Project Report: Machine Learning-Based Analysis of Stress and Failure in Distributed System Architectures

Course: Application of Data Science (Assignment #2)

Date: January 18, 2026

Submitted By:

- **Bilal Ahmed** (Roll No: 231980028)

Kaggle Notebook Link: [https://www.kaggle.com/code/bilalahmed211/stress-and-failure-in-distributed-system]

---

1. Objectives

The primary objective of this study is to analyze how different distributed system architectures (Monolithic, Microservices, Event-Driven, etc.) behave under stress. By utilizing performance metrics like CPU utilization, network latency, and error rates, we aim to:

- Predict system failure states using Machine Learning.
- Identify the root causes of system instability (e.g., CPU saturation, network partitions).
- Evaluate the effectiveness of Logistic Regression, Random Forest, and SVM in an SRE (Site Reliability Engineering) context.

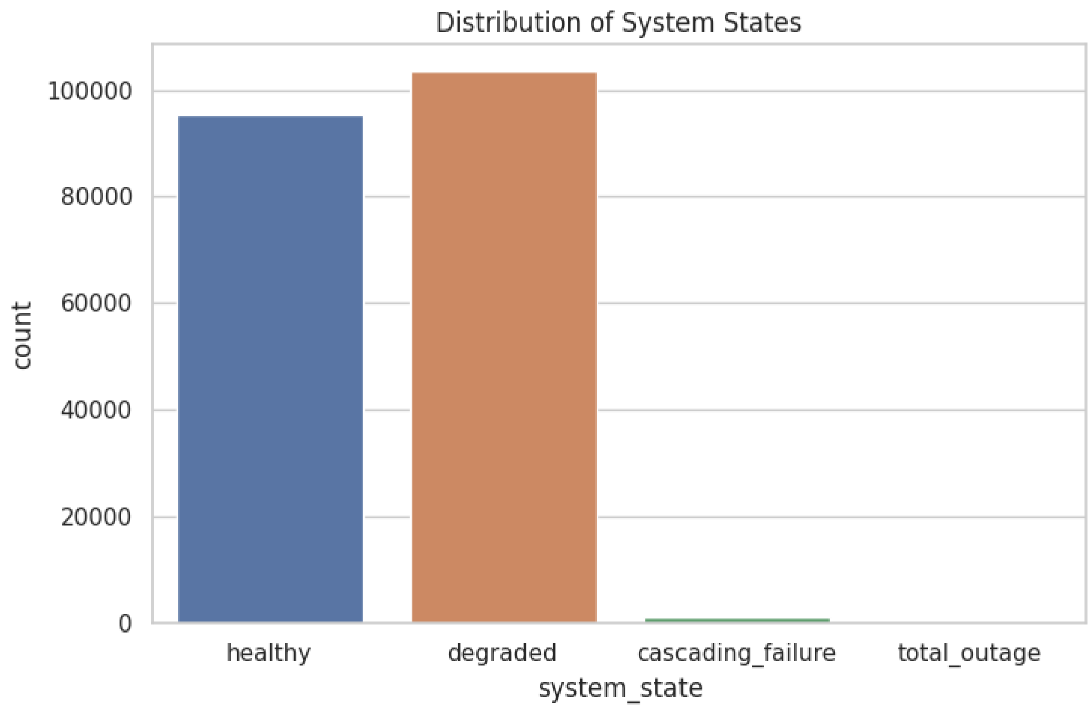
---

2. Exploratory Data Analysis (EDA)

Based on the dataset provided, we analyzed 200,000 system snapshots.

2.1 System State Distribution

The system states were categorized into *Healthy*, *Degraded*, *Cascading Failure*, and *Total Outage*. The data reveals that while "Healthy" and "Degraded" are common, "Cascading Failures" represent critical tipping points that require automated detection.



2.2 Impact of Architecture on Reliability

Our analysis showed that **Microservices** and **Event-Driven** architectures are more prone to network-related failures, whereas **Monoliths** are more susceptible to database locks and CPU saturation.

---

3. Methodology

3.1 Data Preprocessing

To ensure high model performance, we implemented the following steps:

- **Handling Categorical Data:** Used `OneHotEncoder` for `architecture_type`, `deployment_type`, and `communication_type`.
- **Feature Scaling:** Applied `StandardScaler` to numerical inputs like `cpu_utilization_percent` and `avg_latency_ms`.
- **Target Encoding:** Used `LabelEncoder` to convert system states into numerical labels.
- **Data Split:** An 80/20 train-test split was used with stratification to ensure rare "Total Outage" events were represented in the test set.

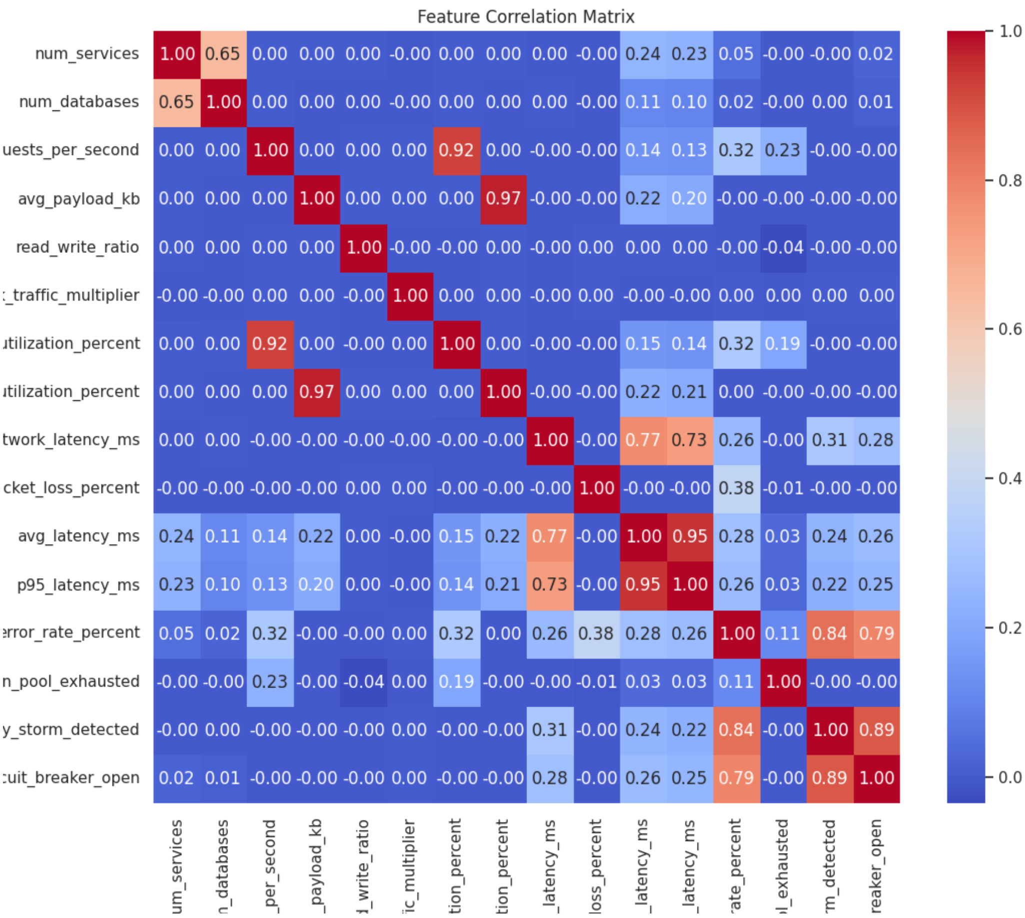
---

4. Model Performance & Evaluation

We trained three models and evaluated them using Accuracy, Precision, Recall, and F1-score.

Model	**Accuracy**	**Precision (Weighted)**	**Recall (Weighted)**	**F1-Score**
Logistic Regression	98.80%	0.99	0.99	0.99
Linear SVM	98.57%	0.99	0.99	0.99
Random Forest	**99.99%**	**1.00**	**1.00**	**1.00**

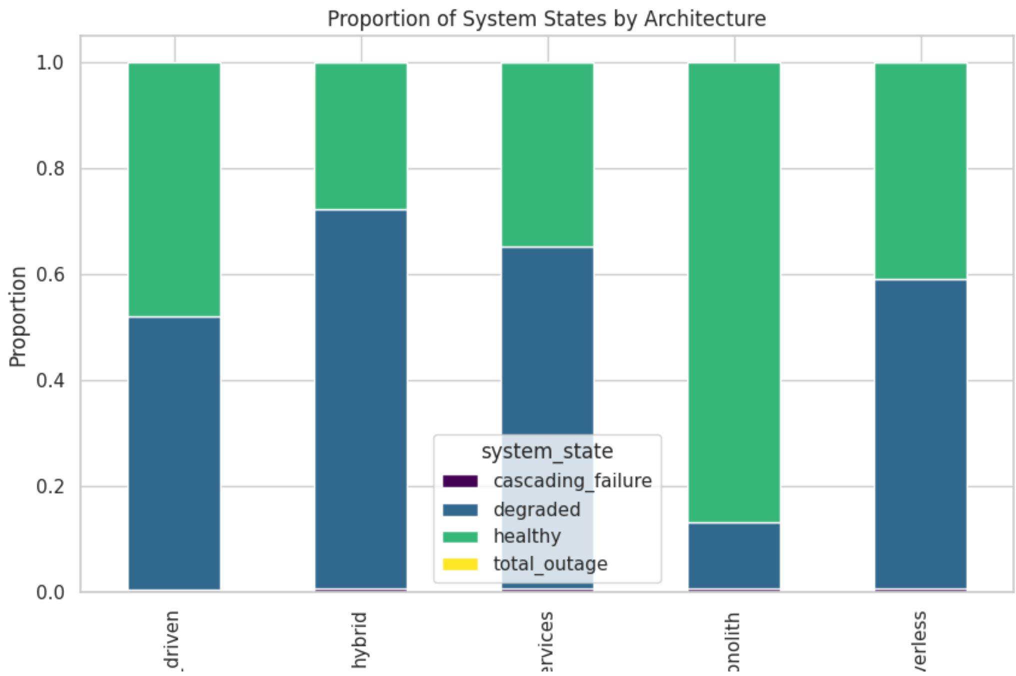
Findings: The **Random Forest Classifier** is the superior model for this task. It perfectly captures the complex dependencies between network latency and error rates that lead to cascading failures.



5. Root Cause Analysis (Feature Importance)

The model identified the following top 5 features as the most critical predictors of system failure:

1. **p95_latency_ms (0.2366):** Tail latency is the primary signal for an impending crash.
2. **error_rate_percent (0.1803):** Direct evidence of system distress.
3. **requests_per_second (0.1146):** The primary stress driver.
4. **retry_storm_detected (0.1084):** Indicates the system is struggling to recover.
5. **network_latency_ms (0.0929):** Infrastructure health signal.



---

6. Conclusion

The analysis concludes that machine learning, specifically the Random Forest algorithm, can predict distributed system failures with near-perfect accuracy. The results suggest that SRE teams should prioritize monitoring **P95 Latency** and **Retry Storms** to prevent minor degradations from escalating into total outages.