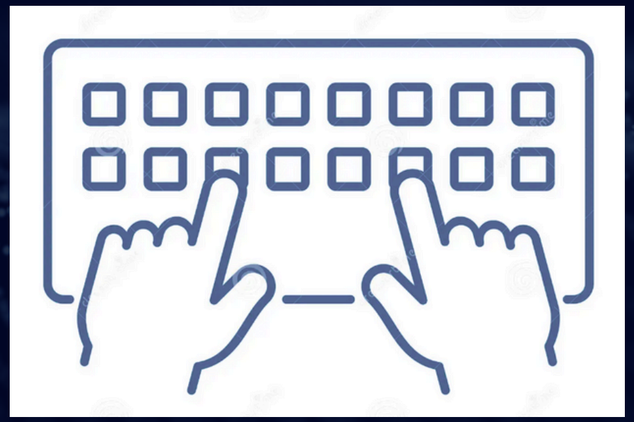
# Explore Transformer-Based Foundation Models for Time Series Predictive Maintenance

Elene Esakia
BSc Computer Science
Constructor University
Supervisor: Hendro Wicaksono, Rahma Dawud
August 2025

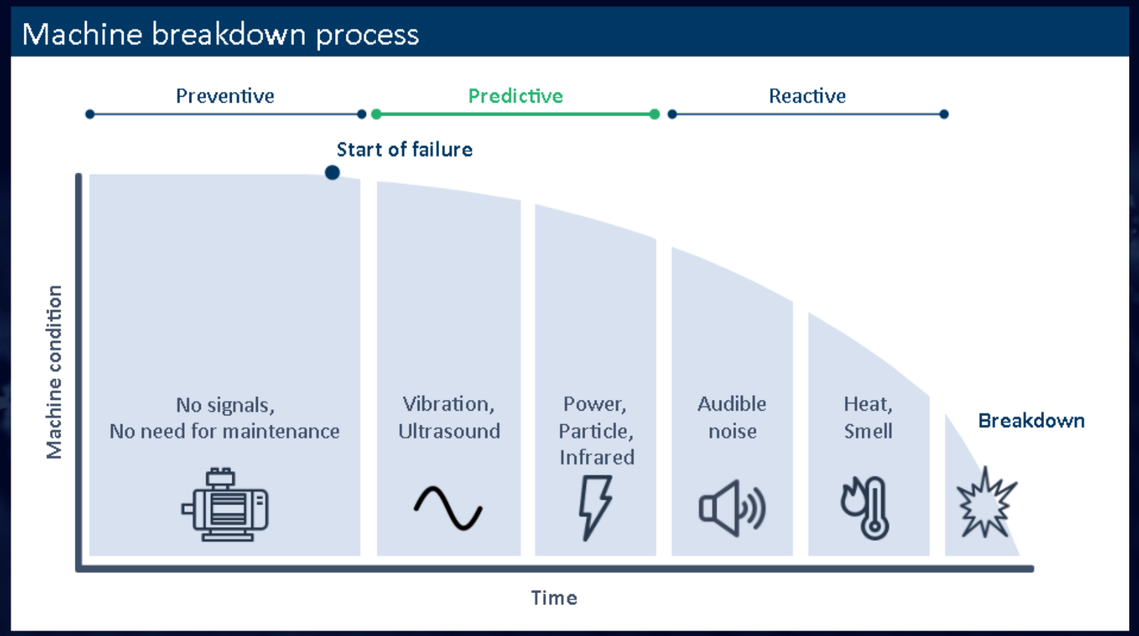
# Content





# Introduction

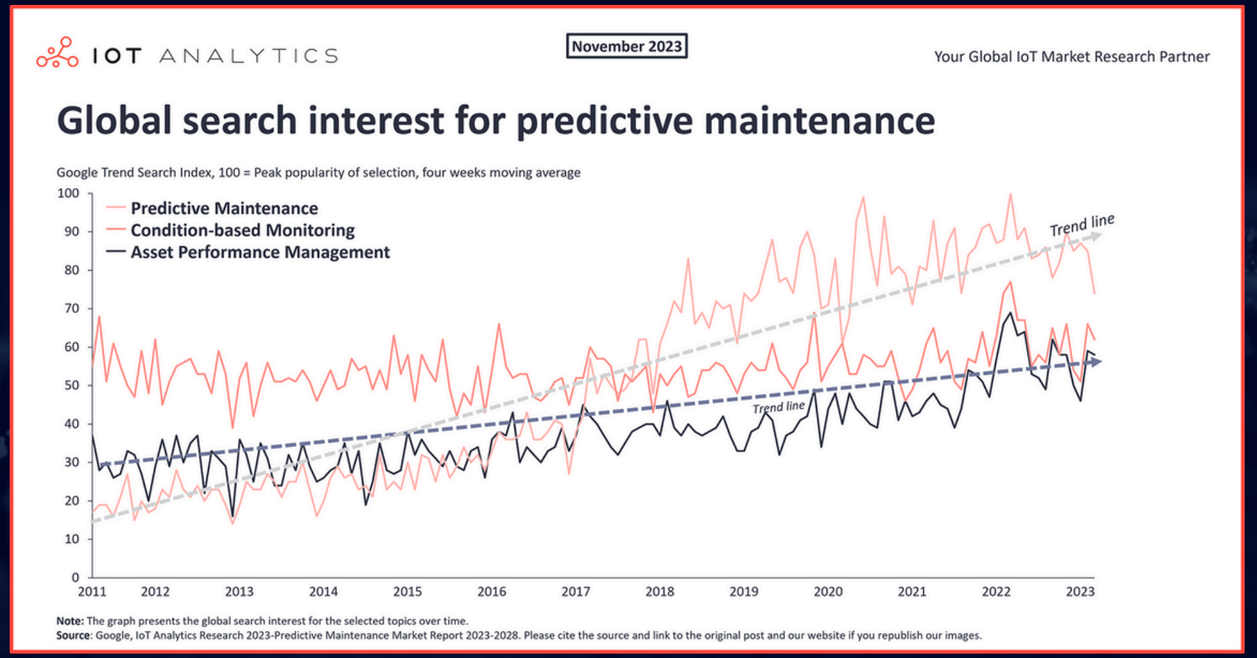
1. Project goals:
    - Investigate transformer-based models for time series predictive maintenance
    - Compare their performance against traditional approaches
2. Importance in Industry:
    - Prevent unexpected machine breakdowns
    - Save time and money on repairs
    - Make machines more reliable and safe
3. Why Transformers:
    - Learn patterns from time series data
    - Handle multiple sensor readings at once
4. Models Explored:
    - PatchTST
    - MOMENT




Elene Esakia | 2025

# Relevant Work

- Predictive maintenance market valued at $5.5B in 2022 (+11% from 2021)
- Unplanned downtime costs ~$125K per hour across industries
- One accurate prediction can save >$100K
- Challenges: Many solutions have <50% accuracy, causing false alarms
- Research gaps:
  - Limited research on transformers for real-world PdM classification
  - Challenge: imbalanced datasets, rare failures, high computational cost
  - Unclear if these models can outperform traditional approaches in practical PdM scenarios



# Methodology

## Step 1: Data Exploration (EDA)

1. Dataset:
   - Microsoft Azure Predictive Maintenance dataset (available on Kaggle)
   - Captures 1 year of hourly operational data for 100 industrial machines
   - 5 CSV files: PdM_telemetry.csv, PdM_failures.csv, PdM_errors.csv, PdM_maint.csv, PdM_machines.csv
2. Key Statistics:
   - 876,100 records
   - Total failures: 761 (98 machines affected)
   - Sensors: 4 features (volt, rotate, pressure, vibration)
   - Missing values: none; duplicates: none
   - Prediction windows: 24h / 48h / 72h (Use Case)


Fig. 1: Sensor Correlation Matrix (telemetry)

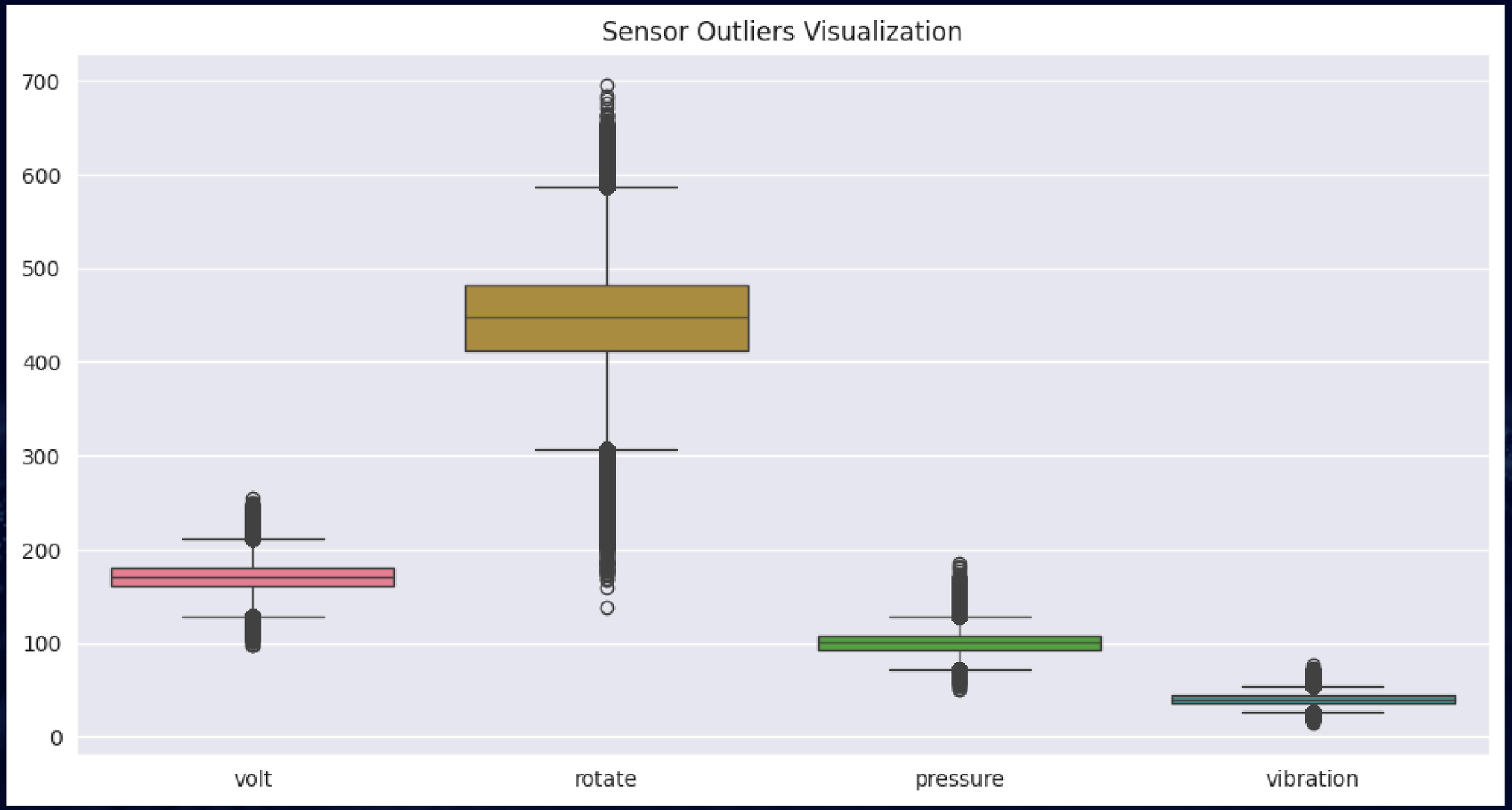



Fig. 2: Sensor Outliers Visualization (telemetry)

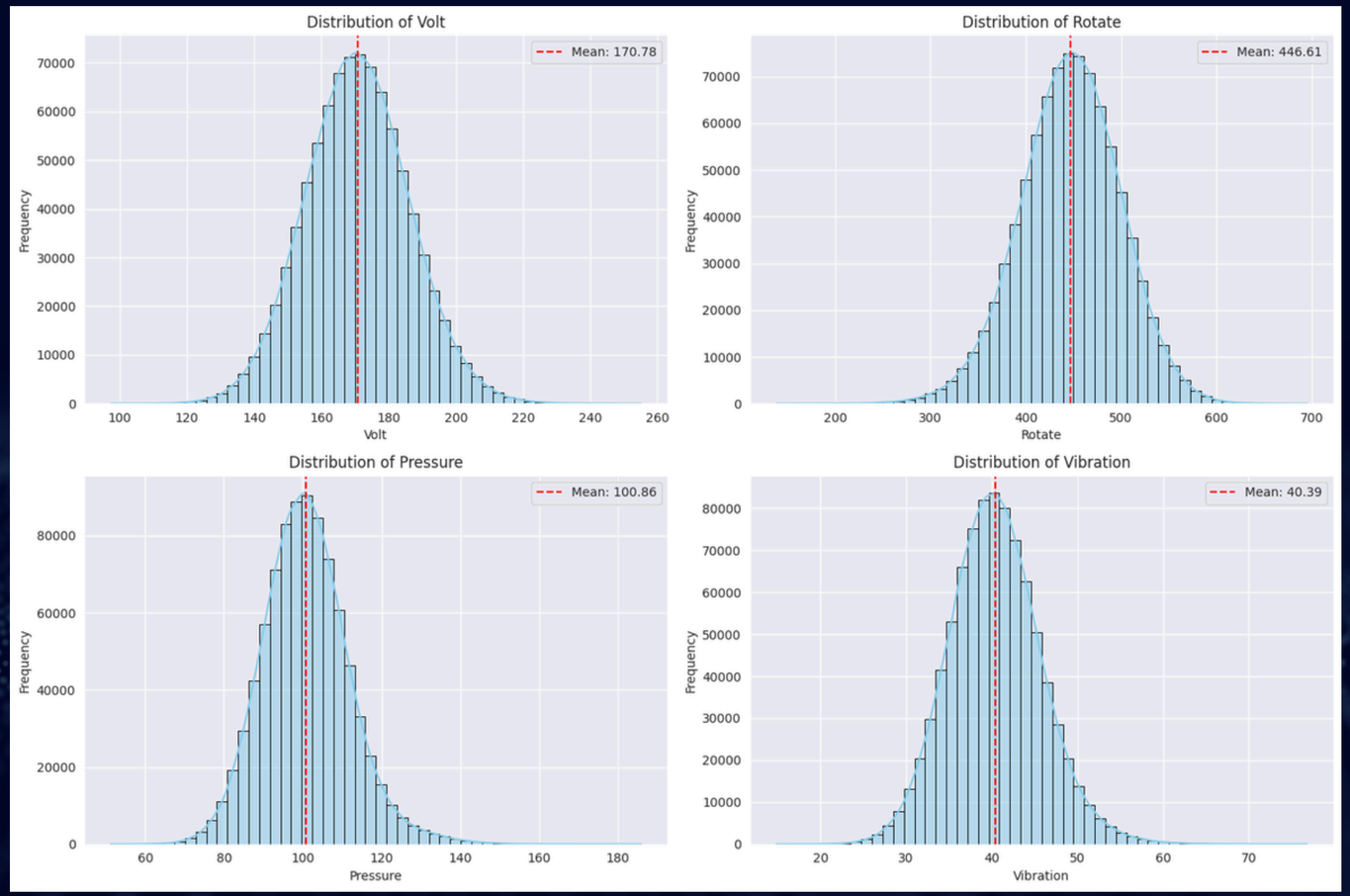



Fig. 3: Sensor Value Distributions with Mean Lines (telemetry)

```
=== Skewness & Kurtosis ===
volt: Skewness = 0.09, Kurtosis = 3.12
rotate: Skewness = -0.14, Kurtosis = 3.20
pressure: Skewness = 0.40, Kurtosis = 3.88
vibration: Skewness = 0.25, Kurtosis = 3.47
```

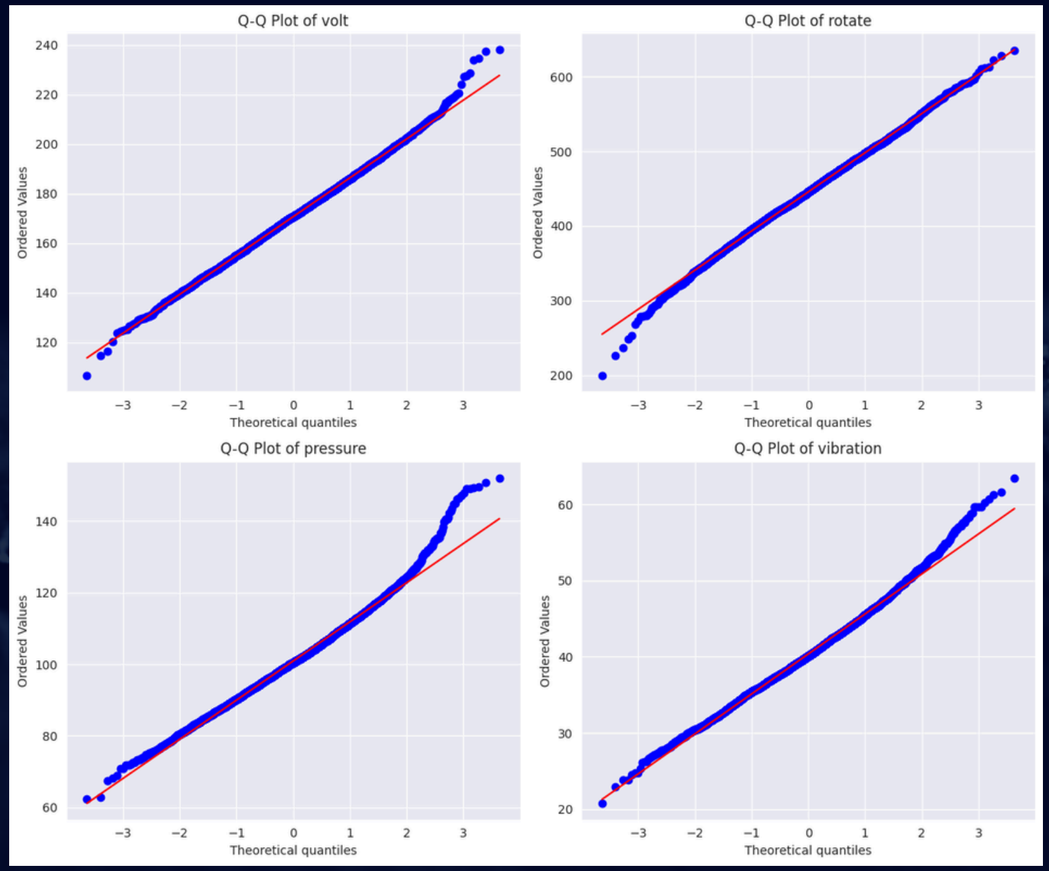



Fig. 4: Skewness, Kurtosis & Q-Q Plots

Elene Esakia | 2025

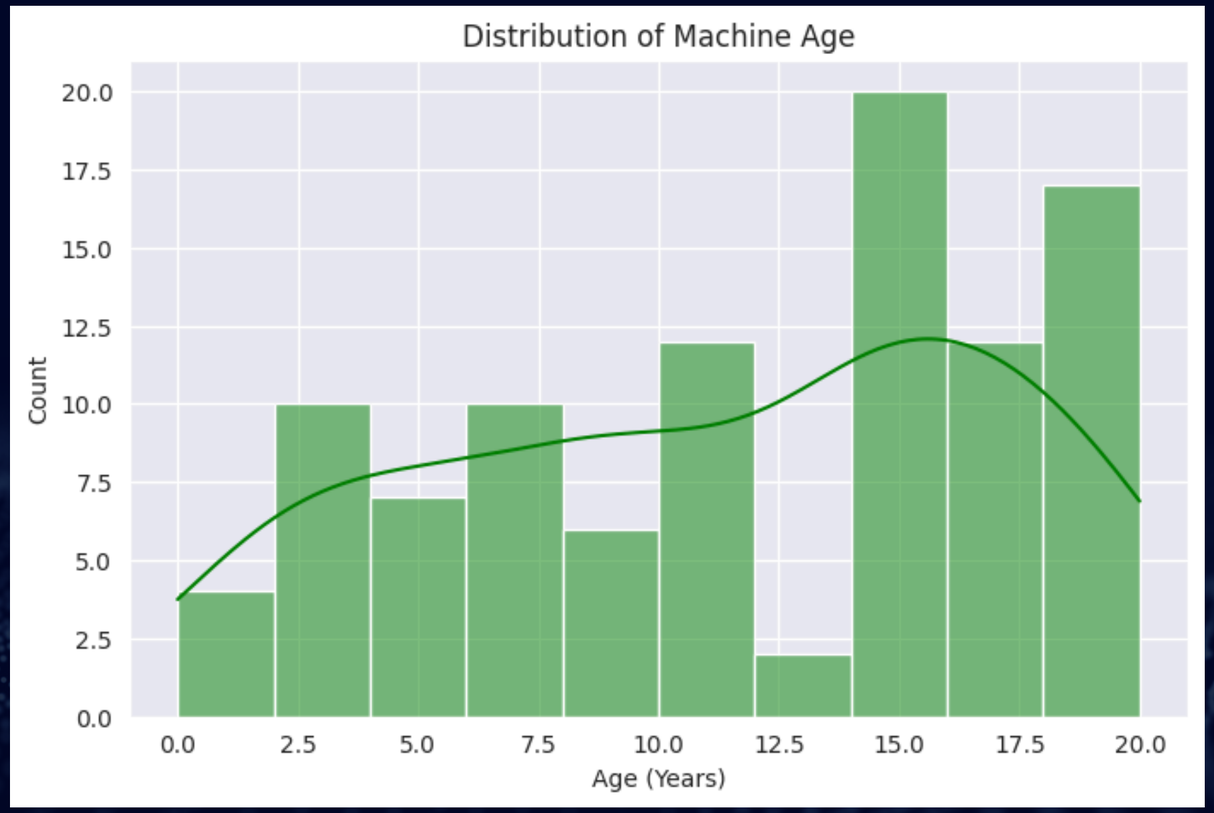



Fig. 5: Machine Age Distribution (machines)

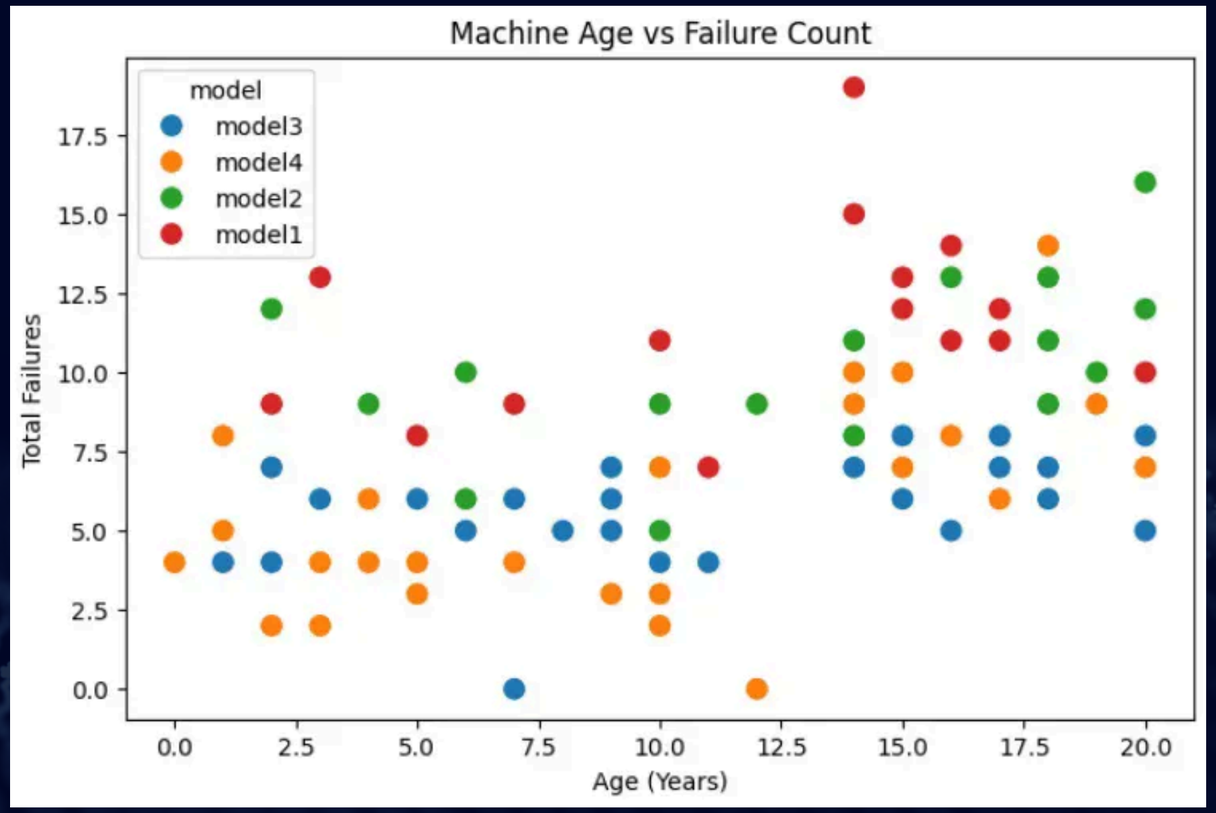



Fig. 6: Machine Age vs Failure Count

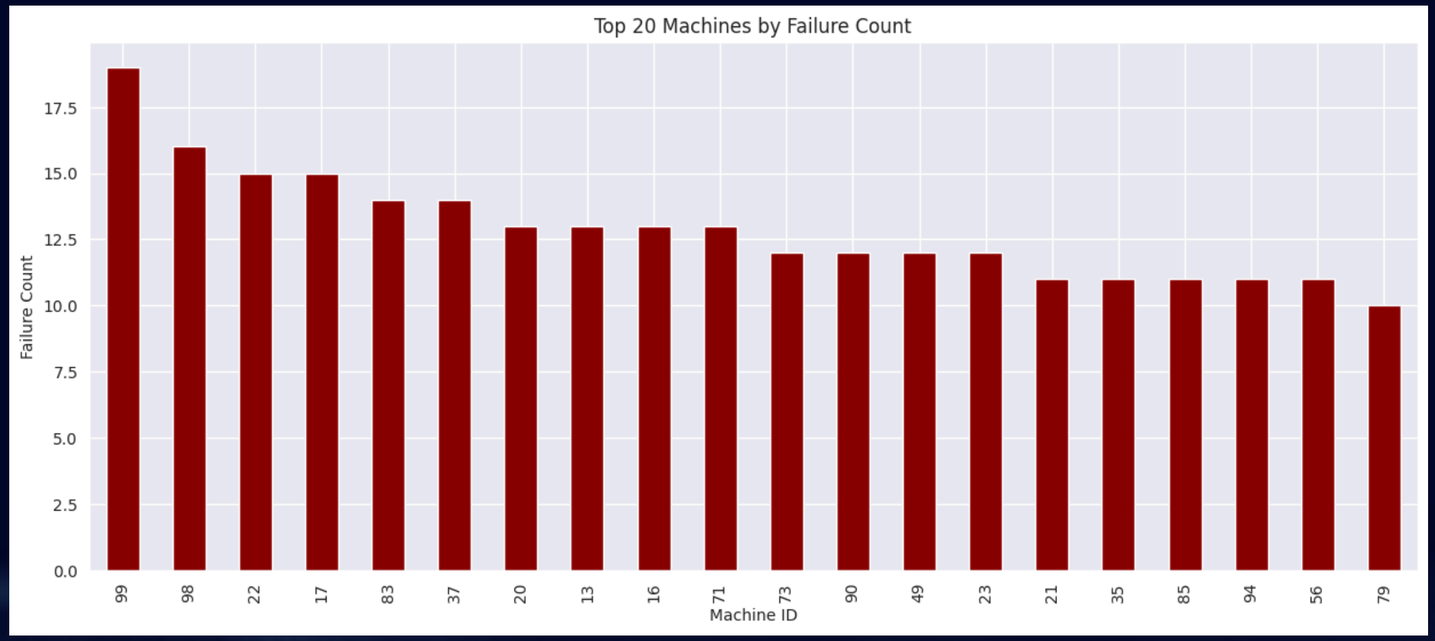



Fig 7: Top 20 Machines by Failrues

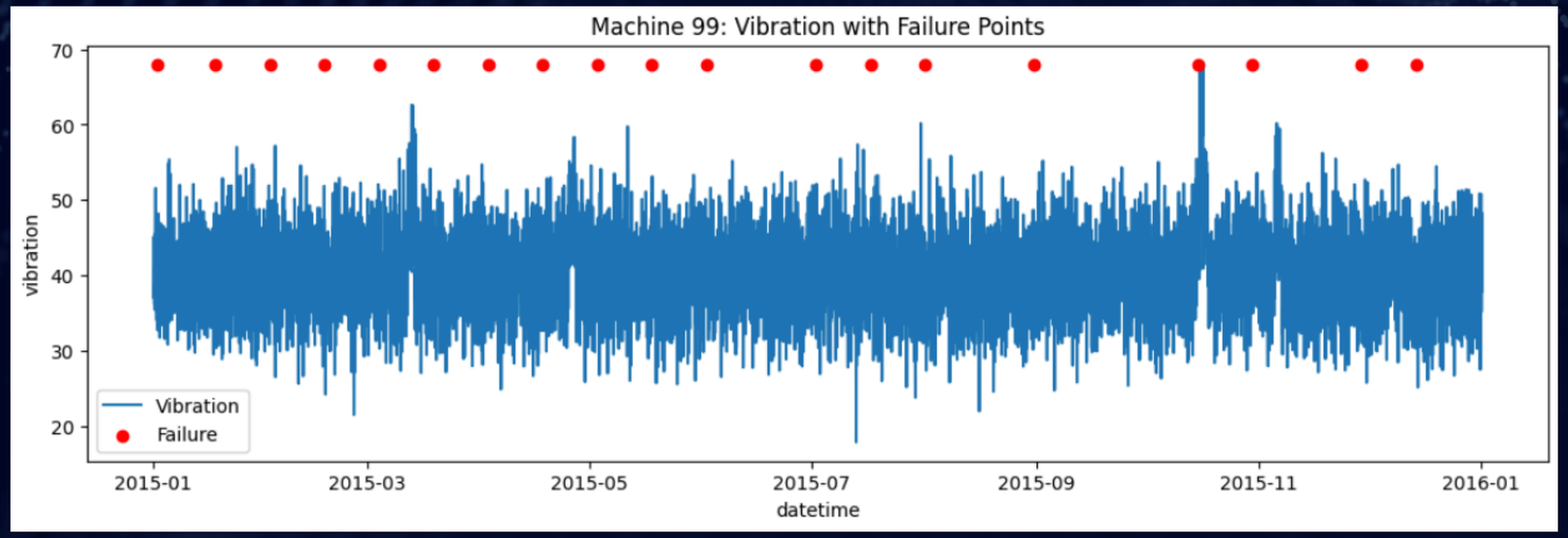
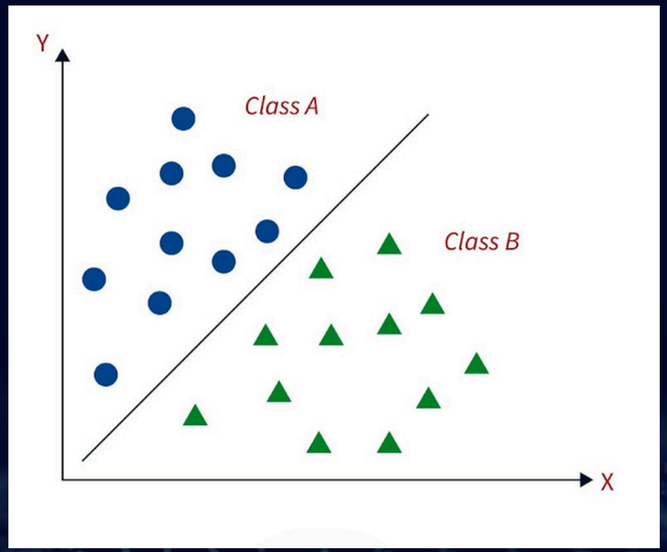



Fig 8: Vibration with Failure Points (machine 99)

## Step 2: Data Preparation

1. Datasets Merge:
   - Combine telemetry and machine metadata using machine ID
   - Convert machine models into separate 0/1 columns (one-hot encoding)
   - Convert error types into 0/1 columns
   - Aggregate error data: Sum errors per machine and timestamp
   - Combine all data into a single dataset
2. Target Variable Creation:
   - Failure alignment: Matched telemetry rows with the next known failure for each machine
   - Aggregated failures by machine and time
   - Created binary labels: will_fail_24h, will_fail_48h, will_fail_72h
   - Final dataset: Combined sensor readings, machine info, error data, and failure labels




```
will_fail_24h    17183
will_fail_48h    33959
will_fail_72h    50735
dtype: int64
will_fail_24h  will_fail_48h  will_fail_72h
0              0              0               825365
1              1              1                17183
0              0              1                16776
               1              1                16776
Name: count, dtype: int64
Percentage of positive cases: 1.96%
Percentage of positive cases: 3.88%
Percentage of positive cases: 5.79%
```

Fig 10: Distribution of Positive vs Negative Cases for Failure Prediction

| | datetime | machineID | volt | rotate | pressure | vibration | age | model1 | model2 | model3 | model4 | error1 | error2 | error3 | error4 | error5 | will_fail_24h | will_fail_48h | will_fail_72h |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2015-01-01 06:00:00 | 1 | 176.217853 | 418.504078 | 113.077935 | 45.087686 | 18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2015-01-01 07:00:00 | 1 | 162.879223 | 402.747490 | 95.460525 | 43.413973 | 18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2015-01-01 08:00:00 | 1 | 170.989902 | 527.349825 | 75.237905 | 34.178847 | 18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2015-01-01 09:00:00 | 1 | 162.462833 | 346.149335 | 109.248561 | 41.122144 | 18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 2015-01-01 10:00:00 | 1 | 157.610021 | 435.376873 | 111.886648 | 25.990511 | 18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
Missing values per column:
datetime         0
machineID        0
volt             0
rotate           0
pressure         0
vibration        0
age              0
model1           0
model2           0
model3           0
model4           0
error1           0
error2           0
error3           0
error4           0
error5           0
will_fail_24h    0
will_fail_48h    0
will_fail_72h    0
dtype: int64

Number of duplicate rows: 0
```

Checked dataset for missing values → none found

Checked for duplicate rows → none found

Ensured dataset is clean and ready for modeling

## Step 3: Feature Engineering

- Focused on sensor data: voltage, rotation, pressure, vibration
- Computed rolling window features (6h, 12h, 24h)
- Mean, Standard Deviation, Min, Max
- Captures short-term and long-term trends in machine behavior
- Missing values handled with forward fill → 0
- Reason: Needed additional features for XGBoost model, which relies on manual feature input

## Step 4: Data Splitting & Model Preparation

- Sorted data by machine ID and datetime
- Time-based split:
  - Train: first 70% of data
  - Validation: next 15% of data
  - Test: final 15% of data
- Checked class imbalance in each split for 24h, 48h, 72h prediction windows

```
train: n=613,300  time 2015-01-01 06:00:00 → 2015-09-13 18:00:00
  will_fail_24h: positives=12,542  rate=2.045%
  will_fail_48h: positives=24,723  rate=4.031%
  will_fail_72h: positives=36,928  rate=6.021%
  val: n=131,400  time 2015-09-13 19:00:00 → 2015-11-07 12:00:00
  will_fail_24h: positives=2,193  rate=1.669%
  will_fail_48h: positives=4,368  rate=3.324%
  will_fail_72h: positives=6,594  rate=5.018%
 test: n=131,400  time 2015-11-07 13:00:00 → 2016-01-01 06:00:00
  will_fail_24h: positives=2,448  rate=1.863%
  will_fail_48h: positives=4,868  rate=3.705%
  will_fail_72h: positives=7,213  rate=5.489%
```

# Step 5: Model Selection

**XGBoost**

- Gradient-boosted decision trees for supervised learning;
- Strong performance on tabular data;
- Fast and interpretable, but needs manual feature creation.

← Baseline Model

**PatchTST**

- Transformer-based model for time series forecasting;
- Learns temporal patterns automatically from raw sequences;
- Effective for multivariate time series and long-term prediction.

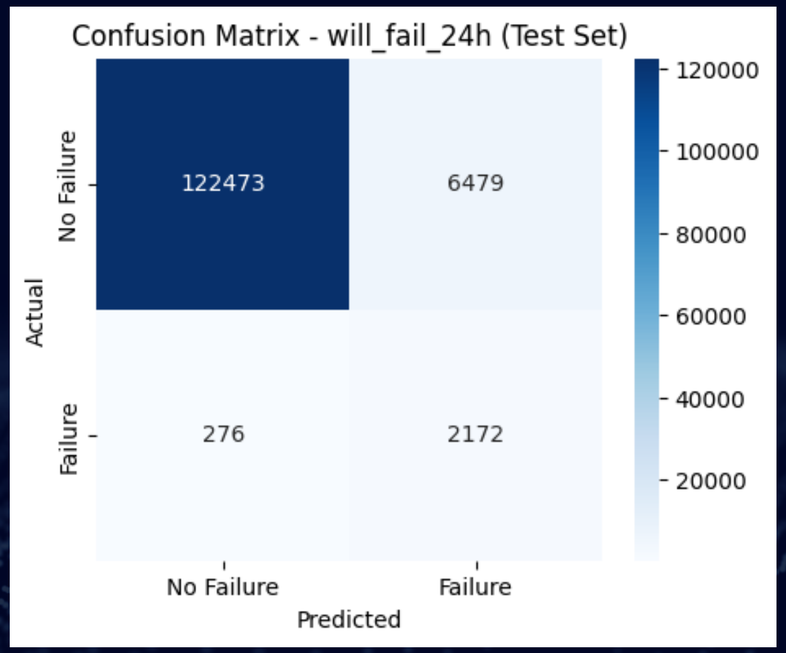
Transformer Models ⟷

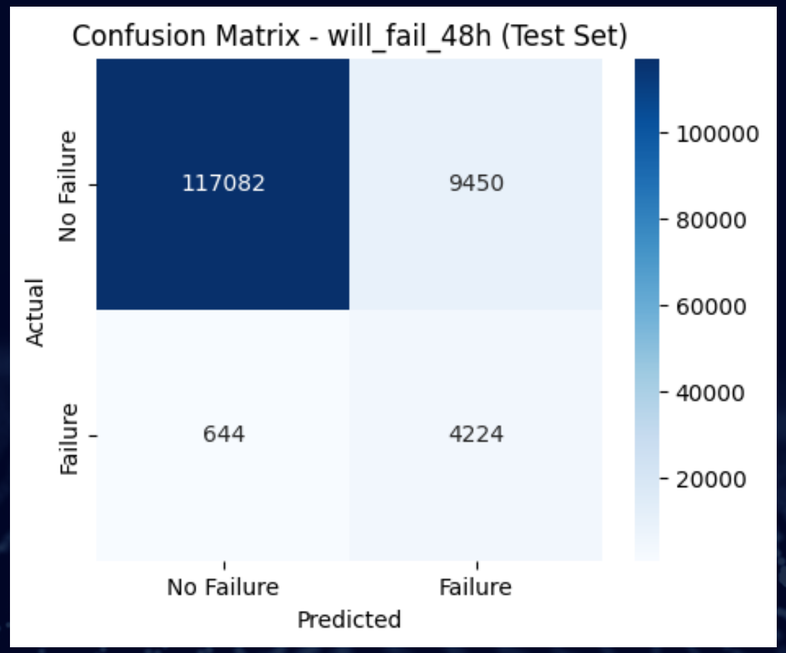
**MOMENT**

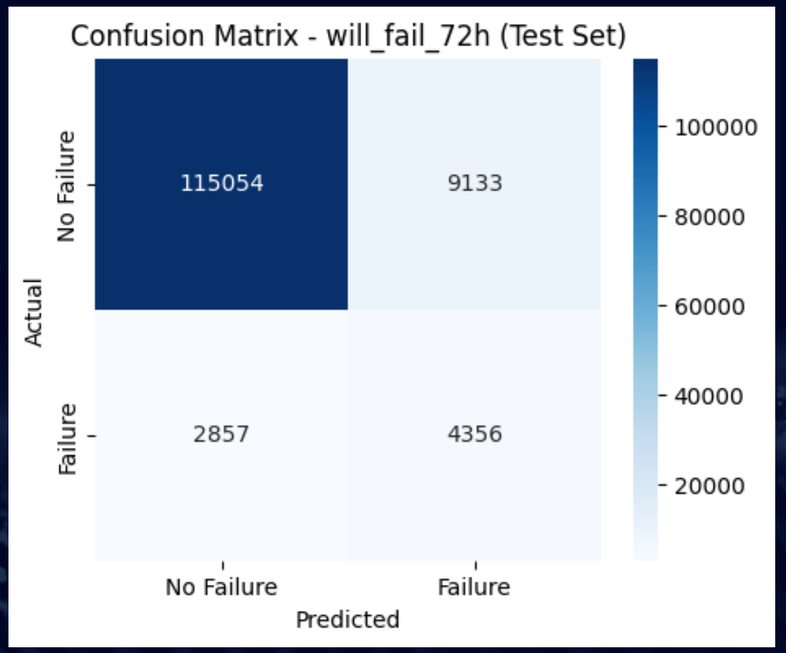
- Large pre-trained transformer foundation model for time series;
- Enables zero-shot and few-shot forecasting, handling multiple tasks and components simultaneously;
- Learns patterns from many machines, working well on new or rare cases.

# Results & Analysis

Model: XGBoost




=== TEST SET REPORT ===

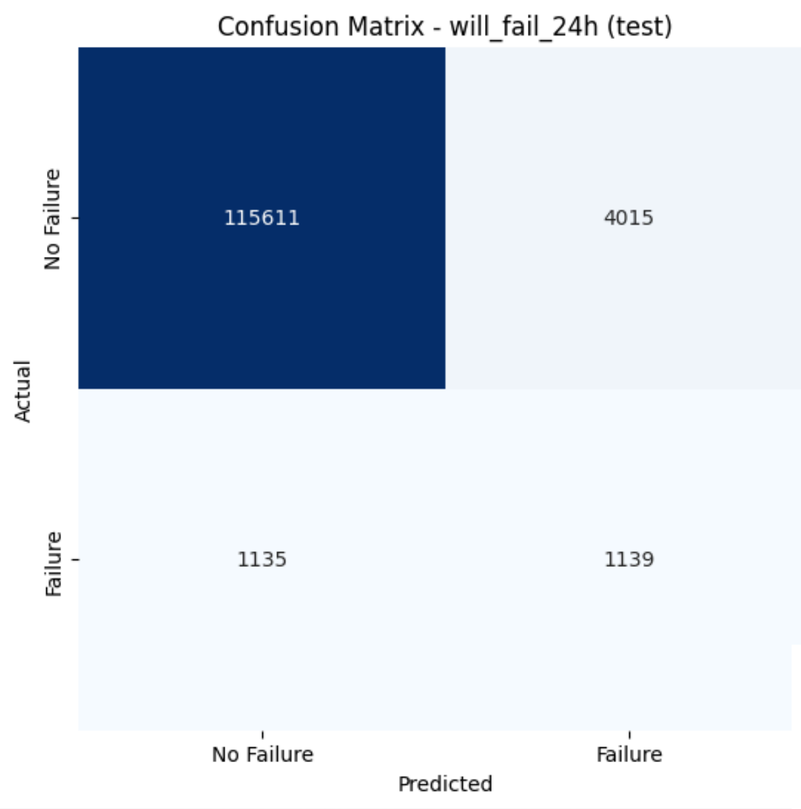
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No Failure | 1.00 | 0.95 | 0.97 | 128952 |
| Failure | 0.25 | 0.89 | 0.39 | 2448 |
| accuracy | | | 0.95 | 131400 |
| macro avg | 0.62 | 0.92 | 0.68 | 131400 |
| weighted avg | 0.98 | 0.95 | 0.96 | 131400 |

=== TEST SET REPORT ===

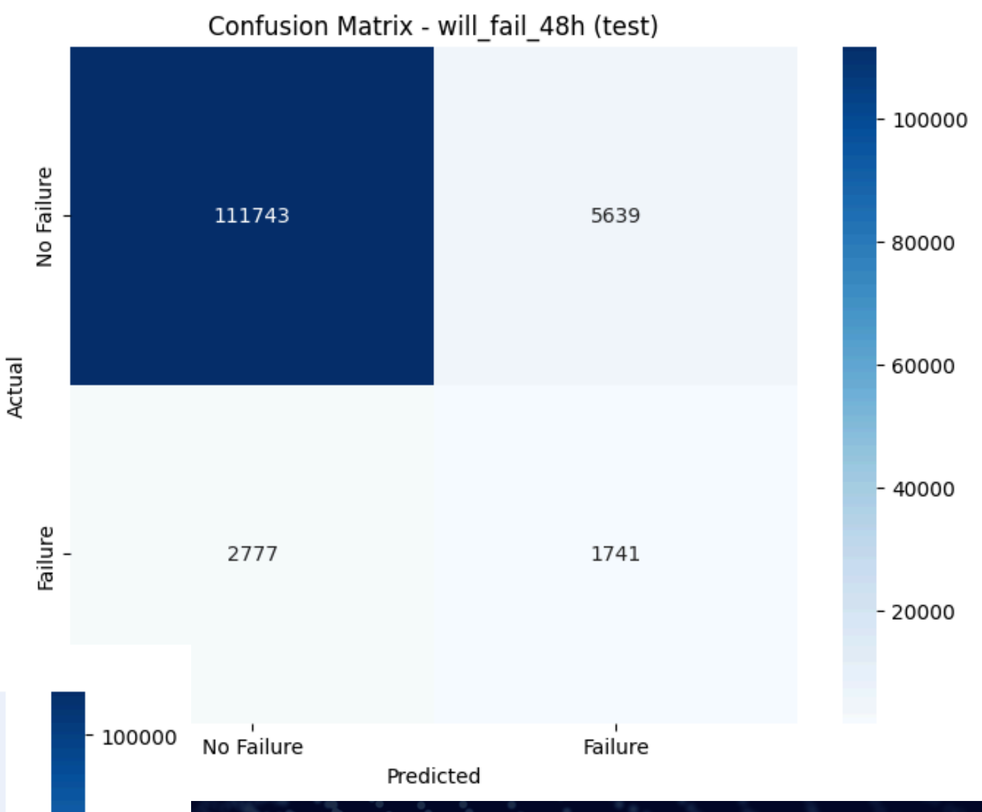
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No Failure | 0.99 | 0.93 | 0.96 | 126532 |
| Failure | 0.31 | 0.87 | 0.46 | 4868 |
| accuracy | | | 0.92 | 131400 |
| macro avg | 0.65 | 0.90 | 0.71 | 131400 |
| weighted avg | 0.97 | 0.92 | 0.94 | 131400 |

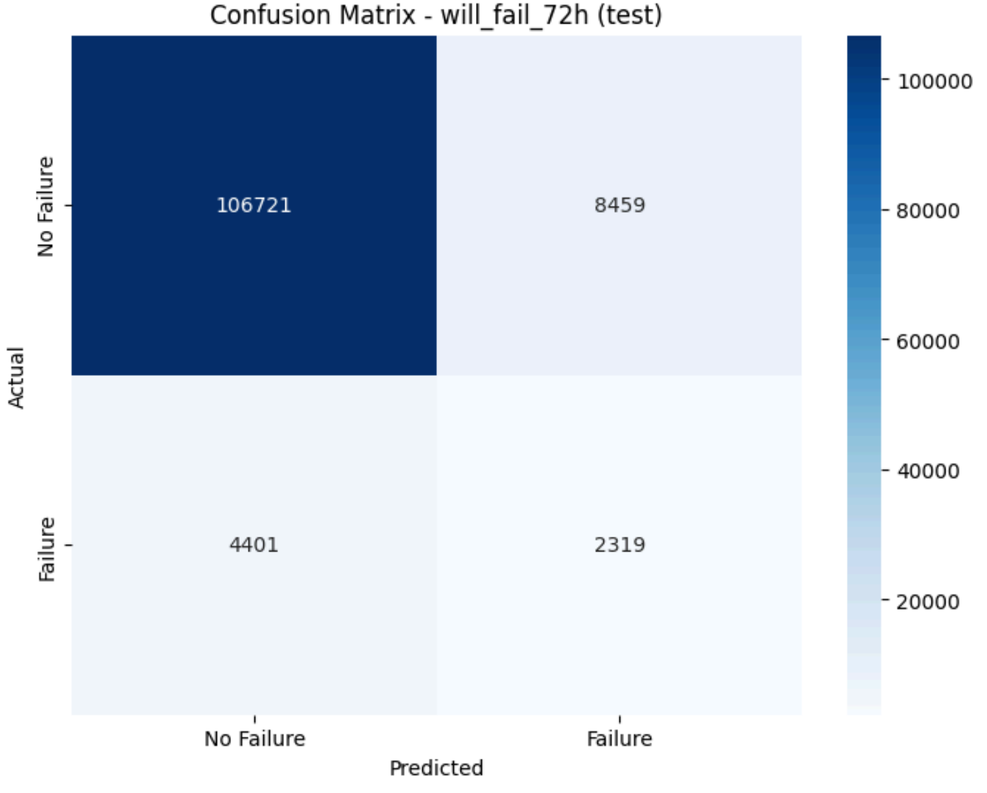
=== TEST SET REPORT ===

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No Failure | 0.98 | 0.93 | 0.95 | 124187 |
| Failure | 0.32 | 0.60 | 0.42 | 7213 |
| accuracy | | | 0.91 | 131400 |
| macro avg | 0.65 | 0.77 | 0.69 | 131400 |
| weighted avg | 0.94 | 0.91 | 0.92 | 131400 |

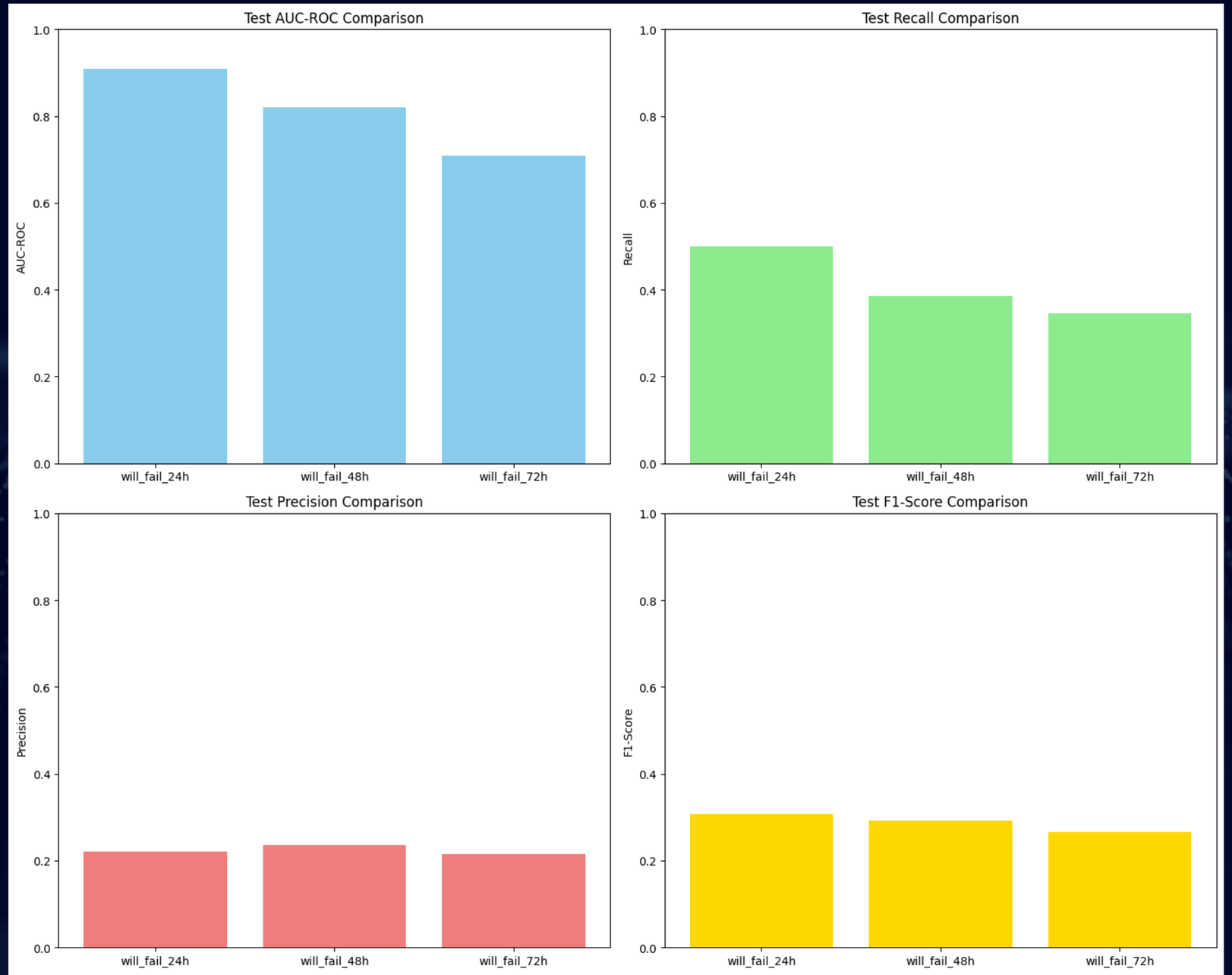



Model: PatchTST

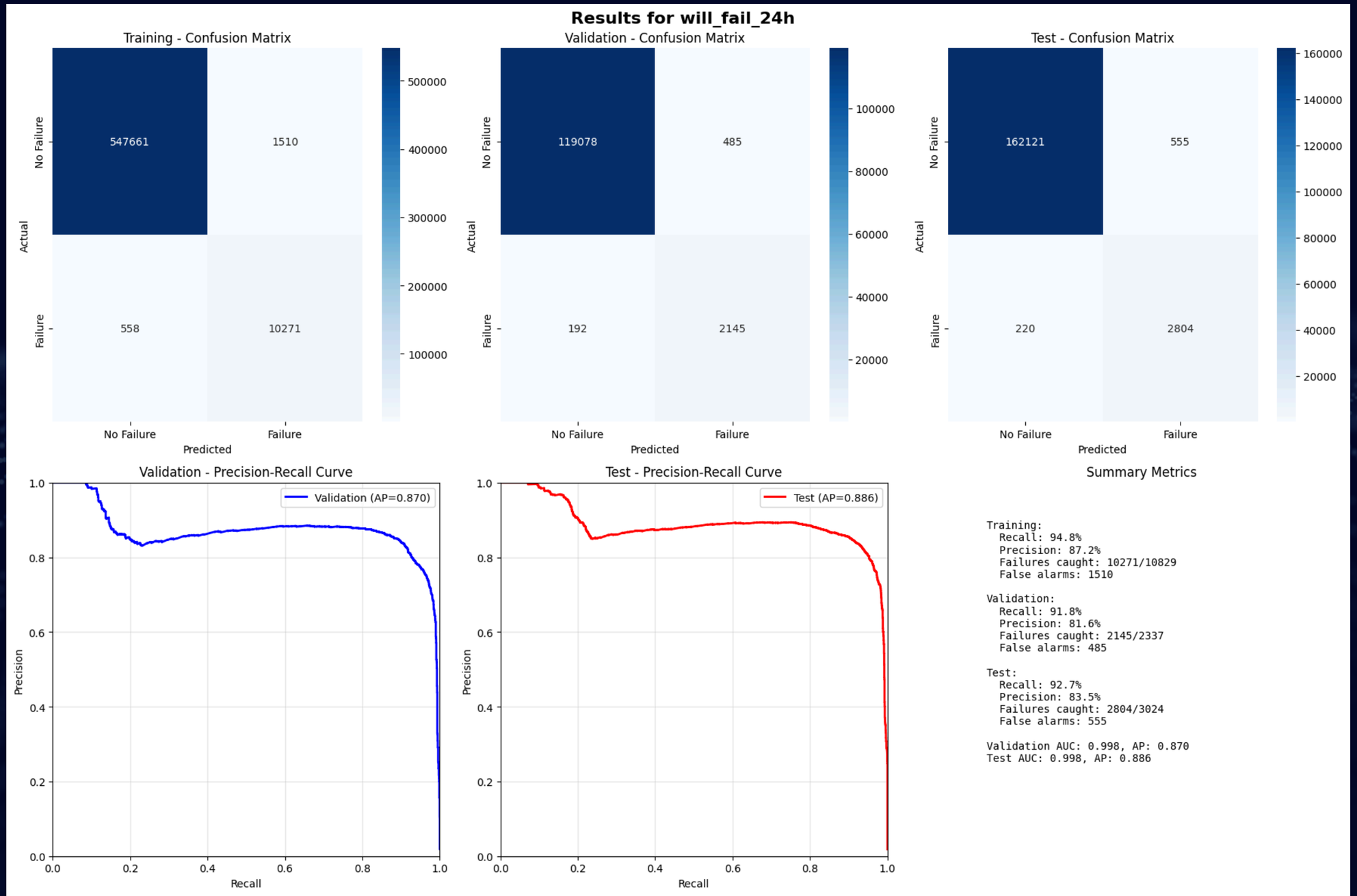


# Model: PatchTST

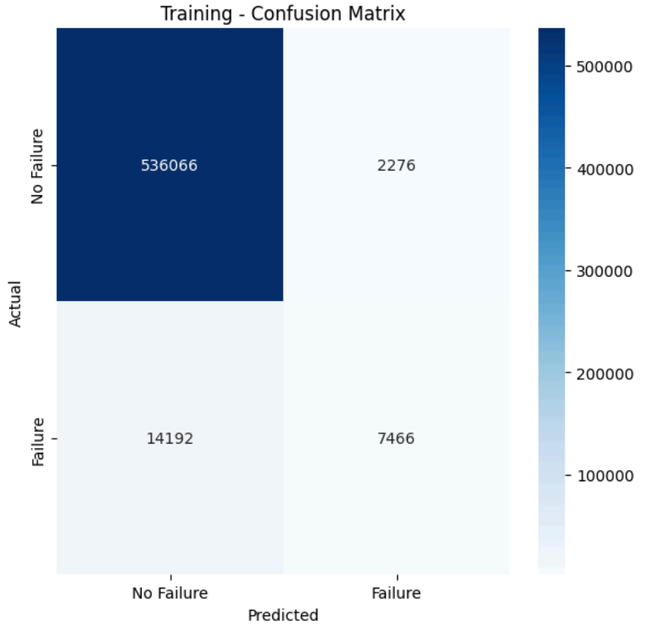


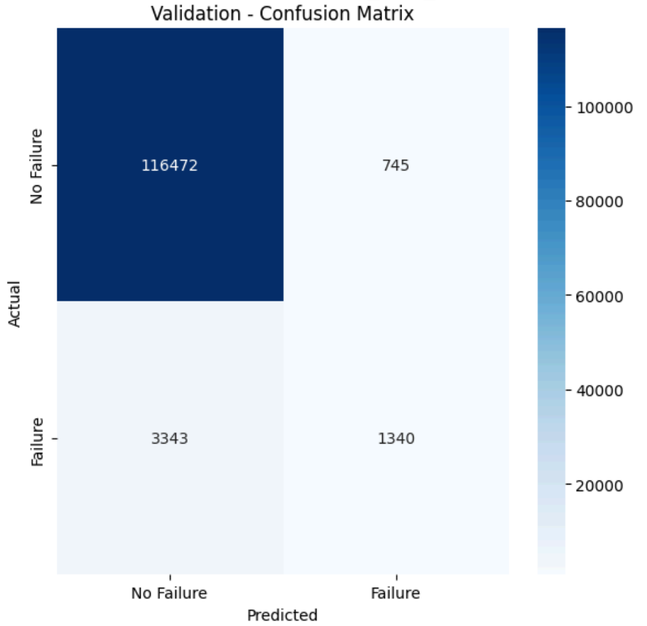
# Model: PatchTST hybrid (24h)

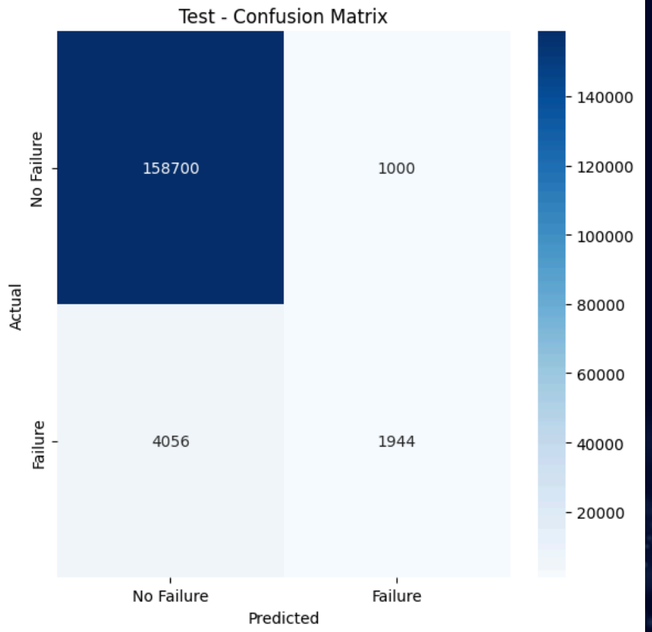


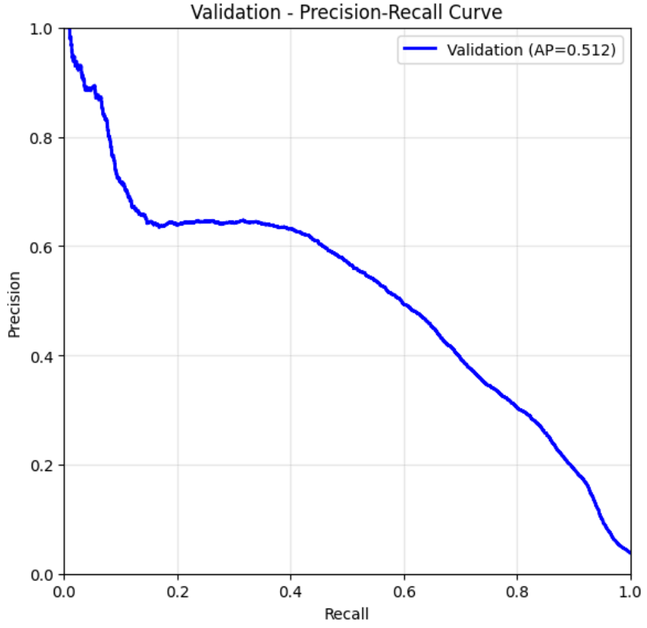
# Model: PatchTST hybrid (48h)

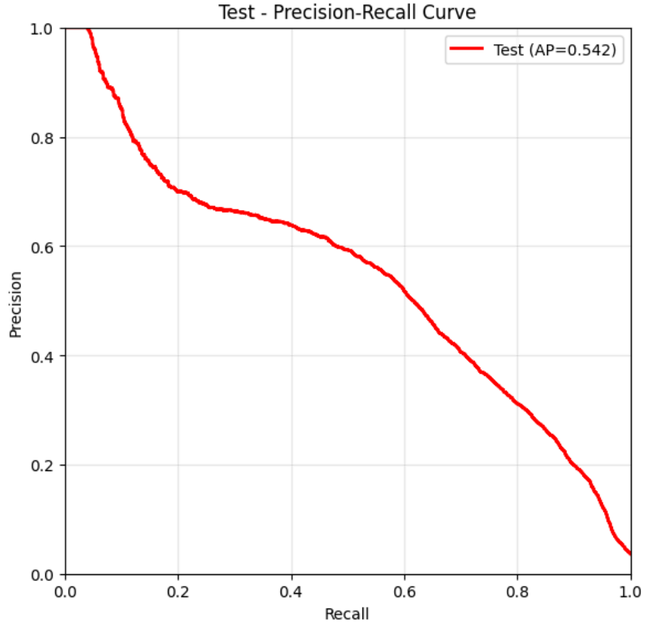
**Results for will_fail_48h**




Summary Metrics

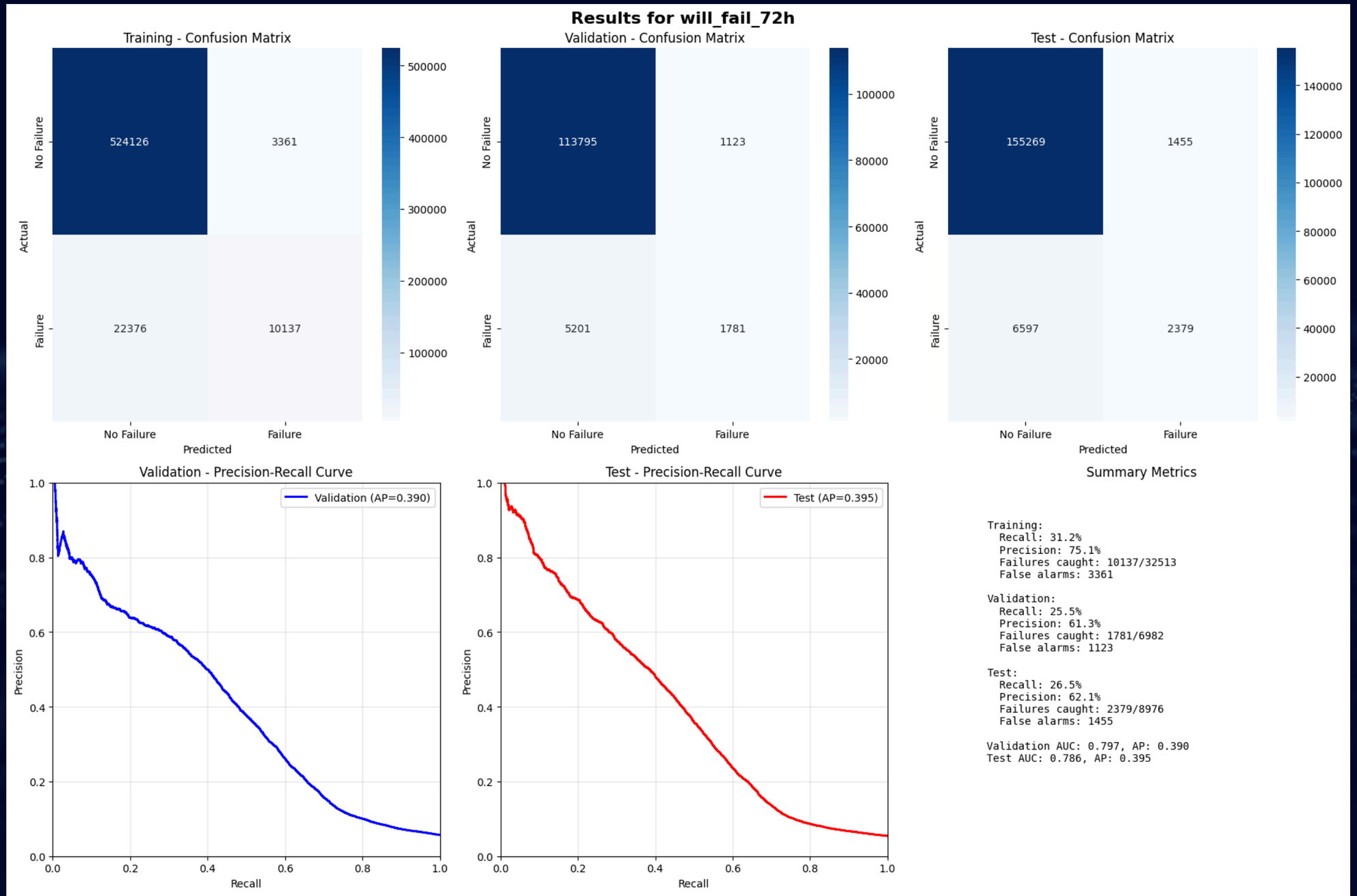
```
Training:
  Recall: 34.5%
  Precision: 76.6%
  Failures caught: 7466/21658
  False alarms: 2276

Validation:
  Recall: 28.6%
  Precision: 64.3%
  Failures caught: 1340/4683
  False alarms: 745

Test:
  Recall: 32.4%
  Precision: 66.0%
  Failures caught: 1944/6000
  False alarms: 1000

Validation AUC: 0.934, AP: 0.512
Test AUC: 0.942, AP: 0.542
```

# Model: PatchTST hybrid (72h)



# Model: SimpleMOMENT

```
=== TEST CLASSIFICATION REPORT (will_fail_24h) ===
              precision    recall  f1-score   support

  No Failure      0.998     0.935     0.965    126676
     Failure      0.208     0.892     0.338      2424

    accuracy                          0.934    129100
   macro avg      0.603     0.913     0.651    129100
weighted avg      0.983     0.934     0.954    129100

=== TEST CLASSIFICATION REPORT (will_fail_48h) ===
              precision    recall  f1-score   support

  No Failure      0.981     0.956     0.968    124330
     Failure      0.313     0.526     0.392      4770

    accuracy                          0.940    129100
   macro avg      0.647     0.741     0.680    129100
weighted avg      0.957     0.940     0.947    129100

=== TEST CLASSIFICATION REPORT (will_fail_72h) ===
              precision    recall  f1-score   support

  No Failure      0.957     0.972     0.965    122014
     Failure      0.347     0.255     0.294      7086

    accuracy                          0.933    129100
   macro avg      0.652     0.614     0.629    129100
weighted avg      0.924     0.933     0.928    129100
```

# Implications and Future Work

- Near-perfect 24h prediction with PatchTST + contextual features (94% recall, 83% precision).
- Class imbalance challenge: failures = only 0.001% of data → rare event detection remains difficult.
- Long-term prediction hard: performance drops significantly beyond 24–48 hours.
- Traditional feature engineering still matters: raw sensor data alone not enough.
- Business value: improved recall/precision can save millions in downtime & false alarms.

Limitations:

- Couldn’t run full MOMENT model due to hardware constraints.
- Transformers mainly built for forecasting, not classification → needed adaptation.

Future work:

- Handle imbalance with SMOTE, focal loss, or other methods.
- Try new transformer architectures (Informer, Autoformer, etc.).
- Improve computational efficiency for foundation models (e.g., full MOMENT).

# Conclusion

- Transformers (esp. PatchTST + context) outperform traditional methods in short-term failure prediction.
- High recall + precision at 24h window = strong business impact.
- Challenges remain: long-term prediction + extreme imbalance.
- Transformers are evolving fast: currently forecasting-focused, but big potential for classification tasks.
- Predictive maintenance future: transformer architectures are a promising and scalable solution.

# Q&A

# References


- https://iot-analytics.com/predictive-maintenance-market/
- https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance
- https://dataloop.ai/library/model/autonlab_moment-1-large/
- https://huggingface.co/docs/transformers/en/model_doc/patchtst
- https://arxiv.org/html/2505.06295v1
- https://www.researchgate.net/publication/345965691_PERFORMANCE_COMPARISON_OF_MACHINE_LEARNING_ALGORITHMS_FOR_PREDICTIVE_MAINTENANCE

# Thank You!

Elene Esakia | 2025