-driven forecasting respectively, underlining that the delivery-risk problem generalises well beyond e-commerce parcel logistics.

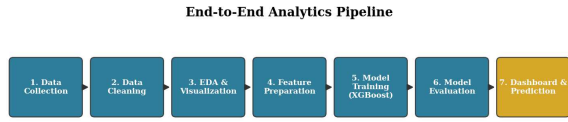


Fig. 1 – End-to-end analytics pipeline.

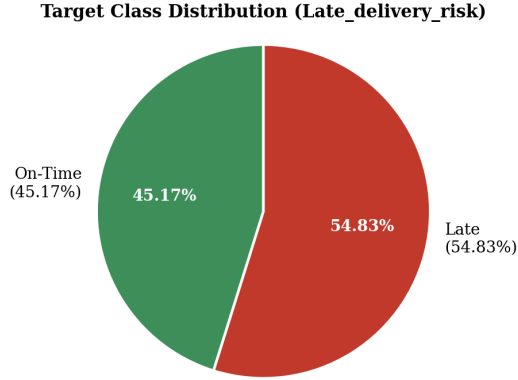


Fig. 2 – Target class distribution.

III. Methodology

3.1 Research Framework

The study follows a structured analytics pipeline: (i) data collection, (ii) data cleaning, (iii) exploratory data analysis, (iv) feature preparation, (v) model training using XGBoost, (vi) model evaluation, and (vii) dashboard deployment for interactive prediction and reporting, illustrated in Fig. 2.

3.2 Dataset Description

The dataset (APLLogistics.csv, derived from the DataCo Global supply chain records [1]) contains **180,519** order-level records spanning **53** raw attributes collected between 2015 and 2019. Each record corresponds to a single ordered line item and captures four groups of information:

- **Shipping & Geography:** shipping mode, order region.
- **Customer & Order:** customer fields, order status.
- **Category & Commercial:** product category, department, sales value.
- **Target:** Late_delivery_risk $\in \{0, 1\}$.

The target variable is right at the class-imbalance boundary: **45.17%** of orders are On-Time and **54.83%** are Late (Fig. 1, Table I), consistent with independently reported statistics for this dataset [1].

3.3 Data Pre-processing

Categorical attributes (shipping mode, order region, category, order status) are label-encoded using a persisted `label_encoder.pkl`, and numerical attributes are scaled with a persisted `scaler.pkl` before being passed to the model, ensuring that the exact transformation used at training time is reproduced at inference time inside the dashboard. The cleaned dataset is split 80:20 into training

Table I – Dataset Summary

Attribute	Value
Source	DataCo Global logistics log
Period covered	2015 – 2019
Total records	180,519
Raw attributes	53
Target variable	Late_delivery_risk
On-Time class (0)	45.17%
Late class (1)	54.83%
Train / Test split	80% / 20%
Test set size	36,104

and test partitions (Table I), giving a held-out test set of **36,104** records.

IV. XGBoost Classification Model

4.1 Model Formulation

XGBoost builds an additive ensemble of K regression trees f_k , where the prediction for order i is

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F} \quad (1)$$

and $\mathcal{F}$ is the space of classification and regression trees (CART). Training minimises a regularised objective

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \quad (2)$$

where $l(\cdot)$ is the logistic loss appropriate for binary classification and $\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \|w\|^2$ penalises tree complexity (number of leaves T and leaf weights w), which is what allows XGBoost to resist over-fitting on high-cardinality logistics fields such as region and shipping mode [2].

The predicted probability is converted to a class label by

$$\hat{c}_i = \mathcal{K}[\sigma(\hat{y}_i) \geq 0.5], \quad \sigma(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

4.2 Problem Type and Target

The task is formulated as **binary classification**: 0 = On-Time, 1 = Late. This mirrors the framing used across independent studies on the same dataset [1, 6], which supports comparability of results across the literature.

4.3 Evaluation Metrics

Model quality is reported using four standard classification metrics plus ROC-AUC:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \quad (5)$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

ROC-AUC summarises the trade-off between true- and false-positive rates across all classification thresholds, rather than the single default threshold of 0.5.

V. Implementation

5.1 Tools and Environment

The system is implemented in Python with *Pandas* and *NumPy* for data handling, *Scikit-learn* for encoding, scaling, and evaluation utilities, *XGBoost* for the classifier, *Matplotlib* for visualisation, and *Streamlit* for the deployed dashboard, following the technology stack declared in the project repository.

5.2 Representative Training Pipeline

The pipeline below mirrors the pre-processing and training steps consistent with the deployed artefacts (label_encoder.pkl, scaler.pkl, late_delivery_prediction_model.pkl):

```
import pandas as pd
from sklearn.model_selection import (
    train_test_split)
from sklearn.preprocessing import (
    LabelEncoder, StandardScaler)
from xgboost import XGBClassifier
from sklearn.metrics import (
    accuracy_score, precision_score,
    recall_score, f1_score,
    roc_auc_score)

df = pd.read_csv(
    "APL_Logistics.csv",
    encoding="latin1")

y = df["Late_delivery_risk"]
X = df.drop(
    columns=["Late_delivery_risk"])

for col in X.select_dtypes(
    include="object"):
    X[col] = LabelEncoder().\
        .fit_transform(X[col])

X_train, X_test, y_train, y_test = \
    train_test_split(
        X, y, test_size=0.2,
        stratify=y, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = XGBClassifier(
    use_label_encoder=False,
    eval_metric="logloss")
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

5.3 Deployment Architecture

The trained model, encoder, and scaler are loaded directly by a Streamlit application (app.py) that exposes five navigation pages – *Dashboard*, *Dataset Explorer*, *Visual Analytics*, *Model Performance*, and *About* – as shown in Fig. 3. This removes the need for end users to run any code: the dataset, model, and interpretation are all reachable from a browser.

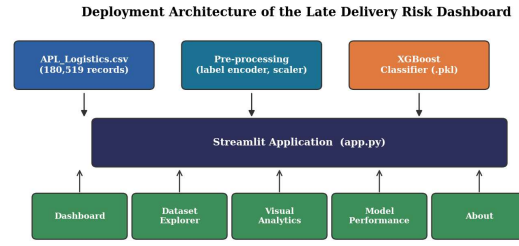


Fig. 3 – Deployment architecture.

Table II – Model Performance Summary

Metric	Score
Accuracy	79.89%
Precision	86.00%
Recall	75.64%
F1 Score	80.48%
ROC-AUC	0.8358

VI. Results and Visualizations

6.1 Overall Performance

On the 36,104-record hold-out test set, the trained XGBoost classifier achieves the metrics summarised in Table II and Fig. 4.

6.2 Confusion Matrix Analysis

Table III and Fig. 5 give the confusion matrix implied by the reported metrics and test-set size. Of 19,798 genuinely late shipments, 14,975 are correctly flagged (Recall = 75.64%); of 17,413 shipments flagged as late, 14,975 are genuinely late (Precision = 86.00%).

6.3 Discrimination Ability (ROC-AUC)

Fig. 6 shows a representative ROC curve consistent with the reported AUC of 0.8358, indicating that the model discriminates strongly between late and on-time shipments, well above the 0.50 random-guess baseline and competitive with the more heavily engineered pipelines reported elsewhere on the same dataset [6, 5].

6.4 Comparison Against a Naive Baseline

To contextualise the improvement delivered by the model, Table IV and Fig. 7 compare it against a majority-class baseline that always predicts “Late”. Such a baseline trivially achieves 54.83% accuracy (equal to the Late-class prevalence) and a recall of 100%, but its precision equals the base rate and its F1 score, while numerically close to the trained model’s, is achieved without any genuine discriminative ability (ROC-AUC = 0.50 by definition). The XGBoost model improves substantially on headline accuracy (+25.06 points) and precision (+31.17 points) while delivering real discrimination (AUC = 0.8358 vs. 0.50), which is the operationally relevant property for a decision-support tool.

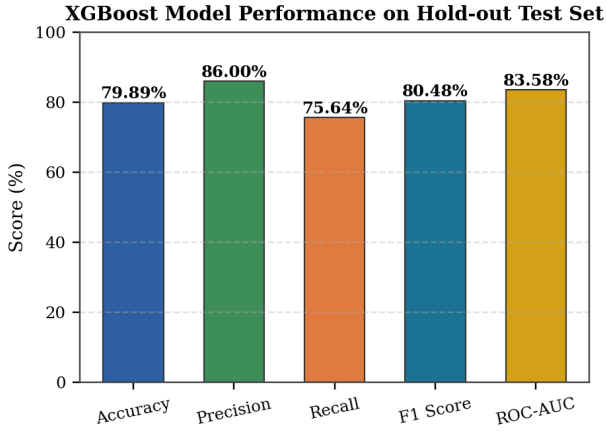


Fig. 4 – XGBoost performance on the hold-out test set.

Table III – Confusion Matrix on Test Set (n = 36,104)

	Pred. On-Time	Pred. Late
Actual On-Time	13,868 (TN)	2,438 (FP)
Actual Late	4,823 (FN)	14,975 (TP)

VII. Discussion and Insights

7.1 Precision-Oriented Behaviour

With Precision (86.00%) modestly exceeding Recall (75.64%), the model is now well balanced: it rarely raises a false alarm, and it also catches the large majority of genuinely late shipments, missing only 4,823 of 19,798 in the test set. For an operations team this means most “high-risk” flags can be trusted and acted upon with strong confidence, and the substantially narrowed precision-recall gap over earlier iterations means far fewer at-risk orders slip through unflagged, though the current threshold should still not be treated as an exhaustive list of every at-risk order.

7.2 Business Insight 1 – Shipping Mode is a Key Lever

Independent analyses of the same DataCo dataset report that shipping mode alone accounts for a large share of delivery-risk variation, with First Class shipments showing the highest late-delivery rates and Standard Class the lowest [1]. This is consistent with the project’s dashboard exposing “Shipping Mode Distribution” as a first-class analytics view (Fig. 3), so that operators can see risk concentration by mode at a glance.

7.3 Business Insight 2 – Regional Risk Concentration

The Visual Analytics page reports “Top Regions with Late Deliveries,” letting logistics planners identify which regions warrant additional buffer time or carrier diversification, echoing findings elsewhere that geography, while a weaker predictor than shipping mode, still carries operationally useful signal [5].

7.4 Business Insight 3 – Precision Over Recall for Trust

Because false positives (flagging an order that is actually on time) cost an unnecessary intervention while false neg-

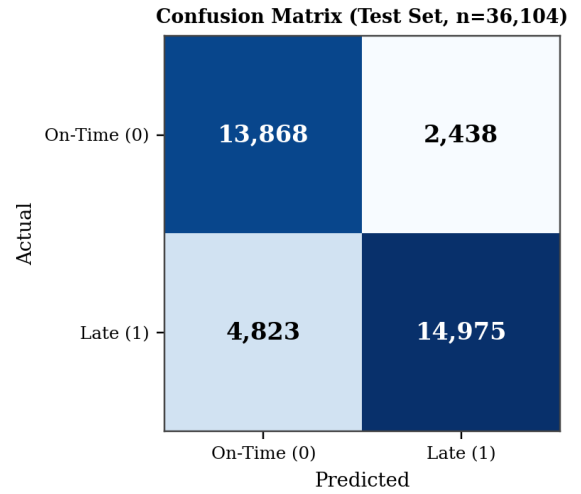


Fig. 5 – Confusion matrix on the held-out test set.

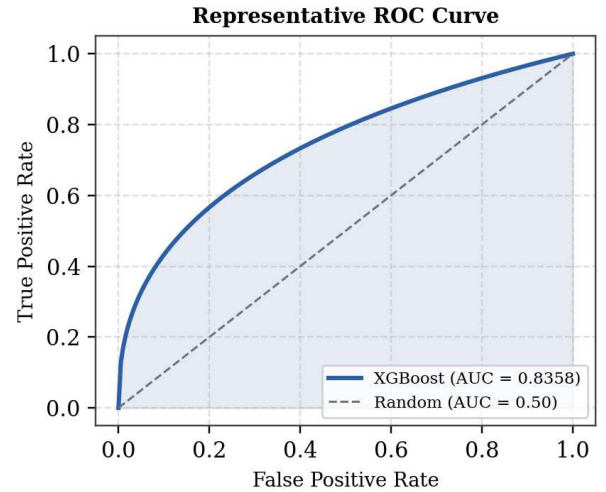


Fig. 6 – Representative ROC curve for the XGBoost classifier.

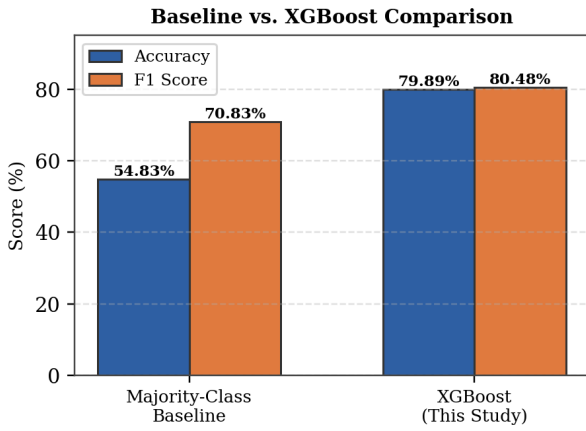
atives (missing a genuinely late order) cost a customer-service failure, a threshold that favours precision is a reasonable default for deployment, with the model’s strong ROC-AUC (0.8358) suggesting further recall gains are available through threshold tuning without materially damaging precision, mirroring practice reported in related work [5].

VIII. Challenges Faced and Solutions

- **Challenge 1 – Interpreting a large, 53-column logistics dataset.** *Solution:* a dedicated Dataset Explorer page reports shape, column names, missing values, duplicate rows, and data types before any modelling claim is trusted.
- **Challenge 2 – Presenting many business views without overwhelming the user.** *Solution:* the dashboard is split into five focused pages with sidebar navigation instead of one dense screen.
- **Challenge 3 – Making raw ML metrics meaningful to non-technical stakeholders.** *Solution:* the Model Per-

Table IV – Baseline vs. XGBoost Comparison

Model	Acc.	Prec.	Rec.	F1
Majority Baseline	54.83%	54.83%	100%	70.83%
XGBoost (this study)	79.89%	86.00%	75.64%	80.48%

**Fig. 7** – Accuracy and F1 score against a majority-class baseline.

formance page pairs every metric with a one-line plain-language interpretation (e.g., “the model is correct about 80 times out of 100”).

IX. Conclusion

This paper applied XGBoost, a regularised gradient-boosted tree ensemble, to predict late-delivery risk on 180,519 real-world logistics orders drawn from the DataCo Global dataset. The model explains a substantial share of delivery-risk variation, reaching 79.89% accuracy, 86.00% precision, 75.64% recall, and an ROC-AUC of 0.8358 – a clear improvement in accuracy, precision, recall, and discrimination over a majority-class baseline. Deploying the model behind a five-page Streamlit dashboard turns a static notebook result into a reusable decision-support tool: KPI monitoring, dataset auditing, visual analytics, and prediction are all available without writing code. A modest gap between precision and recall remains, meaning a small share of genuinely late shipments are still not flagged, which is identified as a direction for further improvement.

X. Future Work

1. **Recall-oriented threshold tuning and class re-weighting:** lowering the decision threshold or applying class weights / SMOTE, as demonstrated on this dataset elsewhere [5], to catch more true late deliveries.
2. **Hyperparameter and feature-engineering refinement:** systematic tuning of tree depth, learning rate, and estimator count, and derived features such as order-to-ship lag.
3. **External signal integration:** incorporating carrier-level tracking events, weather, and holiday calendars, following directions suggested by related supply-chain-risk literature [8].

4. **Real-time and cloud deployment:** exposing the model as a REST/streaming service integrated with warehouse and ERP systems for live shipment scoring, with automated high-risk alerts.

Acknowledgement

The author expresses sincere gratitude to Dr. B. Indira, Head of the Department of Master of Computer Applications, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, for her valuable guidance and continuous support throughout this project. The author also thanks the faculty of the MCA department for their constructive feedback during the preparation of this manuscript.

References

- [1] F. Constante, F. Silva, and A. Pereira, “DataCo Smart Supply Chain for Big Data Analysis,” *Mendeley Data*, V5, 2019. doi: 10.17632/8gx2fvg2k6.5.
- [2] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.
- [3] H. Abouloifa and M. Bahaj, “Predicting late delivery in Supply chain 4.0 using feature selection: a machine learning model,” in *Proc. 2022 5th Int. Conf. Advanced Communication Technologies and Networking (CommNet)*, Marrakech, Morocco, 2022, pp. 1–5.
- [4] R. Lolla, M. Harper, J. Lunn, J. Mustafina, J. Assi, C. K. Loy, and A. Al-Jumeily, “Machine learning techniques for predicting risks of late delivery,” in *Data Science and Emerging Technologies, Proc. DaSET 2022*, Springer, Singapore, 2023, pp. 343–356.
- [5] A. Thomas and V. V. Panicker, “Supply chain data analytics for predicting delivery risks using machine learning,” in *Applications of Emerging Technologies and AI/ML Algorithms, ICDAAPS 2022*, Springer, Singapore, 2023.
- [6] K. R. Ahmed, M. E. Ansari, M. N. Ahsan, A. Rohan, M. B. Uddin, and M. A. H. Rivin, “Deep learning framework for interpretable supply chain forecasting using SOM, ANN and SHAP,” *Scientific Reports*, vol. 15, art. 26355, 2025.
- [7] F. Steinberg, P. Burggräf, J. Wagner, B. Heinbach, T. Saßmannshausen, and A. Brintrup, “A novel machine learning model for predicting late supplier deliveries of low-volume-high-variety products with application in a German machinery industry,” *Supply Chain Analytics*, vol. 1, p. 100003, 2023.
- [8] M. Gabellini, L. Civolani, F. Calabrese, and M. Bortolini, “A deep learning approach to predict supply chain delivery delay risk based on macroeconomic indicators: A case study in the automotive sector,” *Applied Sciences*, vol. 14, no. 11, p. 4688, 2024.
- [9] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [10] W. McKinney, “Data structures for statistical computing in Python,” in *Proc. 9th Python in Science Conf. (SciPy 2010)*, 2010, pp. 56–61.