Crop Yield Prediction using ML

Contents

1. *Introduction*
2. *Problem Statement*
3. *Motivation*
4. *Project Objectives*
5. *Implementation*
6. *Comparision Analysis*
7. *Future Prospects*
8. *Conclusion*
9. *Bibliography and References*

Introduction

Life of a farmer

- Agriculture has always been practiced and been a major source of income in India for centuries due to the country’s gifted fertile soils and terrain.
- About 54% of the population of India is engaged in some form of agricultural practices in the present.
- Each year, farmers put in tremendous efforts in the hopes of gaining a good amount of yield from their harvests.
- Various factors affect crop yield, hence it becomes important for farmers to know how much yield is expected beforehand.

Problem Statement:

To find the best -fit ML model for Crop-Yield Prediction

Motivation

- Farming comes with lot of physical labor and hefty costs of farming equipments.
- The motivation behind this project is to help farmers make right decisions about farming choices and practices.
- This project will help farmers to avoid unforeseen dangers of low yields which can cause their efforts go in vain.
- Instead of compensating with heavy losses caused due to ignorant decisions, this model will try to give accurate predictions about yield produced beforehand.

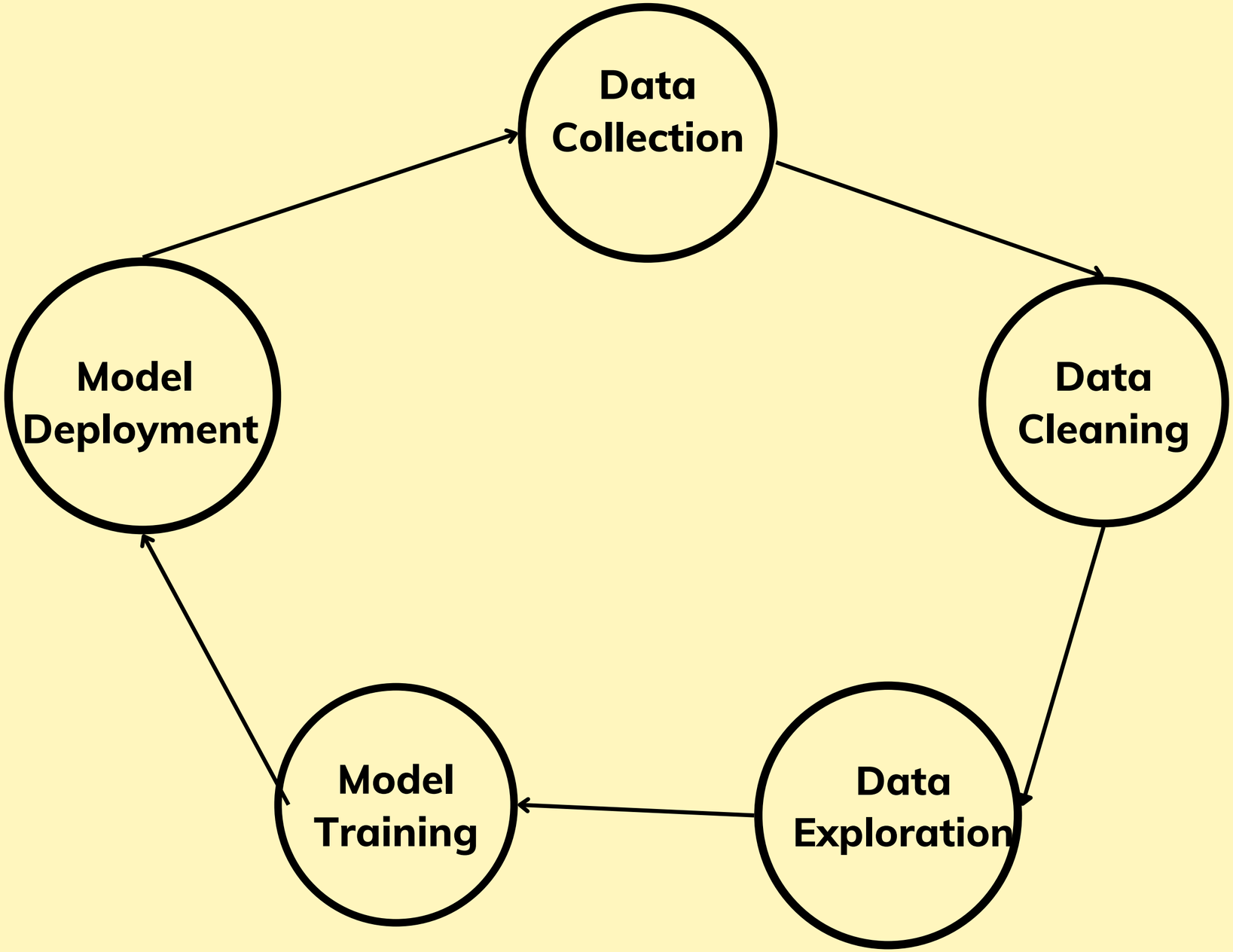
Project Objective

1. Gain higher accuracies on applied ML models.

2. Explore and draw significant conlusions from data

3. Comparision Analysis of ML Algorithms

Data Science Life-Cycle



Tools and Technologies used

- **Programming Language:Python**
- **Libraries used: Pandas, Numpy, Sklearn, MatPlotLib**
- **Data-set: picked from Kaggle**
- **Coding Platform: Google Collab**

Implementation

Step1: Data Collection

- We will be working on a crop yield dataset collected from Indian states . The dataset has been picked from kaggle.

- The dataset consists of 10 features: Crop, Crop_Year, Season, State, Area, Production, Annual_Rainfall, Fertilizer, Pesticide, Yield

Crop	Crop_Year	Season	State	Area	Productio	Annual_Ra	Fertilizer	Pesticide	Yield
Arecanut	1997	Whole Yea	Assam		56708	2051.4		22882.34	0.796087
Arhar/Tur	1997	Kharif	Assam	6637	4685	2051.4	631643.3	2057.47	0.710435
Castor see	1997	Kharif	Assam	796	22	2051.4	75755.32	246.76	0.238333
Coconut	19997	Whole Yea	Assam	19656		2051.4	1870662	6093.36	5238.052
Cotton(lin	1997	Kharif	Assam	1739	794	2051.4	165500.6	539.09	0.420909
Dry chillie	1997	Whole Yea	Assam	13587	9073	2051.4	1293075	4211.97	0.643636
Gram	1997	Rabi	Assam	2979	1507	2051.4	283511.4	923.49	0.465455
Jute	1997	Kharif	Assam	94520	904095	2051.4	8995468	29301.2	9.919565
Linseed	1997	Rabi	Assam	10098	5158	2051.4	961026.7	3130.38	0.461364

step2: Data Cleaning

- A clean dataset helps in increasing productivity and better decision making.
- Unclean data can produce complications making it difficult to work with

Some data cleaning methods used in the project are:

1. Removing Invalid Years
2. Handling NaN values:
    - Replace NaN values in Production and Fertilizer columns with mean
    - replace NAN values in Annual_Rainfall field with value above the current cell
3. Removing duplicate rows.
4. Removing rows with too many missing values.

step3: Data Exploration

Data exploration helps us to draw some conclusions from dataset .

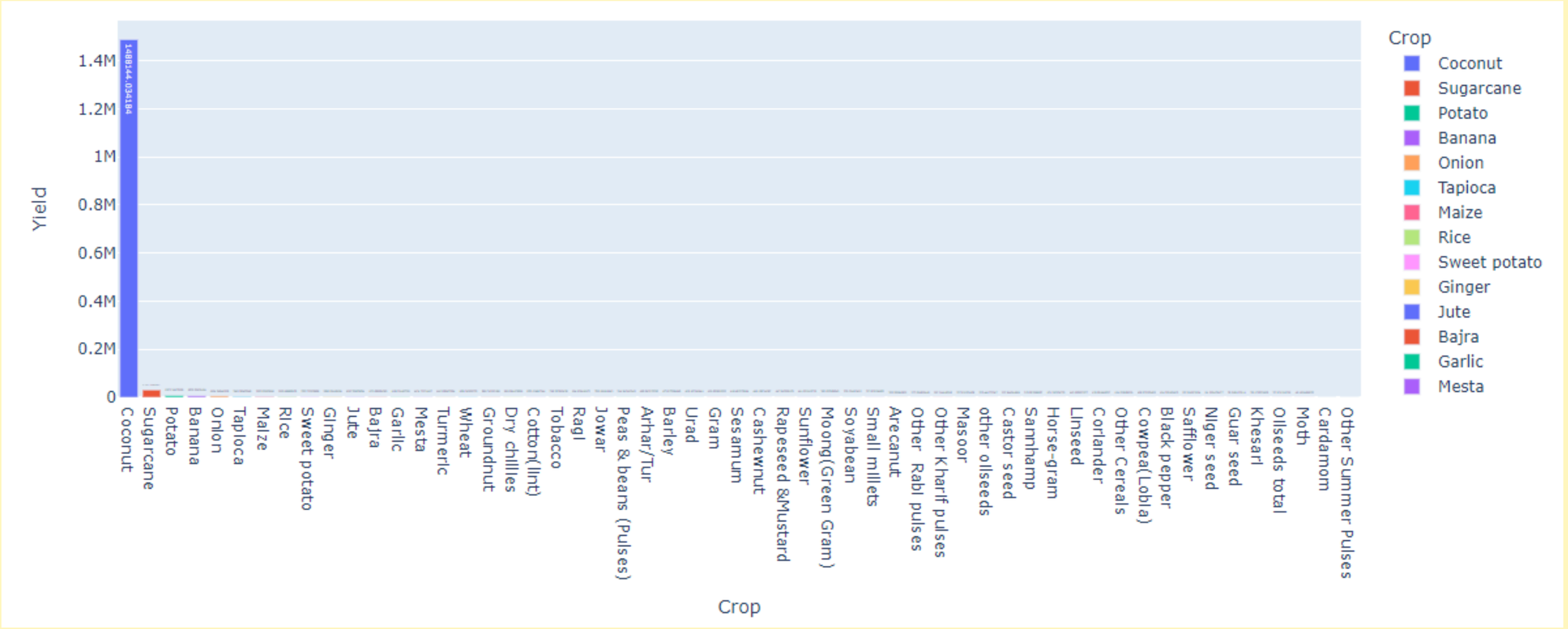
```
df.describe(include='object')
```

	Crop	Season	State
count	19662	19662	19662
unique	55	6	30
top	Rice	Kharif	Karnataka
freq	1194	8224	1432

From the table, we can conclude that:

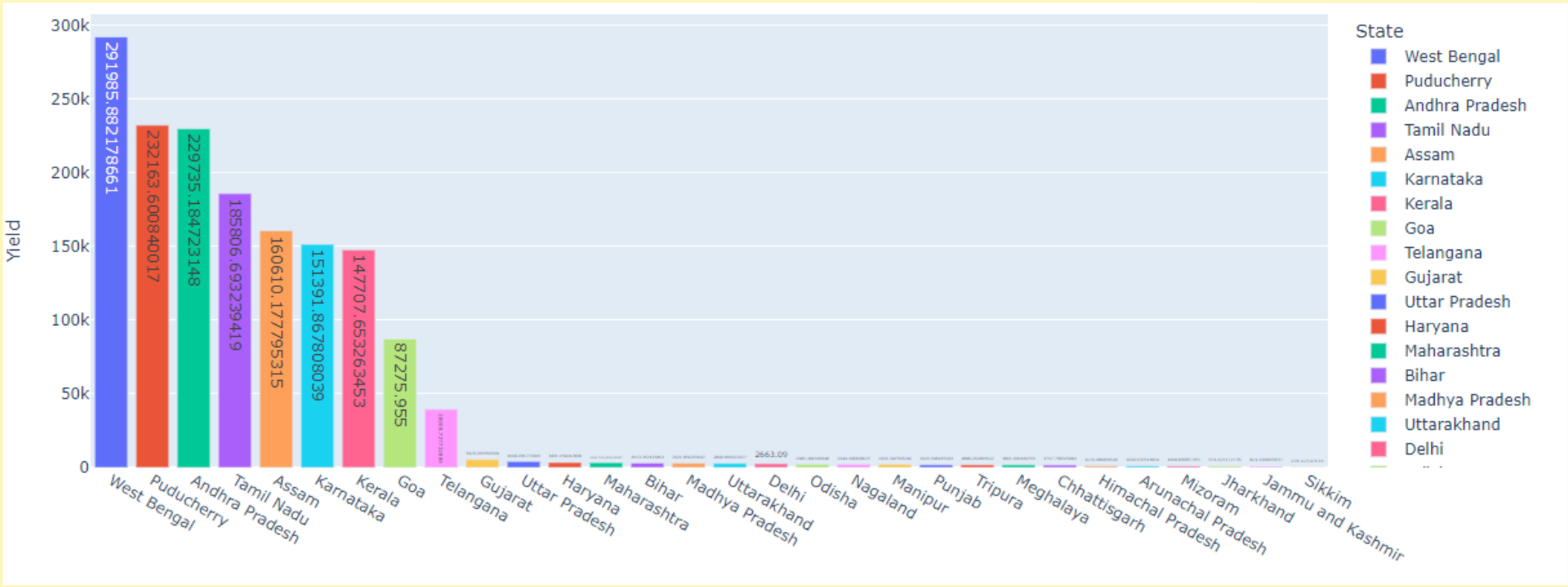
- Rice is the most widely grown crop
- Most crops are grown in the Kharif season

Crop vs Yield



Crop that produced most yield is **Coconut.**

State vs Yield



West Bengal produced the most yields followed by Ponducherry and Andhra Pradesh

step 4. Model Training

We have used the following machine learning algorithms on our dataset.

1. Decision Tree

2. Random Forest

3. Gradient Boosting

4. Linear Regression

1. **Decision Tree**

```
#Decision tree regressor
from sklearn.tree import DecisionTreeRegressor
import numpy as np
from sklearn import metrics
#metrics consists functions for calculating mean_erros,r^2_errors etc
dt=DecisionTreeRegressor(criterion='squared_error',max_depth=8,min_samples_split=10,random_state=42)
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
train_score=dt.score(x_train,y_train)
test_score=dt.score(x_test,y_test)
print("Mean absolute error: ", metrics.mean_absolute_error(y_test,y_pred))
print("root mean squared error: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("train score: ",train_score)
print("test score: ",test_score)
```

```
Mean absolute error:  7.885557105962199
root mean squared error:  92.36191517722625
train score:  0.9950863065267068
test score:  0.9894199256567109
```

Getting a very high score in just one try is questionable , hence we must always consider cases of overfitting before fully trusting the output

K-Fold Cross Validation

One of the solutions to fix overfitting is K-fold Cross Validation

- K-Fold cross validation uses resampling procedure to evaluate on limited data.
- Dataset is divided into K-folds and scores at each fold is calculated and stored
- Mean of the scores at each fold is calculated

```python
#k-fold validation for Decision Tree
from sklearn.model_selection import KFold, cross_val_score
from sklearn.tree import DecisionTreeRegressor
k=10
kf=KFold(n_splits=k,shuffle=True,random_state=42)
fold_scores=[]
dt=DecisionTreeRegressor(criterion='squared_error',max_depth=10,min_samples_split=10,random_state=42)
for train_index, test_index in kf.split(x):
  x_train,x_test=x.loc[train_index],x.loc[test_index]
  y_train,y_test=y[train_index],y[test_index]
  dt.fit(x_train,y_train)
  score=dt.score(x_test,y_test)
  fold_scores.append(score)

i=1
for fold in fold_scores:
  print("fold",i)
  print(fold)
  i=i+1

mean_score=np.mean(fold_scores)
print(" ")
print("mean score")
print(mean_score)
```

```
K-Fold validation on Decision Tree
fold 1
0.958243728216493 2
fold 2
0.9877084191390149
fold 3
0.7606625550234026
fold 4
0.9686386705205299
fold 5
0.8500340220618406
fold 6
0.9720186945218228
fold 7
0.8932188767199122
fold 8
0.9652903367594926
fold 9
0.9562269153804106
fold 10
0.9925593720988285

mean score
0.9304601590441747
```

Although the current score less than previous score, it is more trustable and correct

3. Gradient Boosting

```
mean absolute error:  12.56571340290899
root mean square error:  3.544815002635397
train score:  0.9947521039380619
test score:  0.9786894584626662
```

K - fold validation for Gradient Boosting

```
mean score
0.9640949146872735
```

4. Linear Regression

```
mean absolute error:  141.82411609924586
root mean square error:  11.908993076631033
train score:  0.34155094275859965
test score:  0.33010536112142475
```

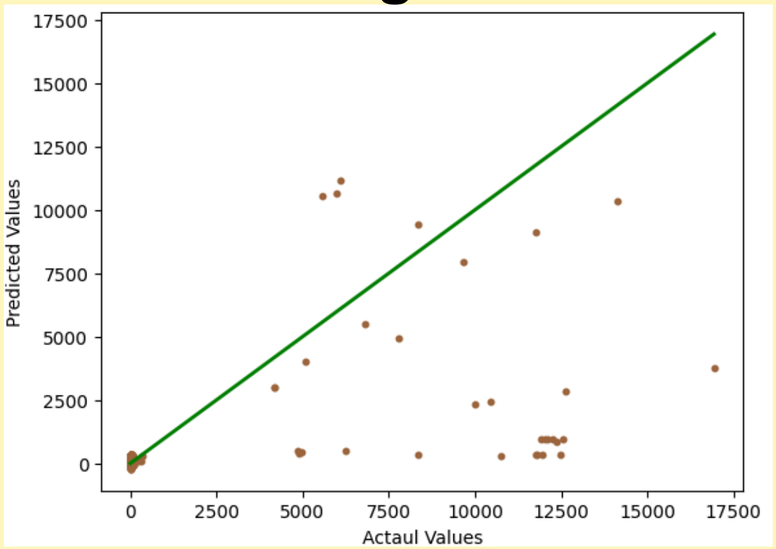
K- fold validation for Linear regression

```
mean score
0.34267923704900644
```

Although the current score is greater than previous score, the model doesn’t perform well in linear regression as the mean score is only 34%.

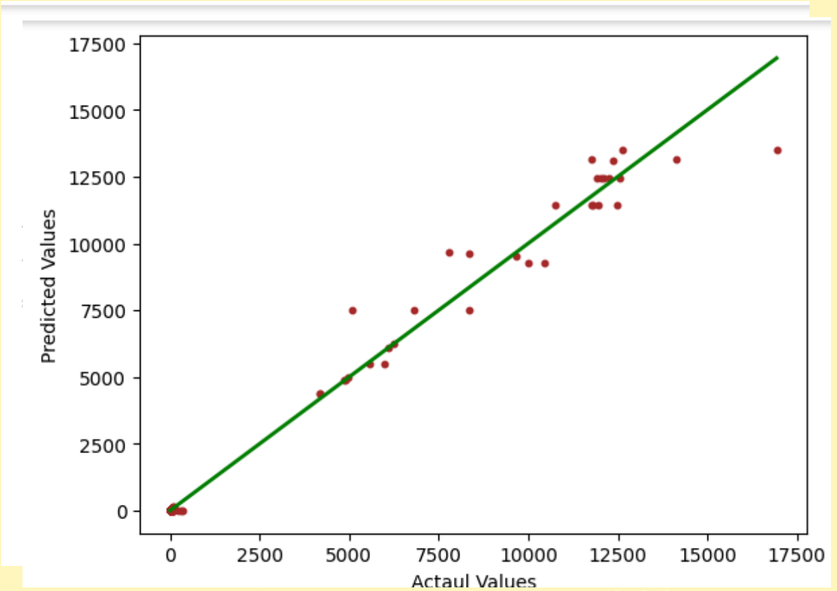
Actual_Yield vs Predicted_Yield

Linear Regrerssion



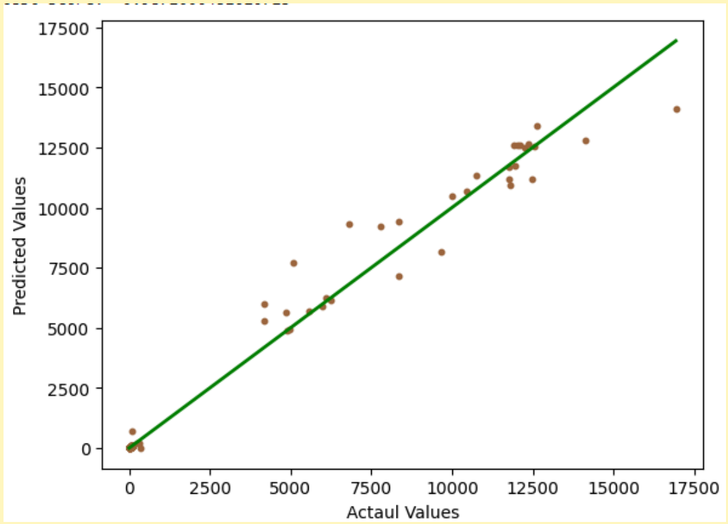
Accuracy Score: 34%

Decision Tree



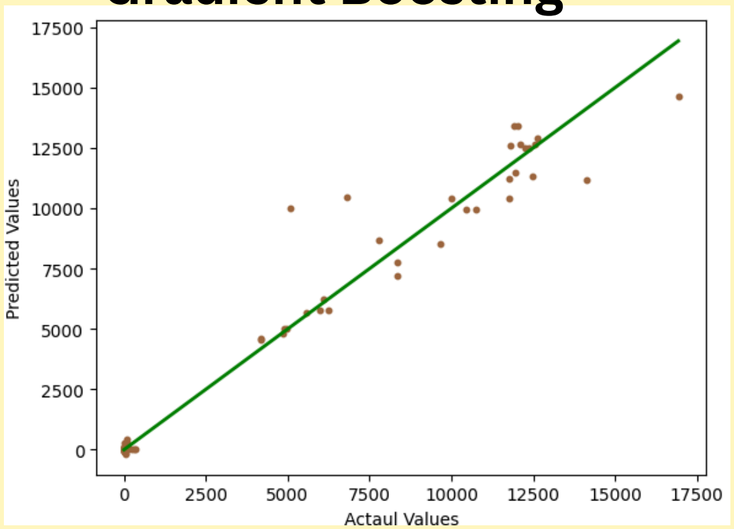
Accuracy Score: 93%

Random Forest



Accuracy Score: 95%

Gradient Boosting



Accuracy Score: 96%

Comparison Analysis

- The algorithms of Decision Tree , Random Forest and Gradient Boosting performs very good on the model with the score of 98.94% , 98.72% and 97.86% respectively . On the other hand, Linear Regression does not perform well with a score of only 34%.

- However, such high scores of Decision tree , Random Forest and Gradient Boosting on first try can be a major sign of overfitting, so we need to perform a check on it.

- Hence , after performing K- fold cross-validation , we get the scores of the algorithms a little less , i.e. , Decision Tree (93.04 %) , Random Forest (95.77 %) and Gradient Boosting (96.40 %) which is more acceptable.

Model	Accuracy Score	Mean Absolute Error	Root Mean Square Error
Linear Regression	0.342	141.82	11.90
Decision Tree	0.936	7.88	92.36
Random Forests	0.957	7.37	101.5
Gradient Boosting	0.964	12.56	3.54

Future Prospects

- We look forward to increase the accuracy scores of the applied algorithms using optimization techniques like feature scaling, gradient decent etc.
- Apply advanced algorithms like Artificial Neural Networks to improve predictive capabilities of the crop yield.
- Apply Deep Learning concepts like CNN for image analysis and recommend crops to plant.

Conclusion

- The application of machine learning techniques in crop yield prediction represents a significant advancement in agricultural science, offering the potential to enhance food security, optimize resource usage, and support economic planning.
- With the help of this project, we can help farmers and others involved in farming to have an idea about expected yield based on their current conditions.
- It can help in taking better decisions and some significant inferences can be made from this prediction system.
- By continuing to refine these techniques and address existing challenges, we can move towards a more resilient and efficient agricultural system, better equipped to meet the demands of a growing global population.

Bibliography and References

1) https://www.kaggle.com/datasets/akshatgupta7/crop-yield-in-indian-states-dataset
2) https://www.fao.org/home/en
3) https://www.analyticsvidhya.com/
4) https://www.w3schools.com/datascience/
5) https://timesofindia.indiatimes.com/readersblog/shameem-ahmad/the-life-of-an-indian-farmer-36314/

-

Research Papers:

- B. N. Singh, A. K. Misra, and R. C. Chaurasia, "Crop yield prediction using machine learning algorithms," International Journal of Computer Applications, vol. 162, no. 8, pp. 21-25, Mar. 2017.
- J. Jeong, J. Resop, N. Mueller, and C. Johnson, "Random forests for global and regional crop yield predictions," PLoS ONE, vol. 11, no. 6, pp. e0156571, Jun. 2016.
- X. You, L. Zhang, and J. Zhang, "A crop yield prediction model based on machine learning," in Proc. 2017 IEEE Int. Conf. Comput. Sci. Eng. (CSE), 2017, pp. 174-179.

THANK YOU