Real-Time Soil Moisture Prediction for Precision Farming

Presented by:
Puneeth kumar Amudala
Dinesh kumar raju kattunga

Introduction

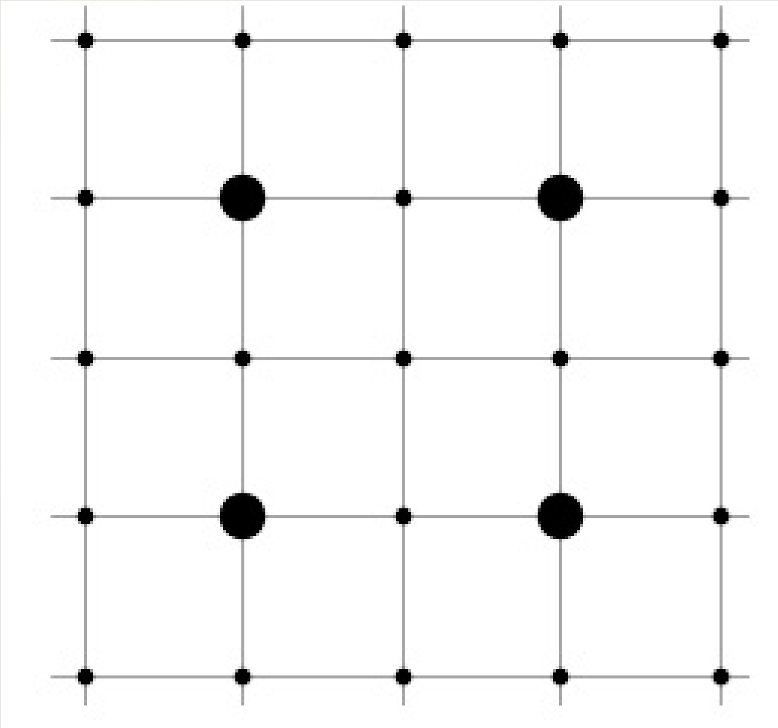
Project Overview:
Scope of work

1. Efficient water management is essential for precision farming.

2. Our project aims to predict soil moisture in unsensed areas using limited sensor data.

3. Machine learning helps reduce the need for many sensors.

4. This makes precision farming more affordable and scalable.



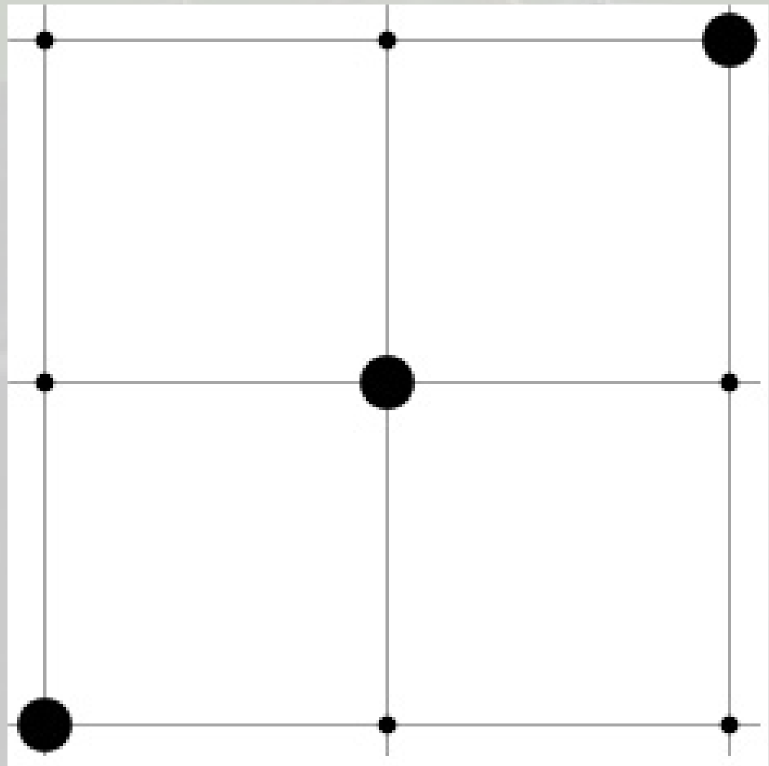
Key Goals:

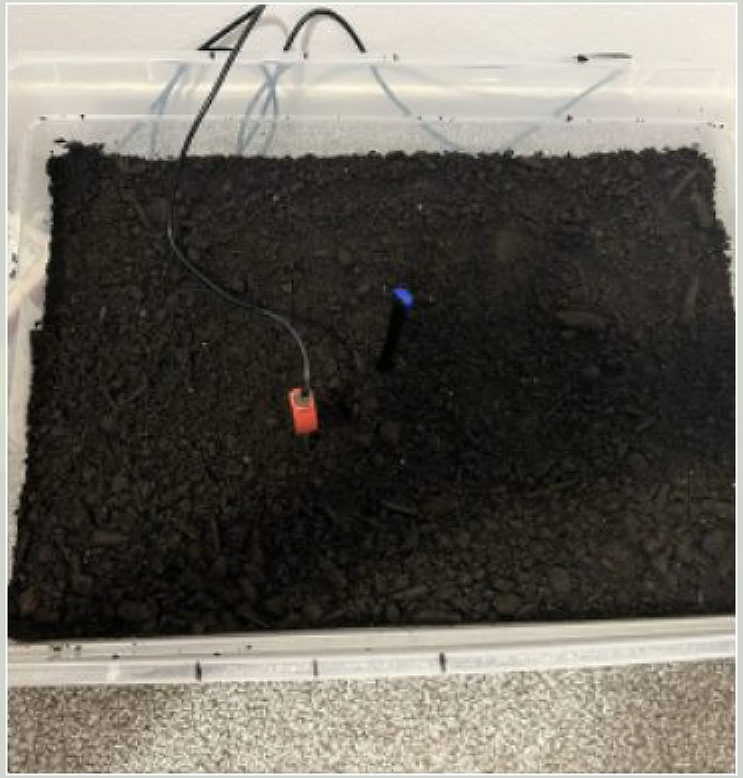
- 1. Predict soil moisture levels at unsensed points using data from central sensors.
- 2. Enhance irrigation planning with accurate, real-time soil moisture estimates.

Project Setup

Marked Coordinates:

1. Center: (11,8)
2. Point1: (13.5,10.5)
3. Point2: (16,13)
4. Collected moisture data at marked points individually on different days using a single sensor.





Data Cleaning and Exploration

1. Splitting the data into manageable datasets.

2. Handle missing values using Mean and Forward Fill (ffill).

```
df_3.head(3)
```

The Number of Columns and rows in df3 :(285, 2)

	timestamp	volumetric_water_content[%]
793	2024-11-18 19:02:56	5.33
794	2024-11-18 19:08:00	5.35
795	2024-11-18 19:13:04	5.35

```
surrounding_values1 = df_3.loc[796:801]
surrounding_values1
```

	timestamp	volumetric_water_content[%]
796	2024-11-18 19:18:08	5.35
797	2024-11-18 19:23:12	5.35
798	2024-11-18 19:28:16	NaN
799	2024-11-18 19:33:24	5.38
800	2024-11-18 19:38:25	5.38

749	2024-11-18 15:20:04	9.60
750	2024-11-18 15:25:08	9.60
751	2024-11-18 15:30:13	9.60
752	2024-11-18 15:35:17	9.60
753	2024-11-18 15:40:21	NaN

Feature Engineering

Calibration of Time:

- Align the timestamps across all datasets.
- Combine datasets into a single table based on the common time column.

Calibration of Soil Moisture:

- Split data into training and test sets.
- Compute the mean of training data and normalize both train and test data.

	Time(Min)	Moisture-1	Moisture-2	Moisture-3
0	0	22.84	8.36	5.33
1	5	20.86	11.26	5.35
2	10	20.01	11.55	5.35
3	15	19.69	11.61	5.35
4	20	19.45	11.61	5.35

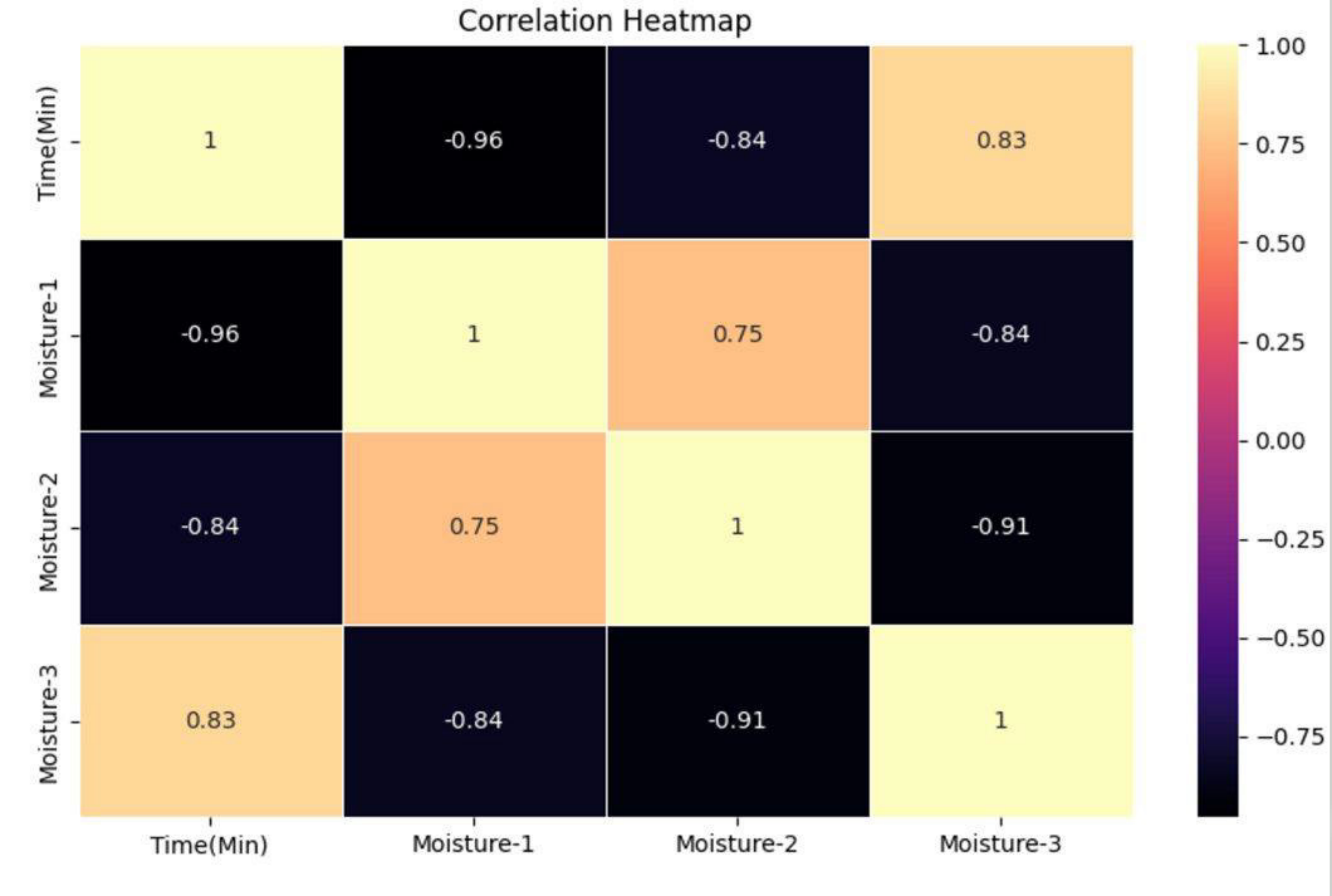
```
train_df.head()
```

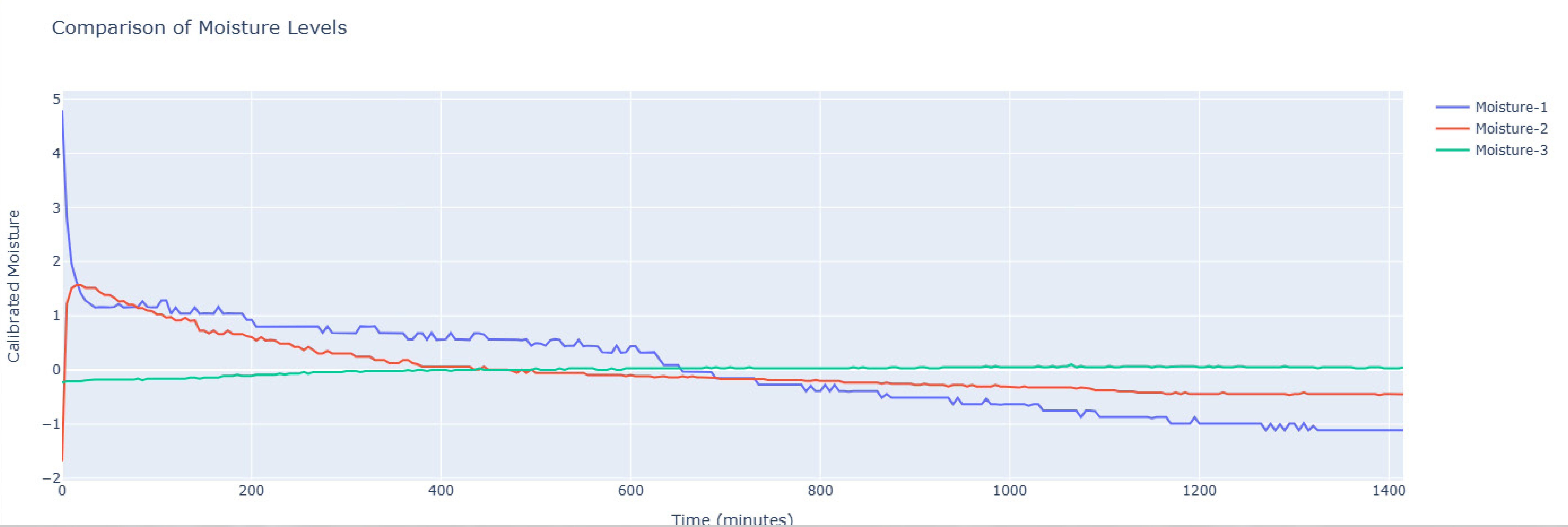
	Time(Min)	Moisture-1	Moisture-2	Moisture-3
268	1340	-1.125903	-0.444626	0.05163
25	125	1.024097	0.915374	-0.15837
86	430	0.544097	0.065374	0.00163
144	720	-0.165903	-0.164626	0.03163
137	685	-0.055903	-0.134626	0.03163

```
train_moisture_means
```

	0
Moisture-1	18.055903
Moisture-2	10.044626
Moisture-3	5.558370

Data Visualisation





How Our Model Works

Model Training:

1. Using data from these marked points , we trained a machine learning model, such as the **Random Forest Regressor**,LinearRegression,DecisionTree and XGBoost.
2. **Train-Test Split:** The dataset was divided into training and testing sets to evaluate the model's performance.

```
X_train=train_df[['Time(Min)', 'Moisture-1']]
y_train=train_df[['Moisture-2','Moisture-3']]
```

```
X_test=test_df[['Time(Min)', 'Moisture-1']]
y_test=test_df[['Moisture-2','Moisture-2']]
```

```
X_train.head()
```

	Time(Min)	Moisture-1
268	1340	-1.125903
25	125	1.024097
86	430	0.544097
144	720	-0.165903
137	685	-0.055903

```
y_train.head()
```

	Moisture-2	Moisture-3
268	-0.444626	0.05163
25	0.915374	-0.15837
86	0.065374	0.00163
144	-0.164626	0.03163
137	-0.134626	0.03163

```
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(random_state=42, n_estimators=100),
    #"Gradient Boosting": GradientBoostingRegressor(random_state=42),
    "XGBoost": XGBRegressor(random_state=42)
}
results = {}
```

How Our Model Works

```python
def get_cv_scores(model, X, y, scoring='neg_mean_squared_error'):
    kf = KFold(n_splits=5, shuffle=True, random_state=42)
    cv_scores = cross_val_score(model, X_train, y_train, cv=kf, scoring=scoring)
    return cv_scores
```

Evaluation:

- The model's performance was assessed using Mean Squared Error (MSE) and R-Squared values, with the aim of minimizing error and increasing prediction accuracy for moisture management in real agricultural fields.

```
Cross-validation scores for Linear Regression:

MSE scores: [0.05127959 0.04714992 0.01858615 0.02266745 0.15090443]
Mean MSE: 0.05811750793360314
Std MSE: 0.0481578500297779
R-squared scores: [0.72978175 0.71899462 0.77578391 0.69818396 0.28457833]
Mean R-squared: 0.6414645134557825
Std R-squared: 0.18024283517996872
----------------------------------------

Cross-validation scores for Decision Tree:

MSE scores: [0.00097609 0.00194348 0.0005     0.00043556 0.09429889]
Mean MSE: 0.019630801932367162
Std MSE: 0.03733793803969098
R-squared scores: [0.98491663 0.98305332 0.96888365 0.96796186 0.6503396 ]
Mean R-squared: 0.9110310120984956
Std R-squared: 0.13053302859758412
----------------------------------------

Cross-validation scores for Random Forest:

MSE scores: [0.00156472 0.00528905 0.00036141 0.00023962 0.10146982]
Mean MSE: 0.021784921724637696
Std MSE: 0.039884382897729255
R-squared scores: [0.98662447 0.97419862 0.97952681 0.97797001 0.62838389]
Mean R-squared: 0.9093407628982788
Std R-squared: 0.1405362098457169
----------------------------------------

Cross-validation scores for XGBoost:

MSE scores: [0.00142014 0.09236248 0.00051744 0.0003334  0.0941941 ]
Mean MSE: 0.037765513972655075
Std MSE: 0.04533118746834057
R-squared scores: [0.98574167 0.67167962 0.97231293 0.97265792 0.65371323]
Mean R-squared: 0.8512210726737977
Std R-squared: 0.1541106234764069
```

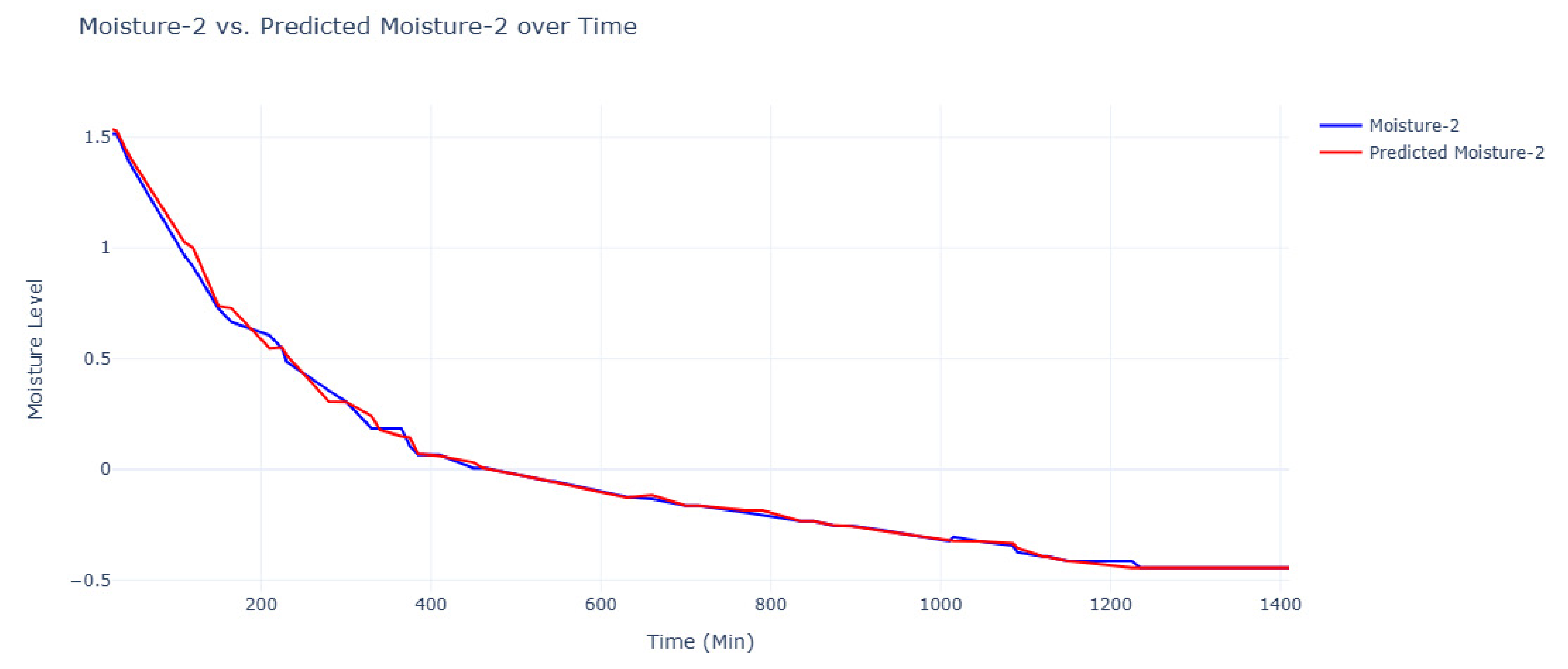
How Our Model Works

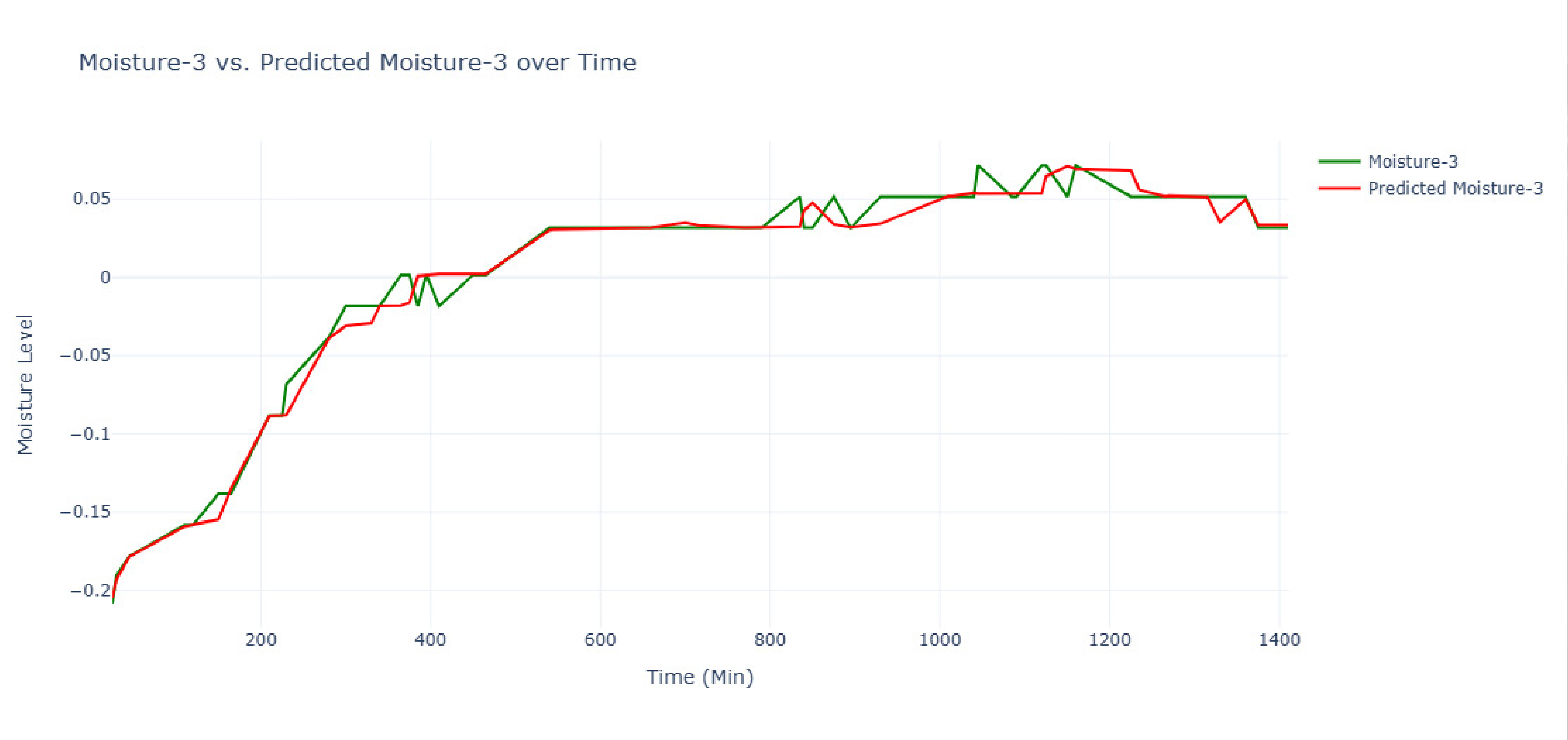
Actual vs Predicted

Model Training:

1. Using data from these marked points (**time, moisture**), we trained a machine learning model, using VotingRegressor.

```
model = VotingRegressor(
    estimators=[
        ('DecisionTree', DecisionTreeRegressor(random_state=42)),
        ('RandomForest', RandomForestRegressor(random_state=42)),
        ('XGBoost', XGBRegressor(objective='reg:squarederror', random_state=42))
    ]

        )

mod1=model.fit(X_train, train_df['Moisture-2'])

mod2=model.fit(X_train, train_df['Moisture-3'])


test_df_with_preds = test_df.copy()

for col in ['Moisture-2', 'Moisture-3']:
    # Train the model
    model = model.fit(X_train, y_train[col])


    y_pred = model.predict(X_test)


    test_df_with_preds[f'Predicted-{col}'] = y_pred
```





How Our Model Works

Recalibrating Predicted Values :

1. Using moisture means which we calculated for the train data.
2. we have added those means to the predicted values to get the Actual Moisture

	Time(Min)	Moisture-1	Moisture-2	Predicted-Moisture-2	Moisture-3	Predicted-Moisture-3
5	25	19.32	11.56	11.580166	5.350	5.353302
6	30	19.26	11.56	11.573500	5.368	5.365349
9	45	19.20	11.43	11.459768	5.380	5.379765
22	110	19.33	11.01	11.069612	5.400	5.398990
24	120	19.20	10.96	11.045908	5.400	5.400291

Simulating Moisture Prediction Over Time

Model -2 Training:

1. Using data from these marked points (**time, moisture, distance**), we trained a machine learning model, such as the **Random Forest Regressor**.
2. The goal was to **predict moisture levels at points not directly measured**, using only a small number of sensors (2 or 3).
3. **Train-Test Split:** The dataset was divided into training and testing sets to evaluate the model's performance.

Time(Min)	Moisture-1	Moisture_Point	Predicted_Moisture	Distance
0	4.797113	Moisture-2	-1.684754	3.535534
0	4.797113	Moisture-3	-0.228620	6.403124
5	2.817113	Moisture-3	-0.208620	6.403124
5	2.817113	Moisture-2	1.215246	3.535534
10	1.967113	Moisture-2	1.505246	3.535534

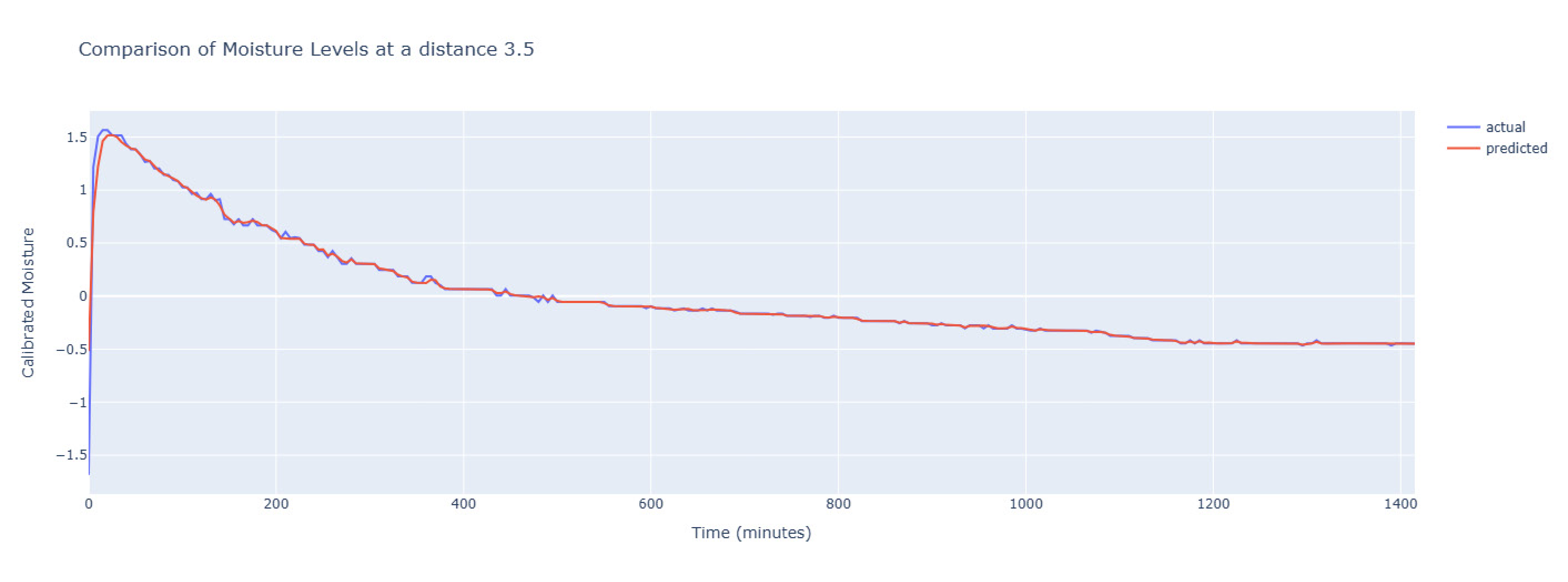
Time(Min)	Moisture-1	Distance
0	4.797113	3.535534
0	4.797113	6.403124
5	2.817113	6.403124
5	2.817113	3.535534
10	1.967113	3.535534

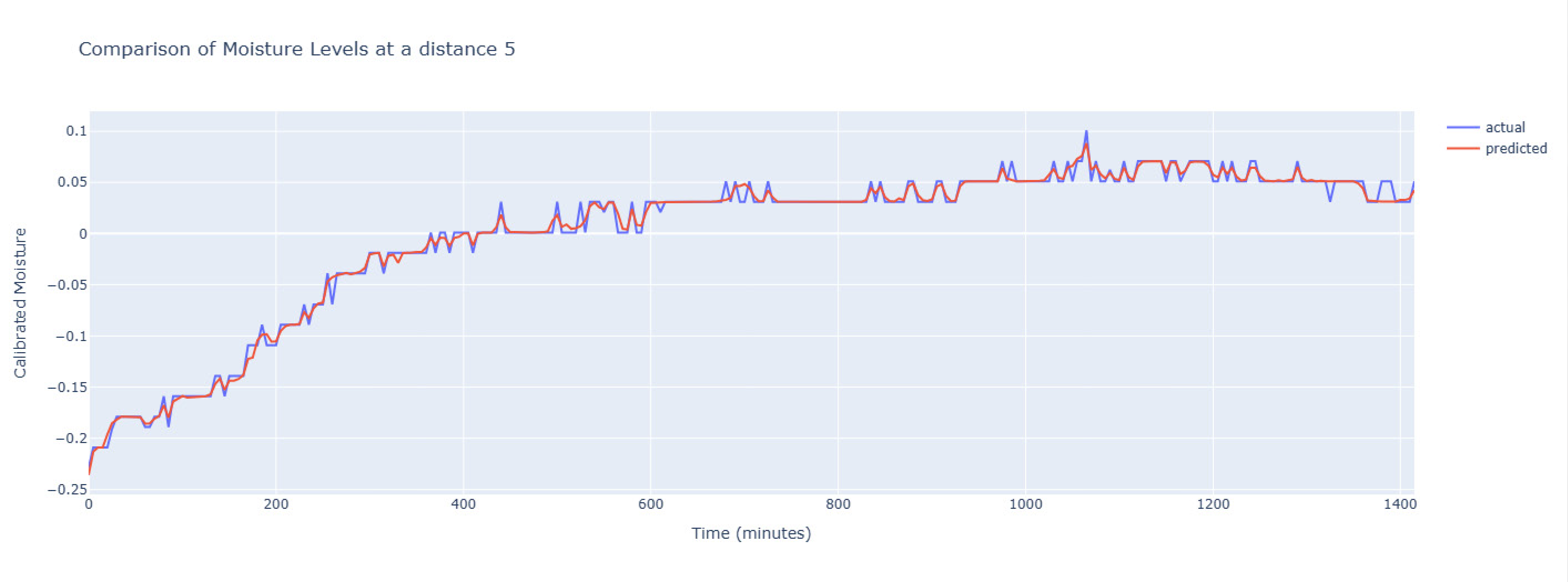
Predicted_Moisture
-1.684754
-0.228620
-0.208620
1.215246
1.505246

```
# Train a model (e.g., Random Forest Regressor)
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

```
Mean Squared Error: 0.0003300785368421089
Predicted Moisture: -0.5164535211267623
Mean Squared Error: 0.0003300785368421089
R-squared: 0.9975462733352372
```

Simulating Moisture Prediction Over Time





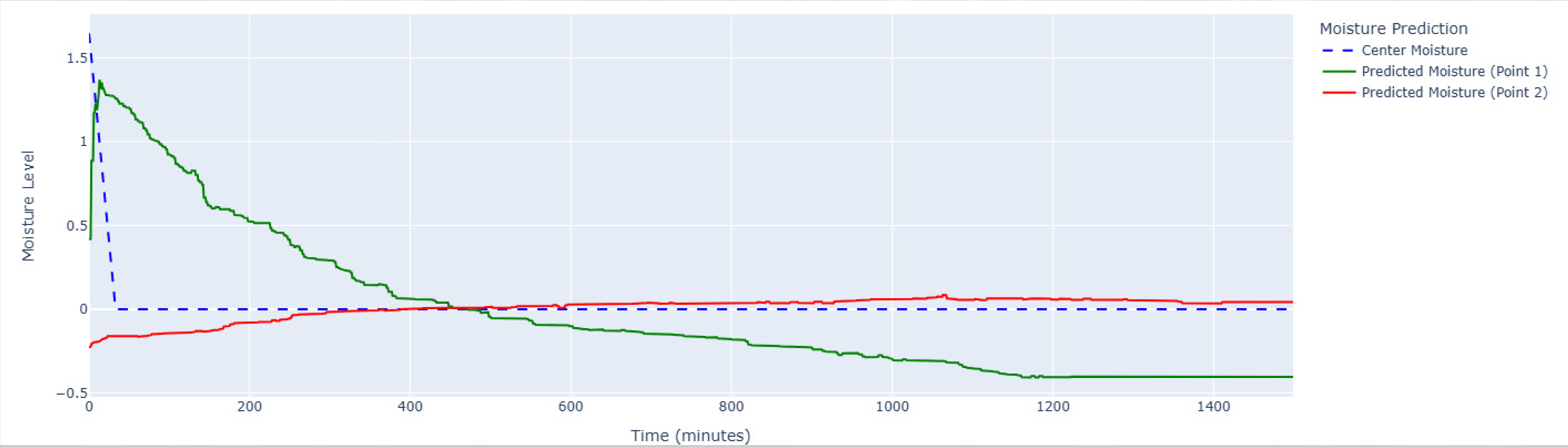
Simulating Moisture Prediction Over Time

Simulated Data for Moisture Prediction:

- we simulate the moisture behavior over time at both the center of the soil and two sensor points (Point 1 and Point 2).
- The center moisture decreases linearly over time, simulating the gradual absorption and evaporation of moisture.
- The distances to the sensor points are **fixed** at 3.5 and 6.0 units.

Explanation:

- **Initial Moisture at Center:** Starting moisture at the center is set to **1.647113**, decreasing by 0.05 per minute over time.
- **Prediction for Points 1 and 2:** The model predicts moisture levels at fixed distances (**3.5 and 6.0 units**) as time progresses.
- **Plotting the Results:** The plot shows moisture levels at the **center** (dashed line) and **predicted moisture** at two points (solid lines in different colors).



Future Scope and Enhancements

Model Expansion:

- Extend the model to handle larger, diverse agricultural fields.
- Incorporate additional variables like temperature, humidity, and soil type for more precise predictions.

Automation and Integration:

- Automate the deployment of optimized sensor grids using drones or robots.
- Integrate real-time data collection for dynamic adjustments.

Data-Driven Insights:

- Utilize advanced machine learning algorithms to improve prediction accuracy.
- Develop user-friendly dashboards for farmers to monitor soil conditions.

Scalability:

- Scale the solution for use in industrial agriculture.
- Test with different crops and environmental conditions.

Sustainability Focus:

- Reduce energy consumption by minimizing sensor usage.
- Promote sustainable farming practices through precision agriculture.

THANK YOU