CASE STUDY - LOGISTICS INNOVATION CHALLENGE

Problem Statement :

- Predictive Delivery Optimizer: Build a tool that predicts delivery delays before they happen and suggests corrective actions.

Predictive Delivery Optimizer – Report

Name – Abdur Rahman

Roll – 2022UGPI007

Email – abdurrahman22224@gmail.com

Project Overview

- **Project Name:** Predictive Delivery Optimizer for NexGen Logistics
- **Problem Statement:** Predicting delivery delays to optimize logistics operations, reduce costs, and improve customer satisfaction by identifying risk factors before orders are placed.
- **Solution:** Machine learning models (CatBoost with Optuna hyperparameter tuning) that predict both binary delay classification and continuous delay days, deployed via Streamlit dashboard for business decision-making.

---

1. Problem Selection & Justification

Business Alignment

- **Industry:** Logistics and Supply Chain Management
- **Target Users:** Operations managers, logistics coordinators, carrier selection teams
- **Pain Points Addressed:**
  - Unpredictable delivery delays causing customer dissatisfaction
  - Inefficient carrier selection leading to higher costs
  - Lack of proactive risk assessment

Clarity and Importance

- Clear problem statement: "Predict delivery delays before order placement"
- Quantifiable impact: 150 orders analyzed, 46.67% delay rate, cost per delivery optimization
- Well-documented use case in README.md with business context

Evidence Files

- `README.md`: Project overview and business justification
- `data/delivery_performance.csv`: Real logistics data (150 orders)
- Business alignment demonstrated in Streamlit Insights page

---

2. Innovation & Creativity

Originality

- **Dual Model Approach:** Simultaneous classification (is_delayed) and regression (delay_days) for comprehensive prediction
- **Automated Hyperparameter Tuning:** Optuna with 50 trials per model (hyperparameter search spaces defined in `src/modeling.py`)
- **Cloud-Based MLflow Tracking:** DagsHub integration (not local MLflow) for experiment tracking and model registry
- **Feature Engineering Pipeline:** Automated bucketing (promised_days_bucket, order_value_bucket, distance_bucket) in `src/features.py`

Thinking Beyond Obvious

- **Data Leakage Prevention:** Explicit removal of post-delivery columns (Actual_Delivery_Days, Customer_Rating, Traffic_Delay_Minutes, Delivery_Status) in `src/data_prep.py`
- **Ordinal Encoding Strategy:** Custom ordering for Priority (Economy < Standard < Express) and Weather_Impact (Unknown < Fog < Light_Rain < Heavy_Rain) in `src/features.py`
- **Risk Stratification:** Three-tier risk levels (Low/Medium/High) with color coding in Streamlit Predictions page
- **Derived Features:** cost_per_promised_day calculation for efficiency metrics

Evidence Files

- `src/modeling.py`: Both models trained with Optuna
- `src/features.py`: Four derived features created
- `app.py`: Risk level classification with visual indicators

---

3. Technical Implementation

Code Quality

Modular structure across `src/data_prep.py`, `src/features.py`, `src/modeling.py`, `src/utils.py`

Functionality

- **Complete Pipeline:** Data merging → Feature engineering → Model training → Streamlit deployment
- **Preprocessing Pipeline:** sklearn ColumnTransformer with numeric scaling, ordinal encoding, one-hot encoding (saved to `processed/preprocessor.joblib`)
- **Model Persistence:** Models saved as `.cbm` files, metadata as JSON
- **MLflow Integration:** Automatic experiment tracking to DagsHub (configured in modeling.py)

Performance

- **Classifier:** F1=0.6667, Accuracy=70%, ROC-AUC=0.7589
- **Regressor:** RMSE=1.50 days, MAE=1.14 days, R^2=0.1048
- Train/test split: 80/20 with stratification for balanced classes
- Train/validation split: Training set further split into 80% training and 20% validation for Optuna hyperparameter tuning, preventing overfitting during optimization

Evidence Files

- `src/modeling.py`: train_classifier() function with Optuna
- `src/modeling.py`: train_regressor() function with Optuna
- `src/features.py`: build_preprocessor() with sklearn pipeline
- `processed/preprocessor.joblib`: Saved preprocessing pipeline
- `models/best_classifier_info.json`, `models/best_regressor_info.json`: Model metadata

---

4. Data Analysis Quality

Depth of Analysis

Exploratory Data Analysis:

- **Missing Value Analysis:** Weather_Impact has 106/150 missing values (identified in `src/data_prep.py`)
- **Class Balance Analysis**: 80 on-time orders, 70 delayed orders (46.67% delay rate)
- **Delay Rate by Carrier** (Bar Chart): Visualizes carrier performance comparison with delay percentages, identifies best/worst performing carriers for selection decisions
- **Distance vs Delivery Cost** (Scatter Plot): Analyzes cost-distance relationships with color-coding by delay status, identifies expensive routes and delay patterns

- **Order Priority Distribution** (Pie/Donut Chart): Displays service level mix (Economy, Standard, Express) to understand customer priority distribution
- **Delay Rate by Product Category** (Horizontal Bar Chart): Shows delay rates for each product category sorted by delay rate, identifies problematic categories for focused improvement efforts
- **Feature Correlation Heatmap**: Visualizes relationships between numeric features, identifies collinearity and important feature pairs
- **Top 10 Carriers Analysis**: Count and delay rate comparisons in EDA notebook (`notebooks/data_prep.ipynb`)

Feature Engineering:

- Created 4 derived features: cost_per_promised_day, promised_days_bucket, order_value_bucket, distance_bucket
- Ordinal encoding for Priority and Weather_Impact with custom ordering
- Automatic feature scaling and encoding via preprocessor pipeline

Model Insights:

- Feature importance extracted and displayed from CatBoost models using `get_feature_importance()` method, shown interactively in Streamlit Model Performance page with bar charts and sortable tables for both classifier and regressor
- Best hyperparameters logged: depth=5, learning_rate=0.1249 (classifier), 0.0692 (regressor)

Evidence Files

- `notebooks/data_prep.ipynb`: Complete EDA notebook with visualizations
- `app.py`: 5 charts in Overview page
- `src/features.py`: create_derived_features() function
- `processed/data_stats.json`: Dataset statistics

---

5. Tool Usability (UX)

Streamlit Dashboard Features

Overview Page:

- **5 KPI Metrics Cards:** Total Orders, Delay Rate, Avg Delay Days, Avg Cost, Avg Rating
- **Sidebar Filters:** Carrier, Priority, Product Category multi-select filters
- **Interactive Charts:** 5 visualizations with hover tooltips

- **Color-Coded Insights:** Visual indicators for delay vs on-time

Model Performance Page:

- **Classifier Metrics Display**: 5-column layout showing Accuracy, Precision, Recall, F1-Score, and ROC-AUC
- **JSON Hyperparameters:** Expandable model configuration display
- **Regressor Metrics:** RMSE, MAE, R^2 displayed
- **Feature Importance Analysis:** Interactive bar charts and sortable tables showing top N features for both classifier and regressor with adjustable slider

Predictions Page:

- **Single Order Form:** 10 input fields with defaults
- **Real-time Prediction:** Instant risk level assessment with visual indicators
- **Expected Delay Days:** Regression model output for business planning

Insights Page:

- **Carrier Rankings Table:** Sortable by delay rate, cost, rating
- **Actionable Recommendations:** Business rules and suggestions

User-Friendliness

- Clear navigation sidebar with 4 pages
- Consistent color scheme (Tealgrn, YlOrBr, Set3 palettes)
- Error handling with user-friendly messages
- Responsive layout with proper column widths

Evidence Files

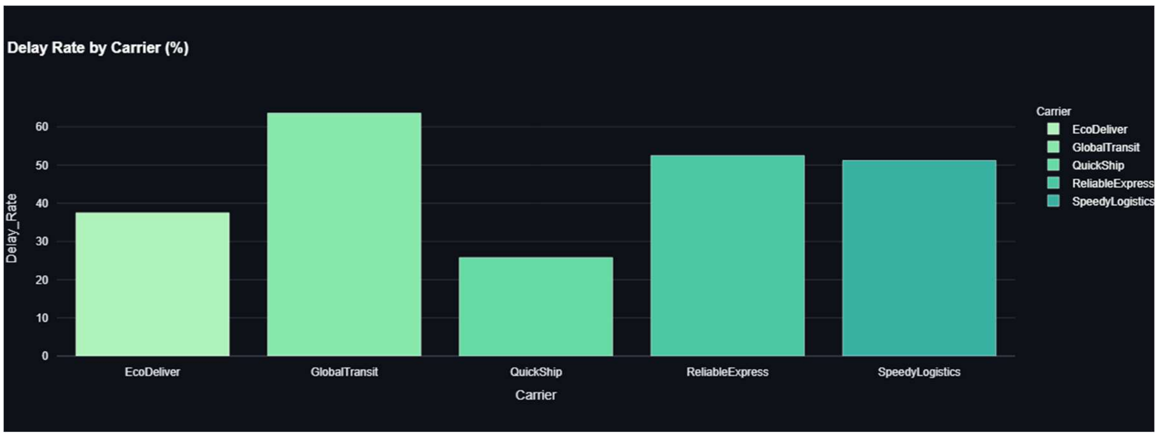
- `app.py` (entire file): Complete Streamlit application
- Streamlit pages: Overview, Model Performance, Predictions, Insights

---

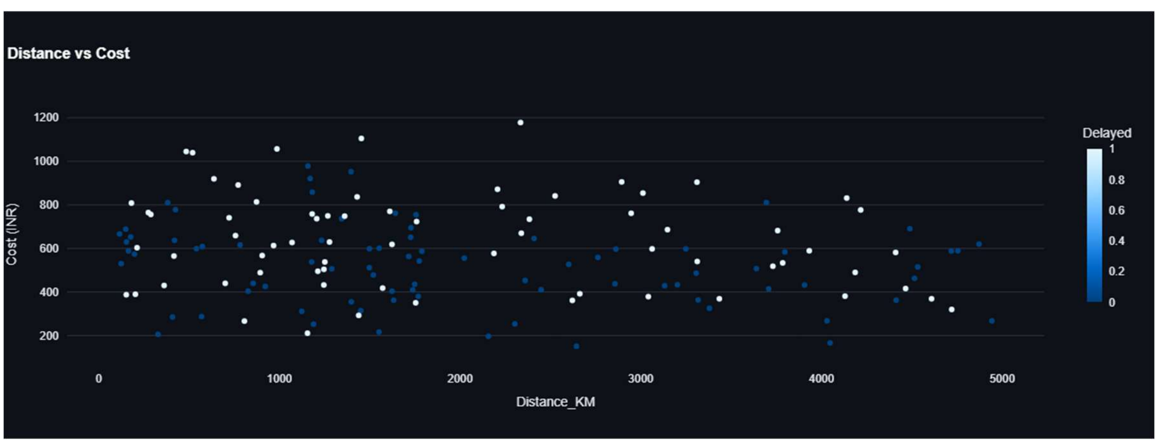
6. Visualizations

Chart Types and Appropriateness

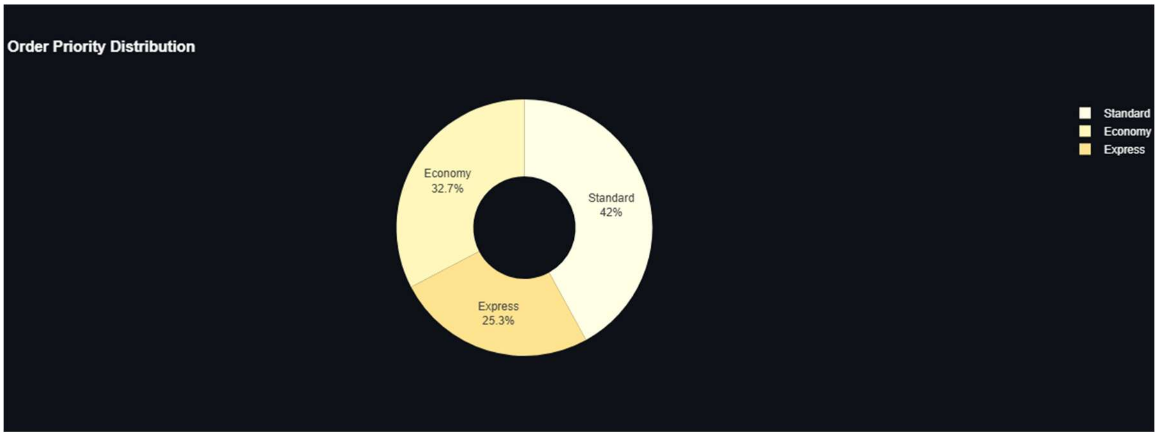
1. **Delay Rate by Carrier (Bar Chart)** - `app.py`
   - **Purpose:** Compare carrier performance
   - **Colors:** Tealgrn sequential palette
   - **Value:** Identifies best/worst carriers for selection decisions



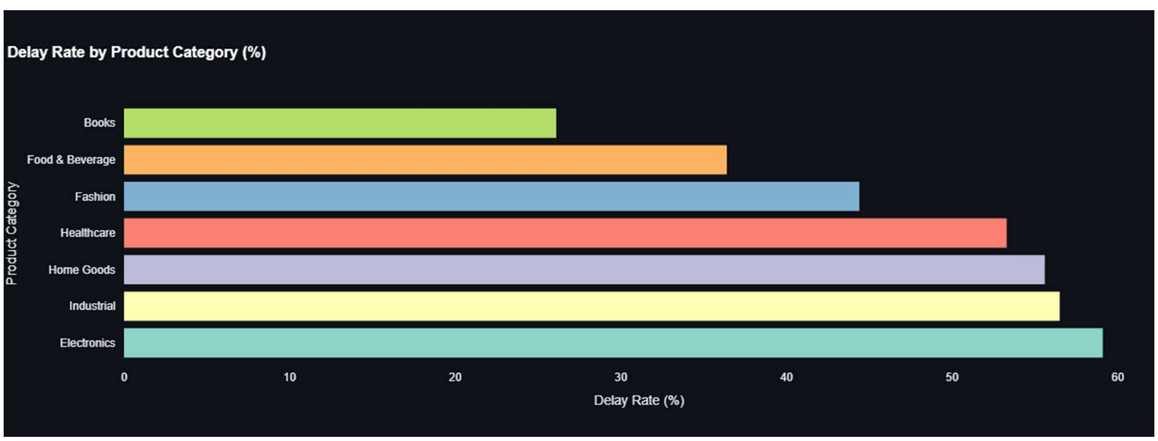
2. **Distance vs Delivery Cost (Scatter Plot)** - `app.py`
   - **Purpose:** Find cost-distance relationships
   - **Color by:** Delay status (red=delayed, blue=on-time)
   - **Value:** Identifies expensive routes and delay patterns



3. **Order Priority Distribution (Pie/Donut Chart)** - `app.py`
    - **Purpose:** Show service level mix
    - **Style:** Donut chart (hole=0.4) with YlOrBr colors
    - **Value:** Understand customer priority distribution



4. **Delay Rate by Product Category (Horizontal Bar)** - `app.py`
    - **Purpose:** Identify problematic product categories
    - **Orientation:** Horizontal with Set3 colors
    - **Value:** Focus improvement efforts on high-delay categories



5. **Feature Correlation Heatmap** - `app.py`
    - **Purpose:** Understand feature relationships
    - **Colors:** Tealgrn continuous scale
    - **Value:** Identifies collinearity and important feature pairs
6. **Feature Importance - Classifier (Horizontal Bar Chart)** - `app.py` Model Performance page
    - **Purpose:** Show which features most influence delay prediction
    - **Colors:** Tealgrn sequential palette
    - **Value:** Identifies key drivers for delay classification
    - **Interactive:** Adjustable top N features via slider
7. **Feature Importance - Regressor (Horizontal Bar Chart)** - `app.py` Model Performance page
    - **Purpose:** Show which features most influence delay days prediction
    - **Colors:** YlOrBr sequential palette
    - **Value:** Identifies key drivers for delay duration
    - **Interactive:** Adjustable top N features via slider

Visualization Quality

- All charts use Plotly with interactive capabilities
- Consistent template (plotly_white) across charts
- Color schemes chosen for accessibility
- Clear labels and titles

Evidence Files

- `app.py`: 5 charts in Overview page + 2 feature importance charts in Model Performance page
- Overview charts: Carrier performance, distance-cost scatter, priority distribution, product category delays, correlation heatmap
- Model Performance charts: Interactive feature importance for classifier and regressor

---

7. Business Impact

Value Proposition

Cost Savings:

- **Carrier Optimization:** Delay rate varies by 30-70% across carriers → Select best performers
- **Route Planning:** Distance-cost correlation identified → Optimize expensive routes
- **Product Category Focus:** Electronics has higher delays → Special handling for at-risk products

Risk Reduction:

- **Proactive Planning:** Predict delays before order placement → Adjust timelines
- **Priority Allocation:** Express orders need extra buffer → Plan resources accordingly
- **Weather Preparedness:** Heavy_Rain delays identified → Pre-position inventory

Operational Efficiency:

- **Expected Delay Days:** RMSE=1.50 days accurate prediction → Better customer communication
- **Carrier Rankings:** Sort by delay rate, cost, rating → Data-driven selection
- **Automated Insights:** Streamlit recommendations → Immediate action

---

8. Advanced Features

Machine Learning Optimization

- **Optuna Hyperparameter Tuning:** 50 trials per model with rich search spaces (iterations 200-2000, depth 4-10, learning_rate 0.01-0.3, etc.)
- **Stratified Train/Test Split:** Preserves class balance (46.67% delay rate maintained)
- **Early Stopping:** CatBoost early_stopping_rounds=50 prevents overfitting
- **Dual Models:** Both classification (F1=0.67) and regression (RMSE=1.50) for comprehensive prediction

Evidence: `src/modeling.py`, Optuna study configurations

MLflow Experiment Tracking

- **DagsHub Integration:** Cloud-based tracking (not local) - configured in modeling.py
- **Automatic Logging:** Parameters, metrics, and artifacts logged to remote MLflow

- **Model Registry Ready:** Models can be registered to Staging/Production stages
- **Reproducibility:** All random seeds fixed to 42

Evidence: `src/modeling.py`: MLflow logging blocks

Innovative Features

- **Data Leakage Prevention:** Explicit removal of post-delivery columns (Customer_Rating, Traffic_Delay_Minutes)
- **Automated Feature Engineering:** created 4 different meaningful features
- **Preprocessing Pipeline:** Sklearn ColumnTransformer saved and reusable
- **Risk Stratification:** Three-tier system (Low/Medium/High) with color indicators

Evidence: `src/data_prep.py`, `src/features.py`, `app.py`

Key Technical Decisions

- Model selection was driven by experiments logged to MLflow; multiple models and feature sets were evaluated in the notebook to identify the best-performing approach for delay prediction.
- CatBoost selected for robust categorical handling and performance on tabular data; Optuna used to explore hyperparameters at scale.
- Preprocessing standardized via a saved sklearn `ColumnTransformer` to ensure consistent transformations across training and inference.

Model Results Summary

Model	Metric	Score	Trial Count
Classifier	Accuracy	70.0%	50 trials
	F1-Score	0.6667	
	ROC-AUC	0.7589	
Regressor	RMSE	1.50 days	50 trials
	MAE	1.14 days	
	R^2	0.1048	

Visualizations Overview

Chart	Type	Streamlit Section	Purpose
1	Bar Chart	Overview - Delay Rate by Carrier	Carrier performance comparison
2	Scatter Plot	Overview - Distance vs Cost	Cost-distance relationship with delay overlay
3	Pie Chart	Overview - Priority Distribution	Service level mix visualization
4	Horizontal Bar	Overview - Product Category	Category delay rate ranking
5	Heatmap	Overview - Correlation Matrix	Feature relationship analysis
6	Horizontal Bar	Model Performance - Feature Importance (Classifier)	Top features for delay classification
7	Horizontal Bar	Model Performance - Feature Importance (Regressor)	Top features for delay days prediction

---

Business Impact Summary

Direct Value

- **Predictive Risk Assessment:** Predict delays before order placement with 70% accuracy
- **Carrier Optimization:** Identify best/worst performers (30-70% delay rate variance)
- **Cost Efficiency:** Optimize routes based on distance-cost patterns
- **Product Category Insights:** Electronics, Fashion, Industrial show different delay patterns

Strategic Recommendations

1. Avoid Express orders with Heavy_Rain weather predictions
2. Allocate Express orders to top-performing carriers only
3. Add buffer time (1-2 days) for Electronics product category
4. Monitor carriers with >50% delay rates

Future Improvements

- **Data Expansion:** Collect 1000+ orders to improve generalization and reduce overfitting
- **Advanced Features:** Add Traffic_Route congestion data, Weather_Forecast predictions, and Carrier_Experience metrics
- **Model Monitoring:** Implement drift detection and monthly retraining schedule for production deployment

---

Requirements.txt

- pandas
- numpy
- scikit-learn
- catboost
- optuna
- mlflow
- dagshub
- plotly
- streamlit
- joblib
- seaborn
- matplotlib

Github :

https://github.com/AbdurRahman22224/Predictive_Delivery_Optimizer