README

predictive-maintenance

Simple pipeline for anomaly detection on machine data. CPU only. Config-driven and reproducible.

What this does

- Load common datasets (IMS, CWRU, C-MAPSS, AI4I).
- Build rolling-window features from signals.
- Train lightweight models (Isolation Forest, kNN-LOF, One-Class SVM, optional Autoencoder).
- Set alert thresholds to a target false-alarm rate.
- Evaluate with clear metrics and save reports.
- Explain top drivers per alert.

Folder layout

```
predictive-maintenance/
├─ README.md              # what this project is and how to run it
├─ requirements.txt       # Python packages to install
├─ configs/               # settings for data and models
├─ data/                  # your files
│  ├─ raw/                # original files you downloaded
│  ├─ clean/              # cleaned versions
│  └─ features/           # numbers made from the data
├─ src/                   # the Python code
├─ scripts/               # small commands you run (prepare, train, evaluate)
├─ notebooks/             # Jupyter/Colab experiments and charts
├─ results/               # what the runs produce
│  ├─ models/             # saved trained models
│  └─ reports/            # metrics, plots, run logs
└─ docs/                  # short guides and notes
```

Install

```
python -m venv .venv
src .venv/bin/activate        # Windows: .venv\Scripts\activate
pip install -U pip
pip install -r requirements.txt
```

`requirements.txt`:
`numpy pandas scikit-learn scipy pyarrow matplotlib plotly shap joblib pyyaml rich streamlit`

(Torch optional for the autoencoder.)

Quick start (IMS example)

1. Place files under `data/raw/ims/`. See `docs/datasets.md`.
2. Run:

```
python scripts/prep_data.py --config configs/datasets/ims.yaml
python scripts/make_features.py --config configs/datasets/ims.yaml
python scripts/train.py --config configs/models/isolation_forest.yaml
python scripts/threshold.py --target_far 0.1/week
python scripts/evaluate.py --report artifacts/reports/ims_iforest/
```

Score a new CSV:

```
python scripts/score_batch.py --config configs/datasets/ims.yaml   --model
artifacts/models/ims_iforest.joblib   --input data/processed/ims/test.csv
--output artifacts/reports/ims_iforest/scores.csv
```

Configs

- `configs/datasets/*.yaml`: paths, splits, rate, window, overlap.
- `configs/models/*.yaml`: model, params, scaler, features.

Edit configs, not code.

Features

- Time: mean, std, RMS, peak-to-peak, kurtosis, skew, crest factor.
- Optional bands: simple frequency energies.

- Rolling windows with overlap. Robust scaling (median/MAD).

Models

- Isolation Forest
- kNN-LOF
- One-Class SVM
- Autoencoder (optional)

Thresholds

`threshold.py` fits a cutoff to meet a target false-alarm rate.
Outputs the threshold and sensitivity curves.

Metrics

- PR-AUC, ROC-AUC (if labels exist)
- Lead time to failure
- Mean time between false alarms
- Runtime and memory
  Saved under `artifacts/reports/<run>/`.

Explainability

SHAP summaries and top feature drivers per alert.

Reproducibility

- Log configs and git commit to `artifacts/reports/<run>/run.json`.
- Fixed seeds where possible.

Tests

```
pytest -q
```

Dashboard (optional)

```
streamlit run dashboards/app.py
```

Goals

- Laptop-friendly.
- Clear steps: prepare → features → train → threshold → evaluate → score.
- Config first.