Machine Learning Models to Predict Soil Moisture for Irrigation Schedule

Md Nahin Islam
Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
mdnislam@stud.fra-uas.de

Doina Logofătu
Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
logofatu@fb2.fra-uas.de

***Abstract*—The agriculture industry must alter its operations in the context of climate change. Farmers can plan their irrigation operations more effectively and efficiently with the exact measurement and forecast of moisture content in their fields. Sensor-based irrigation and machine learning algorithms have the potential to facilitate farmers with significantly effective water management solutions. However, today's machine learning methods based on sensor data necessitate a huge quantity of data for effective training, which poses a number of challenges including affordability, battery life, internet availability, evaporation issues, and other factors. The purpose of this report is to find an efficient machine learning model by doing metric evaluation and comparing the R-squared value, that can predict soil humidity within a certain crop field scenario for the next couple of days using historical data.**

Keywords: Soil Moisture, Regression Model, Metric Evaluation, R-squared Score

I. Introduction

Farming lands, world's largest source of food supply, hold 4,889 million hectares out of 13,003 million hectares of the total land area, classified as 'agricultural area' which is a total of 37.6% of land area on earth [13]. These lands require intensive maintenance and cannot be left unsupervised due to the lack of regular checks, drought, global warming, and the frequency and intensity of extreme weather events, which can be detrimental to crop health and quality due to the lack of of moisture in the soil that might further lead to financial losses. To overcome this situation, sensor-based irrigation and machine learning algorithms might provide efficient and effective solutions for managing water usage.

In this study, we tried to find a machine learning model capable of making short-term predictions of soil humidity for particular crop fields based on recent historical data. A part of the challenge is to design a resilient algorithm that can transform the raw data into a form that can be used to train machine learning models. To achieve the goal, we first worked on the data points and then split the data into training set (90%) and test set (10%) to measure the score of the models. Having fed the training data into 3 different Regression models namely, Ridge, Random Forest, and XGBoost, we have then measured four types of evaluation metrics - MSE, RMSE, MAE, and R-squared value to acquire a better understanding of each model's performance. The resulting best model will enable farmers to anticipate water needs for the crop fields and prepare their irrigation schedules.

The rest of the main sections are organized as follows: Section II is about related work, Section III discusses the methodological approach with a workflow diagram, Section IV is about dataset analysis, Section V is about data preprocessing, Section VI is about data preparation for prediction, Section VII discusses the regression models and evaluation metrics that are used, Section VIII discusses and compares the results of the models, and Section IX is about a final summary.

II. Related Work

Due to the recent development of data science, a large number of studies and research have been conducted on Soil Moisture Prediction using various forms of machine learning algorithms.

Marwa M.A. et al. [11] presented a promising work on this very topic using the SALTMED mathematical model that predicts soil moisture. With high coefficients of determination, RMCE, and CRM values for soil moisture, calibration and validation of the SALTMED model showed that there were only small differences between what was seen and what was predicted.

James Brinkhoff et al. [1] used a Linear Regression model to predict soil moisture for the next 7 days. In their study, they demonstrated a platform to ingest soil moisture, weather, and crop vigor (NDVI) data in real-time.

Nemanja Filipovic et al. [6] designed a Long Short-Term Memory (LSTM) network and trained it at a regional scale using the data from the 2011–2016 period at 28 locations covering four major soil types.

Huihui Feng et al. [5] had made an study that provides an effective approach to investigate the combined effects of precipitation and air temperature on soil moisture for various land covers in the Poyang Lake Basin in China from 2003 to 2009.

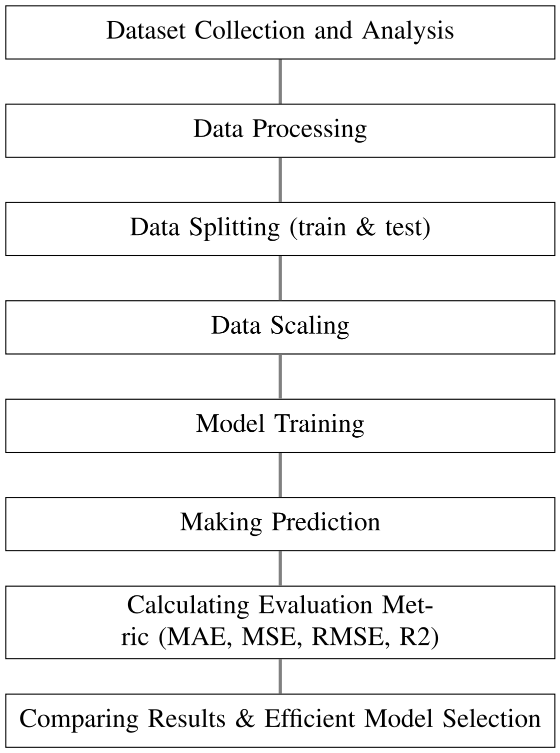

Dataset Collection and Analysis
Data Processing
Data Splitting (train & test)
Data Scaling
Model Training
Making Prediction
Calculating Evaluation Metric (MAE, MSE, RMSE, R2)
Comparing Results & Efficient Model Selection

Fig. 1: Model Workflow

III. Methodological Approach

Fig 1 depicts the approach we have taken in this experiment. The workflow shows the experimental dataset going through a processing pipeline until an efficient machine learning model is found to be best suited by comparing the evaluation metric. In Section IV, the dataset is analyzed thoroughly and visualized. Details about data processing, data splitting (train & test), and data scaling in the workflow diagram are well discussed in Sections V and VI. In Section VII, we describe our models and evaluation metrics. Lastly, Section VIII shows comparisons by the R-Squared value.

It is essential to discuss evaluation metrics and how they help to decide the performance of a trained model.

Mean Absolute Error (MAE) is an widely used metric for evaluation with regression models. It is the average of the absolute values of the individual prediction errors with regard to the test data. MAE is defined by the following equation [14]:

$$mae = \frac{\sum_{i=1}^{n} abs\left(y_i - \lambda(x_i)\right)}{n} \quad (1)$$

here y_i represents the actual target value for test instance x_i , $\lambda(x_i)$ is the predicted target value for x_i , and n is the number of test instances.

Mean Squared Error (MSE) is another widely used metric for evaluation with regression models. It is the average of the squared values of the prediction errors concerning the test data. The error is the difference between the prediction values and the test set. MSE is defined by the following equation [15]:

$$mse = \frac{\sum_{i=1}^{n} (y_i - \lambda(x_i))^2}{n} \quad (2)$$

here y_i represents the actual target value for test instance x_i , $\lambda(x_i)$ is the predicted target value for x_i , and n is the number of test instances.

Root Mean Squared Error (RMSE) is only the root square of the MSE. The error is the difference between the prediction values and the test set. The MSE is conventionally presented by the following equation:

$$rmse = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \lambda(x_i))^2}{n}} \quad (3)$$

here y_i represents the actual target value for test instance x_i , $\lambda(x_i)$ is the predicted target value for x_i , and n is the number of test instances.

R-Squared (R^2) is a measure in statistics that represents the goodness-of-fit of a regression model. R2 value is measured between 0 and 1. Usually, a value closer to 1 is regarded as good quality, and value closer to 0 as bad. However, the decision may vary on different contexts. R^2 is calculted using the following equation,

$$R^2 = \frac{SS_{regression}}{SS_{total}} \quad (4)$$

here, $SS_{regression}$ is the sum of squares due to regression which determins how accurate the regression model represents the modeling data.SS_{total} is the total sum of squares which calculates how much the observed data has varied.

IV. Dataset Analysis

The dataset was taken from the Zindi Platform website with some modifications [4]. Originally, the dataset was collected as a part of an experiment conducted by the University of Gaston Berger [7]. The shape of the dataset is (28049, 14) where 28049 is the number of data rows and 14 is the number of columns. In the following section, a short explanation is given for each of the features of the dataset:

Soil humidity: This will be used both as a training feature and as a target for predictions. The number attached to the name indicates the corresponding field from which the data was acquired.
Air temperature (C): Air temperature plays a vital role in soil moisture. The average soil moisture increases by 0.04% with every increase in precipitation and decreases by 0.12% with every increase in air temperature. Here it is measured in Celsius [5].
Air humidity (%): Changes in atmospheric humidity can alter soil moisture, which we should consider. It is expressed as a percentage. The higher the proportion, the drier the air.

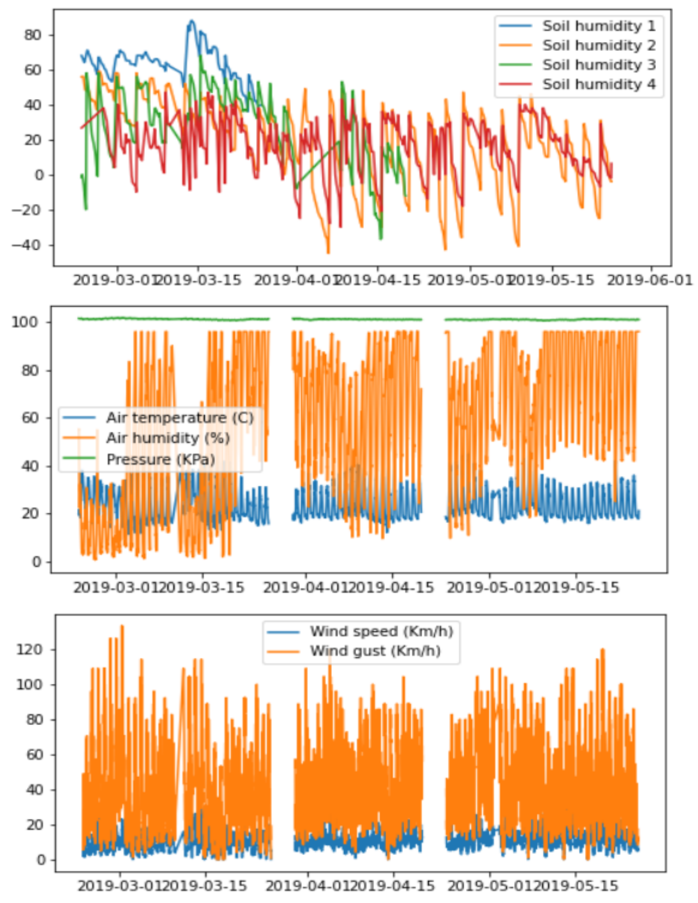
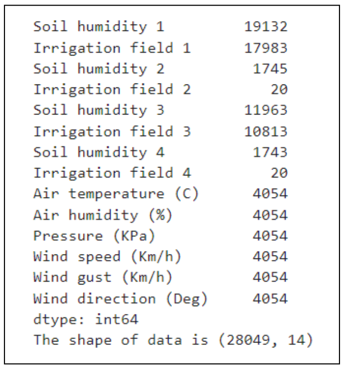
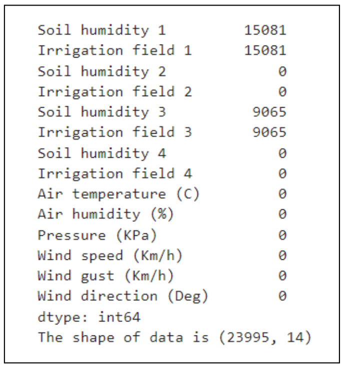


Fig. 2: Data Distribution



```
Soil humidity 1          19132
Irrigation field 1       17983
Soil humidity 2           1745
Irrigation field 2          20
Soil humidity 3          11963
Irrigation field 3       10813
Soil humidity 4           1743
Irrigation field 4          20
Air temperature (C)       4054
Air humidity (%)          4054
Pressure (KPa)            4054
Wind speed (Km/h)         4054
Wind gust (Km/h)          4054
Wind direction (Deg)      4054
dtype: int64
The shape of data is (28049, 14)
```

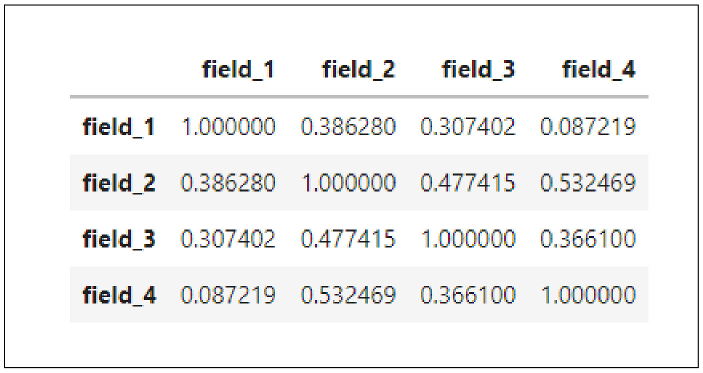
(a) before cleaning



```
Soil humidity 1          15081
Irrigation field 1       15081
Soil humidity 2              0
Irrigation field 2           0
Soil humidity 3           9065
Irrigation field 3        9065
Soil humidity 4              0
Irrigation field 4           0
Air temperature (C)          0
Air humidity (%)             0
Pressure (KPa)               0
Wind speed (Km/h)            0
Wind gust (Km/h)             0
Wind direction (Deg)         0
dtype: int64
The shape of data is (23995, 14)
```

(b) after cleaning

Fig. 3: Missing Value Information



	field_1	field_2	field_3	field_4
field_1	1.000000	0.386280	0.307402	0.087219
field_2	0.386280	1.000000	0.477415	0.532469
field_3	0.307402	0.477415	1.000000	0.366100
field_4	0.087219	0.532469	0.366100	1.000000

Fig. 4: Correlation Matrix of segregated data

Pressure (KPa): Capillary pressures are lowered (become more negative) by an equivalent amount to the soil air pressure when air pressures occur in the soil [6]. The kilopascal (KPa) unit is used here.
Wind speed (Km/h): The speed of the wind in kilometers per hour.
Wind gust (Km/h): A sudden brief increase in the wind speed measured in kilometers per hour.
Wind direction (Deg): It is determined by the origin of wind flow and measured in degrees.
Irrigation: The irrigation variable is set to 1 when the irrigation is turned on and the soil moisture is rising. It is set to 0 when the irrigation is turned off.

Fig 2 explains the data distribution in line graph form. It is noticeable in the figure that $Soil\ Humidity2$ and $Soil\ Humidity4$ spread through the months of February and May in 2019 while $Soil\ Humitdity1$ has no data after March and for $Soil\ Humidity3$, it is end of April. Also there is some information of other features throughout the time duration.

V. Data Preprocessing

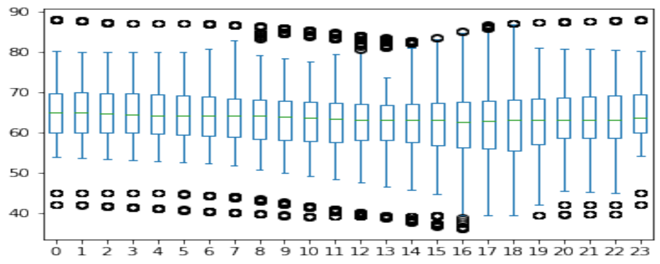
In machine learning, "data preprocessing" is the process of getting raw data ready so that it can be used to build and train machine learning models.

A. Data Cleaning

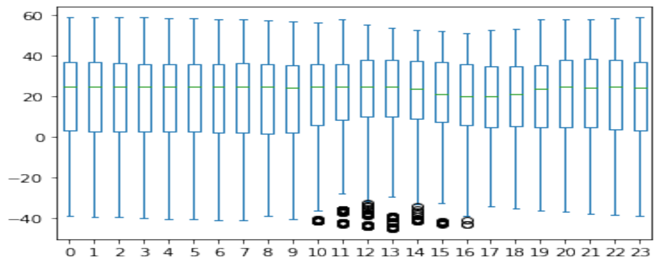
The (28049, 14)-shaped dataset has a good number of missing values, which requires data cleaning to obtain good results. One noticeable thing about the missing values is that most of them exist in the last portion of each column. So in our data cleaning approach, we have dropped an entire row containing the missing values, starting from the last index of the 28049 rows, until a value is found in each column. After the cleaning, the shape of the dataset becomes (23995, 14). Fig 3 shows the information of the missing values before and after the cleaning.

B. Data Segregation and Correlation

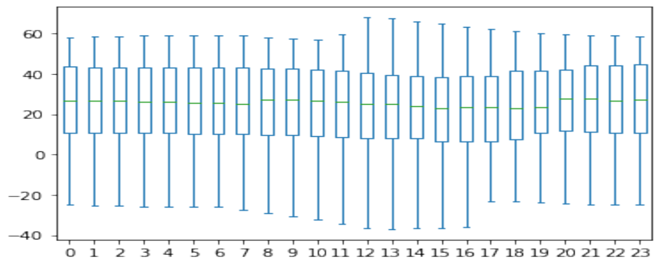
After data cleaning, we have then segregated the features with their associated numbers separately into four groups namely $field_1$, $field_2$, $field_3$, $field_4$ for four different fields. Then the correlation among the sampled data is checked based on the soil humidity. Fig 4 shows $field_2$ and $field_4$ have correlation above 50% where $field_2$ and $field_3$ have a little less than 50% correlation.



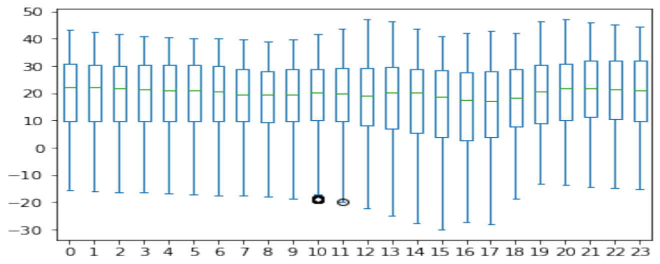
(a) $field_1$



(b) $field_2$



(c) $field_3$



(d) variation-based visualization of $field_4$

Fig. 5: Variation-based explanation

C. Variation-based Explanation and Data Imputation

Fig 5 shows hourly data variation in humidity for each field. $field_1$ has a significant number of outliers in the dataset and the highest number is found during the 16th hour, where $field_2$ has the outliers between the 10th and 16th hour. A small number of outliers are also found in $field_4$ during the 10th hour and $field_3$ does not have any outliers. To overcome these outliers, performing imputation of the mean values of the sampled data (achieved through the subsections A and B) on an hourly basis is found to be an optimal solution.

VI. Data Preparation

The following subsections discuss how the cleaned and sampled data have been transformed into a form that can be used to train machine learning models for prediction.

A. Taking Differentials

The differentials of the features - $Pressure(KPa)$, and $Airtemperature$ are taken by shifting the features by 1 and then subtracting the previous values and assigning then to the variable $target[diff\_pressure]$ and $target[diff\_temperature]$ accordingly.

B. Cyclic Encoding

In the training dataset, we have found a cyclical feature represented by the time attributes, in this case - *hours* and *months*. For example, in the real world, the time gap between 22:00 and 23:00 is one hour, which can be easily measured by subtraction. However, when we consider 23:00 and 00:00, the jump discontinuity occurs, and even though the difference is one hour, without the encoded feature, the absolute difference in the feature is twenty-three. To overcome this issue in machine learning, by applying both $sinencoding$ and $cosencoding$ to the data frame, we will be able to achieve a complete cycle in two dimensions. Cyclic encoding is done using the equation (5) [16].

$$x_{sin} = \sin\left(\frac{2 * \pi * x}{\max(x)}\right) \qquad x_{cos} = \cos\left(\frac{2 * \pi * x}{\max(x)}\right) \tag{5}$$

In equation (5) x represents the target values (hours and months) and $max(x)$ represents the maximum number in the paradigm of x, i.e. 23 hours or 12 months respectively.

C. Deletion & Value Assertion

After taking the differentials and cyclic encoding, rest of the values of the $target < list >$ variable is deleted.
Then the difference in $soilhumidity$ is taken as the prediction target, by convention it is called y and the rest of the differentials are considered as features which is, by convention, presented as X.

D. Data Splitting

For splitting the data into training set and test set we used $train\_test\_split()$ function from the "sklearn.model_selection" library in Python. This function takes features **X** and prediction target **y** as parameters among other parameter with default values. We have also set the $test\_size$ parameter to **0.1** which indicates the percentage of the test set from the whole dataset. In this case, 90% of the data will be used for training and 10% for testing purpose. This split occurs on a random basis. To get the same split all the time some measures can be taken. The function then returns the split data and keeps them in variables X_train, X_test, y_train, and y_test accordingly.

E. Data Scaling

In machine learning, data scaling is used to normalize a dataset by standardizing the range of features. It is very important and good practice to take all data into the same scale. Among many popular methods of data scaling, **min-max scaling** method is taken into consideration for our experiment as it is exceedingly biased by the maximum and minimum values in our data that might still contain outliers. The min-max method scales all the data features between 0 and 1 [9]. The mathematical formula of min-max scaling (0,1) is as follows:

$$X_{norm} = \frac{x - min(x)}{max(x) - min(x))} \tag{6}$$

Algorithm 1: Data Preparation

```
Result: $X\_train$, $X\_test$, $y\_train$, $y\_test$
1 Function: prepare_data ($target\_field$,$window\_size$ ):
2 compute differentials of $Pressure(KPa)$ and
    $Airtemperature(C)$ ;
3 perform cyclic encoding on $month$ and $hour$ ;
4 keep differentials from step 2 and remove
    $Airtemperature(C), Pressure(KPa), Hour, Month$
    from $target\_field$
5 prepare features $X$ and prediction target $y$
6 split data into train set and test set
7 normalize splitted data with $MinMaxScaling$
    function
8 return $X\_train$, $X\_test$, $y\_train$, $y\_test$
```

Algorithm 1 describes the data processing discussed in subsection D of Section V and represents it as a function. The $prepare\_data$ function takes two parameters: the first parameter $target\_field$ represents the sample data of an individual field ($field_1/field_2/field_3/field_4$) and the second parameter $window\_size$ represents the number of days for the prediction to be done. In step 2, all the features are extracted first, and then their differentials are calculated using shifting method as described in subsection *A*. Then the cyclic encoding is performed in step 3 according to subsection *B*. After deleting the prior values in step 4, the new features **X** and prediction target **y** are prepared in step 5, which is discussed in subsection *C*. In step 6, the data is split into two parts: the training set and the test set. The split data are then normalized through step 7 using the $MinMaxScaler()$ function. In the end, at step 8, the split data X_train, X_test, y_train, y_test is returned as ready to be used for training machine learning models.

VII. Regression Models

A. Ridge Regression

Ridge regression is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This modification is done by adding a penalty parameter that is equivalent to the square of the magnitude of the coefficients [12]:

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2 \tag{7}$$

TABLE I: Evaluation Metrics of Ridge Regression

Evaluation Metrics	Obtained Score			
	field_1	**field_2**	**field_3**	**field_4**
Mean Absolute Error	0.073101	0.137389	0.110351	0.106709
Mean Squared Error	0.013011	0.029838	0.021302	0.021528
Root Mean Squared Error	0.114064	0.172737	0.145953	0.146723

α is the complexity parameter which controls the shrinkage amount and $\alpha \geq 0$. The larger α is, the greater the amount of shrinkage. w is the regression coefficient, X is the input, and y is the independent output.

We keep the Ridge parameters at their default values except for the $random\_state$ parameter. The value is set to 123 to fit the model with stochastic gradient descent. Then the model is wrapped with MultiOutput Regressor. Typically, regression models predict a single numerical value, but in this experiment, it is required to make four different predictions at a time for four different fields, and to attain this, data sampling has been done earlier as discussed in subsection B of section V. The evaluation metrics are then collected and put in Table I.

B. Random Forest Regression

In Decision trees, each prediction is based on historical data from only a few houses at each leaf, so a deep tree with too many leaves would overfit, whereas a shallow tree with few leaves would perform badly since it wouldn't be able to collect as many differences in the raw data. On the other hand, Random Forest Regression (RFR) is an extended version of tree-based algorithms that uses the quality features of multiple decision trees for making decisions [8]. Therefore, it is referred to as a "forest" of trees, hence the name "Random Forest". It is called random because it creates the decision trees randomly.

RFR is essentially a bagging approach, also known as bootstrap aggregation, in which numerous trees are generated randomly. Each node is separated into proper segregation of features, and each tree is built out of a different sample of rows. Each tree produces a distinct prediction. After that, the average of these estimations is utilized to provide a single result. This technique beats a single decision tree in terms of accuracy and overfitting due to averaging. A random forest regressor forecast is guaranteed to be an average of the estimations produced by the forest's trees [2]. In our case, the following hyperparameters are tuned, and the rest are set to their default values: $'criterion'$: 'mse', $'n\_estimators'$: [10,20, 25], $'max\_depth'$: [15, 10 ,15], $'min\_samples\_split'$: [10, 15], $'min\_samples\_leaf'$: [2,5], $'bootstrap'$: [True, False]. Then we used the GridSearchCV function to loop through the hyperparameters and fit the estimator (model) on the training set. The idea is to select the best combination of the hyperparameters. We also set cv value to 5, to determine the cross-validation splitting strategy over the training set, and

TABLE II: Evaluation Metrics of Random Forest Regression

Evaluation Metrics	**Obtained Score**			
	field_1	**field_2**	**field_3**	**field_4**
Mean Absolute Error	0.044142	0.085916	0.063216	0.050694
Mean Squared Error	0.004732	0.014802	0.007034	0.006547
Root Mean Squared Error	0.068779	0.121665	0.083866	0.080913

TABLE III: Evaluation Metrics of XGBoost Regression

Evaluation Metrics	**Obtained Score**			
	field_1	**field_2**	**field_3**	**field_4**
Mean Absolute Error	0.044706	0.088209	0.060037	0.046813
Mean Squared Error	0.005602	0.018685	0.007530	0.006569
Root Mean Squared Error	0.074843	0.016695	0.086777	0.081048

set n_{jobs} to -1 for using all the processors in parallel. Table II shows the evaluation metric calculations.

C. *XGBoost Regression*

XGBoost is an optimized implementation of the Gradient Boosting method where the boosted decision tree models are created and trained in a sequential form. Instead of training the models separately, each new model is trained to correct the errors of the previous ones. In this sequential process, each time the outcomes with assigned weights are carried out into the next decision tree until the final prediction is yielded. The weights are increased if the outcomes are predicted incorrectly and decreased for correct prediction [10].

In this study, the Scikit-Learn wrapper interface for XGBoost [3] is used for training the model. We have set the $n\_estimator$ parameter to 2000 and the $learning\_rate$ to 0.01 for the best possible result. This hyperparameter is used to weigh each model that has been mentioned earlier in this section. The default value is 0.3. An increase in the maximum depth of a tree shows better performance but may cause extra complexity and overfitting. To avoid this, we set the max_depth value to 20. Then the metric calculation (Table III) on each of the fields is collected.

VIII. Result

In this section, we present the graphical representation of the difference between the predictions and the subsequent test data. Finally, a comparison of models by their R-Squared scores are shown in tabular form and depicted in a graph.

Fig 6 shows the differences between the predictions made by the Ridge Model and the test data for each crop fields, range from -0.5 to 0.41. Most of these differences range between -0.2 to 0.2.

Fig 7 shows the Random Forest Model result. The value differences are mostly spread through the range of -0.1 to 0.1.

In Fig 8 the value-differences are found residing between -0.2 to 0.2.

Finally, Table IV, along with Figure 9, show a comparative performance evaluation of the regression models for each of the crop fields. It turns out, the **Random Forest Regression**

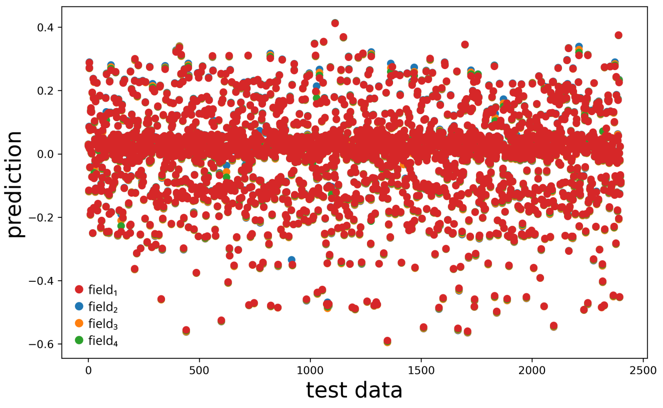


Fig. 6: Ridge Model Result

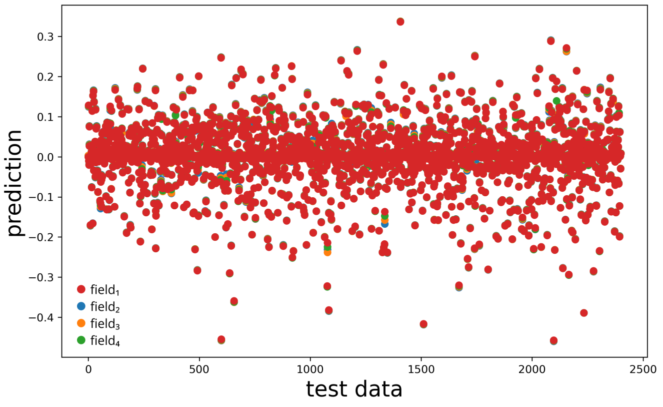


Fig. 7: Random Forest Model Result

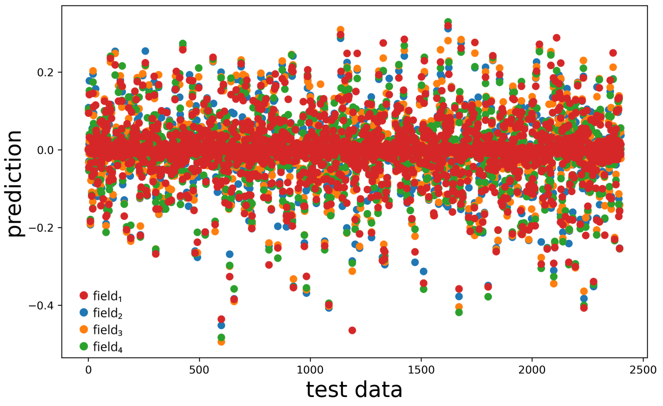


Fig. 8: XGBoost Regression Model Result

TABLE IV: R-Squared Score of Regression Models

Regression Model	**R-Squared**			
	field$_1$	**field$_2$**	**field$_3$**	**field$_4$**
Ridge	0.116720	0.398450	0.168636	0.114326
Random Forest	0.678838	0.701575	0.725507	0.730726
XGBoost	0.619698	0.623315	0.706119	0.729837

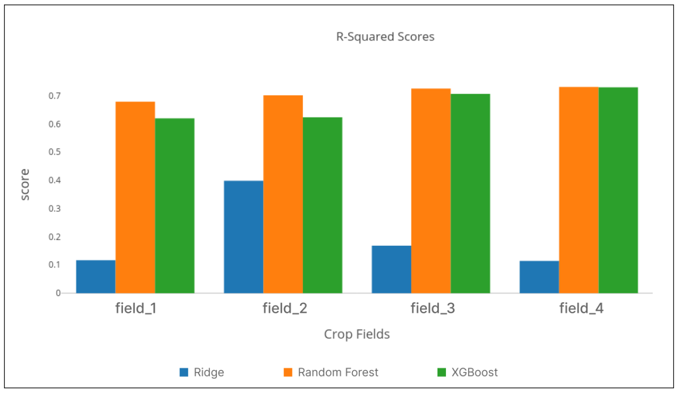


Fig. 9: R-squared Scores comparison among models grouped by field.

model fits well with the experimental dataset. Furthermore, a comparison among the error metrics from Table I, Table II, and Table III shows that the random forest model has the lowest error among all other regression models used in this experiment.

IX. Conclusion

In this experiment, we have made soil moisture predictions on four different crop fields for irrigation schedules using three machine learning models: Ridge Regression, Random Forest Regression, and XGBoost Regression. After going through the data processing pipeline, we trained the models with the transformed data to generate predictions. Then the predictions were tested, followed by the metric evaluation. Finally, based on the evaluation scores, **Random Forest Regression Model** is found to be the most efficient model to work with the subjected dataset. Since there are few variations between the predictions and actual data, we may infer that the data are broadly distributed and fit the tree model well. This initiative also assists the farmers in making their schedules for irrigation in the crop fields. Future scope might include an approach to predicting how much water is needed for irrigation based on the prediction results of this experiment.

X. ACKNOWLEDGEMENTS

This experiment is done as a part of the academic curriculum for the master's degree program in FRA-UAS and is supported by Waziup e.V., a fast growing IoT solution company based in Dresden, Germany. We also want to thank our colleague, Mr. Sheikh Sharfuddin Mim, who is a master's degree (MEng., IT) student at FRA-UAS.

References

[1] James Brinkhoff, John Hornbuckle, and Carlos Ballester Lurbe. Soil moisture forecasting for irrigation recommendation. *IFAC-PapersOnLine*, 52(30):385–390, 2019.

[2] Afroz Chakure. Implementing random forest regression in python: An introduction. https://builtin.com/data-science/random-forest-python, 2022.

[3] dmlc XGBoost. Xgboost documentation. https://xgboost.readthedocs.io/en/latest/python/python$_api.htmlmodule-xgboost.sklearn.$

[4] Waziub e.V. Wazihub soil moisture prediction challenge. https://zindi.africa/competitions/wazihub-soil-moisture-prediction-challenge/data, 2019.

[5] Huihui Feng and Yuanbo Liu. Combined effects of precipitation and air temperature on soil moisture in different land covers in a humid basin. *Journal of Hydrology*, 531:1129–1140, 2015.

[6] Nemanja Filipović, Sanja Brdar, Gordan Mimić, Oskar Marko, and Vladimir Crnojević. Regional soil moisture prediction system based on long short-term memory network. *Biosystems Engineering*, 213:30–38, 2022.

[7] Youssouph Gueye and Maïssa Mbaye. efarm-lab: Edge ai-iot framework for agronomic labs experiments. In *International Conference on Research in Computer Science and its Applications*, pages 101–112. Springer, 2021.

[8] Tae-Hwy Lee, Aman Ullah, and Ran Wang. Bootstrap aggregating and random forest. In *Macroeconomic Forecasting in the Era of Big Data*, pages 389–429. Springer, 2020.

[9] Clare Liu. Data transformation: Standardization vs normalization. https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html, 2022.

[10] David Martins. Xgboost: A complete guide to fine-tune and optimize your model. 2021.

[11] MA Marwa, AF El-Shafie, OM Dewedar, JM Molina-Martinez, and R Ragab. Predicting the water requirement, soil moisture distribution, yield, water productivity of peas and impact of climate change using saltmed model. *Plant Archives*, 20(1):3673–3689, 2020.

[12] ScikitLearn Org. Linear model. https://scikit-learn.org/stable/modules/linear$_model.htmlridge-regression-and-classification.$

[13] Hannah Ritchie and Max Roser. Land use. *Our world in data*, 2013.

[14] Claude Sammut and Geoffrey I. Webb, editors. *Mean Absolute Error*, pages 652–652. Springer US, Boston, MA, 2010.

[15] Claude Sammut and Geoffrey I. Webb, editors. *Mean Squared Error*, pages 653–653. Springer US, Boston, MA, 2010.

[16] Andrich van Wyk. Encoding cyclical features for deep learning. *EPI-USE Lab, Pretoria, South Africa, Tech. Rep*, 2018.