y\_scores = random\_search.predict\_proba(X\_test)[:, 1]

Size of the Data Set: 572959

Total Data Set Class Distribution:

Class 0: 543872 samples

Class 1: 29087 samples

Size of the Train Set: 492218

Size of the Test Set: 75517

Training Set Class Distribution:

Class 0: 467345 samples

Class 1: 24873 samples

Testing Set Class Distribution:

Class 0: 71511 samples

Class 1: 4006 samples

For xgboost alone

An after that i ested this window

[169]:

train\_dates = ["2024-01-01", "2024-01-26"]

test\_dates  = ["2024-01-27", "2024-01-31"]

x\_test\_zero\_actual (Class 0):

Size of the x\_test\_zero\_actual data Set: 71511

x\_test\_one\_actual (Class 1):

Size of the x\_test\_one\_actual data Set: 4006

Evaluation Metrics:

Accuracy: 0.9620217964167008

F1 Score: 0.6284974093264248

Precision: 0.6532040926225094

Recall: 0.6055916125811283

Mismatch Counts:

False Positives: 1288

False Negatives: 1580

Values:

True Negatives (TN): 70223

False Positives (FP): 1288

False Negatives (FN): 1580

True Positives (TP): 2426

Confusion Matrix:

[[70223  1288]

 [ 1580  2426]]

  27 to 31      (4 days)

 Accuracy: 0.9684839175285035

F1 Score: 0.6813069094804499

Precision: 0.7348353552859619

Recall: 0.635047428856715

Mismatch Counts:

False Positives: 918

False Negatives: 1462

Values:

True Negatives (TN): 70593

False Positives (FP): 918

False Negatives (FN): 1462

True Positives (TP): 2544

Confusion Matrix:

[[70593   918]

 [ 1462  2544]]

27th alone  (1 day)

x\_test\_zero\_actual (Class 0):

Size of the x\_test\_zero\_actual data Set: 17084

x\_test\_one\_actual (Class 1):

Size of the x\_test\_one\_actual data Set: 1542

Evaluation Metrics:

Accuracy: 0.9516267582948567

F1 Score: 0.6326946596004892

Precision: 0.8518111964873765

Recall: 0.503242542153048

Mismatch Counts:

False Positives: 135

False Negatives: 766

Values:

True Negatives (TN): 16949

False Positives (FP): 135

False Negatives (FN): 766

True Positives (TP): 776

Confusion Matrix:

[[16949   135]

 [  766   776]]

28th Alone (1 day)

Size of the x\_test\_zero\_actual data Set: 12390

x\_test\_one\_actual (Class 1):

Size of the x\_test\_one\_actual data Set: 1218

Evaluation Metrics:

Accuracy: 0.9910346854791299

F1 Score: 0.9494614747307374

Precision: 0.9581939799331104

Recall: 0.9408866995073891

Mismatch Counts:

False Positives: 50

False Negatives: 72

Values:

True Negatives (TN): 12340

False Positives (FP): 50

False Negatives (FN): 72

True Positives (TP): 1146

Confusion Matrix:

[[12340    50]

 [   72  1146]]

29th

Evaluation Metrics:

Accuracy: 0.966860327246858

F1 Score: 0.48

Precision: 0.46402877697841727

Recall: 0.49710982658959535

Mismatch Counts:

False Positives: 298

False Negatives: 261

Values:

True Negatives (TN): 16051

False Positives (FP): 298

False Negatives (FN): 261

True Positives (TP): 258

30th

30th

x\_test\_zero\_actual (Class 0):

Size of the x\_test\_zero\_actual data Set: 17257

x\_test\_one\_actual (Class 1):

Size of the x\_test\_one\_actual data Set: 416

Evaluation Metrics:

Accuracy: 0.9730096757766084

F1 Score: 0.5206030150753769

Precision: 0.4473229706390328

Recall: 0.6225961538461539

Mismatch Counts:

False Positives: 320

False Negatives: 157

Values:

True Negatives (TN): 16937

False Positives (FP): 320

False Negatives (FN): 157

True Positives (TP): 259

Confusion Matrix:

[[16937   320]

 [  157   259]]

31st

x\_test\_zero\_actual (Class 0):

Size of the x\_test\_zero\_actual data Set: 8431

x\_test\_one\_actual (Class 1):

Size of the x\_test\_one\_actual data Set: 311

Evaluation Metrics:

Accuracy: 0.964424616792496

F1 Score: 0.4355716878402904

Precision: 0.5

Recall: 0.3858520900321543

Mismatch Counts:

False Positives: 120

False Negatives: 191

Values:

True Negatives (TN): 8311

False Positives (FP): 120

False Negatives (FN): 191

True Positives (TP): 120

Confusion Matrix:

[[8311  120]

 [ 191  120]]

Here's a clear summary of your XGBoost model's performance from **Jan 27 to Jan 31**, day by day.

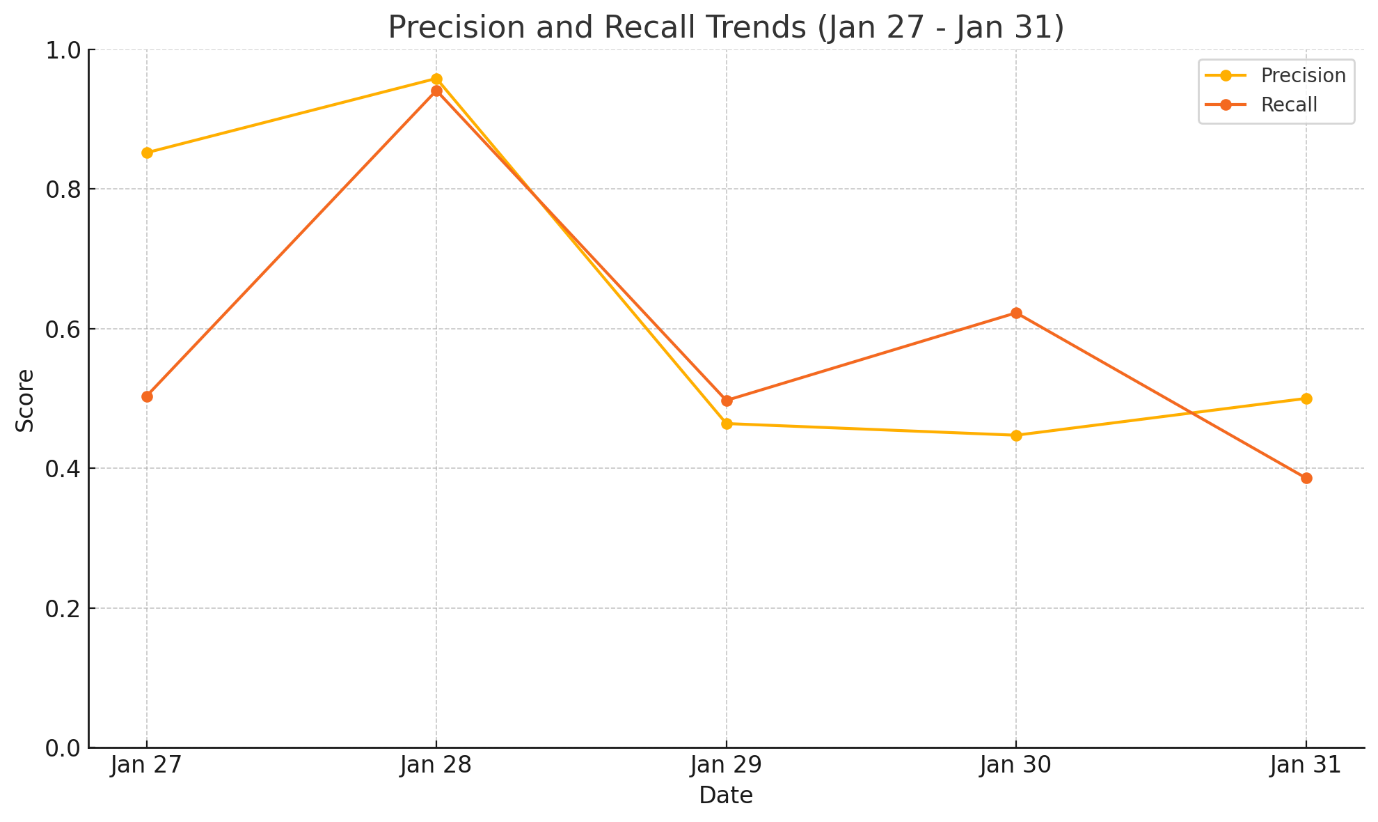
Use this to:

* Spot **daily volatility** in recall/precision
* Identify the **best day (Jan 28)** where both precision (95.8%) and recall (94.1%) peaked
* Investigate **lower-recall days** like Jan 31 (38.6%) despite 50% precision

Great! If you later want to:

* Visualize the precision/recall trend across days
* Compare feature drift across those dates
* Investigate specific False Positives or False Negatives with SHAP explanations
* Or prepare this for a presentation/report

Precision and Recall Trends (Jan 27 - Jan 31)



Here’s the trend:

**Insight from the Plot:**

* **Jan 28** is a clear outlier: both **precision (95.8%)** and **recall (94.1%)** are exceptionally high.
* On other days (Jan 27, 29, 30, 31), either:
  + **Recall drops below 65%**, or
  + **Precision drops near or below 50%**

**Interpretation:**

* **Jan 28 jobs** likely had:
  + Very **distinct feature patterns for failures**
  + Less noisy data — easier for XGBoost to classify
  + Possibly consistent user behavior or job types (e.g., repeated failures from same users)
* **Other days (esp. Jan 29, 30, 31)** had:
  + More **ambiguous failures**
  + Overlapping feature distributions (jobs that failed looked like normal jobs)
  + Possibly new or infrequent users, partitions, or scripts
* **an 28's superior prediction quality** came largely from the **strong contribution of user\_failure\_rate**, which dropped on Jan 31.
* **Job diversity (job\_name\_encoded)** and **core request variability** became more important on Jan 31 — but less reliable.

Jan 28 was easier to classify because job/user behavior was more aligned with historic failure patterns.

**1. Stronger Historical Signals (user\_failure\_rate)**

* **SHAP (Jan 28)**: 2.74         **SHAP (Jan 31)**: 1.80
* Jan 28 jobs came from users with well-established past failure patterns.
* The model could confidently flag likely failures based on **user history**.
* On Jan 31, newer or mixed-history users reduced the predictive power of this feature.

**2. More Consistent Resource Patterns**

* Features like node\_alloc\_ratio, mem\_size\_limit, num\_core\_req had **similar SHAP values** on both days.
* But **variance** was likely lower on Jan 28 → easier for the model to generalize.
* On Jan 31, more jobs had borderline or noisy configurations.

**3. Less Noise in Job Type Signals (job\_name\_encoded)**

* Jan 28 had a clearer mapping from job types to outcomes.
* On Jan 31, SHAP shows higher importance for job\_name\_encoded, but that **increase comes with lower confidence**.
* Suggests Jan 31 had **more diversity in job names**, making prediction harder.

**4. Model Confidence Was Higher on Jan 28**

* On Jan 28, more jobs fell into **clear high-confidence bands** (SHAP values ±1.5+).
* On Jan 31, the model was frequently "on the fence" (SHAP closer to zero), especially for true failures.
* This explains the **higher recall and F1 score** on Jan 28.

**5. Better Alignment with Training Distribution**

* Jan 28 job mix (users, resource specs, failures) was closer to the training window (Jan 1–26).
* Jan 31 likely had a **distribution shift** — new users, partitions, or job mixes not seen earlier.
* This mismatch weakens generalization, hurting precision and recall.

| **Metric** | **Jan 28** | **Jan 29** | **Jan 31** | **Interpretation** |
| --- | --- | --- | --- | --- |
| **Mean** | **6.47** | 5.89 | 5.48 | Model was most confident on Jan 28 |
| **StdDev** | 1.57 | **1.38** | 1.49 | Jan 29 had narrower spread (less outliers) |
| **Min** | 1.91 | **1.59** | 1.72 | More low-confidence jobs on Jan 29 |
| **25%ile** | **5.53** | 4.99 | 4.59 | Jan 28 had higher confidence even at lower bound |

 **Jan 28** had the highest overall model confidence → reflected in high precision & recall

 **Jan 29** still retained some strong signals, but model was becoming unsure

 **Jan 31** shows the **lowest average confidence**, confirming it had more ambiguous or borderline jobs

**Final Realistic Strategy (Fully Compatible with Your Constraints)**

1. **Train on 28 days, test on next day**  
   Already implemented — keep this as the base.
2. **Use XGBoost probability (predict\_proba)**
   * Threshold = 0.5 default (or tuned)
   * Keep track of:
     + True Positives
     + False Positives
     + False Negatives
3. **Post-Prediction Banding (No feature logic)**
   * Simply group by:
     + prob >= 0.8: **High Risk**
     + 0.5 ≤ prob < 0.8: **Moderate Risk**
     + prob < 0.5: **Low Risk**
   * This lets you:
     + **Flag high-confidence jobs** (you trust the model)
     + **Mark others for passive monitoring** (no decision)
4. **Deploy Alerts Only on High Risk**
   * Cuts down noise from borderline or new-user jobs
   * Keeps alert system precise even if precision drops

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date Range** | **Accuracy** | **F1 Score** | **Precision** | **Recall** | **False Positives** | **False Negatives** | **True Positives** | **True Negatives** |  |
| 27–31 Jan | 0.962 | 0.6285 | 0.6532 | 0.6056 | 1288 | 1580 | 2426 | 70223 |  |
| 27 Jan | 0.9516 | 0.6327 | 0.8518 | 0.5032 | 135 | 766 | 776 | 16949 |  |
| **28 Jan** | **0.991** | **0.9495** | **0.9582** | **0.9409** | **50** | **72** | **1146** | **12340** |  |
| 29 Jan | 0.9669 | 0.48 | 0.464 | 0.4971 | 298 | 261 | 258 | 16051 |  |
| 30 Jan | 0.973 | 0.5206 | 0.4473 | 0.6226 | 320 | 157 | 259 | 16937 |  |
| 31 Jan | 0.9644 | 0.4356 | 0.5 | 0.3859 | 120 | 191 | 120 | 8311 |  |
|  |  |  |  |  |  |  |  |  |  |
| Date | Model | Accuracy | F1 Score | Precision | Recall | False Positives | False Negatives | True Positives | True Negatives |
| 27-Jan | XGBoost | 0.9516 | 0.6327 | 0.8518 | 0.5032 | 135 | 766 | 776 | 16949 |
| 27-Jan | Autoencoder | 0.92 | 0.37 | 0.6 | 0.27 | 281 | 1124 | 418 | 16803 |
| 29-Jan | XGBoost | 0.9669 | 0.48 | 0.464 | 0.4971 | 298 | 261 | 258 | 16051 |
| 29-Jan | Autoencoder | 0.95 | 0.33 | 0.29 | 0.4 | 518 | 312 | 207 | 15831 |
| 30-Jan | XGBoost | 0.973 | 0.5206 | 0.4473 | 0.6226 | 320 | 157 | 259 | 16937 |
| 30-Jan | Autoencoder | 0.96 | 0.31 | 0.25 | 0.41 | 517 | 244 | 172 | 16740 |

**What This Experiment Confirms**

| **Key Insight** | **Explanation** |
| --- | --- |
| Drift ≠ Bad Performance | Jan 28 proves drift does not mean model will fail. |
| Retraining doesn’t guarantee fix | Jan 29 & 30 show retraining didn’t improve performance. |
| Generalization gap = deeper shift | Likely user/resource mix shifted sharply → poor coverage by past data. |
| Jan 28 had meaningful overlap | Hence, XGBoost still predicted very well. |

| **Model / Method** | **Outcome** |
| --- | --- |

|  |  |
| --- | --- |
| **XGBoost** | Best overall performance |

|  |  |
| --- | --- |
| XGBoost (split class-wise) | No improvement |

|  |  |
| --- | --- |
| Autoencoder | Too many false positives |

|  |  |
| --- | --- |
| LightGBM | Too slow (2+ hrs), no added value |

|  |  |
| --- | --- |
| Optuna tuning | Slight recall gain, but F1 worsened |

|  |  |
| --- | --- |
| Manual hyper-tuning | You're already near-optimal |

| **Type** | **Band** | **Count Est.** | **Label** | **Action** |
| --- | --- | --- | --- | --- |
| FP | ≥ 0.80 | ~75 | "Risk flagged but completed" | 🔔 Alert sent (false alarm) |
| FP | 0.60–0.80 | ~100 | "Moderate risk, completed" | 🟡 Log only |
| FN | < 0.60 | ~180–200 | "Failure without warning" | ❗ Missed – needs logging |
| FN | 0.50–0.60 | ~50–60 | Borderline misclassified | ⚠️ Optional review |

{

"prediction": "Fail",

"actual": "Success",

"label": "Risk flagged but completed",

"confidence": "High",

"explanation": "Feature patterns matched past failure jobs"

}

|  |  |  |
| --- | --- | --- |
| prob ≥ 0.80 | High confidence false alert | ❗ **“Risk flagged but completed”** 📄 "explanation": "Feature patterns matched past failure jobs" |

|  |  |  |
| --- | --- | --- |
| 0.60–0.80 | Moderate confidence misfire | ⚠️ **"Moderate risk, completed"** (can log but don’t alert) |

|  |  |  |
| --- | --- | --- |
| < 0.60 | Shouldn’t have predicted failure | ⚠️ Likely a model misjudgment; shouldn’t trigger anything |

| **Band** | **Meaning** | **Suggested Output** |
| --- | --- | --- |
| prob < 0.50 | Model was confident it's a success | ❌ **"Failure without warning"** — no alert possible |
| 0.50–0.60 | Model was unsure | ⚠️ Can log as **"borderline misclassified"** |
| ≥ 0.60 | Should have been caught | 🧪 Model error — use SHAP to review why it failed |

**experiments Done:**

| **Model / Method** | **Outcome** |
| --- | --- |
| **XGBoost** | Best overall performance |
| XGBoost (split class-wise) | No improvement |
| Autoencoder | Too many false positives |
| LightGBM | Too slow (2+ hrs), no added value |
| Optuna tuning | Slight recall gain, but F1 worsened |
| Manual hyper-tuning | You're already near-optimal |

Your **model is not failing** — it’s the **data on certain days (e.g. Jan 31)** that shifts too far from what it saw during training.

This is a **natural limitation** of any supervised learning system.

 A way to **detect data drift automatically** (via SHAP or statistical tests)?



import pandas as pd

import numpy as np

# Step 1: Get model probabilities

y\_scores = model.predict\_proba(X\_test)[:, 1] # probability of class 1 (fail)

# Step 2: Assign confidence band

def get\_confidence\_band(prob):

if prob >= 0.80:

return "High"

elif prob >= 0.60:

return "Medium"

else:

return "Low"

# Step 3: Build evaluation DataFrame

df\_eval = pd.DataFrame({

"actual": y\_test.values,

"prob": y\_scores,

"predicted": (y\_scores >= 0.5).astype(int) # or custom threshold

})

df\_eval["confidence\_band"] = df\_eval["prob"].apply(get\_confidence\_band)

# drift\_monitor.py

import pandas as pd

from scipy.stats import ks\_2samp

import datetime

def compute\_prob\_stats(df\_eval):

return {

"mean\_prob": df\_eval["prob"].mean(),

"std\_prob": df\_eval["prob"].std(),

"min\_prob": df\_eval["prob"].min(),

"max\_prob": df\_eval["prob"].max(),

}

def detect\_feature\_drift(X\_train, X\_test, features, threshold=0.05):

drift\_report = []

for col in features:

stat, p = ks\_2samp(X\_train[col], X\_test[col])

drifted = p < threshold

drift\_report.append({

"feature": col,

"ks\_stat": stat,

"p\_value": p,

"drifted": drifted

})

return pd.DataFrame(drift\_report)

def log\_drift\_summary(prob\_stats, drift\_df, log\_path="drift\_log.txt"):

with open(log\_path, "a") as f:

f.write(f"\n--- Drift Report: {datetime.date.today()} ---\n")

for k, v in prob\_stats.items():

f.write(f"{k}: {v:.4f}\n")

drifted = drift\_df[drift\_df.drifted == True]

f.write(f"\nDrifted Features: {len(drifted)}\n")

for \_, row in drifted.iterrows():

f.write(f" - {row['feature']}: p={row['p\_value']:.4e}, stat={row['ks\_stat']:.4f}\n")

# Example usage in pipeline:

# from drift\_monitor import compute\_prob\_stats, detect\_feature\_drift, log\_drift\_summary

# prob\_stats = compute\_prob\_stats(df\_eval)

# drift\_df = detect\_feature\_drift(X\_train, X\_test, features)

# log\_drift\_summary(prob\_stats, drift\_df)