Ferma: Fermentation Monitoring & Yield Predictor

Dataset link we use to web scrapping the datas and convert to CSV:

Bulk data (OD_{600nm}; substrate concentration; pH) recorded in the CDC biofilm reactors

Ofiteru, Irina (2024). Bulk data (OD600nm; substrate concentration; pH) recorded in the CDC biofilm reactors. Newcastle University. Dataset. https://doi.org/10.25405/data.ncl.24851244.v1

- **Problem Validation (20 pts)**:
  Biomanufacturing labs industry struggle with contamination, inconsistent pH, variable yields. Directly connected to national biomanufacturing priorities. Our sfotware platform helps evaluate qualities of all the bio samples so it can be ready for

  NSCEB_WP_Biomanufacturing
  (Stages I–IV: feedstock → outputs → end products).

- **Functional Prototype (25 pts)**:
  You can build a **working MVP** fast:
  - Upload fermentation time-series data (public datasets exist)
  - Model predicts **yield**, **contamination risk**, **optimal harvest time**
  - UI shows trend + risk score + recommended action

1. Tech Stack

To move fastest, treat this as a “single-page app” with a thin separation:

- Language: Python
- Core framework: **Streamlit** (front end + backend in one file, but logically separated)
- Libraries:
    - Data: `pandas`, `numpy`
    - ML: `scikit-learn`
    - Plots: built-in Streamlit charts or `plotly.express`
- Data format: CSV upload

Run-Time Flow (End-to-End)

1. User opens link .
2. User uploads CSV or uses sample data.
3. Frontend sends data to preprocessing function.
4. Backend:
    - Cleans data, computes features.
    - Runs yield model → predicted final yield.
    - Runs risk logic → batch risk score.
    - Computes optimal harvest time.
    - Generates chart-ready data and text recommendations.
5. Frontend:
    - Renders KPI cards.
    - Draws charts with vertical line for harvest time.
    - Shows diagnostics table and analyzed features

    - Shows recommendation bullet points.

`realistic_bioreactor_dataset.csv`

- Synthetic-but-realistic fermentation bioreactor time-series
    - Process conditions: `temp_c`, `pH`, `od600`, `substrate_conc`, `dissolved_o2`
    - Timeline: `timestamp`, `time_hours`
    - Output: `product_conc` (what you model and monitor)

2. Full Back-end (do via Goggle Colab)

How we build our model

Imports
Brings in NumPy, pandas, scikit-learn, Plotly.

Load data

- Reads `realistic_bioreactor_dataset.csv`
- Sorts by time, forward-fills missing values
- Creates `time_hours`

Feature engineering

- Slopes (`prod_slope`, `od_slope`)
- Rolling means and stds
- Deviations from setpoints (temp & pH)

- Lag features (previous OD, previous product)

ML model (**this is the main ML part**):

Does full Feature engineering:

ML Yield Prediction: Trains RandomForest regression

Compute risk, harvest time,
Generate recommendations

Dashboard visualization: Shows KPIs + Displays the Plotly dashboard inline

- Selects `FEATURE_COLS` and target (`product_conc`)
- Drops rows with NaNs
- Trains `RandomForestRegressor`
- Evaluates MAE and R^2
- Creates `product_pred` for entire time series

Final yield prediction

- Takes last predicted product and last actual product

Risk score

- Computes mean deviations from ideal temp and pH
- Counts sharp OD drops
- Estimates DO slope
- Aggregates into a penalty and converts to `risk_score`

Harvest time

- Uses plateau detection based on smoothed `prod_slope` to choose harvest time

Recommendations (**this is where recommendations live**)

- List `recs` is built based on temp deviation, pH deviation, OD drops, risk score
- If all good, adds "Batch appears stable and healthy."

Printing summary + dashboard

- Prints all metrics to console
- Builds Plotly figure and shows it

Additional bioprocess analysis

- `mu`, `doubling_time`, `growth_phase`, `oxygen_limitation`, `substrate_empty`, `productivity`, `stability_index`, `time_peak_rate`

Feature	Library / Method
Yield Prediction	**Scikit-Learn (RandomForestRegressor)**
Risk Score	**Python logic + anomaly detection**
Harvest Time	**Slope-based plateau detection**
App	**Streamlit**
Charts	**Plotly**