Solar Panel Efficiency Prediction

A Machine Learning Approach for Predicting Photovoltaic Performance

---

1. Project Overview

This project focuses on predicting solar panel efficiency using a wide range of environmental and technical parameters. It integrates a sophisticated machine learning pipeline encompassing:

- Advanced feature engineering
- Ensemble learning strategies
- Hyperparameter optimization
- Model explainability using SHAP

The goal is to enable proactive maintenance and optimize energy yield in solar energy infrastructure.

2. Tools and Technologies

Language: Python 3.x

Core Libraries and Frameworks:

- `pandas`, `numpy` – Data manipulation and numerical operations
- `scikit-learn` – ML algorithms and utilities
- `XGBoost`, `LightGBM` – Gradient boosting with GPU support
- `Optuna` – Hyperparameter optimization
- `SHAP` – Model interpretability
- `matplotlib`, `seaborn` – Data visualization

3. Feature Engineering

Implemented in: *`src/feature_engineering.py`*

a) Data Cleaning

- Handle missing values using median (numerics) and mode (categoricals)
- Convert string numerics to appropriate types
- Treat outliers using IQR and Z-score

b) Advanced Feature Creation

Power-Based Features:

- $power\_output = voltage \times current$
- $power\_per\_area = \frac{power\_output}{irradiance}$

Temperature Efficiency Features:

- $temp\_efficiency\_ratio = \frac{module\_temperature}{temperature + 273.15}$
- $temp\_difference = module\_temperature - temperature$

Environmental Interactions:

- $irradiance \times module\_temperature$
- $humidity \times temperature$
- $\frac{cloud\_coverage}{irradiance}$

Degradation Indicators:

- $\frac{panel\_age}{maintenance\_count}$
- $soiling\_ratio \times panel\_age$

Cooling Effect:

- $\frac{wind\_speed}{module\_temperature}$

Efficiency Indicators:

- $irradiance \times (1 - soiling\_ratio) \times (1 - \frac{cloud\_coverage}{100})$

Polynomial Features:

- $temperature^2$, $irradiance^2$, $humidity^2$

4. Model Architecture

Implemented in: `src/model_training.py`

Base Models:

- XGBoost (GPU)
- LightGBM (GPU)
- Random Forest

Meta-Learner:

- Ridge Regression

Optimization Strategy:

- Hyperparameter tuning using Optuna
- 5-fold cross-validation
- GPU-accelerated training

5. Training Workflow

1. Data preprocessing and feature engineering
2. Hyperparameter optimization
3. 5-fold cross-validation
4. Ensemble model training
5. SHAP-based feature importance analysis
6. Model evaluation and predictions

6. Source Files

- `src/feature_engineering.py` – Feature creation and preprocessing
- `src/model_training.py` – Training pipeline and optimization
- `src/data_exploration.py` – Exploratory analysis and visualization
- `requirements.txt` – Project dependencies
- `README.md` – Project overview and instructions

7. Performance Metrics

- Root Mean Squared Error (RMSE)
- R^2 Score (Coefficient of Determination)
- Custom Score: $100 \times (1 - RMSE)$

8. Key Features

- GPU-accelerated training
- Advanced feature engineering
- Automated hyperparameter tuning
- SHAP-based interpretability
- Robust cross-validation
- Stacking ensemble modeling

9. Future Enhancements

- Implement advanced feature selection techniques (e.g., Boruta)
- Explore deep learning models (e.g., TabNet, DNN)
- Enhance ensemble methods (e.g., blending, voting)
- Create an automated retraining pipeline
- Develop real-time prediction APIs