Predicting High Operational Risk Days Using Machine Learning Techniques

Pranav Senthilkumaran (ps1471@scarletmail.rutgers.edu)

Master's in Data Science, Rutgers University

1) Abstract

Operational risk can lead to production delays, quality issues, and increased downtime if not identified early. The goal of this project is to build a machine learning model that can predict high-risk operational days using historical daily operational data collected across multiple locations.

In this project, the data was first explored to understand patterns, distributions, and class imbalance. Data preprocessing and transformation steps such as **date conversion**, **categorical encoding**, and **feature scaling** were applied to make the data suitable for machine learning models. Several meaningful features were engineered to better represent operational stress, such as rejection rate, downtime per ticket, and tickets per unit..

Multiple classification models were trained, starting with simple baseline models like **Logistic Regression** and **Linear SVM**, followed by more advanced tree-based and non-linear models such as **Random Forest** and **CatBoost**. The models were evaluated using standard metrics including Precision, Recall, F1-score, ROC-AUC, and confusion matrices.

Finally, model interpretation techniques were used to identify key factors associated with high operational risk. While the models highlight strong associations, the results are predictive and **correlational** rather than causal. Overall, the project demonstrates how machine learning can be used as an early warning system to flag risky operational days and support proactive decision-making.

Keywords: Date Conversion, Categorical Encoding, Feature Scaling, Logistic Regression, Linear SVM, Random Forest, CatBoost, Correlation.

2) Introduction

Operational efficiency is critical for maintaining consistent production quality and minimizing downtime across multiple locations. Leveraging historical operational data enables organizations to move from reactive monitoring to proactive risk prediction.

2.1) Problem Statement

Operational systems generate large amounts of daily data related to production, downtime, and issue handling. However, identifying which days are likely to be operationally risky is not always straightforward.

High-risk days often occur due to a combination of factors such as:

- Increased downtime
- Higher defect rates
- Longer issue resolution times
- Operational stress during certain shifts or days

Manual monitoring or rule-based systems may miss these patterns. This project aims to use machine learning to automatically learn from historical data and predict high-risk days before problems escalate.

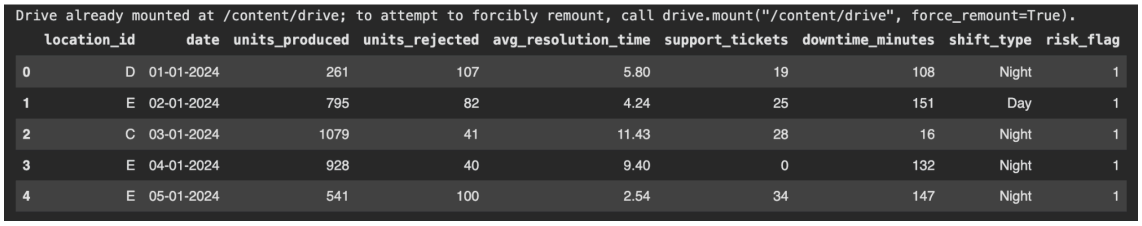
2.2) Objective

The **main objective** of this project is to predict whether a given operational day is **high risk** or **normal** using historical operational data and identify key drivers of risk.

This project focuses on a supervised machine learning approach to predict operational risk. The problem is formulated as a **binary classification task (Multiclass classification with 2 classes)**, where each data record represents the operational metrics for a **specific location on a given day**, and the goal is to classify the day as either high risk or normal.

3) Dataset Description

This project uses a structured historical operational dataset (`operations_ml.csv`) that captures **daily operational performance across multiple locations**. The dataset is designed to reflect real-world manufacturing and operational environments where production efficiency, quality issues, and downtime contribute to operational risk.



```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

	location_id	date	units_produced	units_rejected	avg_resolution_time	support_tickets	downtime_minutes	shift_type	risk_flag
0	D	01-01-2024	261	107	5.80	19	108	Night	1
1	E	02-01-2024	795	82	4.24	25	151	Day	1
2	C	03-01-2024	1079	41	11.43	28	16	Night	1
3	E	04-01-2024	928	40	9.40	0	132	Night	1
4	E	05-01-2024	541	100	2.54	34	147	Night	1

Each row in the dataset represents **one operational location on one specific day**, making the data suitable for supervised machine learning and time-aware analysis.

The data was loaded into the analysis environment using **Google Colab**, with files accessed directly from **Google Drive**. Mounting Google Drive enables seamless access to cloud-stored datasets without manually uploading files each time, ensuring reproducibility and efficient experimentation.

3.1) Dataset Overview and General Relationship

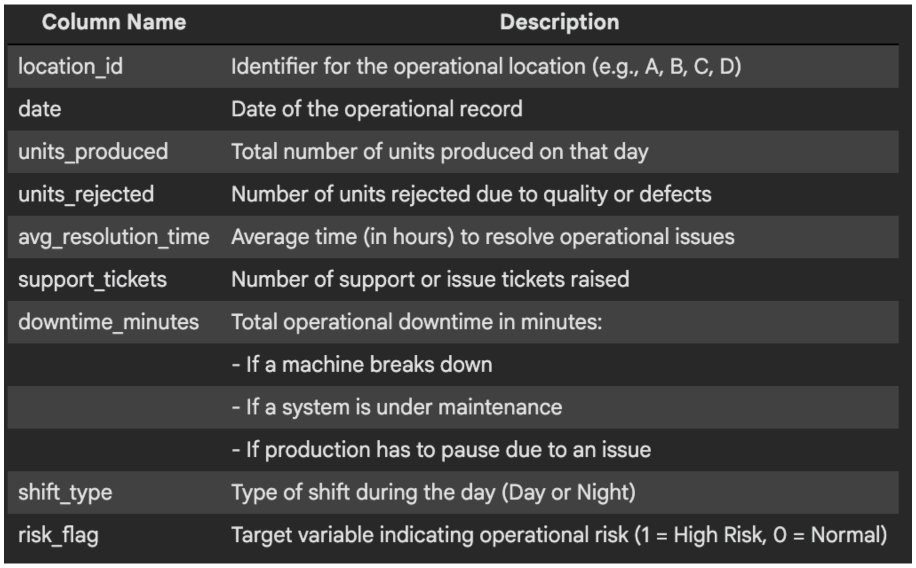
- The dataset contains **500 rows**, where each row represents one location on one day.
- There are **9 original columns**, including operational metrics, categorical variables, and the target variable.

In supervised classification, the relationship between predictor variables (**X**) and the target variable (**y**) is modeled as: $y = f(X) + \varepsilon$

- Xrepresents the set of input features (operational metrics),
- y represents the target variable (`risk_flag`),
- $f(X)$is the learned machine learning model,
- ε represents noise or unexplained variation.

The model learns patterns in historical data to estimate the probability that a given operational day will be **high risk** based on its input features.

3.2) Input (Predictor) Features and Target Variables Description



Column Name	Description
location_id	Identifier for the operational location (e.g., A, B, C, D)
date	Date of the operational record
units_produced	Total number of units produced on that day
units_rejected	Number of units rejected due to quality or defects
avg_resolution_time	Average time (in hours) to resolve operational issues
support_tickets	Number of support or issue tickets raised
downtime_minutes	Total operational downtime in minutes:
	- If a machine breaks down
	- If a system is under maintenance
	- If production has to pause due to an issue
shift_type	Type of shift during the day (Day or Night)
risk_flag	Target variable indicating operational risk (1 = High Risk, 0 = Normal)

The dataset (`operations_ml.csv`) contains daily operational metrics collected across multiple locations, including production volume, unit rejections, issue resolution time, support tickets, downtime, and shift type. These variables serve as **input features** that help describe operational conditions for each day. The **target variable**, `risk_flag`, indicates whether a given operational day is classified as **high risk (1)** or **normal (0)**.

4) Data Preprocessing and Transformation

Before building machine learning models, the dataset was carefully prepared to ensure data quality and validation.

4.1) Data Preprocessing

The dataset was first checked for basic data quality issues:

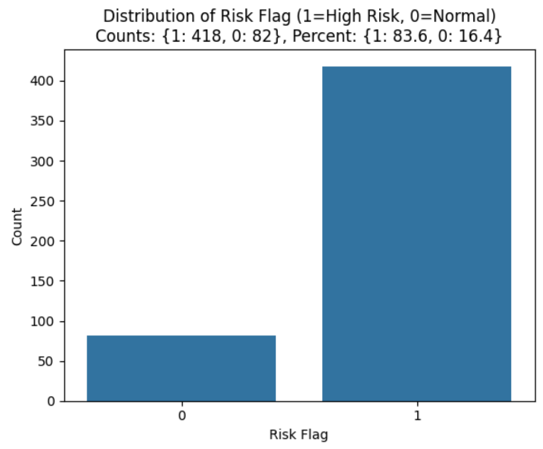
- **Missing values:**
  All columns were checked for missing values, and none were found.
- **Duplicate records:**
  Duplicate rows were verified and none were present.

Since the dataset was already clean and contained **no missing values**, **no imputation techniques** (such as mean, median, mode, or KNN imputation) were required.

4.2) Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed to understand the structure, quality, and patterns in the dataset before building machine learning models. This step helps identify class imbalance, feature distributions, and potential relationships with the target variable.

4.2.1) Target Variable Analysis

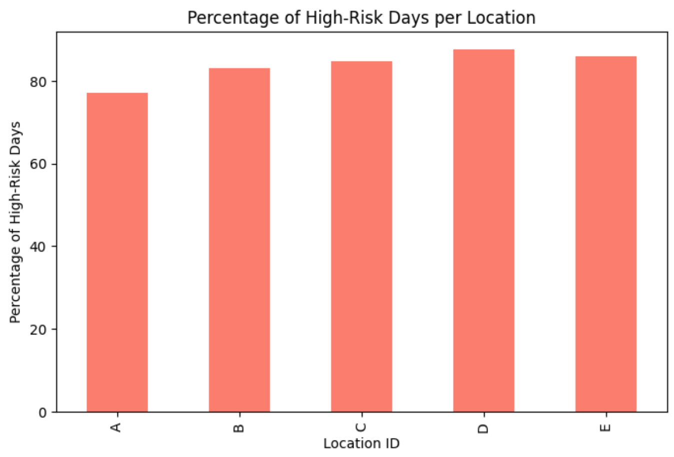


The target variable `risk_flag` shows **class imbalance**:

- **High Risk (1): 418 days (83.6%)**
- **Normal (0): 82 days (16.4%)**

This indicates that most days are labeled as high-risk.

4.2.2) Categorical Feature Analysis

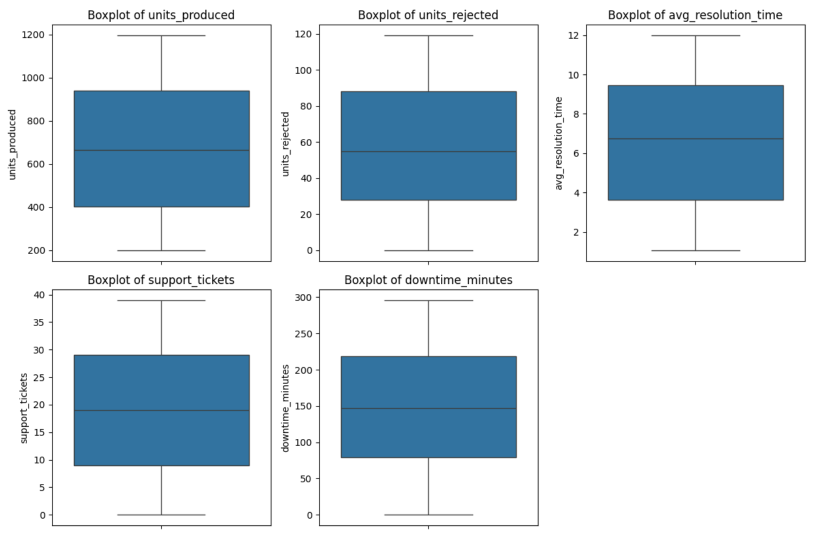


Operational risk was analyzed across different locations and all locations show a high proportion of high-risk days, indicating that operational risk is a system-wide issue rather than being isolated to a single site.

- **Location D** has the highest percentage of high-risk days (~87.5%), suggesting higher operational stress at this location.
- **Location A** has the lowest high-risk percentage (~77%), but still exhibits a majority of risky days.

4.2.3) Numerical Feature Analysis

The following numerical features were analyzed using boxplots and distribution plots: `units_produced, units_rejected, avg_resolution_time, support_tickets, downtime_minutes.`



units_produced shows a wide spread, indicating significant variation in daily production volumes across locations and days.

units_rejected also has high variability, suggesting that defect levels fluctuate substantially and may contribute to operational risk.

avg_resolution_time displays a broad range, implying inconsistent issue resolution efficiency on different days.

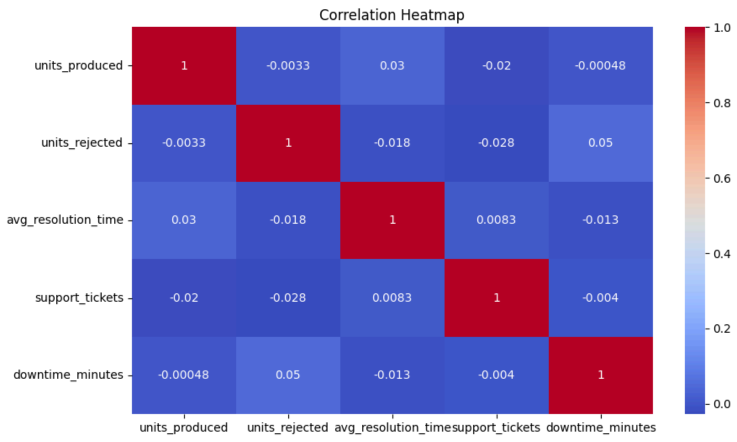
support_tickets varies from very low to high counts, reflecting fluctuating operational pressure and incident frequency.

downtime_minutes shows large dispersion, with some days experiencing extremely high downtime, which is a strong indicator of operational stress.

4.2.4) Correlation Analysis

Correlation analysis among numerical features showed **low linear correlation values**. This indicates that:

- No single numerical feature strongly explains operational risk by itself.
- Risk is influenced by combined effects of multiple variables.



The correlations between numerical features are **very low**, which means there are **no strong linear relationships** between them.

- This indicates **low multicollinearity**, which is good for:
  - Linear models like **Logistic Regression** and **Linear SVM**
  - Stable and easy-to-interpret model results

Since the features are not highly correlated, they can all be safely used in the model without causing instability and weak linear correlations explain why **tree-based models** (Random Forest, CatBoost) perform well as they can learn **non-linear patterns**. This also justifies **feature engineering**, such as creating ratios like **rejection rate** and **downtime per ticket**, to capture relationships that are not visible through simple correlation analysis.

4.3) Data Transformation

Data transformation is a significant step to make the dataset suitable for machine learning models. It helps to restructure raw data so that patterns can be learned more effectively and model performance is improved, and it also includes **feature engineering**, where new or more meaningful features are created from existing data.

4.3.1) Converting Date Column to Datetime

The `date` column was converted to a datetime format to enable time-based analysis. This was done because:

- It allows extraction of useful time-based features such as **day of week** and **weekend indicators**.
- It supports chronological analysis and trend identification.
- Although the raw date is not used directly as a predictor, converting it to datetime helps derive meaningful temporal patterns.

4.3.2) Feature Engineering

Feature engineering is crucial to help the model capture hidden patterns that indicate high operational risk.

The following features were created:

Rejection Rate: (`units_rejected / units_produced`)

- Captures the proportion of defective units produced, highlighting quality issues.
- Higher rejection rate -> higher likelihood of a high-risk day.

Downtime per Ticket: (`downtime_minutes / (support_tickets + 1)`)

- Measures the operational impact of each support ticket, avoiding division by zero.
- Larger downtime per ticket -> more severe operational problems -> higher risk.

Tickets per Unit: (`support_tickets / units_produced`)

- Represents the frequency of issues relative to production volume.
- More tickets per unit -> fragile operations -> higher risk.

Day of Week: date.dt.dayofweek

- Captures weekly patterns in operational risk.

Is Weekend: 1 if day_of_week is Saturday/Sunday else 0

- Flags potential risk variations on weekends vs weekdays.

4.3.3) Encoding Categorical Variables

Machine learning models generally require **numeric inputs**; therefore, categorical variables must be transformed into numerical representations.

4.3.3.1) Shift Type Encoding

The `shift_type` variable is converted from the categorical values `"Day"` and `"Night"` into binary values `0` and `1`, respectively. This transformation enables the model to directly incorporate shift information into its computations.

4.3.3.2) One-Hot Encoding for Location

The `location_id` variable is a nominal categorical feature. One-hot encoding is applied to create separate binary columns for each location (for example, `location_B`, `location_C`), which prevents the model from assuming any ordinal relationship among locations.

The parameter `drop_first=True` is used to remove one redundant column and avoid multicollinearity. When all location indicator variables are `False`, the observation is interpreted as belonging to the reference category (Location A).

These steps ensure that:
This encoding process ensures that **all features are numeric** and suitable for model training. It also prevents the model from inferring incorrect numeric relationships between categorical values **(false ordering)** and allows it to accurately learn **shift-specific** and **location-specific risk patterns**.

4.4) Defining Features and Target

Before training a machine learning model, the dataset must be divided into **predictor variables (features)** and the **target variable**.

Features (X):

The feature set consists of all variables used by the model to predict risk, including `units_produced`, `units_rejected`, `avg_resolution_time`, `support_tickets`, `downtime_minutes`, `rejection_rate`, `shift_type`, `downtime_per_ticket`, `tickets_per_unit`, `day_of_week`, `is_weekend`, and the **one-hot encoded location variables**. The columns `risk_flag` (the target variable) and `date` are excluded from the feature set, as they are not used directly for prediction.

Why the Date Is Not Used Directly as a Predictor:

The `date` column serves only as a timestamp and does not inherently explain the underlying causes of operational risk. Machine learning models may interpret **dates as arbitrary numerical values**, which can lead to the **learning of meaningless patterns**. Instead, more informative time-based features are derived from the date during feature engineering, such as **day of the week** and a **weekend indicator**. These derived variables capture recurring behavioral and operational patterns over time, making them more relevant for risk prediction than the raw calendar date.

Target (y):

The target variable is `risk_flag`, where a value of **1 indicates a high-risk day** and **0 indicates a normal day**.

Why This Is Needed:

Clearly separating features (X) from the target (y) is essential for supervised machine learning. This separation prevents data leakage by ensuring the model does not have access to the target during training. It also prepares the dataset for subsequent steps such as train–test splitting, model training, and performance evaluation.

4.5) Train–Validation–Test Split

The dataset was split into three parts:

- **Training set (60%)**: 300 samples with 15 features, used to learn patterns from the data.
- **Validation set (20%)**: 100 samples with 15 features, used for hyperparameter tuning and model selection.
- **Test set (20%)**: 100 samples with 15 features, used for final, unbiased performance evaluation.

The split is performed in two stages. First, the data is split into a **training set (60%)** and a **temporary set (40%)**. The temporary set is then equally divided into **validation (20%)** and **test (20%)** sets. Stratified sampling is applied at each step to preserve the class distribution of the target variable across all subsets, ensuring balanced model evaluation.

4.6) Feature Scaling

Numerical features were standardized using **StandardScaler**.

- Features were scaled to have **mean = 0** and **standard deviation = 1**.
- This step is especially important for models such as:
  - Logistic Regression
  - Support Vector Machines (SVM)

The scaler was fitted only on training data and applied to validation and test sets to prevent data leakage.

5) Methodology - Baseline and Advanced Models

The main objective is to predict high operational risk days using historical operational data. The dataset contains daily operational metrics per location, and the target variable is `risk_flag`, where 1 indicates a high-risk day and 0 indicates a normal day. The approach involves building baseline models and advanced models, followed by evaluation and interpretation of results.

5.1) Baseline 1: Logistic Regression

Logistic Regression is a widely used classification model that predicts the probability of a binary outcome. It works well when the relationship between input features and the target is approximately linear.

For each observation $Xi = [x1, x2,..., xp]$, Logistic Regression computes a weighted sum of features plus a bias term: $zi = w0 + w1x1 + w2x2 + ... + wpxp$

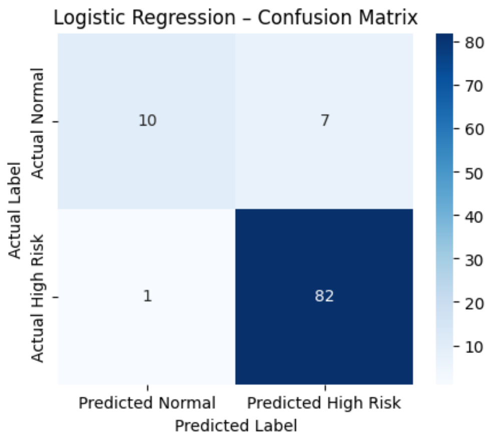
This sum is passed through the **sigmoid function** to produce a probability between 0 and 1:

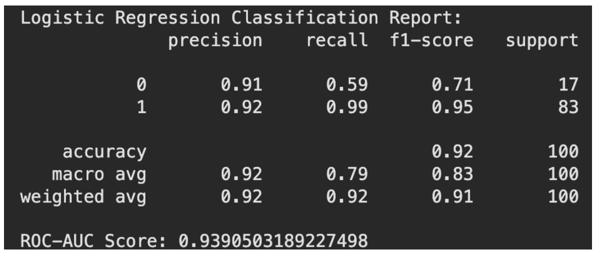
$$y_i = \sigma(z_i) = \frac{1}{1+e^{-z_i}}$$

If $y_i > 0.5$, the day is classified as **high-risk (1)**; otherwise, it is classified as **normal (0)**. The model learns the weights $w0, w1,..., wpw_0, w_1,..., w_pw0, w1,..., wp$from historical data to minimize prediction error.

Evaluation Metrics: Precision, Recall, F1-score, ROC-AUC, and the Confusion Matrix are used to assess model performance.

Results of Logistic Regression:





```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.59      0.71        17
           1       0.92      0.99      0.95        83

    accuracy                           0.92       100
   macro avg       0.92      0.79      0.83       100
weighted avg       0.92      0.92      0.91       100

ROC-AUC Score: 0.9390503189227498
```

5.2) Baseline 2: Linear Support Vector Machine

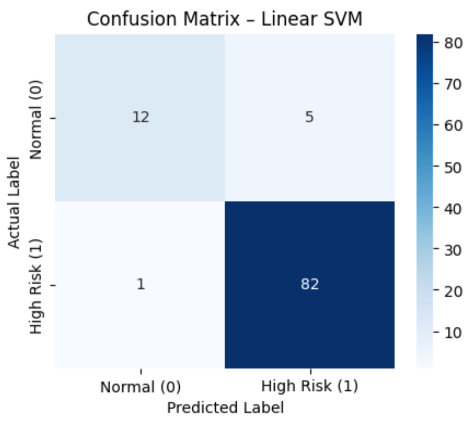
Linear SVM is a supervised classification algorithm that separates high-risk (1) and normal (0) days using a straight line (hyperplane) in the feature space. The main idea is to find the **optimal hyperplane** that maximizes the margin between the nearest points of each class, called **support vectors**.

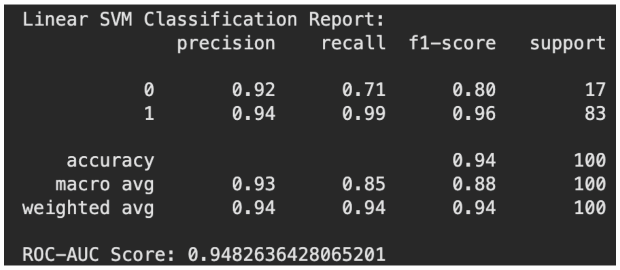
Mathematically, Linear SVM solves: $maximize\, margin\, subject\, to y_i(w \cdot X_i + b) \geq 1$

Here, w represents the feature weights and b is the bias. Predictions are based on the side of the hyperplane the observation falls on: $\hat{y}_i = \{1,\ if w \cdot X_i + b \geq 0;\ \ 0,\ otherwise$

Linear SVM performs well for linearly separable data and can handle high-dimensional datasets efficiently and Linear SVM yields the best performance across evaluation metrics when compared to **Polynomial**, **RBF**, and **Sigmoid** kernels.

Results of Linear SVM:





```
Linear SVM Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.71      0.80        17
           1       0.94      0.99      0.96        83

    accuracy                           0.94       100
   macro avg       0.93      0.85      0.88       100
weighted avg       0.94      0.94      0.94       100

ROC-AUC Score: 0.9482636428065201
```

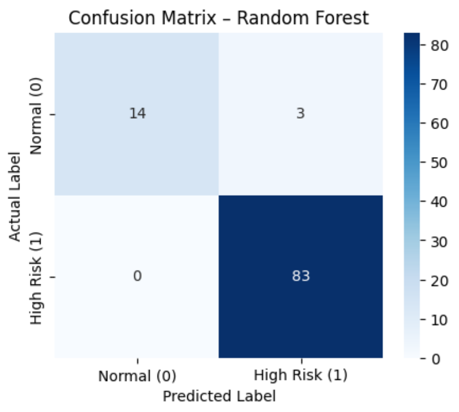
5.3) Advanced Model 1: Random Forest

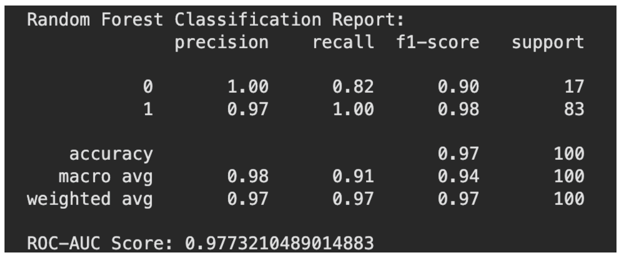
Random Forest is an **ensemble learning model** composed of many **decision trees**. Each tree is trained on a random subset of the training data (bootstrapping) and considers a random subset of features at each split. This randomness reduces overfitting and increases model robustness.

For a new observation, each tree predicts $\hat{y}_i^t \in \{0, 1\}$. The final prediction is based on **majority voting across all trees**: $\hat{y}_i = mode(\hat{y}_{i1}, \hat{y}_{i2}, ..., \hat{y}_{iT})$

Random Forest captures **non-linear relationships** and **feature interactions**, making it robust for tabular datasets with complex patterns.

Results of Random Forest:





```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        17
           1       0.97      1.00      0.98        83

    accuracy                           0.97       100
   macro avg       0.98      0.91      0.94       100
weighted avg       0.97      0.97      0.97       100

ROC-AUC Score: 0.9773210489014883
```

5.4) Advanced Model 2: CatBoost

CatBoost is a **gradient boosting algorithm** specifically designed for tabular data. It builds decision trees **sequentially**, where each new tree focuses on **correcting errors** made by previous trees. Unlike Random Forest, which uses voting, CatBoost combines tree outputs using a **weighted sum**:

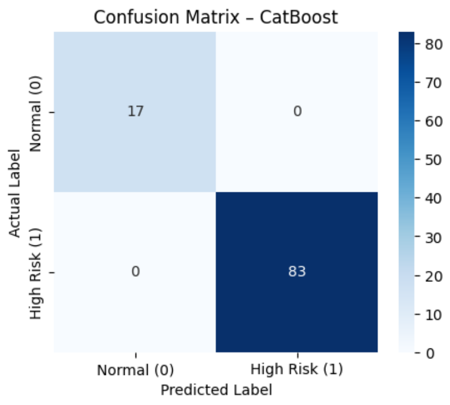
$$\hat{y}_i = \sigma(\sum_{m=1}^{M} \alpha_m h_m(X_i))$$

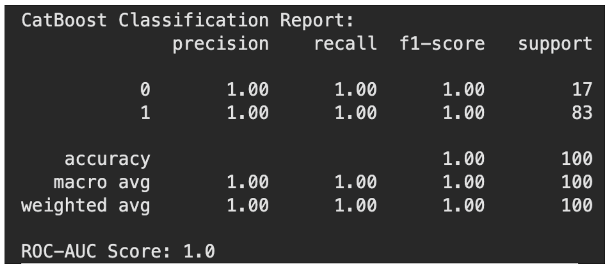
Here:

- $hm(Xi)$is the output of the m-th tree,
- αm is its weight,
- $\sigma$is the sigmoid function, which converts the result into a probability.

CatBoost handles **categorical features natively** without extensive preprocessing, and includes **built-in regularization techniques** to prevent overfitting. It captures **complex non-linear patterns** and **feature interactions**, which makes it highly effective for predicting high operational risk days.

Results of CatBoost:





```
CatBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        83

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100

ROC-AUC Score: 1.0
```

6) Bottlenecks – Misclassifications

Even the best models can sometimes make mistakes. Misclassifications occur when a model predicts a day as high-risk (1) or normal (0) incorrectly. Understanding these mistakes helps improve models and interpret results better.

Logistic Regression:

- **Number of Failed Predictions:** 8
- **Example of Failure:**
  - The model predicted a day as high-risk, but the actual risk was normal.
  - Features: moderate downtime and support tickets led the linear model to push the probability above 0.5.
- **Reason:** Logistic Regression assumes a **linear relationship** between features and risk. In cases where this assumption doesn't hold, it can produce **false positives or false negatives**.

Linear SVM

- **Number of Failed Predictions:** 6
- **Example of Failure:**

    - A day with high rejected units but low downtime was predicted as high-risk, while it was actually normal.

- **Reason:** Linear SVM assumes classes are **linearly separable**. When data points lie **close to the decision boundary** or have conflicting feature indicators, the model can misclassify them.

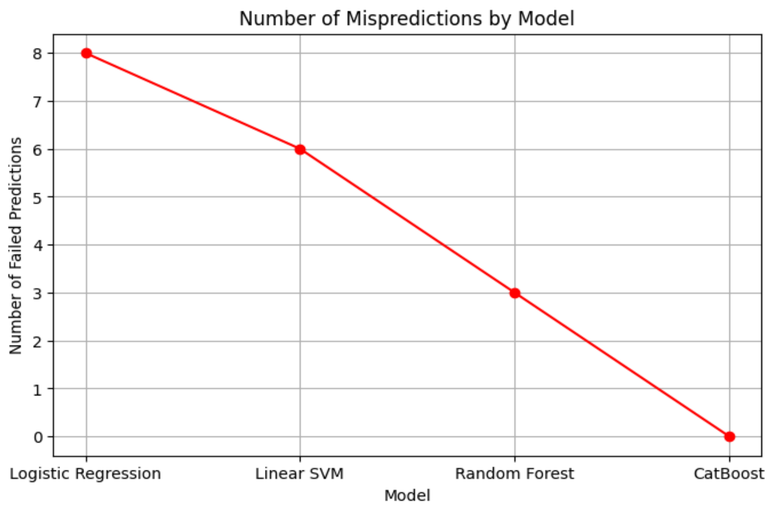
Random Forest

- **Number of Failed Predictions:** 3
- **Example of Failure:**
    - A day with moderately high downtime but low support tickets was predicted as high-risk incorrectly.
- **Reason:** Random Forest handles **non-linear patterns**, but **rare combinations of features** can confuse the trees, causing a few errors but performs better than linear baseline models.

CatBoost (Best Model)

- **Number of Failed Predictions:** 0
- **Reason:** CatBoost uses **boosting**, which sequentially corrects errors from previous trees. It captures **complex feature interactions** effectively, avoiding misclassifications in this dataset.

Line Chart of Misclassifications:



7) Key Drivers of Operational Risk – Insights from All Models

This section explains which factors most strongly influence whether a day is classified as **high operational risk**. The insights are derived from all four models: Logistic Regression, Linear SVM, Random Forest, and CatBoost.

7.1) Strong Risk Drivers

The following features consistently increase the likelihood of a **high-risk day** across all models:

- **Downtime Minutes**
  Longer machine or system downtime strongly increases operational risk. When production is interrupted for extended periods, operations become unstable.
- **Rejection Rate**
  A high proportion of rejected units indicates quality problems. This is one of the strongest indicators of operational risk.
- **Average Resolution Time**
  Slower resolution of operational issues reflects inefficiencies and delays, leading to higher risk.
- **Units Rejected**
  A high number of rejected units directly contributes to risk, even if total production is high.

These factors represent **core operational failures** related to reliability, quality, and response time.

7.2) Moderate and Contextual Risk Drivers

These features influence risk in certain situations but are not dominant on their own:

- **Day of the Week**
  Some weekdays show higher risk patterns, likely due to workload or staffing differences.
- **Shift Type (Night Shift)**
  Night shifts are slightly more prone to risk, potentially due to lower staffing or slower response times.
- **Location (C, D, E)**
  Certain locations show moderate risk differences, indicating location-specific operational challenges when compared to other locations like Location A, B.

7.3) Protective / Risk-Reducing Factors

These features are associated with **lower operational risk**:

- **Units Produced**
  Higher production volume generally indicates smoother and more stable operations.
- **Tickets per Unit**
  Fewer issues relative to production reflect better process control and system reliability.
- **Weekend Indicator**
  Slightly lower risk is observed on weekends, possibly due to reduced operational load.
- **Location B**
  Some locations consistently show lower inherent risk like Location B.

8) Conclusion

8.1) Model Interpretation & Key Driver Insights

Logistic Regression & Linear SVM:

- Identify linear relationships between features and operational risk. High coefficients for downtime_minutes, rejection_rate, and avg_resolution_time highlight the most critical risk drivers. Misclassifications occur for borderline feature combinations due to **linear assumptions**.

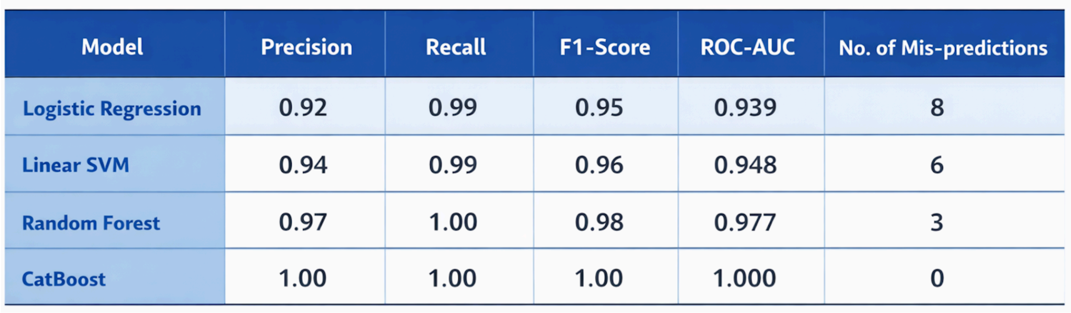
Random Forest:

- Captures **non-linear relationships** and feature interactions, reducing mispredictions. Feature importance confirms downtime, defect rate, and resolution time as dominant drivers. Handles edge cases better than linear models.

CatBoost:

- Gradient boosting captures complex feature interactions perfectly. No mispredictions in the test set indicate **strong generalization**. Reinforces that downtime, rejection rate, and average resolution time are the top risk drivers.

8.2) Model Performance Comparison



Model	Precision	Recall	F1-Score	ROC-AUC	No. of Mis-predictions
Logistic Regression	0.92	0.99	0.95	0.939	8
Linear SVM	0.94	0.99	0.96	0.948	6
Random Forest	0.97	1.00	0.98	0.977	3
CatBoost	1.00	1.00	1.00	1.000	0

The table below summarizes the performance of all four models using Precision, Recall, F1-score, ROC-AUC, and the number of mispredictions on the test set.

- **Logistic Regression** provided a strong baseline with high recall, successfully identifying most high-risk days, but produced more mispredictions due to its linear assumptions.
- **Linear SVM** improved upon Logistic Regression by achieving better overall balance between precision and recall, resulting in fewer misclassifications.
- **Random Forest** significantly outperformed the baseline models by capturing non-linear relationships and feature interactions, achieving an F1-score of 0.98 with only three mispredictions.

- **CatBoost** delivered the best overall performance, achieving perfect scores across all evaluation metrics and zero mispredictions on the test set.

Overall, while **Logistic Regression and Linear SVM** offered interpretable baseline results, **Random Forest** demonstrated stronger performance gains than baselines, and **CatBoost** emerged as the top-performing model, making it the most suitable choice for predicting high operational risk days in this dataset.

8.2) Key Insights

Hyperparameter tuning does not always help, especially when the dataset is small and the model is simple. If the model already performs well with default settings, changing parameters and penalizing coefficients will not lead to noticeable improvement. Focusing on **better features (Feature Engineering)** is often more useful.

Machine learning models like Logistic Regression, SVM, Random Forest, CatBoost learn **patterns from data**, not causation. They predict outcomes based on historical associations and they **cannot prove that changing a feature will change risk**, only that the feature is associated with risk.