Supply Chain Disruption Recovery Analytics

Project Overview

This project focuses on **analyzing, modeling, and predicting supply chain disruption recovery time** using real-world–style operational data. It combines **data engineering, machine learning, and scenario simulation** to help organizations understand how disruptions impact recovery and how strategic decisions (like faster response or backup suppliers) improve resilience.

Problem Statement

Supply chain disruptions caused by geopolitical events, logistics delays, supplier failures, or natural disasters can lead to significant operational and financial losses. Organizations need:

- Accurate estimation of **full recovery time**
- Insights into **key drivers of disruption impact**
- Tools to **simulate what-if scenarios** and improve resilience

This project addresses these needs using machine learning and analytics.

Objectives

- Clean and analyze supply chain disruption data
- Store and manage data using PostgreSQL
- Predict **full recovery days** using ML models
- Classify **response strategies**
- Build a **resilience index**
- Perform **what-if simulations**
- Deploy a reusable ML pipeline for future predictions

Tech Stack

- **Language:** Python
- **Libraries:** pandas, numpy, scikit-learn, sqlalchemy, psycopg2, joblib
- **Database:** PostgreSQL
- **ML Models:** Random Forest Regressor, Random Forest Classifier
- **Techniques:** Feature engineering, One-hot encoding, Pipelines, Scenario simulation

---

Data Pipeline Architecture

1. Load CSV data
2. Data cleaning & validation
3. Store raw data in PostgreSQL
4. Feature engineering
5. Train ML models
6. Evaluate model performance
7. Run simulations & predictions
8. Store predictions back to database

Algorithm (Step-by-Step)

Step 1: Data Ingestion

- Load supply chain disruption data from CSV
- Inspect schema, data types, and missing values

Step 2: Database Storage

- Create PostgreSQL connection using SQLAlchemy
- Store cleaned data into relational tables

Step 3: Feature Selection & Encoding

- Remove identifiers and leakage columns
- Apply one-hot encoding to categorical variables
- Scale numerical features where required

Step 4: Model Training (Regression)

- Split data into training and testing sets
- Train Random Forest Regressor to predict full recovery days
- Evaluate using R^2 and MAE

Step 5: Model Training (Classification)

- Predict disruption response type using Random Forest Classifier
- Evaluate using precision, recall, and F1-score

Step 6: What-if Scenario Simulation

- Modify severity, response time, or backup supplier presence
- Predict impact on average recovery time

Step 7: Resilience Index Calculation

- Normalize disruption severity, production impact, and recovery days
- Combine weighted metrics into a resilience score (0–100)

Step 8: ML Pipeline Creation

- Combine preprocessing and model into a single pipeline
- Enable consistent training and inference

Step 9: Model Persistence

- Save trained pipeline using joblib
- Reload model for future predictions

Pseudocode

```
 LOAD dataset
CLEAN missing values
STORE data in PostgreSQL

FOR regression task:
    SELECT features and target
    ENCODE categorical variables
    SPLIT train-test
    TRAIN RandomForestRegressor
    EVALUATE model

FOR simulation:
    MODIFY scenario parameters
    ALIGN features
    PREDICT recovery time

CALCULATE resilience index
SAVE model and predictions
```

Model Evaluation

- **Regression Metrics:**
    - R^2 Score

    - Mean Absolute Error (MAE)
- **Classification Metrics:**
    - Precision
    - Recall
    - F1-score

Feature importance analysis highlights key drivers such as:

- Disruption severity
- Response time
- Production impact
- Supplier tier

What-if Simulation Insights

The simulation module allows analysis of:

- Higher disruption severity → longer recovery
- Faster response time → reduced recovery days
- Backup suppliers → improved resilience

This helps decision-makers test strategies *before* implementation.

Limitations

- Dataset is historical and static
- External real-time risk signals not included
- Model performance depends on data quality

Future Enhancements

- Integrate real-time APIs (weather, geopolitics)
- Add deep learning models
- Deploy as a web dashboard (Streamlit / Flask)
- Automate retraining using scheduled pipelines

Repository Structure

```
Supply-Chain-Analytics/
│
├── data/
├── notebooks/
├── models/
├── scripts/
├── README.md
└── requirements.txt
```

Supply Chain Recovery Web Application (Streamlit)

Application Overview

This Streamlit application transforms the trained machine learning pipeline into an **interactive executive dashboard**. It allows users to input disruption parameters and instantly receive:

- Predicted recovery time
- Risk classification (Low / Medium / High)
- KPI summary
- Feature importance visualization
- Downloadable executive PDF report

This completes the project by demonstrating **end-to-end deployment capability**.

Application Architecture

1. Load trained ML pipeline (joblib)
2. Capture user inputs via Streamlit form
3. Apply preprocessing + model prediction
4. Classify risk level based on predicted recovery days
5. Display KPIs and feature importance
6. Generate downloadable PDF executive report

Streamlit App Algorithm (Step-by-Step)

Step 1: Load Dependencies

- Import Streamlit, pandas, numpy
- Load trained ML pipeline using joblib
- Import visualization and PDF libraries

Step 2: Cache Model

- Use `@st.cache_resource` to avoid reloading model repeatedly
- Improves performance

Step 3: Collect User Inputs

- Operational inputs (severity, impact, response time, revenue loss)
- Contextual inputs (industry, disruption type, region, supplier size)
- Convert revenue loss using log transformation to match training schema

Step 4: Make Prediction

- Create DataFrame matching training structure
- Call pipeline.predict()
- Extract predicted recovery days

Step 5: Risk Classification Logic

IF predicted_days ≤ 10 → Low Risk
ELSE IF ≤ 30 → Medium Risk
ELSE → High Risk

Step 6: KPI Dashboard

- Display metrics using `st.metric()`
- Show operational summary

Step 7: Feature Importance Extraction

- Extract RandomForest model from pipeline
- Retrieve feature names from preprocessor
- Display top 15 features using bar chart

Step 8: PDF Report Generation

- Create structured executive report using ReportLab
- Include:
    - Input summary

  - Prediction
  - Risk level
  - Feature importance table
- Enable download via `st.download_button()`

Streamlit App Pseudocode

```
 LOAD trained pipeline
DISPLAY input form

IF user clicks predict:
    PREPARE input dataframe
    TRANSFORM revenue_loss using log1p
    PREDICT recovery time
    CLASSIFY risk level
    DISPLAY KPIs
    PLOT feature importance
    GENERATE PDF report
    ENABLE download
```

Business Value

This dashboard enables:

- Real-time decision support
- Executive-level reporting
- Scenario-based operational planning
- Risk visualization for leadership teams

It demonstrates skills in:

- ML deployment
- Dashboard design
- Business communication
- Automated reporting