Crop Yield Prediction in India

BY: - RISHABH AWASTHY

Targeted Problem: - The targeted problem is to predict crop yields per acre of rice or wheat crops in India, The goal is to provide accurate crop yield predictions to empower these farmers and alleviate poverty and malnutrition.

Research Question: -

Can we predict crop yield per acre, using data on farming practices and environmental conditions?

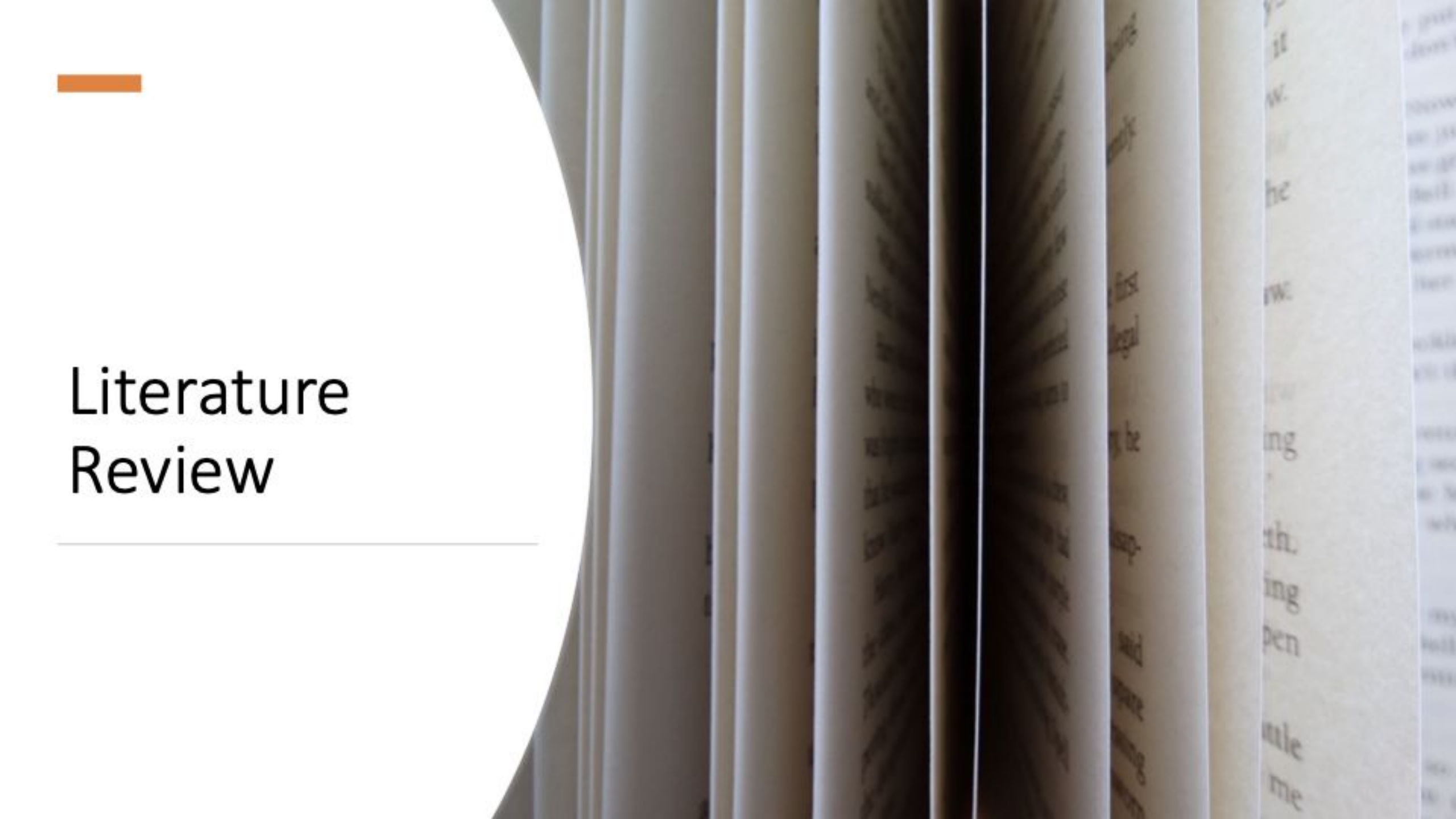
Dataset: -

1. **Source:** The dataset used for this project is collected by Digital Green through surveys conducted among smallholder farmers in India. The data includes information on farming practices, environmental conditions, crop types (rice or wheat), and historical crop yields.
   Dataset link: https://zindi.africa/competitions/digital-green-crop-yield-estimate-challenge/data
2. **Short Description:** The dataset is designed to capture the specific challenges and conditions faced by smallholder farmers in India. It provides valuable insights into the factors influencing crop yields and allows for a targeted approach to predicting rice and wheat crop yields per acre.
3. **Size:** The dataset is substantial, covering a significant number of smallholder farmers over multiple agricultural seasons. It encompasses a diverse range of features, allowing for a comprehensive analysis.

Motivation: -

- **Technical Motivation:** The development of a crop yield prediction model tailored to smallholder farmers in India represents an opportunity to apply machine learning and data science for social impact. It involves addressing real-world challenges by harnessing data to improve the livelihoods of vulnerable populations.
- **Personal Motivation:** The prospect of contributing to breaking the cycle of poverty and malnutrition among smallholder farmers in India is a compelling motivation. This project offers a chance to make a tangible difference in the lives of these farmers and advance global food security in a meaningful and impactful way.

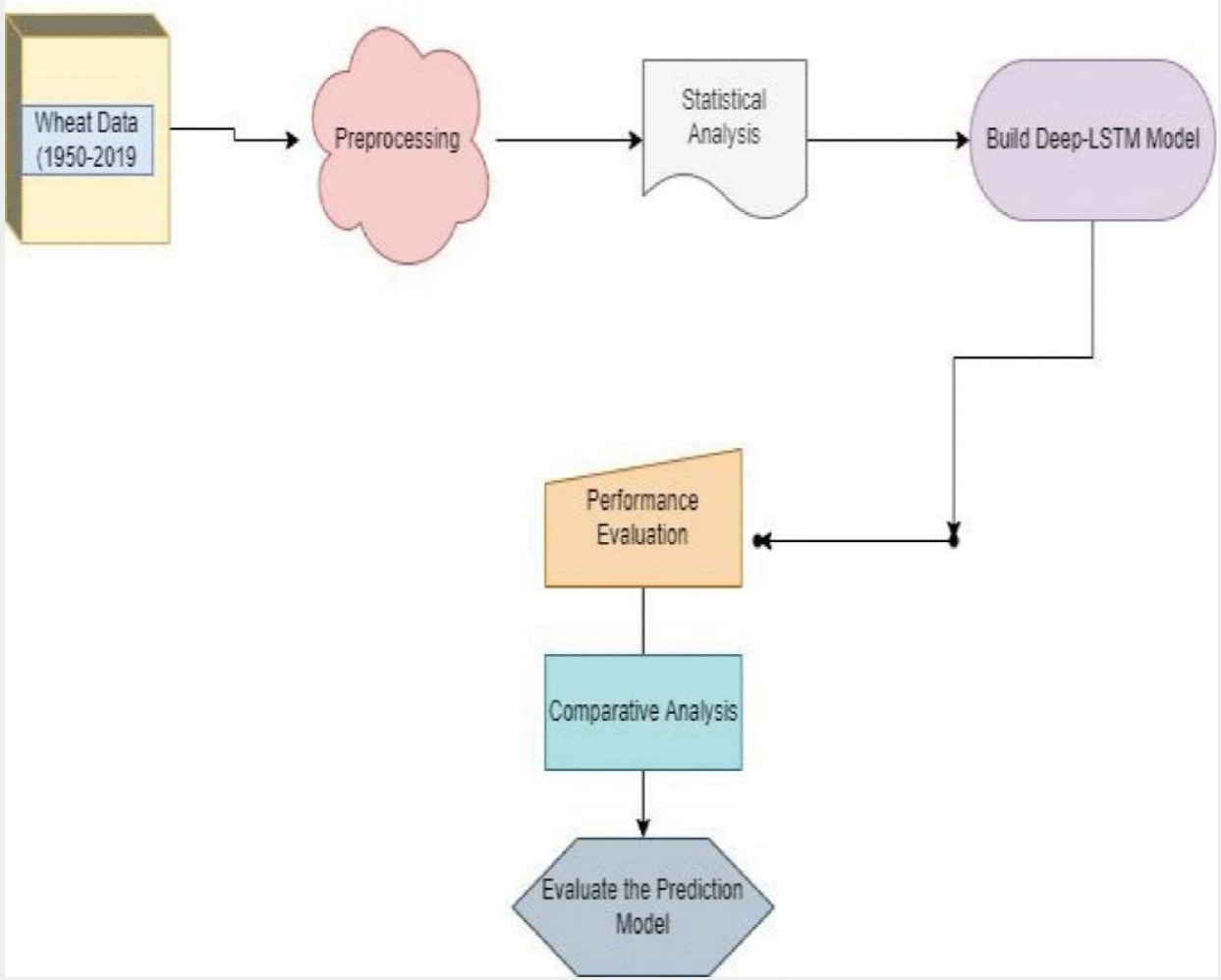
Literature Review



Deep-LSTM Model for Wheat Crop Yield Prediction in India

By: - Preeti Saini; Bharti Nagpal

Article Link : - https://ieeexplore.ieee.org/document/9913604



Goal: - The Article focuses on the Yield prediction of Wheat Crops in India using a Deep-LSTM model, where results were evaluated using Root Mean Square Error, and Mean Square Error and Compared with the existing machine learning methods.

Dataset: - Dataset was collected from the Directorate of Economics and Statistics for the 1950-2019 years.

Methodology: - The LSTM architecture includes three gates to process the input to the output stage.

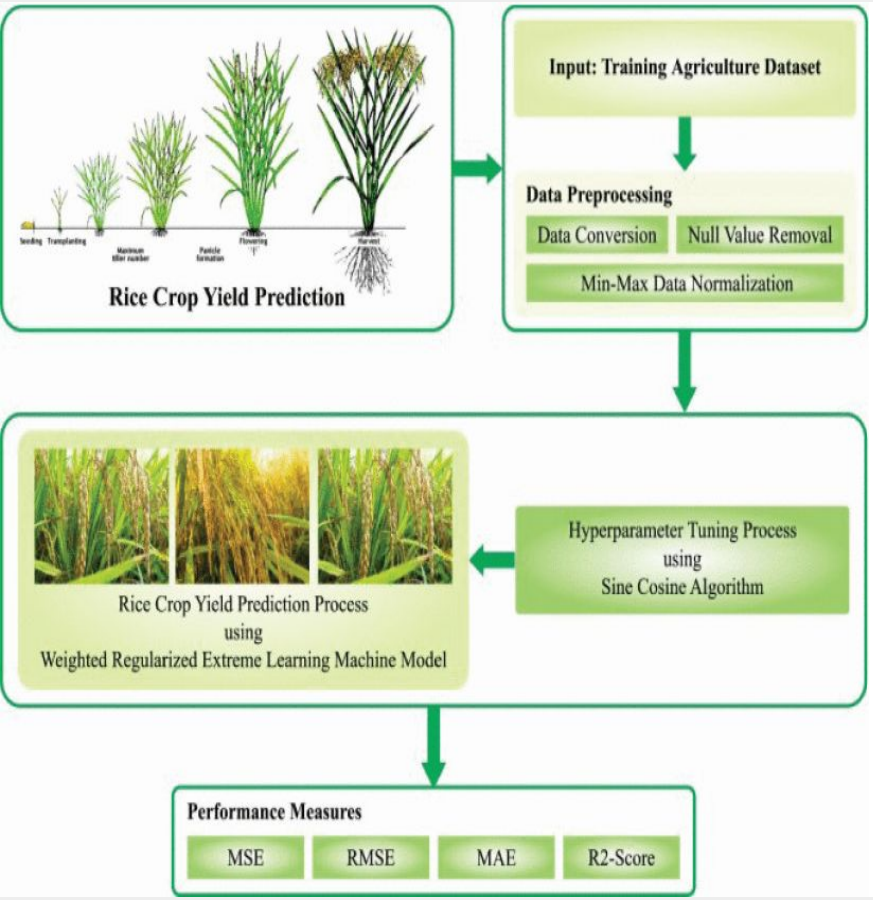
- Forget Gate : Initially, this gate decides which data in the sequence is to forget and retain with the network in the form of [0,1].
- Input Gate : In the next step, the input gate decides which new information needs to be added to the network.
- Output Gate : At this gate, decide the new hidden state of the Cell using the Sigmoid activation function

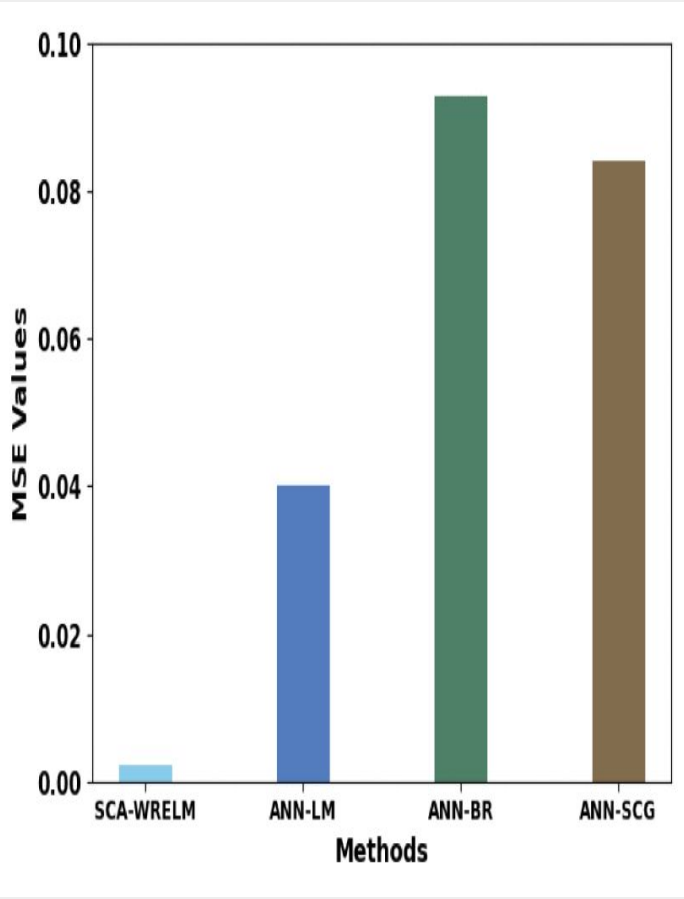
Result: - The results conclude that the proposed model outperforms the existing GPR and Holt-Winter methods with a lower RMSE Value of 0.20, 0. 51 & 0.61 respectively.

Automated Rice Crop Yield Prediction using Sine Cosine Algorithm with Weighted Regularized Extreme Learning Machine

By: - S. Thirumal; R. Latha

Article link: - https://ieeexplore.ieee.org/document/10142403/





Goal: - Proposed SCA-WRELM technique aims to forecast the rice crop productivity effectually. To do this, the SCA-WRELM technique performs min-max data normalization process to scale the data into uniform format. For yield prediction, the SCA-WRELM technique uses WRELM model to forecast the rice productivity. Finally, the SCA is exploited for the optimal parameter tuning of the WRELM model and it results in improved predictive results.

Dataset: - Using previous SCA-Wrelm dataset

Methodology: - The SCA-WRELM technique uses WRELM model to forecast the rice productivity. Huang developed the ELM, it is a kind of SLFN [16]. In ELM, the fundamental concept was that weights in-between hidden and input layers were made arbitrarily rather than iterative computation method, and weights between output and hidden layers (HLs) were calculated in least square model.

Result: - The simulation values of the SCA-WRELM approach is tested on rice yield dataset and the outcomes demonstrate the improved outcomes of the SCA-WRELM approach over other existing methods.

Optimizing Crop Yield in Agriculture using Data Mining and Machine Learning Techniques

By: - Rashmi Welekar, Charanjeet Dadiyala

Article link: - https://ieeexplore.ieee.org/document/10170493

Sr. No.	ML Algorithm	Accuracy
1.	KNN (K-Nearest Neighbours)	75%
2.	Logistic Regression	70%
3.	SVM (Support Vector Machine)	55%
4.	Gaussian Naïve Bayes	45%

Goal : - The use of data mining and machine learning techniques can aid in predicting the most suitable crops to cultivate based on a variety of factors such as soil, weather, and nutrient conditions.

Dataset: - The dataset collected was from various sources namely:

- https://www.kaggle.com/
- https://plants.usda.gov/npk/main
- https://www.cropnutrition.com/nutrient-knowledge
- https://soilhealth.dac.gov.in/

Methodology: - This project focuses on analyzing agricultural conditions and scenarios to find optimal parameters and data to maximize crop yield using data mining and machine learning techniques like k-Nearest Neighbors (k-NN), Naïve Bayes, Support Vector Machine (SVM), Linear Regression, etc. By mining crop, nutrient, soil, location, and climatic data and analyzing new, non-experimental data, the project aims to optimize yield and production, and make the agricultural sector more resilient to climatic change.

Result: - The research is a helpful instrument for farmers and governments to forecast crop yields and make wise choices regarding agricultural practises.

Estimation of Crop Yield From Combined Optical and SAR Imagery Using Gaussian Kernel Regression

By: - Yeshanbele Alebele; Wenhui Wang; Weiguo Yu; Xue Zhang; Xia Yao; Yongchao Tian; Yan Zhu

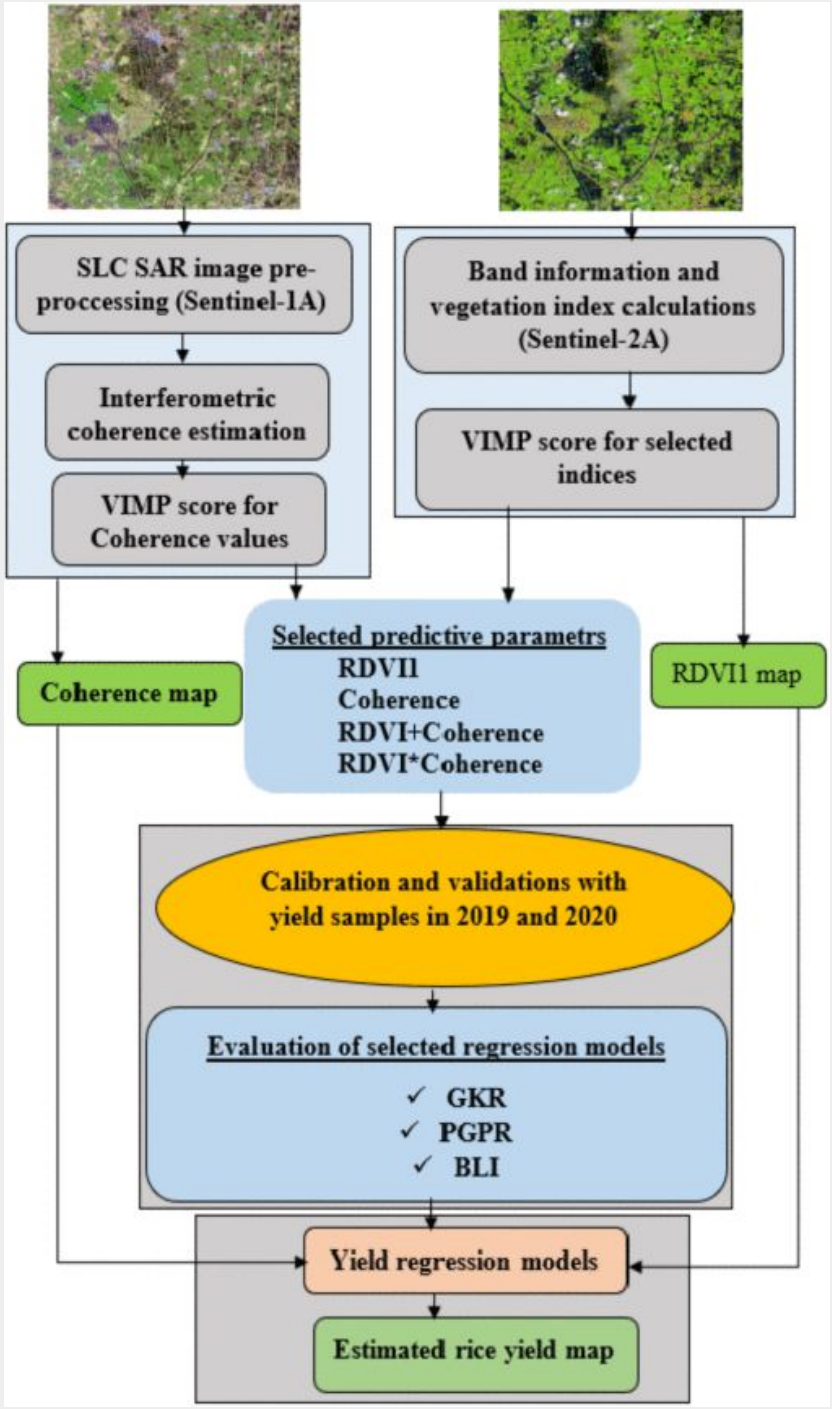
Article link : - https://ieeexplore.ieee.org/document/9565348

Goal : - Objective was to investigate the synergetic use of Sentinel-2 vegetation indices and Sentinel-1 interferometric coherence data through Gaussian kernel regression for estimating rice grain yield.

Dataset: - Data from the European radar *C*-band imaging system

Methodology: - introduced two Gaussian-based regression approaches, Gaussian kernel regression based on kernel values as weighing functions, and PGPR based on MCMC sampling. The two Gaussian-based approaches were compared with their linear extension, Bayesian linear inference regression to relate SAR and optical derived metrics with field-measured crop yield.

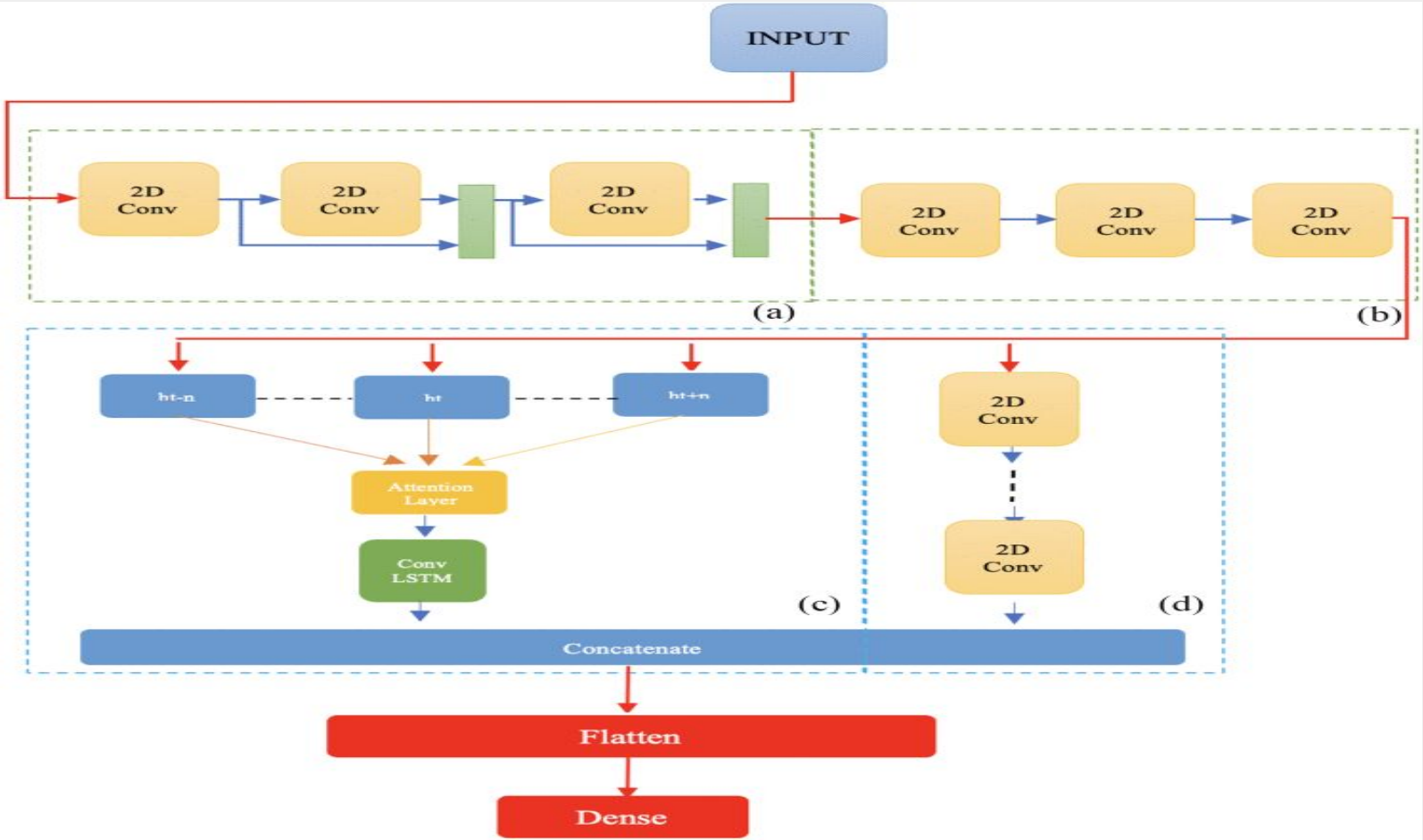
Result: - This study investigated the capacity of interferometric coherence to complement the information from optical data. From the piecewise linear estimation of variable importance, the best score was observed at the heading stage (RDVI1 45% and interferometric coherence_VH 44%) and used as input for the regression models.



Multispectral Crop Yield Prediction Using 3D-Convolutional Neural Networks and Attention Convolutional LSTM Approaches

By: - Seyed Mahdi Mirhoseini Nejad; Dariush Abbasi-Moghadam; Alireza Sharifi; Nizom Farmonov;

Article Link :- https://ieeexplore.ieee.org/document/9956008



Goal: - This study proposed two architecture. The first model includes 2D-CNN, skip connections, and LSTM-Attentions. The second model comprises 3D-CNN, skip connections, and ConvLSTM Attention

Dataset: - MODIS-Land-surface from 2003 to 2018

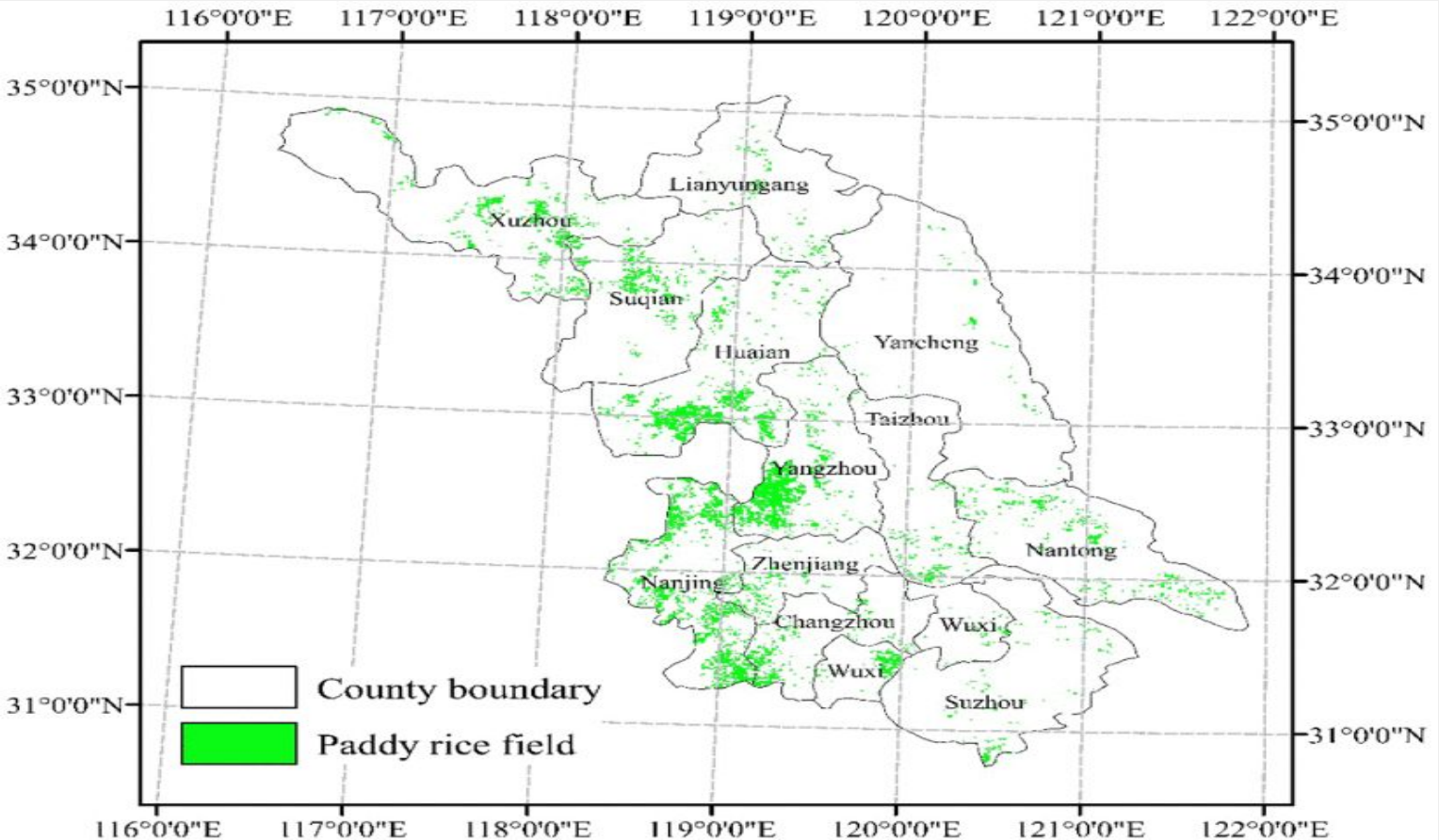
Methodology: - Two novel methods have been proposed by this article, which are the combination of the 2D-CNN and LSTM attention as the first model, and the usage of 3D-CNN and ConvLSTM instated of 2D-CNN and single LSTM as the second model for county-level crop yield prediction. As the first step, multi-2D-CNNs are used with help of the skip connection to extract features. After that, the outputs of the previous step are used for attention LSTM and multicascaded CNN parallelly.

Result: - It is found that a model with three layers of CNN is the most effective forecasting system. Both proposed models have been evaluated with relevant works, including DeepYeild, ConvLSTM, 3D-CNN, and CNN-LSTM, which are tested from 2016 to 2018 in the identical conditions to have a fair comparison. It is finally discovered that the second proposed model achieved the highest score in comparison with the other methods.

Rice Yield Estimation Based on an NPP Model With a Changing Harvest Index

By: - Fumin Wang; Feilong Wang; Jinghui Hu; Lili Xie; Xiaoping Yao

Article Link: - https://ieeexplore.ieee.org/document/9103206



Goal: - Developed an NPP-based rice yield estimation model with HI being a changing parameter.

Dataset: - Jiangsu meteorological bureau & China Meteorological Forcing Dataset

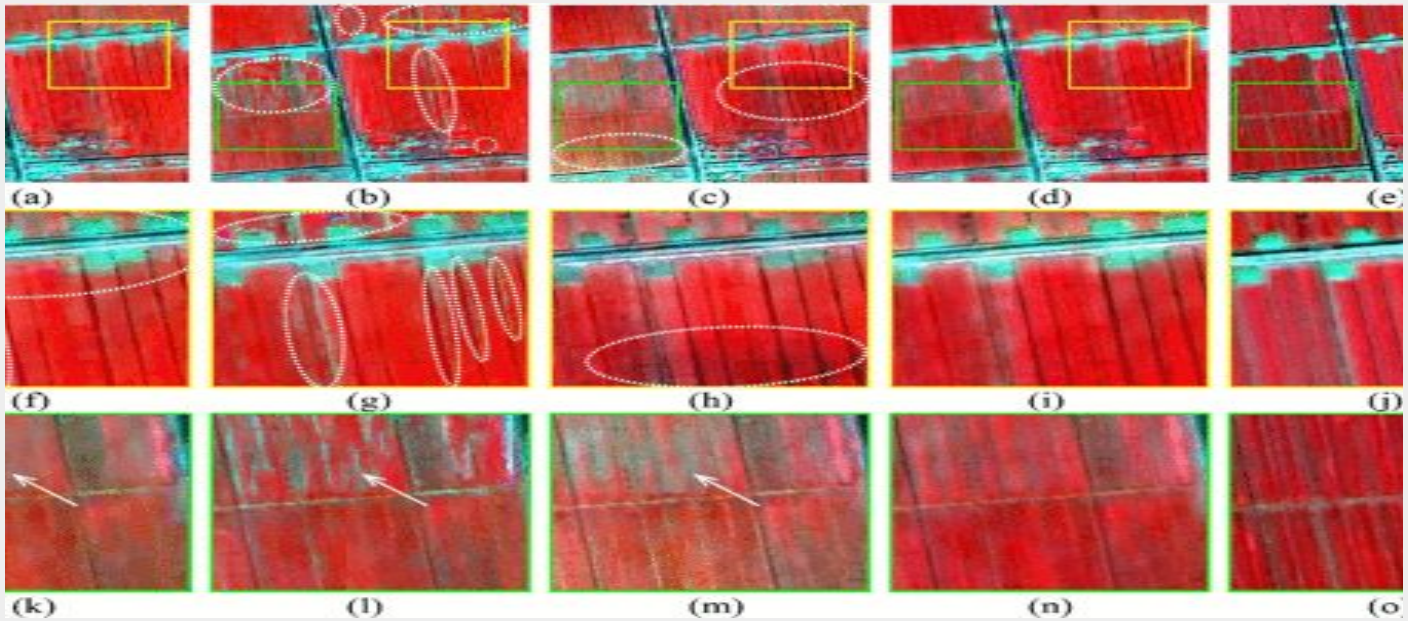
Methodology: - Developed a method for rice yield estimation with an NPP model and a changing HI. Through the NPP model, the rice yield of Jiangsu Province from 2004 to 2014 were obtained with a fixed HI first. However, the overestimation and underestimation of rice yield constantly occur, and the overall estimation accuracy is low. Since a linear increasing trend in the HI was observed, a changing HI was then considered in the NPP model. The evaluation shown that by incorporating the linearly changing HI, the proportion of the relative errors within ±5% is 1.64 times more than those obtained with a fixed HI.

Result: - The conclusion convinced that the improved NPP model with a changing HI can be a powerful method for rice yield estimation by satellite remote sensing.

Spatiotemporal Image Fusion Method for High-Resolution Monitoring of Crops at the Subfield Level

By: - Jiale Jiang; Qiaofeng Zhang; Xia Yao; Yongchao Tian; Yan Zhu; Weixing Cao; Tao Cheng

Article Link: - https://ieeexplore.ieee.org/document/9165877



Goal : -Satellite-based time-series crop monitoring at the subfield level for the efficient implementation of precision crop management.

Data set: - One from real Gaofen-1 (GF-1) and simulated Landsat-like/Sentinel-like images, and the other from real GF-1 and real Landsat/Sentinel-2 data on two sites.

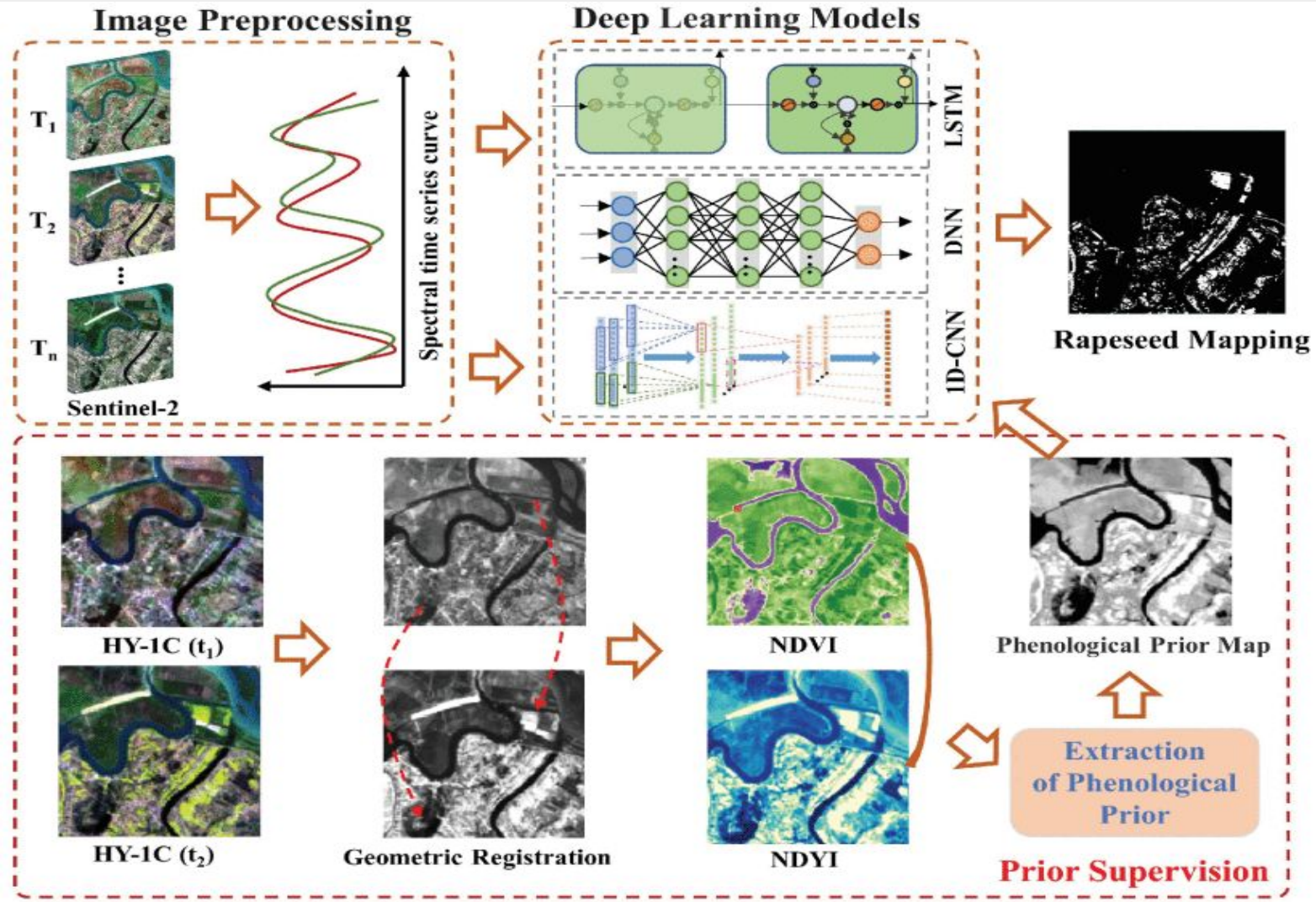
Methodology: - This study proposed a high-resolution spatiotemporal image fusion method (HISTIF) consisting of filtering for cross-scale spatial matching (FCSM) and multiplicative modulation of temporal change (MMTC). In FCSM, we considered both point spread function effect and geo-registration errors between fine and coarse resolution images. Subsequently, MMTC used pixel-based multiplicative factors to estimate the temporal change between reference and prediction dates without image classification.

Result: - The results demonstrated that HISTIF produced substantial reduction in the fusion error from cross-scale spatial mismatch and accurate reconstruction in spatial details within fields, regardless of simulated or real data. The images predicted by STARFM exhibited pronounced blocky artifacts. While the images predicted by HISTIF and Fit-FC both showed clear within-field variability patterns

PPCE: A Practical Loss for Crop Mapping Using Phenological Prior

By: - Bin Yang; Jinyuan Guo; Jianqiang Liu; Xin Ye

Article Link: - https://ieeexplore.ieee.org/document/9991949



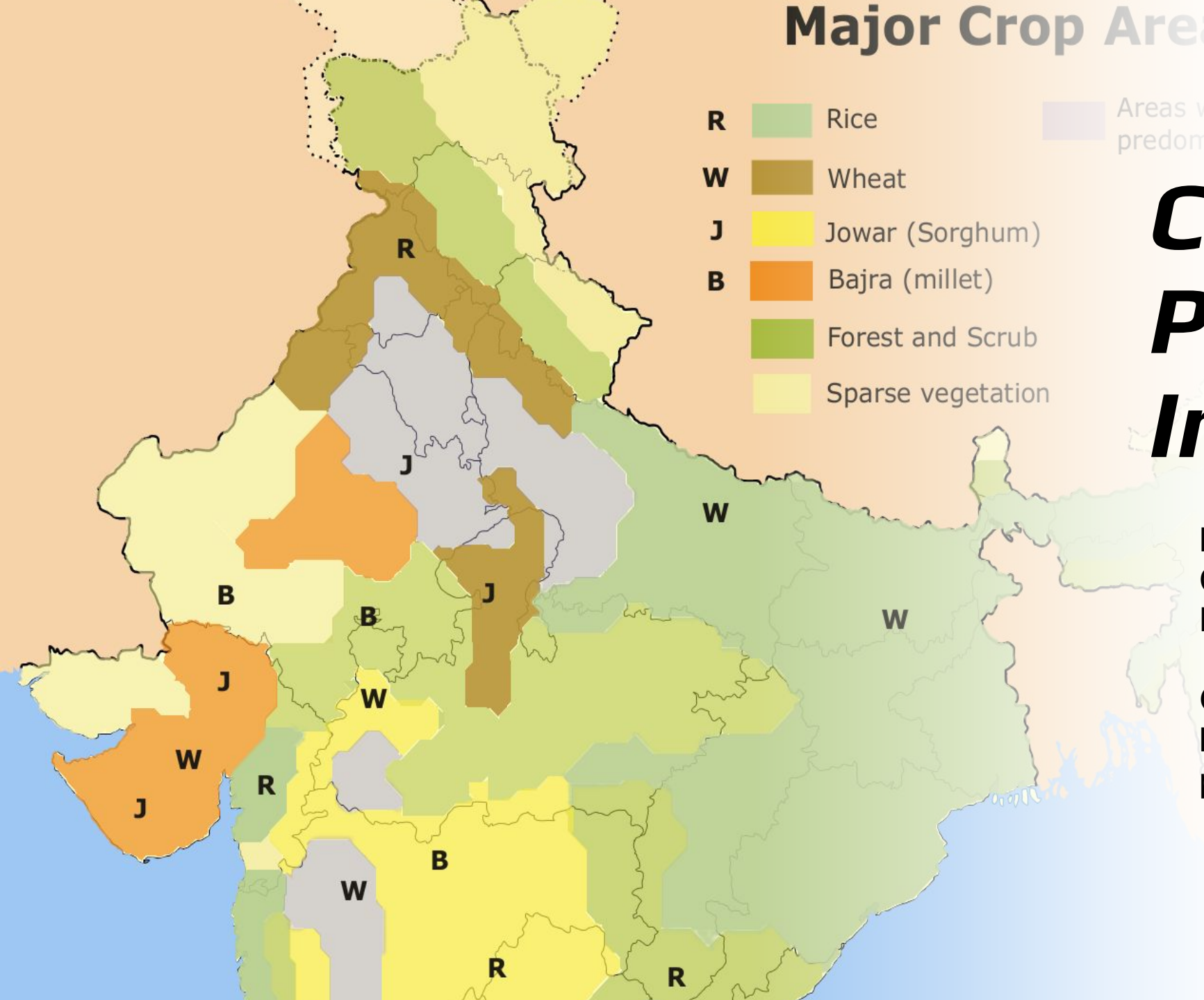
Goal: - Proposes a crop phenological prior cross entropy loss (PPCE) function, which focuses on guiding the training processing in the direction where crops can be better identified.

Dataset: - Data of a particular city named Changde city, China is used in this research

Methodology: - This letter proposes a novel loss function called PPCE loss to improve the rapeseed mapping performance using time series images. It involves rapeseed phenological prior, which is described by NDYI and NDVI, to quantify the prior probability of a pixel being rapeseed. The phenological prior helps to train the network in the direction where rapeseed pixels can be accurately distinguished from other pixels.

Result: - It provides a new tool for the supervision of deep learning models to improve the accuracy of crop mapping. The experimental results compared with five widely used loss functions over three typical study areas using three widely used deep learning models indicate the superior performance of PPCE loss for rapeseed mapping.


Mid-Sem
Presentation



Crop Yield Prediction in India 2022

By: - Rishabh Awasthy
CS-668
Email: - ra35976n@pace.edu

Github: - https://github.com/awsthy/Cs-668-Capstone-Project

Research Question: -

Can we predict crop yield per acre, using data on farming practices and environmental conditions ?

Literature Review

There are total of 8 pages in the literature review from which i learn various technique and the methods for my project. The model include involves employing a Deep-LSTM model with forget, input, and output gates to process the data. The results indicate that the proposed model outperforms existing methods, with a lower RMSE value, showcasing its effectiveness.
There are also data mining and machine learning techniques that are employed to predict optimal crop cultivation based on factors such as soil, weather, and nutrient conditions. The dataset is collected from various sources, including Kaggle and agricultural databases.
Various machine learning techniques like k-Nearest Neighbors, Naïve Bayes, Support Vector Machine, and Linear Regression are used. The research aims to provide valuable insights for farmers and governments to enhance agricultural practices and yield optimization.
And SCA-WRELM technique, targeting efficient rice crop productivity forecasting. The dataset used is a previous SCA-WRELM dataset, and the methodology involves data normalization and the utilization of the WRELM model, optimized using SCA for parameter tuning. The results demonstrate improved predictive outcomes compared to existing methods.

Dataset: -

- **Link to the data: -** https://zindi.africa/competitions/digital-green-crop-yield-estimate-challenge
- **Description : -** This dataset capture detailed information about various agricultural activities in **India year 2022**, including land preparation, cultivation methods, irrigation, fertilization, harvesting, and related factors. It could be useful for analyzing and optimizing agricultural practices in the specified districts and blocks.
- **Size: -** There are 3870 rows and 45 columns in the dataset.

```
[30] data =pd.read_csv("/content/Train (2).csv")
```

```
data.head()
```

ıte	...	Harv_method	Harv_date	Harv_hand_rent	Threshing_date	Threshing_method	Residue_length	Residue_perc	Stubble_use	Acre
-27	...	machine	2022-11-16	NaN	2022-11-16	machine	30	40	plowed_in_soil	0.312500
-20	...	hand	2022-11-25	3.0	2022-12-24	machine	24	10	plowed_in_soil	0.312500
-20	...	hand	2022-12-12	480.0	2023-01-11	machine	30	10	plowed_in_soil	0.148148
-17	...	hand	2022-12-02	240.0	2022-12-29	hand	26	10	plowed_in_soil	0.222222
-21	...	machine	2022-11-30	NaN	2022-12-02	machine	24	40	plowed_in_soil	0.468750

EDA and Methodology

Null values and Data Overview

```
missing_data = data.isnull().sum()
columns_with_missing_data = missing_data[missing_data > 0]
print(columns_with_missing_data)
```

```
RcNursEstDate                          83
SeedlingsPerPit                       289
NursDetFactor                         289
TransDetFactor                        289
TransplantingIrrigationHours          193
TransplantingIrrigationSource         115
TransplantingIrrigationPowerSource    503
TransIrriCost                         882
StandingWater                         238
OrgFertilizers                       1335
Ganaura                              2417
CropOrgFYM                           2674
PCropSolidOrgFertAppMethod           1337
CropbasalFerts                        188
BasalDAP                              543
BasalUrea                            1704
FirstTopDressFert                     485
1tdUrea                               556
1appDaysUrea                          556
2tdUrea                              2694
2appDaysUrea                         2700
MineralFertAppMethod.1                481
Harv_hand_rent                        252
dtype: int64
```

Overview

Overview | Alerts 62 | Reproduction

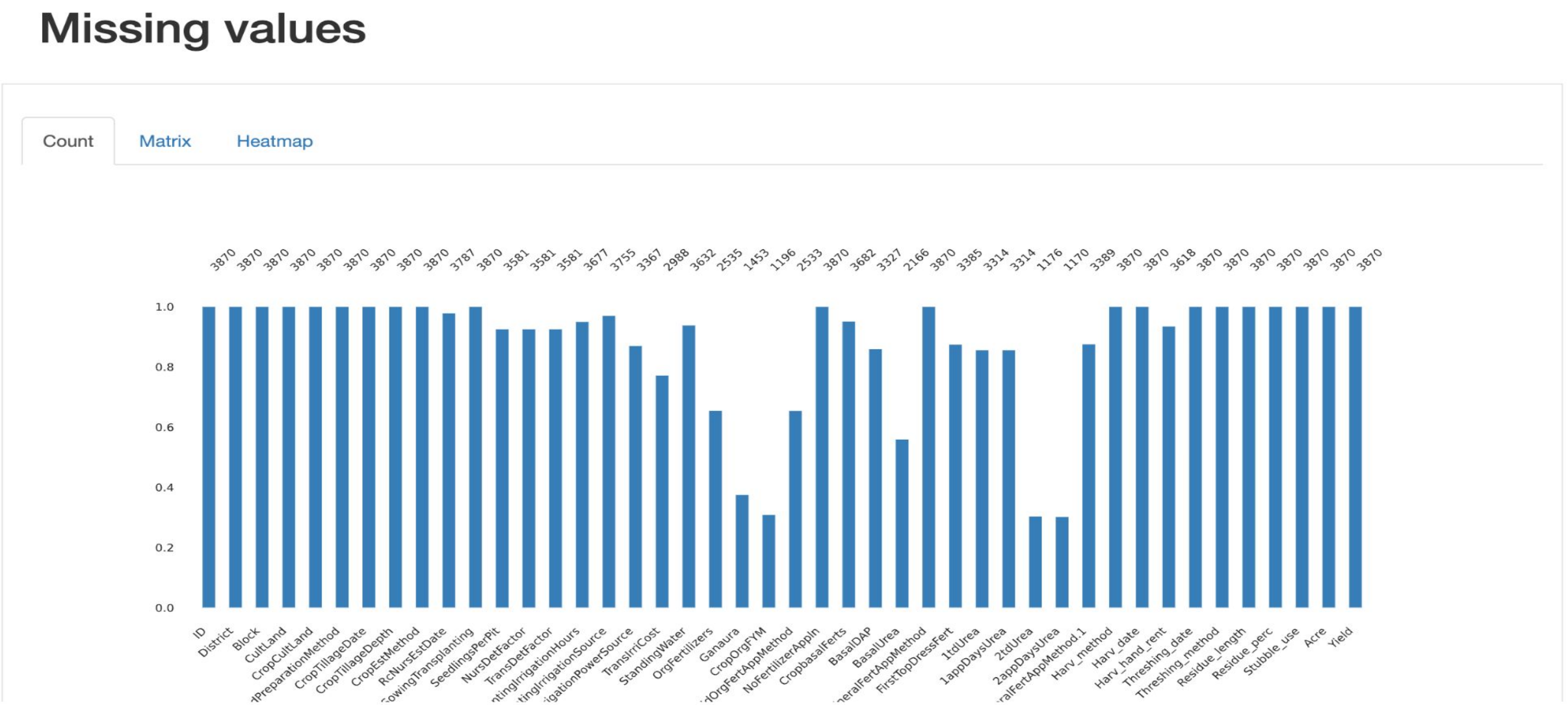
Dataset statistics

Number of variables	44
Number of observations	3870
Missing cells	20803
Missing cells (%)	12.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.3 MiB
Average record size in memory	352.0 B

Variable types

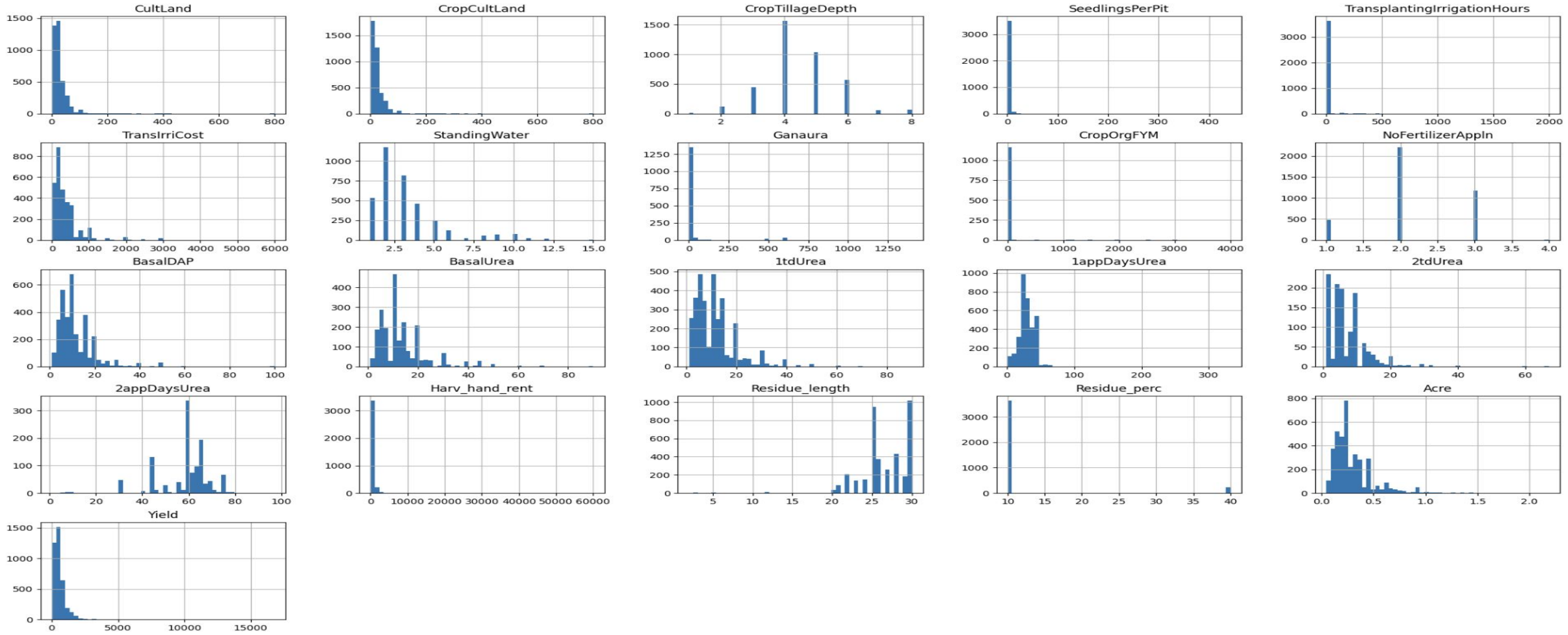
Text	3
Categorical	17
Numeric	19
DateTime	5

Missing Value Graph: -



Distribution of the dataset

```
%matplotlib inline
import matplotlib.pyplot as plt
data.hist(bins=50, figsize=(25,15))
plt.show()
```



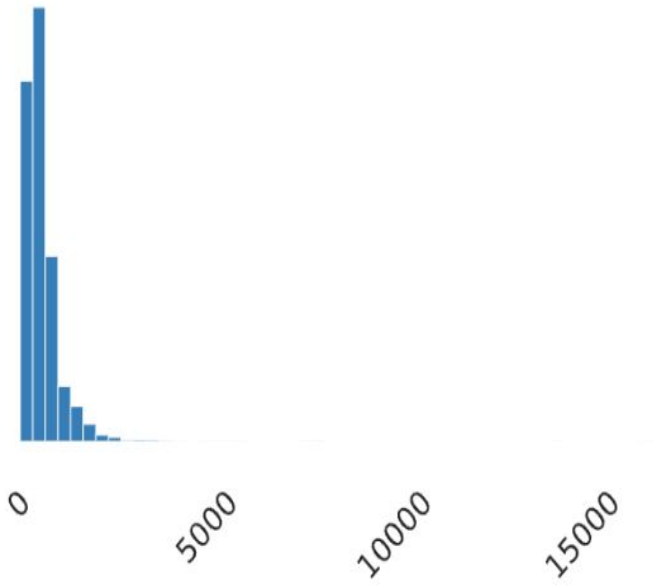
Distribution Of Target Value: -

Yield

Real number (ℝ)

HIGH CORRELATION

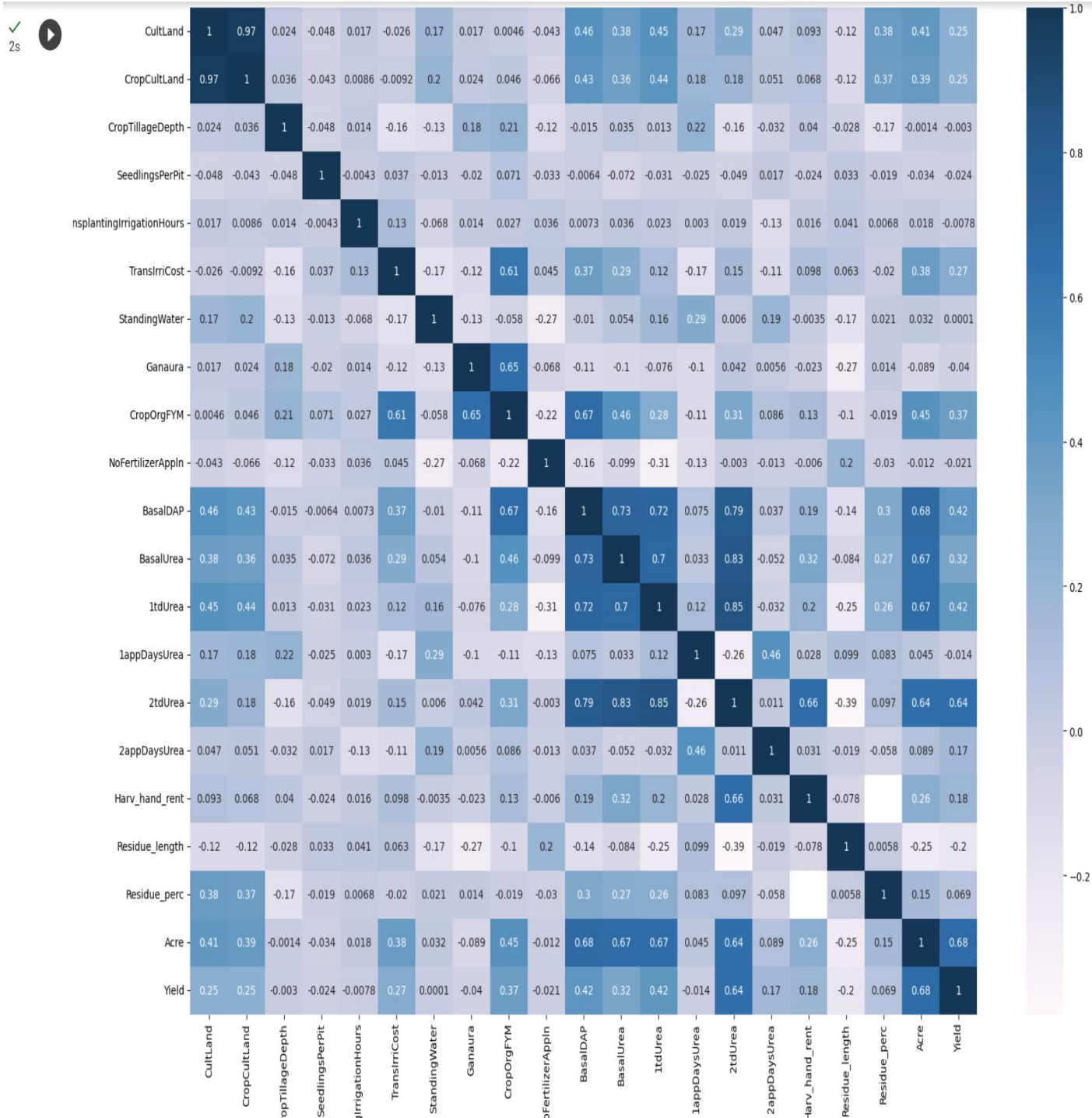
Distinct	379	**Minimum**	4
Distinct (%)	9.8%	**Maximum**	16800
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	594.26925	**Memory size**	30.4 KiB



More details

Correlation Heat-Map & Table with most correlated features with Target value

Yield	1.000000
Acre	0.676554
2tdUrea	0.644100
BasalDAP	0.419260
1tdUrea	0.417841
CropOrgFYM	0.372322
BasalUrea	0.323633
TransIrriCost	0.274761
CultLand	0.254201
CropCultLand	0.246221
Harv_hand_rent	0.181270
2appDaysUrea	0.167423
Residue_perc	0.068579
StandingWater	0.000103
CropTillageDepth	−0.002953
TransplantingIrrigationHours	−0.007763
1appDaysUrea	−0.014110
NoFertilizerAppln	−0.020704
SeedlingsPerPit	−0.023557
Ganaura	−0.040204
Residue_length	−0.204293



NEXT STEPS

- As we can see in the dataset there are some null values, and we can not drop all the values, cause if we do that, lot of information will be lost, so we replace these values either by the values of the columns which are most correlated or else we will drop the values.
- As we can see there are many columns which are string type and hold values, which can affect the the prediction result, so we’ll use the Label encoder

References: -

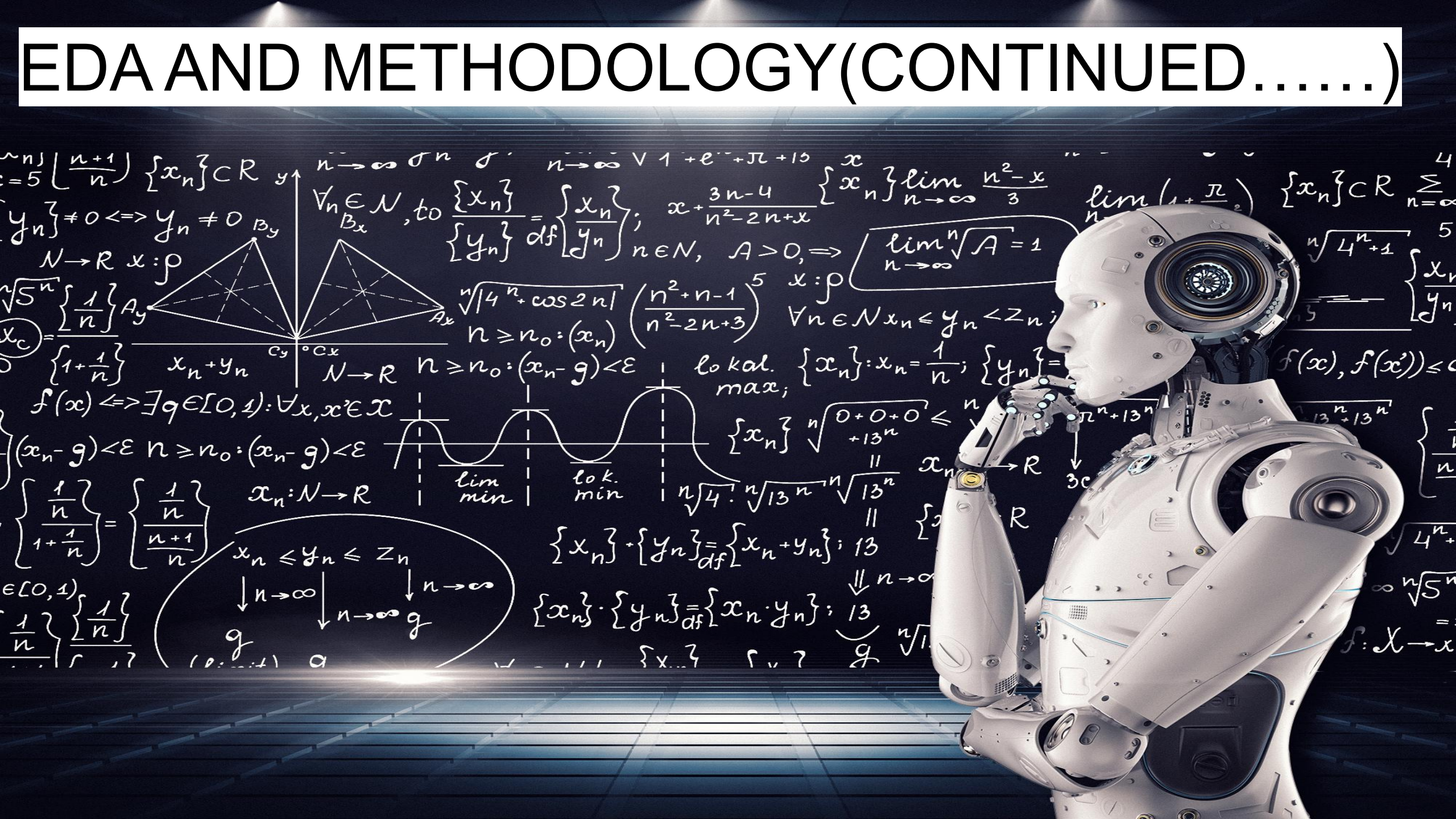
- S. M. M. Nejad, D. Abbasi-Moghadam, A. Sharifi, N. Farmonov, K. Amankulova and M. Lászlź, "Multispectral Crop Yield Prediction Using 3D-Convolutional Neural Networks and Attention Convolutional LSTM Approaches," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 254-266, 2023, doi: 10.1109/JSTARS.2022.3223423.

- P. Saini and B. Nagpal, "Deep-LSTM Model for Wheat Crop Yield Prediction in India," *2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, Sonepat, India, 2022, pp. 73-78, doi: 10.1109/CCiCT56684.2022.00025.

- R. Welekar and C. Dadiyala, "Optimizing Crop Yield in Agriculture using Data Mining and Machine Learning Techniques," *2023 4th International Conference for Emerging Technology (INCET)*, Belgaum, India, 2023, pp. 1-7, doi: 10.1109/INCET57972.2023.10170493.

- S. Thirumal and R. Latha, "Automated Rice Crop Yield Prediction using Sine Cosine Algorithm with Weighted Regularized Extreme Learning Machine," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 35-40, doi: 10.1109/ICICCS56967.2023.10142403.
  https://ieeexplore.ieee.org/document/10142403/

EDA AND METHODOLOGY(CONTINUED……)



Here we replace columns NAN values with other most related column: -

Replacing Null Values by comparing the Correlation of features with each other

```
[419] # Replace missing values in '2tdUrea' with values from '1tdUrea'
      data['2tdUrea'].fillna(data['1tdUrea'], inplace=True)
```

```
[420] # Replace missing values in '2tdUrea' with values from 'BasalUrea'
      data['2tdUrea'].fillna(data['BasalUrea'], inplace=True)
```

```
[421] # Replace missing values in '2tdUrea' with values from 'BasalDAP'
      data['2tdUrea'].fillna(data['BasalDAP'], inplace=True)
```

```
[422] # Replace missing values in '1tdUrea' with values from '2tdUrea'
      data['1tdUrea'].fillna(data['2tdUrea'], inplace=True)
```

```
[423] # Replace missing values in 'BasalDAP' with values from '2tdUrea'
      data['BasalDAP'].fillna(data['2tdUrea'], inplace=True)
```

```
[424] # Replace missing values in 'BasalUrea' with values from '2tdUrea'
      data['BasalUrea'].fillna(data['2tdUrea'], inplace=True)
```

```
[428] data['TransIrriCost'].fillna(data['CropOrgFYM'], inplace=True)
```

0s

Setting a Correlation Threshold of 20% , So that we feed features only that are important to Target Value : -

Keeping Columns That have correlation with target value more than 20%, This step is necessary for Dimensional reduction

```
[435] import pandas as pd

    # Assuming data is your DataFrame
    corr_matrix = data.corr()
    correlation_threshold = 0.20  # Set your desired correlation threshold

    # Get the names of features to keep based on the correlation
    features_to_keep = corr_matrix.columns[corr_matrix["Yield"].abs() >= correlation_threshold].tolist()

    # Display the names of features to keep
    print(features_to_keep)
```

```
['CultLand', 'CropCultLand', 'TransIrriCost', 'BasalDAP', 'BasalUrea', '1tdUrea', '2tdUrea', 'Residue_length', 'Acre', 'Y
```

After Replacing & the setting threshold we check for null values: -

```
data.isnull().sum()
```

```
CultLand          0
CropCultLand      0
TransIrriCost     0
CropOrgFYM        0
BasalDAP          0
BasalUrea         0
1tdUrea           0
2tdUrea           0
Residue_length    0
Acre              0
Yield             0
dtype: int64
```

```
[194] data.head()
```

	CultLand	CropCultLand	TransIrriCost	CropOrgFYM	BasalDAP	BasalUrea	1tdUrea	2tdUrea	2appDaysUrea	Harv_hand_rent	F
0	45	40	200.0	NaN	15.0	20.0	15.0	15.0	NaN	NaN	
1	26	26	125.0	NaN	15.0	10.0	20.0	20.0	NaN	3.0	
2	10	10	80.0	1.0	4.0	5.0	5.0	5.0	NaN	480.0	
3	15	15	NaN	NaN	6.0	3.0	5.0	5.0	NaN	240.0	
4	60	60	300.0	NaN	15.0	30.0	30.0	30.0	NaN	NaN	


BUILDING MODELS

DEEP NEURAL NETWORK USING TENSORFLOW WITH 500 EPOCHS

DNN Using TensorFlow

```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
# Select features and target variable
features = data_set.drop(['Yield'], axis=1)  # Exclude 'ID' from features
target = data_set['Yield']

# One-hot encode categorical variables if needed
# features = pd.get_dummies(features)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)


# Build the neural network model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='linear')  # Assuming a regression task, adjust for classification
])

# Compile the model
model.compile(optimizer='adam', loss='mae')  # Adjust for your specific problem

# Train the model
model.fit(X_train, y_train, epochs=500, batch_size=32, validation_data=(X_test, y_test))
```

DEEP NEURAL NETWORK USING PYTORCH

DNN Using PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import TensorDataset, DataLoader
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
import pandas as pd
import numpy as np
# Select features and target variable
features = data_set.drop(['Yield'], axis=1)  # Exclude 'ID' from features
target = data_set['Yield']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Standardize features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train_scaled, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)  # Reshape to (num_samples, 1)

X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)  # Reshape to (num_samples, 1)

# Build the neural network model
class NeuralNetwork(nn.Module):
    def __init__(self, input_size):
```

K-NEAREST NEIGHBOUR MODEL WITH VARING VALUE OF K FROM 0 TO 3000

KNN Model

```
[477] import numpy as np
      from sklearn.neighbors import KNeighborsRegressor
      from sklearn.metrics import mean_absolute_error
      from sklearn.model_selection import train_test_split, cross_val_score
      import pandas as pd

      # Implement custom Lp norm distance function
      def custom_lp_distance(x, y, p=2):
          return np.power(np.sum(np.abs(x - y) ** p), 1/p)

      # Split the dataset into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

      # Define a range of p values to try
      p_values = [2]

      # Initialize variables to store the best p, k, and minimum MAE
      best_p = None
      best_k = None
      min_mae = float('inf')

      # Perform cross-validation for each p value
      for p_value in p_values:
          # Define a range of k values to try
          k_values = [50, 100, 200, 400, 800, 1600, 3000]

          # Perform cross-validation for each k value
          for k in k_values:
              knn_model = KNeighborsRegressor(n_neighbors=k, metric=custom_lp_distance,weights='distance', p=p_value)
              cv_scores = cross_val_score(knn_model, features, target, cv=5, scoring='neg_mean_absolute_error')
```

ELASTIC-NET MODEL USING L1 RATIO OF 0.5

ElasticNet model

```
[473] from sklearn.linear_model import ElasticNet
      from sklearn.metrics import mean_absolute_error
      from sklearn.preprocessing import StandardScaler
      from sklearn.model_selection import train_test_split
      import pandas as pd
      # Build and train the ElasticNet model
      elastic_net_model = ElasticNet(alpha=1.0, l1_ratio=0.5)  # You can adjust alpha and l1_ratio
      elastic_net_model.fit(X_train_scaled, y_train)

      # Make predictions on the test set
      predictions = elastic_net_model.predict(X_test_scaled)

      # Display the results
      results = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
      print(results)

      # Evaluate the model
      mae = mean_absolute_error(y_test, predictions)
      print(f'Elastic Net Model Mean Absolute Error on Test Set: {mae}')
```

RANDOM FOREST MODEL WITH VALUE OF N =100

Random Forest model

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd

# Build and train the Random Forest model
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can adjust the number of estimator
random_forest_model.fit(X_train_scaled, y_train)

# Make predictions on the test set
Rpredictions = random_forest_model.predict(X_test_scaled)

# Display the results
results = pd.DataFrame({'Actual': y_test, 'Predicted': Rpredictions})
print(results)

# Evaluate the model
mae = mean_absolute_error(y_test, Rpredictions)
print(f'Random Forest Mean Absolute Error on Test Set: {mae}')
```

DECISION TREE MODEL WITH RANDOM STATE =42

Decision Tree model

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
# Build and train the Decision Tree model
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train_scaled, y_train)

# Make predictions on the test set
predictions = decision_tree_model.predict(X_test_scaled)

# Display the results
results = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
print(results)

# Evaluate the model
mae = mean_absolute_error(y_test, predictions)
print(f'Decision Tree Mean Absolute Error on Test Set: {mae}')
```

SUPPORT VECTOR REGRESSOR MODEL USING RBF KERNEL

Support Vector Regressor Model

```python
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd

# Build and train the SVR model
svr_model = SVR(kernel='rbf')  # 'rbf' stands for radial basis function, you can experiment with other kernels
svr_model.fit(X_train_scaled, y_train)

# Make predictions on the test set
predictions = svr_model.predict(X_test_scaled)

# Display the results
results = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
print(results)

# Evaluate the model
mae = mean_absolute_error(y_test, predictions)
print(f'SVR Mean Absolute Error on Test Set: {mae}')
```

LINEAR REGRESSION MODEL

Linear Regression model

```
[469] from sklearn.linear_model import LinearRegression
      from sklearn.metrics import mean_absolute_error
      from sklearn.preprocessing import StandardScaler
      from sklearn.model_selection import train_test_split
      import pandas as pd
      # Build and train the Linear Regression model
      linear_model = LinearRegression()
      linear_model.fit(X_train_scaled, y_train)

      # Make predictions on the test set
      predictions = linear_model.predict(X_test_scaled)

      # Display the results
      results = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
      print(results)

      # Evaluate the model
      mae = mean_absolute_error(y_test, predictions)
      print(f'Linear Regression Mean Absolute Error on Test Set: {mae}')
```

WE USE TOTAL 8 MODELS TO COMPARE THE PERFORMANCE

MEAN ABSOLUTE ERROR OF ALL THE MODELS

Decision Tree Mean Absolute Error on Test Set= 158.6299095607235

DNN Using TensorFlow Mean Absolute Error on Test Set= 144.74821615711 97

KNN Using PyTorch Mean Absolute Error on Test Set= 142.15101623535156

KNN Mean Absolute Error= 245.63562130710153

Elastic Net Model Mean Absolute Error on Test Set= 214.25858467801345

Random Forest Mean Absolute Error on Test Set= 141.68252056813716

SVR Mean Absolute Error on Test Set= 274.74520402721856

Linear Regression Mean Absolute Error on Test Set= 173.46452090283591

TOP THREE MODELS

FROM MAE RESULTS TOP 3 MODELS WITH LOWER MAE FOR OUR DATA ARE: -

1. RANDOM FOREST WITH MAE =141.68
2. DNN USING PYTORCH WITH MAE = 142.15
3. DNN USING TENSORFLOW WITH MAE = 144.74

Conclusion, FutureWork & Limitation

***So, Now we can say, Yes we can predict Yield per Acre using data on farming practices and environmental conditions like, CultLand, CropCultLand, TransIrriCost, BasalDAP, BasalUrea, 1tdUrea, 2tdUrea, Residue_length*.**

Future works include refining the predictive model, exploring new features, addressing regional specificity, collaborating with experts, and extending the scope to other crops. However, **limitations** such as data quality, environmental dynamics, and generalization challenges must be considered. Overall, the project holds promise for improving the livelihoods of smallholder farmers, with ongoing efforts needed for model enhancement, adaptability to regional nuances, and addressing inherent limitations.

LINK TO THE NOTEBOOK : -
https://colab.research.google.com/drive/1DwNXXqAsxq-X4uAEWbezfLuJFrkQPTVr?usp=sharing

References: -

- S. M. M. Nejad, D. Abbasi-Moghadam, A. Sharifi, N. Farmonov, K. Amankulova and M. Lászlź, "Multispectral Crop Yield Prediction Using 3D-Convolutional Neural Networks and Attention Convolutional LSTM Approaches," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 254-266, 2023, doi: 10.1109/JSTARS.2022.3223423.

- P. Saini and B. Nagpal, "Deep-LSTM Model for Wheat Crop Yield Prediction in India," *2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, Sonepat, India, 2022, pp. 73-78, doi: 10.1109/CCiCT56684.2022.00025.

- R. Welekar and C. Dadiyala, "Optimizing Crop Yield in Agriculture using Data Mining and Machine Learning Techniques," *2023 4th International Conference for Emerging Technology (INCET)*, Belgaum, India, 2023, pp. 1-7, doi: 10.1109/INCET57972.2023.10170493.

- S. Thirumal and R. Latha, "Automated Rice Crop Yield Prediction using Sine Cosine Algorithm with Weighted Regularized Extreme Learning Machine," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 35-40, doi: 10.1109/ICICCS56967.2023.10142403.
  https://ieeexplore.ieee.org/document/10142403/