ML-ENHANCED INDOOR TEMPERATURE PREDICTIONS BASED ON WEATHER CONDITIONS

construction robotics | international M.Sc.programme

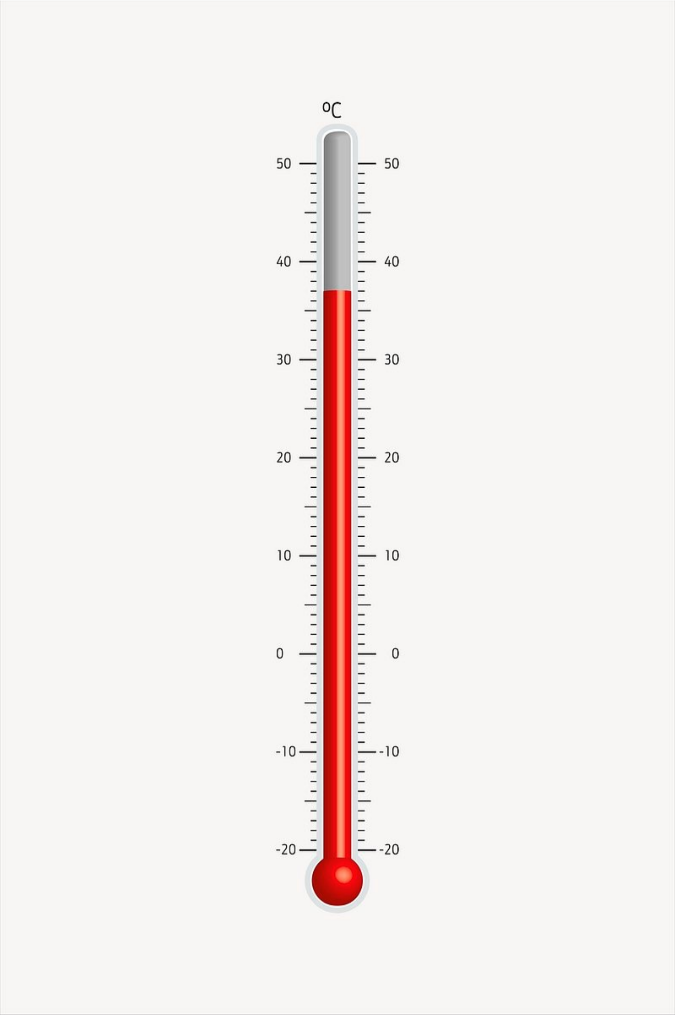
HTOO INZALI | ROBIN BERWEILER
04 AUG 2023

Content

Introduction

Background and Problem Statement :

- Indoor temperature plays a crucial role in our daily lives, directly influencing our comfort, well-being, and productivity.
- Uncomfortable indoor temperatures can lead to decreased productivity, health issues, and increased energy consumption.
- Consequently, there is a **growing need for accurate indoor temperature predictions** to optimize energy management and create comfortable indoor environments.

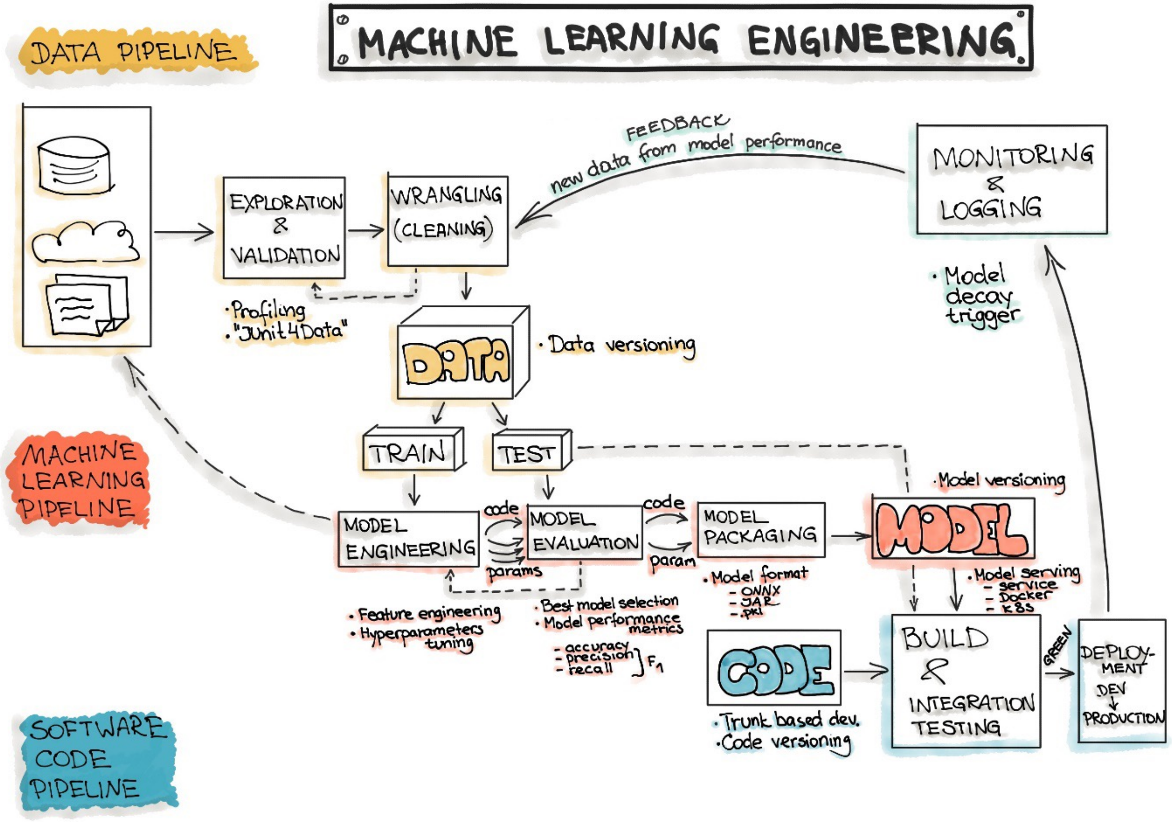


Introduction

Objectives and Scope

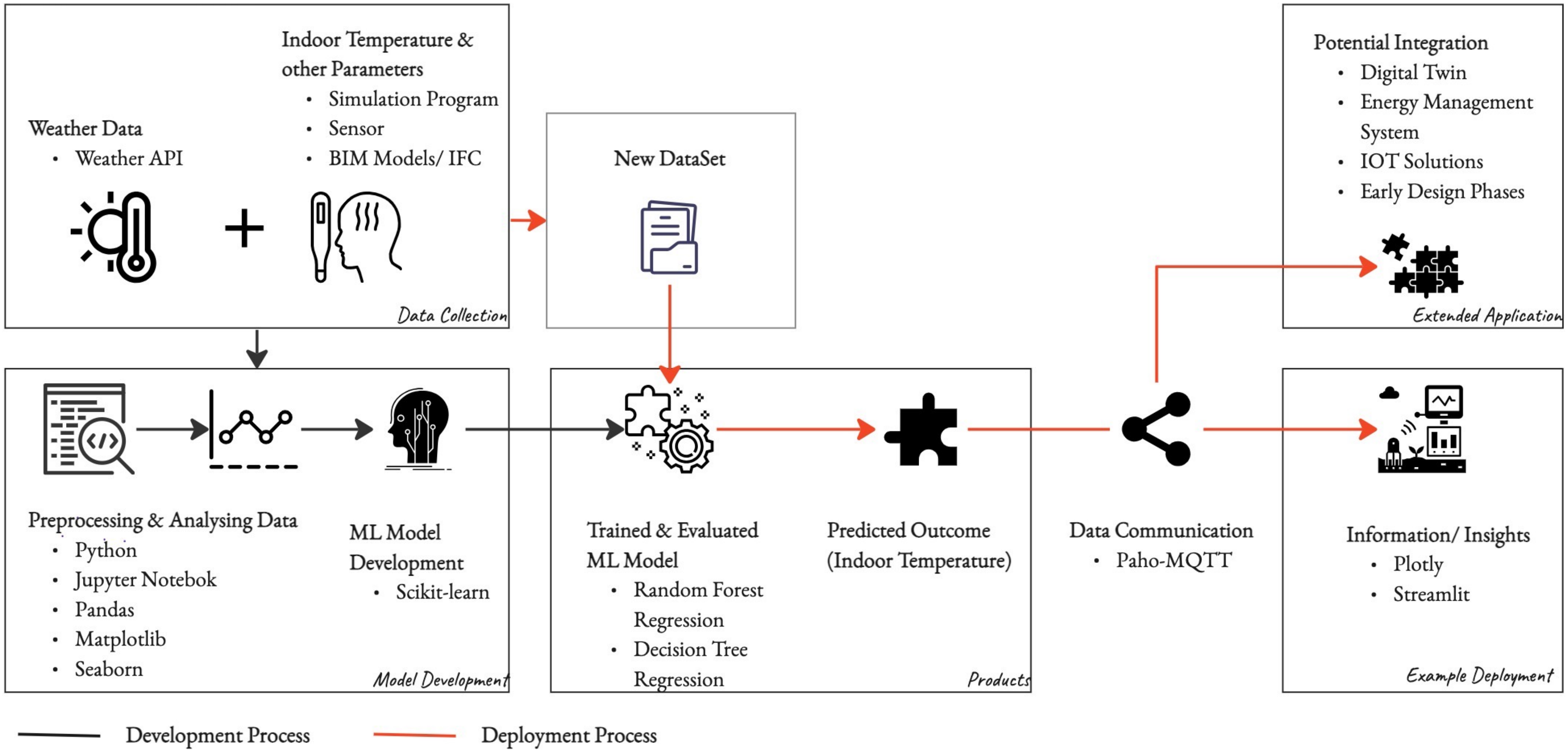
- To develop a **machine learning model for indoor temperature prediction based on the weather conditions** and **MQTT integration** for real-time data communication.

- The scope includes
  - Data collection
  - Data preprocessing
  - Model development,
  - MQTT integration, and
  - Optional user interface development



Development Process

Workflow



Development Process

Data Preprocessing

Getting general insight of the original dataset

In [7]:
```
df.shape
```
Out[7]:
```
(3679952, 13)
```

In [8]:
```
df.describe()
```
Out[8]:

	time	importWeatherModified.min	importWeatherModified.h	importWeatherModified.dm	importWeatherModified.dy	importWeatherModified.FF	impc
count	3.679952e+06	3.679952e+06	3.679952e+06	3.679952e+06	3.679952e+06	3.679952e+06	
mean	1.576800e+07	2.949792e+01	1.149976e+01	1.572055e+01	1.830002e+02	3.180580e+00	
std	9.103665e+06	1.731893e+01	6.921565e+00	8.796301e+00	1.053663e+02	2.369005e+00	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	
25%	7.883988e+06	1.400000e+01	6.000000e+00	8.000000e+00	9.200000e+01	1.400000e+00	
50%	1.576800e+07	2.900000e+01	1.100000e+01	1.600000e+01	1.830000e+02	2.600000e+00	
75%	2.365200e+07	4.400000e+01	1.700000e+01	2.300000e+01	2.740000e+02	4.400000e+00	
max	3.153600e+07	5.900000e+01	2.400000e+01	3.100000e+01	3.650000e+02	1.610000e+01	

In [9]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3679952 entries, 0 to 3679951
Data columns (total 13 columns):
 #   Column                                        Dtype
---  ------                                        -----
 0   time                                          float64
 1   importWeatherModified.min                     int64
 2   importWeatherModified.h                       int64
 3   importWeatherModified.dm                      int64
 4   importWeatherModified.dy                      int64
 5   importWeatherModified.FF                      float64
 6   importWeatherModified.S_height                float64
 7   window_Status_hour_of_day_Winter.Window_Status float64
 8   importWeatherModified.port_a.T                float64
 9   port_a.T                                      float64
 10  window_Status_hour_of_day_Spring.Window_Status float64
 11  window_Status_hour_of_day_Summer.Window_Status float64
 12  window_Status_hour_of_day_Autumn.Window_Status float64
dtypes: float64(9), int64(4)
memory usage: 365.0 MB
```

In [10]:
```
len(df)
```
Out[10]:
```
3679952
```

In [100]:
```
df
```
Out[100]:

	windspeed	sunHeight	weatherTemp	roomTemp	windowStatus	month	season
0.0	0.0	0.0	288.15	289.150000	1	Jan	Winter
0.0	0.0	0.0	288.15	289.150000	1	Jan	Winter
0.0	1.7	-68.4	284.05	289.150000	1	Jan	Winter
10.0	1.7	-68.4	284.05	289.132398	1	Jan	Winter
20.0	1.7	-68.4	284.05	289.115464	1	Jan	Winter
...	...	...	...	...	...	...	...
31536000.0	0.3	-68.5	276.05	283.217353	1	Dec	Winter
31536000.0	0.3	-68.5	276.05	283.215471	1	Dec	Winter
31536000.0	0.3	-68.5	276.05	283.213644	1	Dec	Winter
31536000.0	0.3	-68.5	276.05	283.211865	1	Dec	Winter
31536000.0	0.3	-68.5	276.05	283.210134	1	Dec	Winter

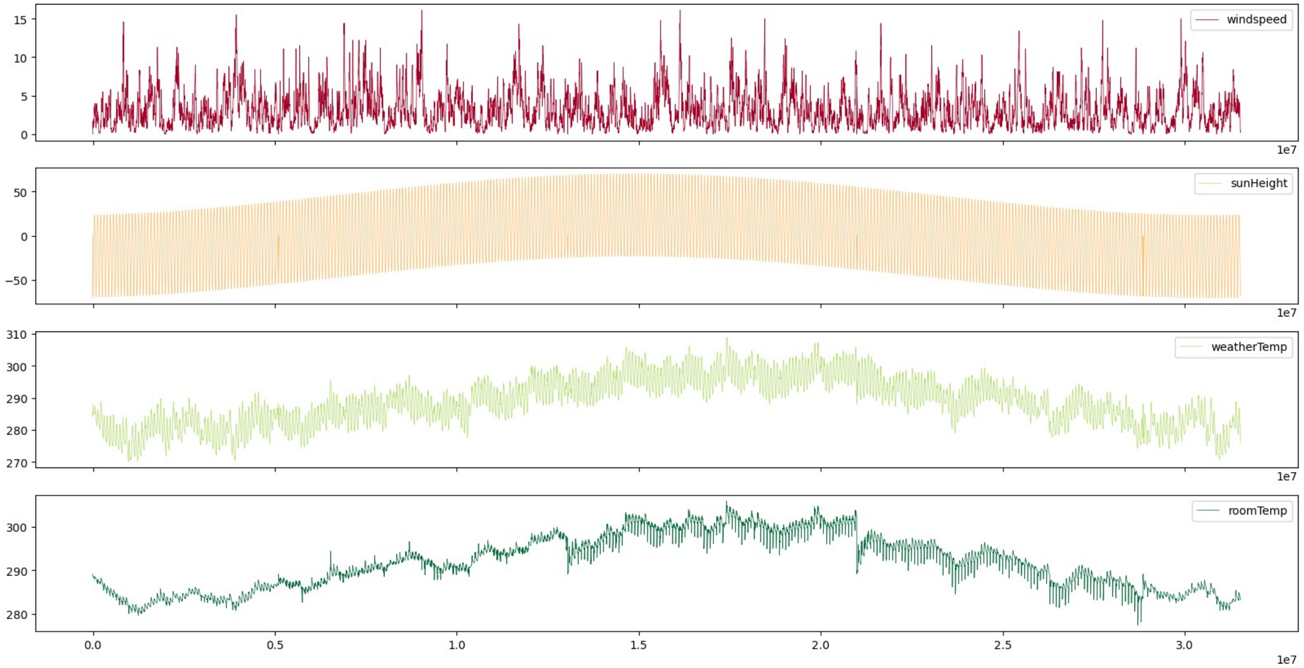
3679952 rows × 7 columns

In [101]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 3679952 entries, 0.0 to 31536000.0
Data columns (total 7 columns):
 #   Column        Dtype
---  ------        -----
 0   windspeed     float64
 1   sunHeight     float64
 2   weatherTemp   float64
 3   roomTemp      float64
 4   windowStatus  int64
 5   month         object
 6   season        object
dtypes: float64(4), int64(1), object(2)
memory usage: 224.6+ MB
```

Development Process

Data Analysis

	windspeed	sunHeight	weatherTemp	roomTemp	windowStatus
windspeed	1.000000	0.160677	0.097283	0.021785	0.101831
sunHeight	0.160677	1.000000	0.540736	0.336121	0.616676
weatherTemp	0.097283	0.540736	1.000000	0.901332	0.409763
roomTemp	0.021785	0.336121	0.901332	1.000000	0.213515
windowStatus	0.101831	0.616676	0.409763	0.213515	1.000000



Highest correlation of 0.90 between weather temperature and indoor temperature.

Model Development

- Based on the prior correlation result between weather temperature and indoor temperature, we tried Simple Linear Regression first.
- Due to unsatisfactory accuracy, we tried other regression models.

Actual	Linear Regression	K-Neighbors	Random Forest	Decision Tree	Ridge
295.851333	295.040644	295.852476	295.852473	295.852476	295.040644
300.972153	297.278215	300.972470	300.972851	300.972787	297.278214
283.531920	285.408346	283.531647	283.531641	283.531647	285.408346
295.265548	294.626400	293.416171	294.017441	294.005441	294.626400
300.644504	300.094993	300.645105	300.645133	300.645105	300.094993
294.333830	291.491844	294.335277	294.335502	294.335501	291.491843
282.848831	285.450205	282.848948	282.848949	282.848948	285.450205
293.419365	292.867465	293.419349	293.419352	293.419350	292.867467
299.594400	295.278716	299.598846	299.598453	299.598442	295.278715
284.760050	286.718683	284.759941	284.759908	284.759941	286.718683

Model Evaluation

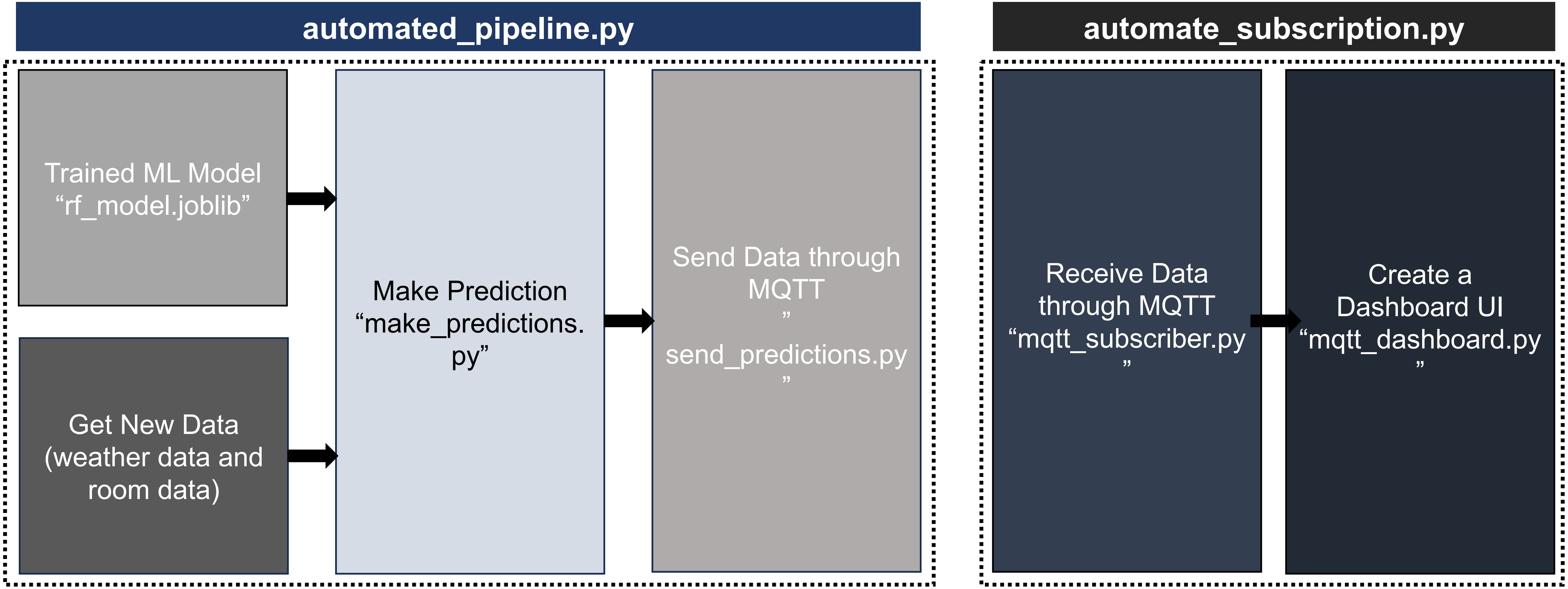
- Mean Absolute Error(**MAE**) evaluates **the prediction error**. The **lower** the value, the better.
- **R-squared** indicates the **proportion of variance** in indoor temperature. The **higher**, the better.

	Model	MSE	R-squared
0	Linear	5.639336	0.854940
1	K-Neighbors	0.100037	0.997427
2	Random Forest	0.061396	0.998421
3	Decision Tree	0.061128	0.998428
4	Ridge	5.639336	0.854940

The Random Forest and Decision Tree with the least MAE and the highest R-squared.
Random Forest Model was selected to use for the later.

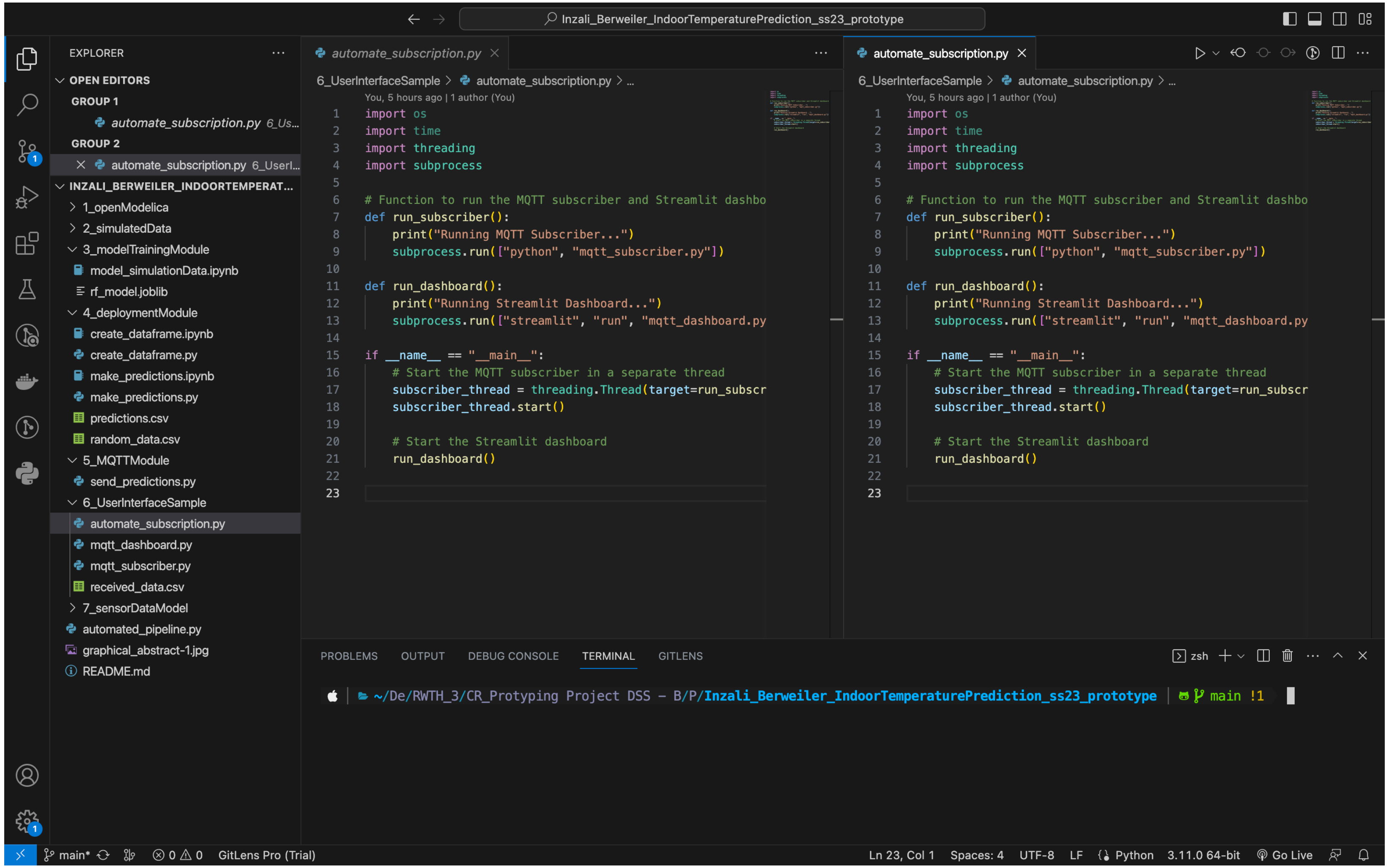
Development Process – Automating the Pipeline

MQTT - Publish **MQTT - Subscribe**



Development Process

Automated Pipeline Script



Product Components

ML Model Development Module

```
Inzali_Berweiler_IndoorTemperaturePrediction_ss23_prototype
    /
|--- 1_openModelica
|
|--- 2_simulatedData
|     |--- autumn_SeptOctNov.csv
|     |--- spring_MarAprMay.csv
|     |--- summer_JunJulAug.csv
|     |--- winter_Dec.csv
|     |--- winter_JanFeb.csv
|
|--- 3_modelTrainingModule
|     |--- model_simulationData.ipynb
|     |--- rf_model.joblib
|
|--- 4_deploymentModule
|     |--- create_dataframe.py
|     |--- make_predictions.py
|     |--- predictions.csv
|     |--- random_data.csv
|
|--- 5_MQTTModule
|     |--- send_predictions.py
|
|--- 6_UserInterfaceSample
|     |--- automate_subscription.py
|     |--- mqtt_dashboard.py
|     |--- mqtt_subscriber.py
|     |--- received_data.csv
|
|--- automated_pipeline.py
|
|--- README.md
```

User Interface - Dashboard

localhost:8501

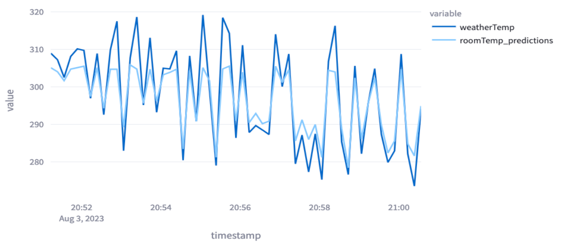
Indoor Temperature Predictions Dashboard

Raw Incoming Data

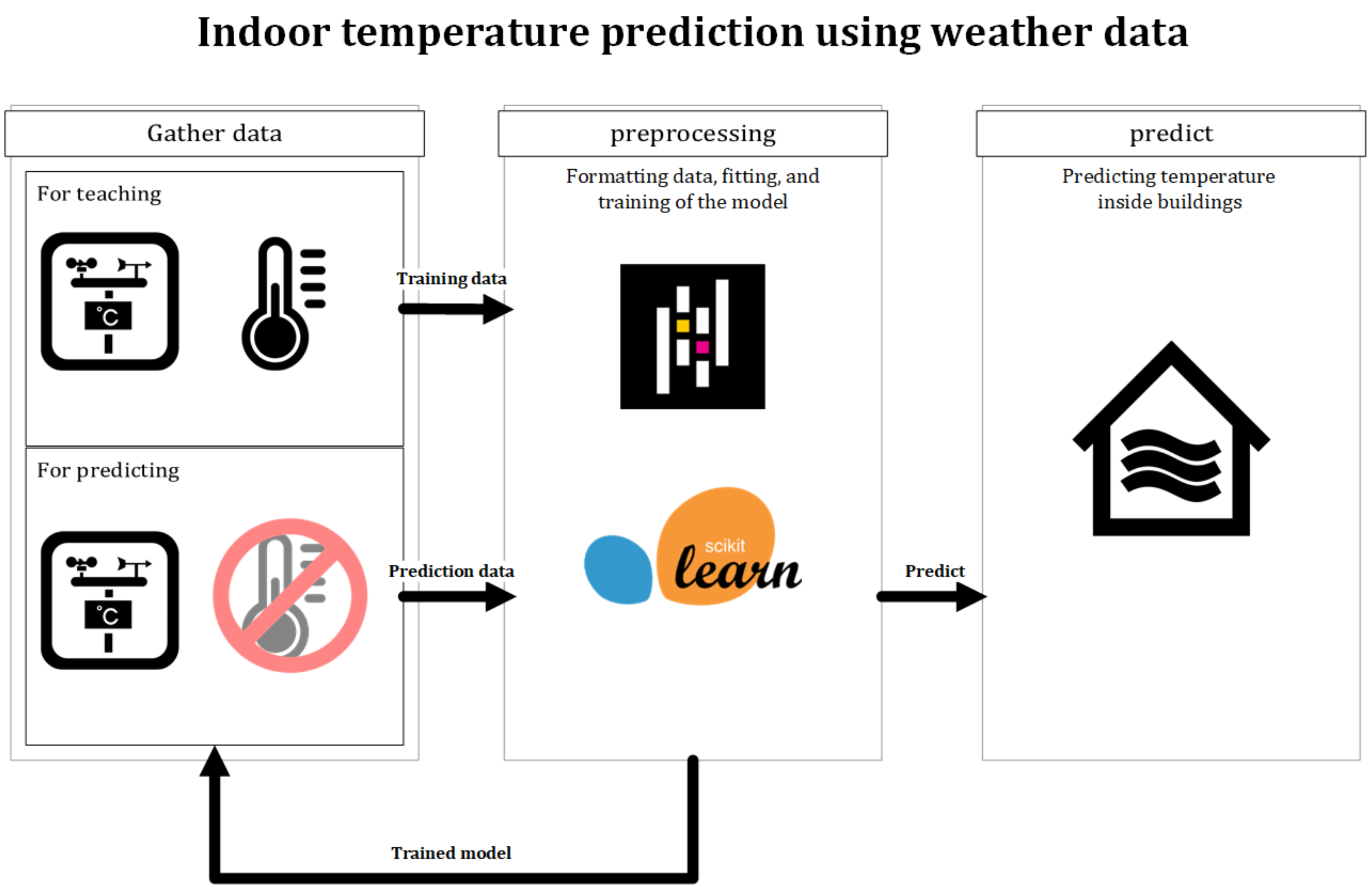
	timestamp	windspeed	sunHeight	weatherTemp	windowStatus	roomTemp_predictions
0	2023-08-03 20:51:13.539527	7.4524	56.637	308.9415	3	305.083
1	2023-08-03 20:51:23.539527	6.7356	77.5518	307.2291	1	304.0276
2	2023-08-03 20:51:33.539527	5.4147	89.4021	302.5057	2	301.6388
3	2023-08-03 20:51:43.539527	3.1398	6.7629	308.1096	2	304.7187
4	2023-08-03 20:51:53.539527	9.8559	0.5558	310.1316	3	305.1231
5	2023-08-03 20:52:03.539527	0.3019	85.987	309.7135	3	305.508
6	2023-08-03 20:52:13.539527	7.1946	89.2368	296.9509	3	297.431
7	2023-08-03 20:52:23.539527	7.1258	26.7952	308.8955	3	305.1319
8	2023-08-03 20:52:33.539527	8.9785	70.2533	292.5576	2	294.2538
9	2023-08-03 20:52:43.539527	6.3347	9.9121	309.8317	1	304.6981

Window Status : 1=closed, 2=tilted, 3=fully opened

Room Temperature Over Time



Prototype Based on Sensor Data
Concept



Training pipeline

To set up this prototype a sufficient trained model is required.
Steps to acquire the model were done in the following order:

- Collect weather & sensor data
- Fit the data for training
- Train a regression model (decisior
- fivefold validation
- Deploy trained model

```
def fivefold_validation():
    X, y, X_train, X_test, y_tra
    model = RandomForestRegresso
    model.fit(X_train, y_train)
    # Splitting the data into 5
    kf = KFold(n_splits=5, shuff

    # Iterate over each fold
    for train_index, test_index

        # Test your model on the
        accuracy = model.score(X

        # Print the accuracy for
        print("Accuracy:", accur
```

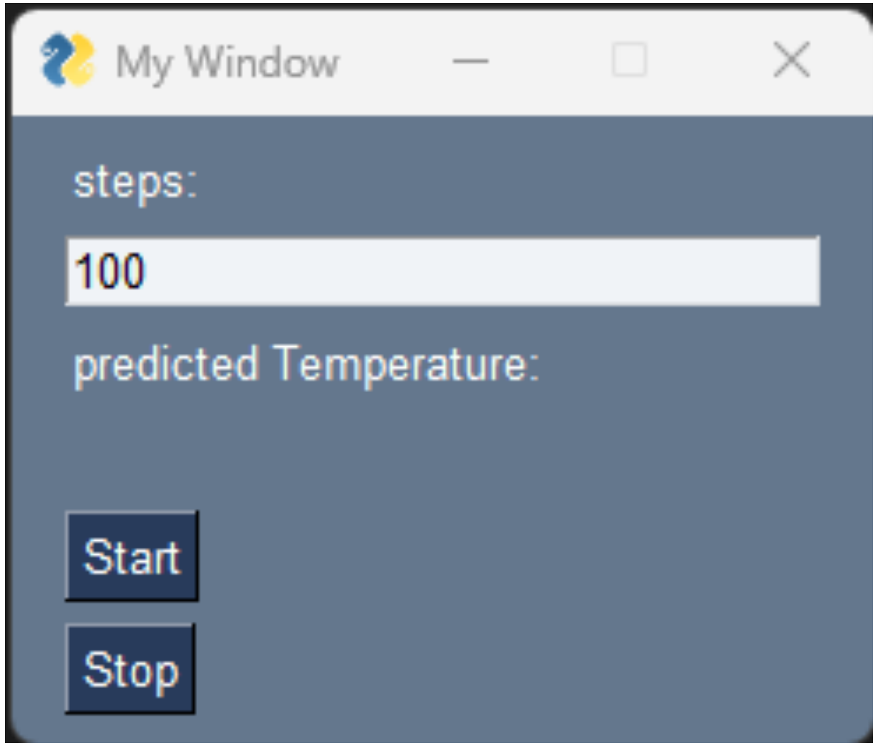
```
def current_data_test(n):

    df = sensor_mqtt.get_sensor_mqtt()

    integer_columns = df.select_dtypes(include='int64').columns
    df[integer_columns] = df[integer_columns].apply(pd.to_numeric)
    df.columns = df.columns.str.strip()
    df = df.drop("RTemperature (C)", axis=1)
    df = df.drop("RTemperature (F)", axis=1)
    df = df.drop("Timestamp", axis=1)
    df = df.drop("Weather Conditions", axis=1)


    X, y, X_train, X_test, y_train, y_test = split_dataset()

    predictions = []
    for _ in range(n):
        model = DecisionTreeRegressor()
        model.fit(X_train, y_train)
        y_pred = model.predict(df)
        predictions.append(y_pred)

    pred_mean = sum(predictions) / len(predictions)
    print(pred_mean)
    return pred_mean
```

Prototype Based on Sensor Data
Frontend

What the user sees

We created a simple UI packaging the main process in one small window

- Takes variable to change accuracy/cpu usage
- Shows predicted value in °C
- Buttons to start or stop the process



Conclusion

Advantages

- Accurate indoor temperature
- Streamline Data Communication
- Scalability and Flexibility (due to modular architecture
- Interactive User Interface

Limitations

- Data Availability
- Data Quality
- Model Generalization
- Predicted vs Actual Data Performance Monitoring
- Latency (MQTT)

Potential Integration

- Using BIM Data to train
- Building Design Optimization
- Integration with Digital Twin Platforms
- Smart Building Management Systems

Thank you