IMPERIAL

Imperial College London
Department of Civil and Environmental Engineering

CIVE70111

Machine Learning for Civil Engineering

Coursework 2025

Professor Panagiotis Angeloudis

Transport Systems & Logistics Laboratory

Academic year 2025-26

Last revision: November 8, 2025 at 14:15

- **Submissions on Friday 12th of December @ 12:00**
- **Presentations on Friday 12th @ afternoon**

1. Overview

Efficient management of solar energy generation is a critical aspect of sustainable power supply, particularly in regions with abundant solar resources. This coursework aims to help improve solar energy production forecasting, identify faulty equipment, and optimise the performance of solar power plants.

You are working with data collected over a 34-day period from two solar power plants. The dataset includes generation data at the inverter level and sensor readings at the plant level. The inverter data capture the electricity generated by each inverter, while the sensor data represents aggregated environmental measurements relevant to the entire plant.

Expected Workload: This coursework is designed to require approximately 40–50 hours of work in total across the entire team. This includes time for data exploration, model development, analysis, documentation, and preparation of your final report and presentation.

A detailed description of the datasets is provided in Section 7.

2. Given Data

1. `Plant_1_Generation_Data.csv`, `Plant_2_Generation_Data.csv`: Power generation at the inverter level for each plant.
2. `Plant_1_Weather_Sensor_Data.csv`, `Plant_2_Weather_Sensor_Data.csv`: Sensor readings collected at a plant level for each plant.

3. Coursework Tasks

Task 1: Problem Formulation *[5 marks]*

Before beginning your analysis, formally define the machine learning problem using the T-P-E framework introduced in Lecture 01.

- **Task (T)**: Clearly define what your models aim to predict or classify. For each of the main problems (power forecasting, operating condition classification, temporal forecasting), specify what is being predicted (target variable), the ML problem type (regression, binary classification, multi-class classification, sequence prediction), and the prediction horizon (real-time, 1-hour ahead, multi-step).
- **Performance (P)**: Define how you will measure success and justify your choice of metrics in terms of business objectives (e.g., minimizing revenue loss from poor forecasts), data characteristics (e.g., class imbalance, rare events), and engineering requirements (e.g., acceptable error tolerances).

- **Experience (E)**: Describe the data available for learning: which features are available and most relevant; the temporal structure and coverage; data quality issues or limitations; and how you will split data for training, validation, and testing while respecting temporal order.

Deliverable: A concise problem formulation section (1-2 pages) in your report explicitly addressing T, P, and E for each major task. This framework should guide all subsequent modeling decisions.

Task 2: Data Exploration, Analysis, and Preprocessing *[15 marks]*

Conduct a comprehensive exploratory analysis of both the power generation and sensor data, derive insights to inform modeling decisions, and prepare the data for machine learning.

Data Quality and Integration

- **Data Quality Assessment**: Identify and document any data quality issues, such as missing values, inconsistencies, or anomalies.
- **Data Handling**: Address missing data and check for outliers.
- **Integration**: Combine the power generation and sensor datasets into a unified dataset, aligning sources correctly (e.g., by timestamp) to enable joint analysis.

Exploratory Data Analysis

- **Statistical Summary**: Compute summary statistics (mean, median, standard deviation, min, max) for both power and sensor readings.
- **Visualizations**: Create visualizations (e.g., histograms, scatter plots, time series plots, correlation matrices) to illustrate distributions, trends, and relationships in the data.
- **Trend Analysis**: Analyse time-based trends in power output, such as daily or weekly cycles, and seasonal patterns.
- **Environmental Impact**: Investigate how changes in conditions (e.g., temperature, irradiance) influence power output through correlation analysis and visualizations.
- **Pattern Identification**: Identify patterns that may be useful for future modelling, such as peak generation times, sudden drops in output, or relationships between inverter performance and weather conditions.

Feature Engineering and Preprocessing

- **Feature Scaling**: Implement and compare scaling methods (standardization and/or min-max normalization). Identify features with very different scales, explain why scaling matters for gradient descent convergence, and document the impact on model training.
- **Feature Selection**: Based on your exploratory analysis, identify which features are most relevant for the modeling tasks ahead.

Hint: Understanding the temporal and environmental relationships in the data will support better forecasting models. Pay attention to correlations between power generation and sensor data, such as irradiance levels and power output, and the range and distribution of different features.

Task 3: Power Forecasting Model *[20 marks]*

Develop models to accurately forecast the power output for each inverter on future days, using environmental conditions such as solar irradiance and temperature. This will help anticipate power production capabilities and optimise the operation of both solar power plants.

- **Model Development**: Implement **at least 3 different regression model types** (e.g., Linear Regression, Ridge/Lasso Regression, Decision Trees, Random Forests, Neural Networks, etc.) to predict power output at the inverter level (DC Power Output) for each plant using environmental conditions. You may also experiment with ensemble methods.
- **Model Evaluation**: Select and apply appropriate metrics, such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), to evaluate the model's accuracy for each plant individually.
- **Model Comparison**: Compare the performance of your different model types and different sets of input features. Build models for predicting both DC power and AC power output. Evaluate which model performs better and discuss the reasons why this might be the case.
- **Bias-Variance Analysis**: Plot learning curves showing training and validation error versus dataset size. Diagnose whether your models suffer from high bias (underfitting) or high variance (overfitting). Demonstrate how regularisation affects the bias-variance trade-off.
- **Training Visualization**: Plot the cost function (loss) evolution over training iterations/epochs for at least one model. Identify convergence points and compare convergence rates across different model architectures or learning rates.

Task 4: Classification of Operating Conditions *[20 marks]*

Reliable solar power generation requires maintaining inverters within optimal operating conditions to maximise efficiency and minimise energy losses. In this task, you will develop a classification model to predict the operating condition of inverters based on environmental and operational data.

Important: This dataset exhibits significant class imbalance (approximately 23% Optimal vs. 77% Suboptimal conditions). You must address this imbalance in your analysis and model development.

- **Data Preparation**: Prepare the dataset for classification by selecting relevant features that might affect an inverter's operational condition. These features could include ambient temperature, solar irradiance, inverter performance metrics (e.g., DC and AC power),

and any other relevant data from the sensor readings. Analyze the class distribution and discuss the implications of class imbalance.

- **Model Development**: Develop and train classification models (e.g., Logistic Regression, Decision Trees, Support Vector Machines, Neural Networks, etc.) to predict the operational condition ("Optimal" or "Suboptimal") of each inverter. Implement at least one technique to handle class imbalance, such as class weighting, SMOTE (Synthetic Minority Over-sampling Technique), or adjusting decision thresholds.
- **Model Evaluation**: Evaluate the classification models using appropriate metrics. *Due to the class imbalance, prioritize F1 Score, Precision, and Recall over Accuracy.* Include a confusion matrix to understand model performance on each class. Use cross-validation or train-test splits to ensure reliable evaluation.
- **Feature Importance**: Analyse which features contribute most significantly to predicting the operational condition of inverters. Discuss the implications of these findings for plant maintenance and operational planning.

Task 5: Model Interpretation *[10 marks]*

Understanding and interpreting machine learning models is crucial for making informed decisions in real-world applications. In this task, you will investigate the inner workings of your models and discuss how they can be applied to improve solar plant operations.

- **Model Interpretation**: Use **at least one** interpretation technique such as SHAP (SHapley Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations), or feature importance plots to interpret your forecasting and classification models. Explain how each feature influences the predictions made by your models.

  Note: Some interpretation methods (particularly SHAP) can be computationally expensive on large datasets. For datasets with hundreds of thousands of rows, it is acceptable and recommended to use a representative sample (e.g., 10% of the data) for interpretation analysis to reduce computation time while still gaining meaningful insights.
- **Insights for Operation**: Based on your model interpretations, discuss specific insights that could be used to optimise the operation of solar plants. For example, identify conditions under which inverters are likely to underperform and suggest proactive maintenance strategies.
- **Limitations and Challenges**: Discuss the limitations of your models and any challenges you encountered during development and evaluation. Consider aspects such as data quality, model assumptions, and potential biases.
- **Recommendations**: Provide recommendations for improving model performance and applying these models in real-world settings. Consider factors such as data collection improvements, alternative modelling approaches, and deployment considerations.

Task 6: Temporal Forecasting *[15 marks]*

Implement a simple **LSTM-based** time-series forecasting model for short-horizon (e.g., 1-hour ahead) power predictions.

- **Sequence Preparation**: Create sliding window sequences from your time-series data using a lookback window (e.g., past 24 hours) to predict the next hour of power output. Ensure temporal order is preserved in all splits.
- **LSTM Implementation**: Build a *single-layer* LSTM with a small hidden size (e.g., 16–64 units) followed by a Dense output layer. Keep the architecture minimal; advanced RNN variants are optional.
- **Baselines**: Compare against a simple persistence or moving-average baseline. A comparison to your best Task 3 model on the same horizon is optional.
- **Evaluation**: Single-horizon evaluation (1-hour ahead) is **required**. Optionally, you may include an additional horizon (e.g., 3 hours ahead) and briefly discuss error growth with horizon.
- **Design Justification**: Briefly justify your lookback window, hidden size, and your loss/optimizer selection.
- **Visualization**: Plot representative days showing actual vs. predicted power. Include at least one day with stable conditions and one with variable/cloudy conditions.

Computational Note: LSTM training can be more computationally intensive than classical models. Using a representative subset or cloud notebooks with GPU is acceptable. Methodology and clarity are prioritized over heavy tuning.

Hint: Respect temporal order in train/validation/test splits to avoid data leakage.

Task 7: Interactive Model Dashboard *[10 marks]*

Develop a Streamlit-based interactive dashboard that allows users to interact with your trained models in real-time. This task allows you to develop some of the basic skills required to deploy machine learning models in a practical, user-friendly interface.

Note: You may use GitHub Copilot or other AI tools to accelerate development, but ensure you understand and can explain all code. Document your use of AI tools (and examples of key prompts that you used) in your README and report.

- **Application Features**: Create an interactive dashboard that lets users input environmental conditions (temperature, irradiation, etc.), select the plant and model type, visualize predictions in real-time, compare outputs across different models, and view performance metrics and feature importance plots.
- **Implementation**: Use Streamlit to build the application. Your dashboard should load your trained models (from pickle or saved model files), provide interactive widgets for user inputs (sliders, dropdowns, number inputs), process inputs with appropriate feature

scaling and encoding, generate predictions, and display visualizations using Plotly or Matplotlib.

- **Code Organization**: Place all Streamlit application code in the `src/` folder of your repository:
  - `src/app.py` - Main Streamlit application
  - `src/utils.py` - Helper functions for model loading and prediction
  - `src/config.py` - Configuration settings (optional)
  - `src/models/` - Saved model files
- **Documentation**: Include a `src/README.md` with installation instructions (`pip install streamlit`), how to run the application locally (`streamlit run src/app.py`), screenshots demonstrating the interface, and a brief description of the features and design decisions.

Task 8: Critical Analysis and Reflection *[5 marks]*

Reflect on the entire coursework process and provide a critical analysis of your approach and findings.

- **Approach Evaluation**: Critically evaluate the methods and models you employed throughout this coursework. Discuss what worked well and what could have been improved.
- **Comparative Analysis**: Compare your results across different tasks (e.g., regression vs. classification, different model types). What patterns emerge? How do different approaches complement each other?
- **Learning Outcomes**: Reflect on what you learned through this coursework. How has your understanding of machine learning applications in civil engineering evolved?
- **Future Directions**: Propose potential extensions or improvements to this analysis. What additional data or methods could provide further insights into solar plant operations?

4. Submission Requirements

Repository Structure

You must use the **Template** available at:

https://github.com/tsl-imperial/project-template

Create a **private** GitHub repository from this template and organize your work according to the provided structure. The template includes:

Required Deliverables

Your GitHub repository must include the following.

- **README.md**: Complete the template (README-EXAMPLE.md) with project title and student details, an executive summary of findings, clear instructions to run your code, and an acknowledgment of any AI tools used (if applicable).
- **Report Notebook**: Provide a comprehensive Jupyter notebook (`notebooks/00-report.ipynb`) that covers all tasks with clear sections, includes narrative markdown explaining your approach and findings, embeds key visualizations and results, and contains proper citations and references. Export to PDF if desired, but submit the `.ipynb`.
- **Analysis Notebooks**: Include task-specific notebooks in `notebooks/` with descriptive filenames and executable cells in order. The report notebook should reference and summarize these. Suggested filenames:
  - `01-data-exploration.ipynb`
  - `02-power-analysis.ipynb`
  - `04-forecasting-model.ipynb`
- **Source Code**: Place reusable Python functions and modules in `src/` if you extract common functionality from notebooks.
- **Data**: Store datasets in `data/`. For files larger than 50MB, add `data/README.md` with download instructions.
- **Trained Models**: Save trained models (e.g., `.pkl`, `.h5`, `.pt`) in `models/` (create if needed). If models are large, provide instructions to regenerate them.
- **Requirements**: Update `requirements.txt` with all Python packages needed to run your code.

Submission Instructions

1. Ensure your repository is **private** and properly organized
2. Verify all notebooks run successfully from top to bottom
3. Complete the pre-submission checklist in `docs/submission-guidelines.md`
4. Submit your GitHub repository URL via Blackboard
5. **Grant access** to the course instructors (usernames will be provided on Blackboard)
6. **Prepare for in-class presentation**: Teams will be asked to showcase their work in class on the afternoon of 12th December

In-Class Presentation and Demonstration

Teams will be required to present and demonstrate their work in class on Friday 12th December in the afternoon, following the submission deadline.

- **Duration**: Approximately 10 minutes per team
- **Format**: Informal demonstration-style presentation (slides optional but not required)
- **Content**: Focus on your key findings, model performance, and interesting insights from your analysis

The presentation does not carry separate marks but provides an opportunity to demonstrate your understanding of the submitted work. Your ability to articulate and defend technical choices, communicate concepts clearly, and answer questions may positively influence your assessment within the Communication and Critical Thinking criteria.

Important Notes

- Your repository **must be private** to comply with academic integrity requirements
- Ensure all team members (if applicable) are added as collaborators
- Git commit history will be used to verify individual contributions
- See `docs/academic-integrity.md` for guidelines on AI tool usage and citations

5. Assessment Criteria

Your coursework will be assessed based on:

- **Technical Competence** (40%): Correctness and appropriateness of methods, code quality, model implementation, and proper use of machine learning techniques covered in the course.
- **Analysis and Interpretation** (30%): Depth of analysis, quality of insights, interpretation of results, bias-variance analysis, and understanding of model behavior.
- **Communication** (20%): Clarity of writing, quality of visualisations, organisation of report, and effective presentation of technical concepts. Your in-class presentation provides an opportunity to demonstrate communication skills and may positively influence this component.
- **Critical Thinking** (10%): Demonstration of independent thought, awareness of limitations, quality of recommendations, and proper problem formulation using the T-P-E framework. Your ability to defend technical choices and answer questions during the presentation may positively influence this component.

6. Academic Integrity

You may discuss general approaches with your peers, but all submitted work must be your own. Use of AI tools (e.g., ChatGPT, GitHub Copilot) is permitted for assistance with coding and concept clarification, but you must acknowledge their use in your report and take full responsibility for all submitted work. Plagiarism will be dealt with according to Imperial College regulations.

7. Data Description

This section provides descriptions of the fields present in two datasets: **Generation Data** and **Weather Sensor Data**. Each dataset records observations related to solar plant operations at 15-minute intervals. Below, each column (field) in the datasets is defined in detail.

Generation Data

This dataset contains information on power generation recorded at a 15-minute interval for each inverter at the solar plant.

- **DATE_TIME**: The date and time of each observation, recorded at 15-minute intervals.
- **PLANT_ID**: A unique identifier for the plant. This ID will remain the same across the entire file as it represents a single plant.
- **SOURCE_KEY**: The unique identifier for the inverter in the plant. Each inverter has its own ID, allowing for tracking of power generation at the inverter level.
- **DC_POWER**: The amount of direct current (DC) power generated by the inverter within the 15-minute interval, measured in kilowatts (kW).
- **AC_POWER**: The amount of alternating current (AC) power generated by the inverter within the 15-minute interval, measured in kilowatts (kW).
- **DAILY_YIELD**: A cumulative sum of the power generated by the inverter for the current day, up to the specified point in time. This resets each day.
- **TOTAL_YIELD**: The cumulative power generated by the inverter since its installation, up to the specified point in time.
- **Operating_Condition**: Indicates the operational status of the inverter at the time of observation. It can be either "Optimal" or "Suboptimal." An "Optimal" condition means the inverter is functioning efficiently within expected parameters, while "Suboptimal" suggests underperformance potentially due to equipment issues or environmental factors impacting efficiency.

Weather Sensor Data

This dataset provides weather-related information relevant to solar energy production, recorded every 15 minutes.

- **DATE_TIME**: The date and time of each observation, recorded at 15-minute intervals.
- **PLANT_ID**: A unique identifier for the plant, consistent across the file.
- **SOURCE_KEY**: The unique identifier for the sensor panel used for recording the environmental conditions. Since there is only one sensor panel per plant, this will remain consistent across the entire file.
- **AMBIENT_TEMPERATURE**: The ambient temperature at the plant.

- **MODULE_TEMPERATURE**: The temperature reading of a solar panel module attached to the sensor panel.
- **IRRADIATION**: The amount of solar irradiation received at the plant during the 15-minute interval, a critical factor for power generation.

Notes

- Both datasets use **DATE_TIME** for aligning observations recorded at the same 15-minute intervals, allowing for combined analysis of generation and weather conditions.
- **PLANT_ID** remains the same for each file as they represent data from a single plant.