NASA Bearing Dataset: Predicting Machine Failure Through Vibration Analysis

Executive Summary

Imagine a factory with heavy machinery. These machines vibrate during normal operation, but as they wear out, the vibration pattern changes. By monitoring these vibrations, we can predict when a machine is about to fail — giving us a heads-up before expensive downtime happens.

This project does exactly that using real vibration data from NASA's bearing experiments. We analyze the vibrations to spot unusual patterns that signal equipment degradation.

What's the Real-World Problem?

The Situation:

- Industrial machines (bearings, motors, pumps) generate vibrations while working.
- Early signs of wear show up in changing vibration patterns before actual failure.
- Unexpected breakdowns cost companies money, time, and can be dangerous.
- Maintenance teams need a way to know "when should I replace this part?"

Our Solution:

Build a system that automatically watches vibration data and alerts when something looks off — giving maintenance teams time to act before failure.

Understanding the Data

What is a vibration snapshot?

You place a sensor on a machine and record how much it shakes for 1 second. The sensor takes 20,000 measurements in that second (sampling at 20,000 times per second). This gives us one "snapshot" file with 20,000 numbers representing shake intensity over time.

The Dataset Structure:

- Multiple tests running for days or weeks until bearing failure.
- Each test has hundreds of snapshot files (one every few minutes).
- Each file named with timestamp: `2003.10.23.00.54.13` means October 23, 2003 at 00:54:13.
- Files stored in folders: `1st test`, `2nd test`, `3rd test` (three separate bearing tests).

Example: If a test runs for 10 days, and we record snapshots every 10 minutes, that's about 1,440 files per test.

Our Approach: Simple Statistics Instead of Raw Numbers

The Challenge:

20,000 numbers per file × hundreds of files = huge amount of data. Hard to spot patterns or find anomalies directly.

Our Smart Solution:

Instead of looking at all 20,000 vibration readings, we compute 4 simple numbers that summarize each snapshot:

1. **Mean (Average)** – What's the typical vibration intensity?
    - Low mean = machine works good
    - High mean = something's working hard or wearing out
2. **Standard Deviation (Spread)** – How much does vibration vary?
    - Low spread = consistent, smooth operation
    - High spread = erratic, unpredictable shaking = could be failing
3. **Minimum** – What's the quietest moment in this snapshot?
4. **Maximum** – What's the strongest shake in this snapshot?

These 4 numbers give us a "health summary" of the bearing at each moment.

Why This Works:

- Easier to visualize 4 numbers over 100 files than 20,000 × 100 numbers
- Captures the essence of vibration behavior
- Can spot trends over time (e.g., mean keeps increasing = degrading health)

How We Detect Anomalies (Unusual Patterns)

What's an Anomaly?

A point in time where the vibration pattern is "weird" or very different from normal. Like if your car normally runs quietly, but one day it starts making strange sounds — that's an anomaly.

Our Detection Method: Isolation Forest

Imagine you're looking at a crowd of people. Most people look normal. Suddenly, someone dressed as a superhero walks by — they stand out! The "Isolation Forest" algorithm works similarly:

- It builds a statistical model of what "normal" vibration looks like based on the majority of data.
- It checks each new snapshot against this normal pattern.
- If a snapshot is very different from normal, it flags it as an anomaly.
- It doesn't need us to tell it what "failure" looks like — it just finds what's different.

Why Isolation Forest?

- Works without labeled data (we don't need to know in advance which snapshots are failures).
- Good for finding outliers in tabular data (our 4 summary numbers per snapshot).
- Fast and interpretable — we can see which snapshots are flagged as unusual.

What We Get Out of This

Visualizations & Outputs:

1. **Time-Series Plot of Mean Vibration**
    - Shows vibration intensity over days/weeks
    - You'll see it mostly flat (healthy), then gradually rising, then sharp spikes (degradation)

- Red dots mark anomalies detected by our model

2. **Distribution Plots**
    - Show how mean, std dev, min, and max are spread across all snapshots
    - Helps understand data range and variability
3. **Anomaly Highlights**
    - A table listing all detected anomalous snapshots with their timestamp
    - Helps maintenance teams know exactly when to investigate

Why This Matters (Business Value)

1. **Predictive Maintenance** – Know problems before they cause failure
2. **Cost Savings** – Avoid unplanned downtime and emergency repairs
3. **Safety** – Prevent accidents from sudden equipment failure
4. **Efficiency** – Schedule maintenance proactively, not reactively
5. **Data-Driven Decisions** – Replace parts based on data, not guesswork

Limitations & Room for Improvement

Current Limitations:

- We only use basic statistics (mean, std, min, max). Richer features from frequency analysis could help.
- Our model ignores time sequences — it treats each snapshot independently. Deep learning could capture "this pattern led to failure last time."
- We don't have labeled failure data — so we're finding statistical outliers, not necessarily failures.
- Model needs tuning for different bearing types.

Future Improvements:

- Add frequency-domain features (analyzing vibration by frequency bands, not just amplitude).
- Train deep learning models (LSTM, autoencoders) that learn temporal patterns.

- Combine with maintenance records to validate predictions against actual failures.
- Test on new bearing types and update model accordingly.
- Integrate real-time monitoring and automated alerts.

How to Use the Code

What You'll Run:

A Jupyter notebook that:

1. Reads all snapshot files from the test folders
2. Extracts the 4 summary statistics per file
3. Creates an organized table of features with timestamps
4. Plots vibration trends over time
5. Runs the Isolation Forest anomaly detector
6. Shows which snapshots are flagged as anomalous
7. Visualizes everything on graphs

Requirements:

- Python 3.7+
- Libraries: pandas, numpy, scikit-learn, matplotlib
- Snapshot files in `archive/1st test/`, `archive/2nd test/`, `archive/3rd test/`

Steps:

1. Open the notebook in Jupyter
2. Run each cell from top to bottom
3. Review the plots and anomaly table
4. Adjust parameters (e.g., contamination rate) if needed
5. Export results for presentation