

Turbofan Engine Remaining Useful Life (RUL) Prediction - MLOps Project

Overview

This project predicts the Remaining Useful Life (RUL) of turbofan engines using sensor data and advanced deep learning models (CNN-LSTM). It is designed as a full MLOps pipeline, leveraging **MLflow** for experiment tracking, **DVC** for data versioning, and **Docker** for reproducible deployment.

Data Description

The dataset is sourced from NASA's CMAPS repository ([behrad3d/nasa-cmaps](#)). It consists of multiple text files:

- **Training Data:** `train_FD00X.txt`
- **Testing Data:** `test_FD00X.txt`
- **RUL Labels:** `RUL_FD00X.txt`

Data Format

- **Train/Test Files:** Each row represents one engine cycle and contains:
 - Engine ID
 - Cycle number
 - 24 sensor readings (float)
 - Operational settings (float)
 - Example: `1 1 -0.0005 0.0004 100.0 518.67 ... 8145.32`
- **RUL Files:** Each row is an integer representing the RUL for a corresponding engine in the test set.

See sample files: - [RUL_FD003.txt](#) - [train_FD003.txt](#)

Data Processing

Data processing is handled in [src/data_operation.py](#):

1. **Extraction:** Raw data files are loaded and merged into Pandas DataFrames.
2. **Cleaning:** Unnecessary columns are dropped, missing values handled.
3. **Feature Engineering:**
 4. Engine IDs are constructed as `file_name_cycle`.
 5. RUL is calculated for each row in train/test sets.
 6. Label encoding for categorical columns.
7. **Normalization:** Features are normalized using `MinMaxScaler`.
8. **Windowing:** Data is split into sequences for time-series modeling (e.g., 10 cycles per window).
9. **Saving:** Processed data is saved to [data/processed_data/](#).

See [src/data_operation.py](#) for implementation details.

Model Training

Model training is orchestrated by `model_training_pipeline.py` and `training_pipeline_runner.py`:

- Uses a CNN-LSTM architecture (see notebook: [smarter-maintenance-with-cnn-lstm-94-accurate.ipynb](#)).
- Hyperparameters (epochs, batch size, validation split) are configurable via CLI.
- Model metrics (loss, MAE, accuracy) are logged to MLflow.
- Model is saved and versioned via MLflow.

MLOps Stack

MLflow

- Tracks experiments, parameters, metrics, and models.
- See usage in [training_pipeline_runner.py](#).

DVC

- Version controls raw and processed data.
- Use `dvc add data/raw_data/` and `dvc add data/processed_data/` to track data changes.
- Pipelines can be defined in `dvc.yaml`.

Docker

- Dockerfile will be provided for reproducible environment.
- Build and run: `sh docker build -t turbofan-rul . docker run -it turbofan-rul`

Project Structure

```

data/
├── raw_data/
├── processed_data/
├── datasets/
├── src/
│   ├── config.py
│   ├── data_operation.py
│   ├── pipelines.py
│   ├── train_model.py
│   ├── test_model.py
│   └── transform_data.py
├── notebooks/
│   └── smarter-maintenance-with-cnn-lstm-94-accurate.ipynb
├── training_pipeline_runner.py
├── model_training_pipeline.py
├── app.log
├── dvc.yaml
├── Dockerfile
└── README.md
```

Getting Started

1. Clone the repository

```
git clone <repo-url>
cd turbofan_engine_life_prediction
```

2. Setup Environment

```
python -m venv venv
source venv/bin/activate # or .\venv\Scripts\activate on Windows
pip install -r requirements.txt
```

3. Data Versioning

```
dvc pull
```

4. Train Model

```
python training_pipeline_runner.py [epochs] [batch_size] [validation_split]
```

5. Track Experiments

- Start MLflow UI: `sh mlflow ui`
- View results at localhost:5000

6. Build Docker Image

```
docker build -t turbofan-rul .
docker run -it turbofan-rul
```

Contributing

Feel free to open issues or submit pull requests for improvements.

License

MIT License

References

- NASA CMAPS Dataset: <https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>
- MLflow: <https://mlflow.org/>
- DVC: <https://dvc.org/>

- Docker: