Analyzing Crop Recommendation Systems with Explainable Artificial Intelligence

A Report Submitted

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Tejash Pratap Singh (20214233)

Yash Kumar Nayak (20214303)

Shashi Raj (20214263)

Shrinivash (20214219)

to the

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY ALLAHABAD PRAYAGRAJ

May, 2024

UNDERTAKING

I declare that the work presented in this report titled "***Analyzing crop recommendation systems with explainable artificial intelligence***", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, for the award of the Bachelor of Technology degree in Computer Science & Engineering, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

May, 2024

Prayagraj

Tejash Pratap Singh (20214233)

Yash Kumar Nayak (20214303)

Shashi Raj (20214263)

Shrinivash (20214219)

CERTIFICATE

Certified that the work contained in the report titled ” ***Analyzing crop recommendation systems with explainable artificial intelligence***”, by

Tejash Pratap Singh 20214233

Yash Kumar Nayak 20214303

Shashi Raj 20214263

Shrinivash 20214219,

has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

May, 2024

Dr. Pragya Dwivedi

Computer Science and Engineering Dept.

M.N.N.I.T, Allahabad

Preface

As part of our B.Tech. curriculum to gain practical knowledge in Explainable AI, we undertook a project to enhance the **crop recommendation systems**. Utilizing advanced machine learning techniques, specifically employing LIME and SHAP technique to understand the black box functionalities of the model. The project encompasses various concepts and implementations detailing the processes involved in explaining crop recommendation through machine learning and explainable AI. By meticulously documenting our methodology and results, we aim to contribute to the expanding knowledge base in XAI CROP. Our primary objective is to bridge the gap between sophisticated machine learning techniques and the practical needs of farmers, ensuring that recommendations are not only precise but also comprehensible. This initiative represents a collective effort to delve deeper into the mechanisms of AI transparency and interpretability, acknowledging the paramount importance of understanding how AI systems arrive at their decisions.

Acknowledgement

We would like to express our sincere gratitude to several individuals for supporting us throughout this project. First, we express our sincere gratitude to our mentor, **Dr. Pragya Dwivedi**, for her enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times during the project. Her immense knowledge and profound experience have been instrumental in the successful completion of this project. Without her support and guidance, this project would not have been possible. We also wish to express our sincere thanks to Motilal Nehru National Institute of Technology, Allahabad, for accepting us into the graduate program. In addition, we are deeply indebted to the Ministry of Education, Government of India, for supporting our college.

Contents

1. Introduction

Explainable AI (XAI) is pivotal for transparent and trustworthy crop recommendation systems in agriculture. By providing clear explanations for recommendations, XAI fosters trust and adoption among farmers. These explanations offer valuable insights into the factors driving recommendations, empowering farmers to make informed decisions aligned with local conditions. Additionally, XAI facilitates the identification of biases and errors in AI models, ensuring more reliable recommendations. Collaborative efforts between AI experts and domain-specific stakeholders are enhanced, enabling a deeper understanding of agricultural data and improving the overall effectiveness of crop recommendation systems. Ultimately, XAI enhances transparency, interpretability, and accountability, thereby maximizing the benefits of AI technology in optimizing agricultural practices and crop yields.

1.1. Motivation

The passage underscores the profound importance of agricultural reform in African nations to combat poverty, hunger, and malnutrition, especially in the face of climate change-induced challenges. These challenges, ranging from shifting rainfall patterns to the spread of pests and diseases, pose significant threats to crop production and food security. Consequently, there is a pressing need to develop robust predictive models capable of anticipating the effects of climate change on crop yields. Machine learning (ML) offers a promising solution, as evidenced by recent studies such as those by Schlenker and Roberts [1], which demonstrated the utility of ML models in disease detection. Additionally, ML techniques, including hyperparameter optimization and ensemble learning, have been successfully applied in predicting heart disease [2]. By leveraging ML algorithms and integrating diverse datasets, researchers can enhance our understanding of agricultural dynamics and devise strategies to mitigate the adverse impacts of climate change on African agriculture, ultimately advancing global food security and socio-economic development.

1.2. Problem Statement

This study aims to analyze a crop dataset to understand the factors influencing crop yields, train machine learning models for accurate prediction, and apply explainable AI (XAI) algorithms like LIME and SHAP to elucidate model predictions. With agriculture facing challenges from climate change, predicting crop yields accurately is crucial. However, the opaque nature of machine learning models hinders understanding. By leveraging XAI techniques, this research seeks to provide transparent and interpretable insights into the decision-making process of crop yield prediction models. Ultimately, this study aims to enhance agricultural decision-making and contribute to sustainable food production practices.

2. Related Algorithms

2.1. Regression Algorithms

Regression algorithms in machine learning aim to predict continuous numeric values based on input features. These algorithms analyze the relationship between independent variables (features) and a dependent variable (target) to make predictions. Popular regression algorithms include Linear Regression, which assumes a linear relationship between features and the target; Decision Trees, which partition feature space into segments; and Support Vector Regression, which finds the optimal hyperplane to separate data points. Additionally, ensemble methods like Random Forest and Gradient Boosting combine multiple models to improve prediction accuracy. Regression algorithms are widely used in various domains, including finance, healthcare, and environmental science, for forecasting and decision-making tasks.

2.1.1. Logical Regression

The linear regression model is a statistical technique used to predict a continuous outcome variable based on its relationship with one or more predictor variables. Imagine a straight line drawn through points on a graph. This line represents the linear relationship between the variables. By analyzing this line's slope and intercept, you can estimate the future value of the outcome variable for new data points. It's a foundational tool for uncovering trends and making predictions in various fields like economics, finance, and machine learning.

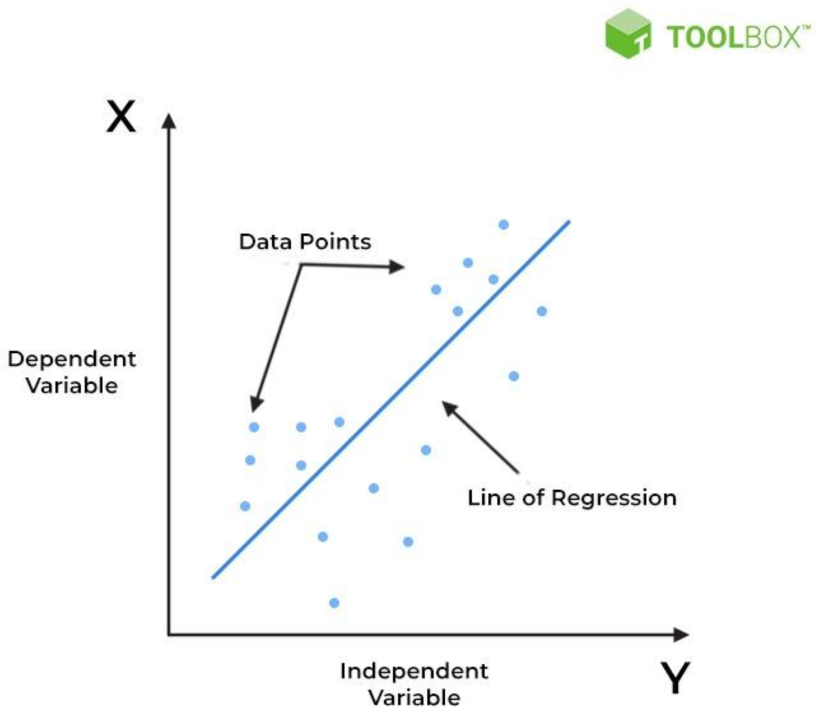


Figure- 1 Linear Regression [3]

2.1.2. Supported Vector Machine

Support Vector Regression (SVR) is a machine learning algorithm used for continuous prediction tasks. Unlike linear regression, it finds a hyperplane that fits the data points within a margin of tolerance, allowing for some error. This approach helps manage non-linear relationships and outliers in the data. SVR utilizes kernel functions to handle complex data patterns, making it a powerful tool for various regression problems.

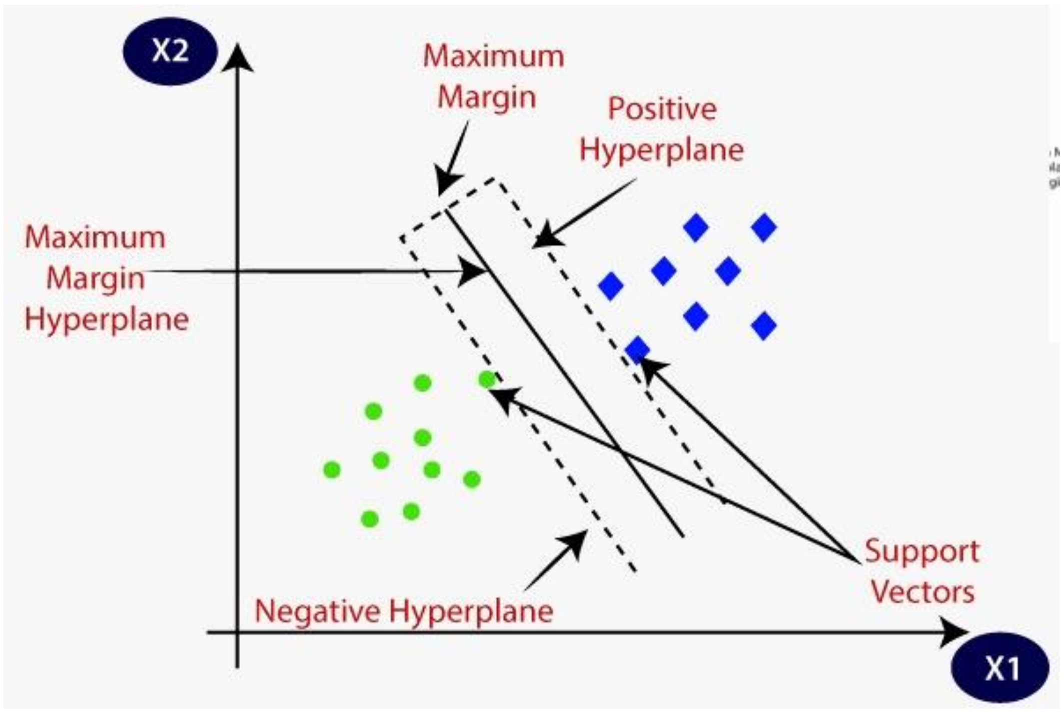
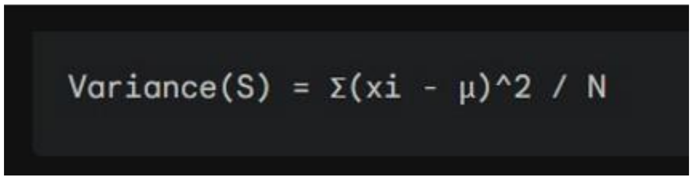


Figure-2 SVM [4]

2.1.3. Decision Tree

A decision tree regression model predicts continuous values by asking a series of yes/no questions about the data. Imagine a flowchart where each split depends on a feature's value. You navigate the tree based on your data, reaching a final prediction based on the terminal node you land on. This approach is interpretable, meaning you can understand the logic behind the predictions. However, decision trees can be prone to overfitting if not carefully grown.

Variance, in the context of decision tree regression, measures the spread of the target variable's values within a node. A lower variance indicates the data points in that node are closer in value, making it a more homogeneous group.



```
Variance(S) = Σ(xi - μ)^2 / N
```

Figure-3 Decision Tree [5]

Where:

S: Set of data points at a particular node.

Xi: Value of the target variable for data point i.

μ: Average value of the target variable in set S.

N: Total number of data point in set S.

2.1.4. Random Forest

Random Forest is a powerful ensemble learning technique in machine learning that combines multiple decision trees to improve predictive performance. Each decision tree in the forest is trained on a random subset of the training data and a random subset of features, leading to diverse trees. During prediction, the output of all trees is averaged (for regression tasks) or aggregated (for classification tasks) to make the final prediction. Random Forest mitigates overfitting, handles high-dimensional data, and provides feature importance measures. Its versatility, robustness, and ability to handle complex datasets make it a popular choice for various classification and regression tasks in fields such as finance, healthcare, and ecology.

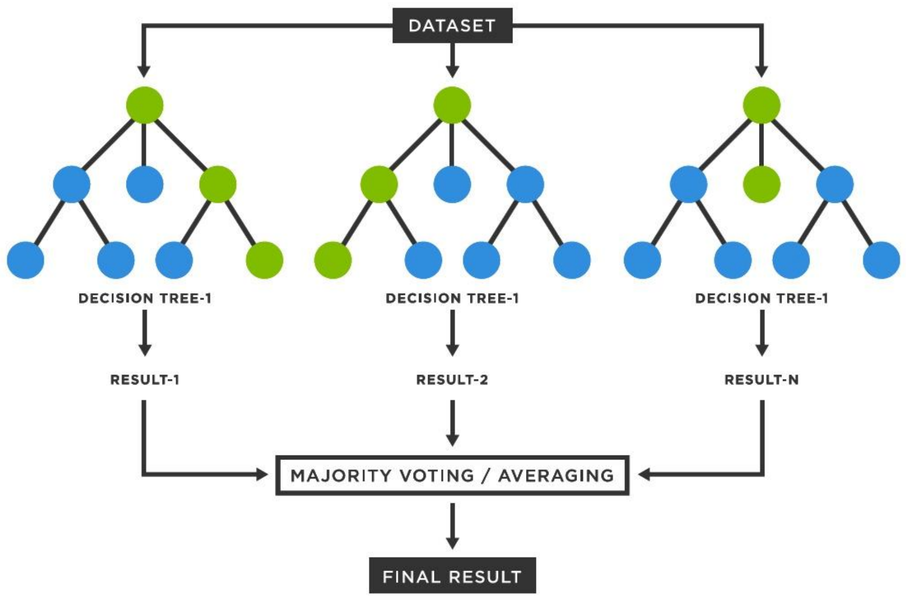


Figure-4 Random Forest [6]

2.1.5. Gradient Boosting

Gradient Boosting is a powerful machine learning technique used for both regression and classification tasks. It builds a strong predictive model by combining multiple weak learners, typically decision trees, in a sequential manner. Each subsequent tree is trained to correct the errors of the previous ones, with a focus on minimizing the loss function. Gradient Boosting iteratively improves the model's predictive accuracy by fitting new trees to the residuals of the previous predictions. This iterative process results in a highly accurate ensemble model capable of capturing complex relationships in the data. Gradient Boosting is known for its robustness and ability to handle large datasets effectively.

2.2. XAI Algorithms

Explainable AI (XAI) algorithms provide interpretable explanations for complex machine learning models' predictions. Techniques like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) highlight the contribution of input features, enhancing transparency and trustworthiness in AI systems, crucial for informed decision-making and regulatory compliance.

2.2.1. LIME

LIME (Local Interpretable Model-agnostic Explanations) is an explainable AI technique that provides local explanations for black-box machine learning models. It works by approximating the complex model's behavior around a specific prediction by generating interpretable explanations. LIME perturbs input data points and observes how the model's predictions change, fitting a simpler, interpretable model to these perturbed data points. By examining the coefficients of this local model, LIME identifies the most influential features for a given prediction, offering insights into why the model made a particular decision. LIME is widely used for enhancing transparency and trustworthiness in AI systems across various domains.

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}(\mathcal{L}(f, g, \pi_x) + \Omega(g)) \qquad (1)$$

Where:

- *G* is a class of potentially interpretable models (e.g, linear regression, decision trees, etc..) whose domain is a binary vector with the same dimension of *X'* and indicates the presence or absent of the *interpretable components.*
- *Ω(g)*, where g belongs to *G*, is a measure of complexity (opposed to interpretability). It might represent, for instance, the number of non-zero weights for a linear model, or the depth of the tree for decision trees.
- *f* is the model being explained.
- $\pi_x(z)$ is a proximity measure between data points, which assumes high values when *x* and *z* are "close", low values if they are "far".
- $\mathcal{L}(f,g,\pi_x)$ is a loss function which evaluates the level of inaccuracy of *g* in approximating *f* within the defined locality π_x.

2.2.2. SHAP

SHAP (SHapley Additive exPlanations) algorithms provide a unified framework for explaining the output of machine learning models. By calculating Shapley values, which quantify the contribution of each feature to the model's prediction, SHAP algorithms offer insights into the importance and impact of individual features. These explanations are not only intuitive but also consistent across different models and prediction instances. SHAP algorithms enable users to understand complex model decisions, identify influential features, and gain actionable insights into the underlying relationships between features and predictions, thereby enhancing model interpretability and trustworthiness.

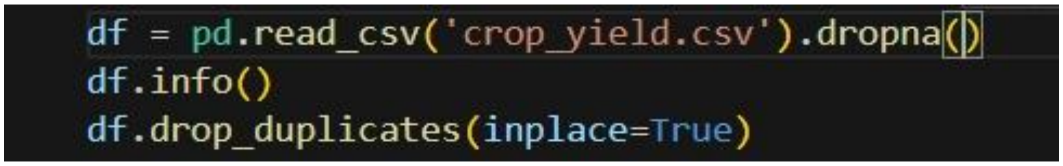
3. MLDLC

MLDLC (Machine Learning Driven Life Cycle) is a framework encompassing the entire lifecycle of a machine learning project, from data collection and preprocessing to model training and deployment. It emphasizes iterative development, collaboration between stakeholders, and continuous monitoring and optimization for effective implementation of ML solutions.

3.1. Data Preprocessing

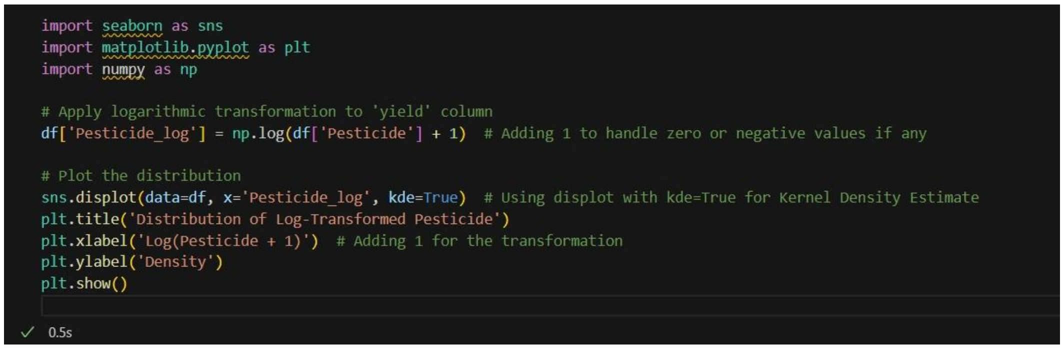
Data preprocessing is a crucial step in machine learning that involves cleaning, transforming, and organizing raw data to make it suitable for analysis and modeling. This process includes handling missing values, encoding categorical variables, scaling numerical features, and splitting data into training and testing sets. By preparing the data appropriately, preprocessing enhances the quality and reliability of machine learning models, leading to more accurate predictions and insights from the data.

Here we removed duplicates and NaN values from input dataframe.



```python
df = pd.read_csv('crop_yield.csv').dropna()
df.info()
df.drop_duplicates(inplace=True)
```

Then we applied Logarithmic transformation on Production, Pesticide, Yield, Area, Fertilizer columns of our dataframe as shown-



```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Apply logarithmic transformation to 'yield' column
df['Pesticide_log'] = np.log(df['Pesticide'] + 1)  # Adding 1 to handle zero or negative values if any

# Plot the distribution
sns.displot(data=df, x='Pesticide_log', kde=True)  # Using displot with kde=True for Kernel Density Estimate
plt.title('Distribution of Log-Transformed Pesticide')
plt.xlabel('Log(Pesticide + 1)')  # Adding 1 for the transformation
plt.ylabel('Density')
plt.show()
```

```
✓ 0.5s
```

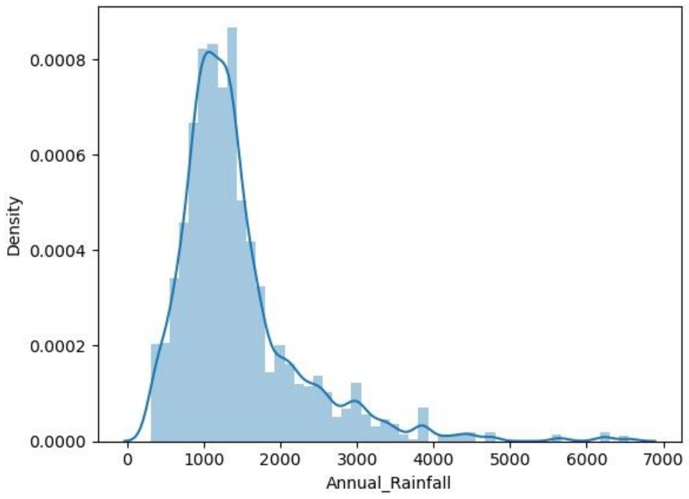


Figure- 5 Before Transformation

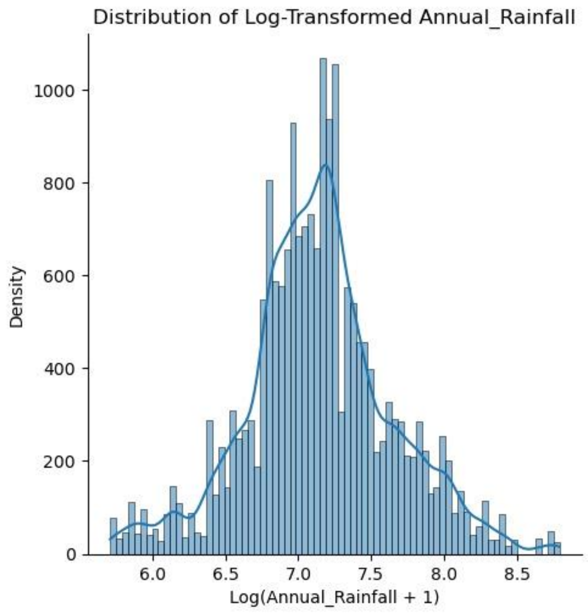
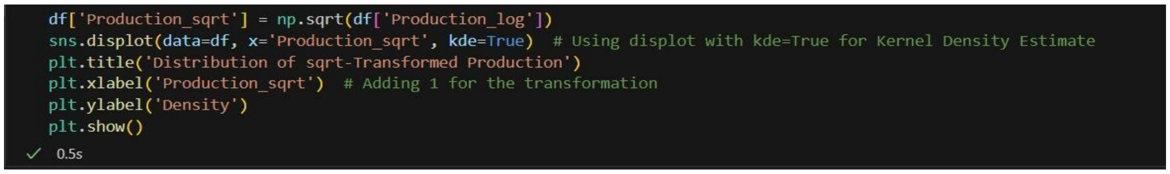


Figure-6 After Transformation



```
df['Production_sqrt'] = np.sqrt(df['Production_log'])
sns.displot(data=df, x='Production_sqrt', kde=True)  # Using displot with kde=True for Kernel Density Estimate
plt.title('Distribution of sqrt-Transformed Production')
plt.xlabel('Production_sqrt')  # Adding 1 for the transformation
plt.ylabel('Density')
plt.show()
```
✓ 0.5s

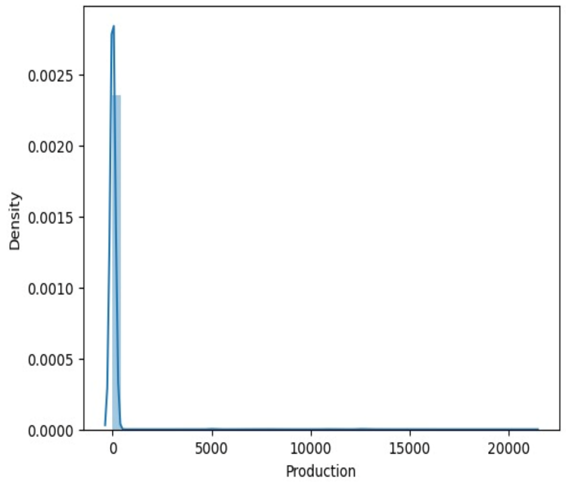


Figure-7 Before transformation

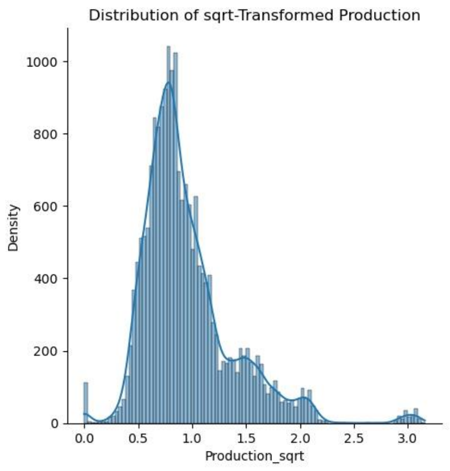
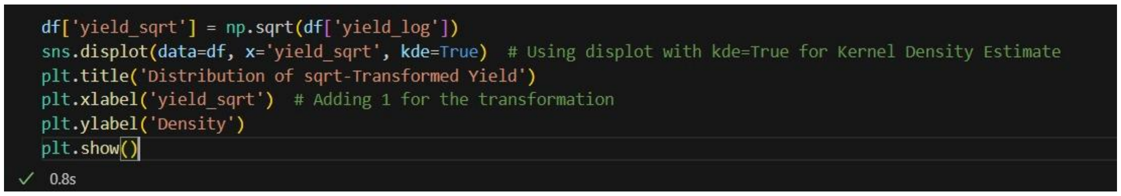
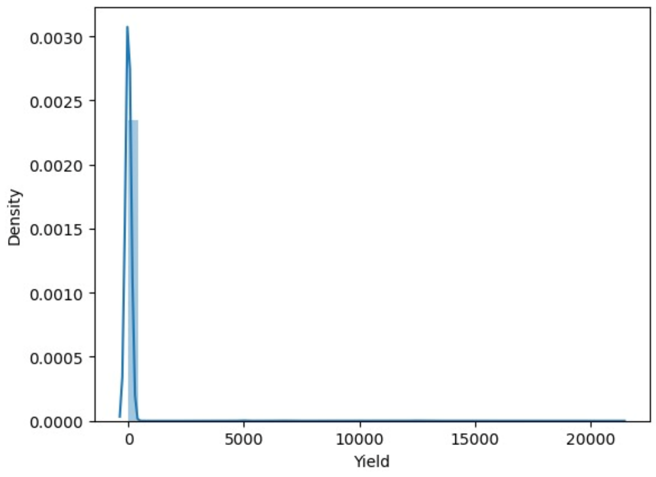


Figure-8 After Transformation



```
df['yield_sqrt'] = np.sqrt(df['yield_log'])
sns.displot(data=df, x='yield_sqrt', kde=True)  # Using displot with kde=True for Kernel Density Estimate
plt.title('Distribution of sqrt-Transformed Yield')
plt.xlabel('yield_sqrt')  # Adding 1 for the transformation
plt.ylabel('Density')
plt.show()
```

✓ 0.8s



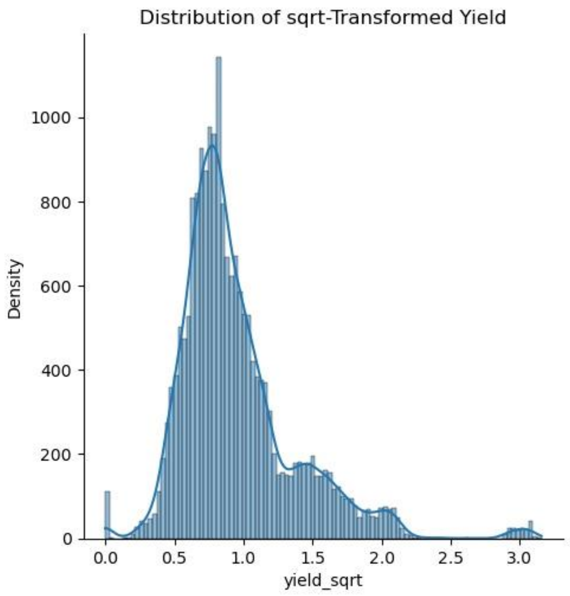
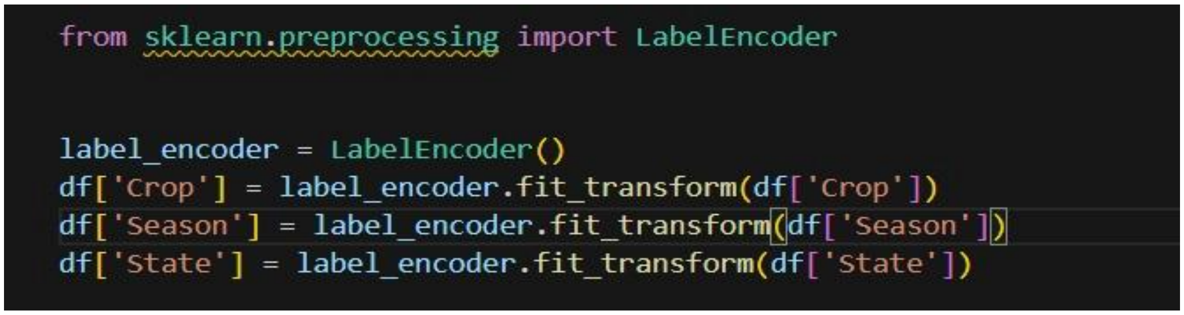


Figure-9 Before Transformation   Figure-10 Before Transformation

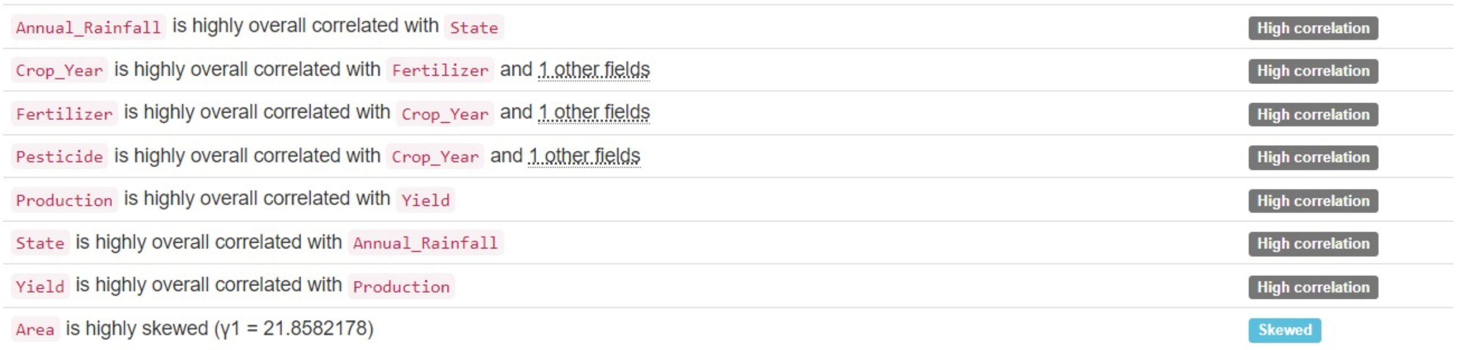
Crop, Season, States were in nominal form so we encoded them into numerical form using Label Encoder.



```
from sklearn.preprocessing import LabelEncoder


label_encoder = LabelEncoder()
df['Crop'] = label_encoder.fit_transform(df['Crop'])
df['Season'] = label_encoder.fit_transform(df['Season'])
df['State'] = label_encoder.fit_transform(df['State'])
```

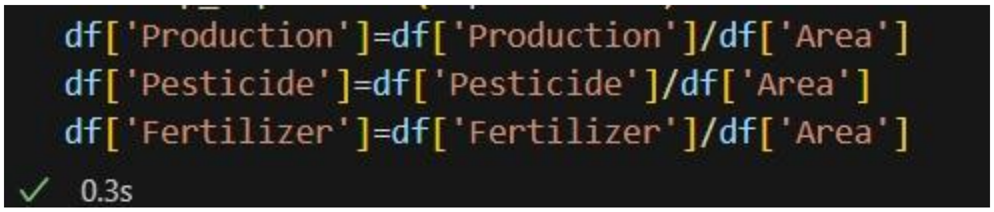
3.2. Feature Selection

Feature selection is the process of identifying and selecting the most relevant features from a dataset to improve model performance and efficiency. By eliminating irrelevant or redundant features, feature selection reduces dimensionality, mitigates overfitting, and enhances model interpretability. Techniques for feature selection include filter methods, wrapper methods, and embedded methods, which evaluate feature importance based on statistical metrics, model performance, or intrinsic properties of the learning algorithm, respectively.



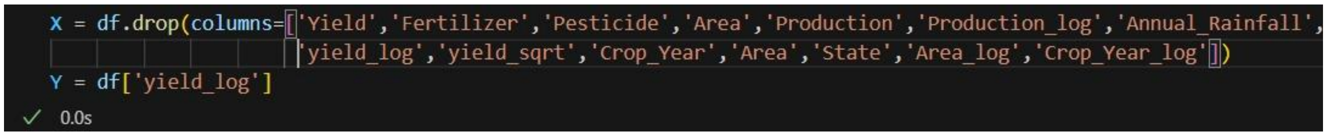
Annual_Rainfall is highly overall correlated with State	High correlation
Crop_Year is highly overall correlated with Fertilizer and 1 other fields	High correlation
Fertilizer is highly overall correlated with Crop_Year and 1 other fields	High correlation
Pesticide is highly overall correlated with Crop_Year and 1 other fields	High correlation
Production is highly overall correlated with Yield	High correlation
State is highly overall correlated with Annual_Rainfall	High correlation
Yield is highly overall correlated with Production	High correlation
Area is highly skewed (γ1 = 21.8582178)	Skewed

Since Area was in high correlation with Production, Pesticide, Fertilizer so we carried out dimensionality reduction by combining it with three as shown.

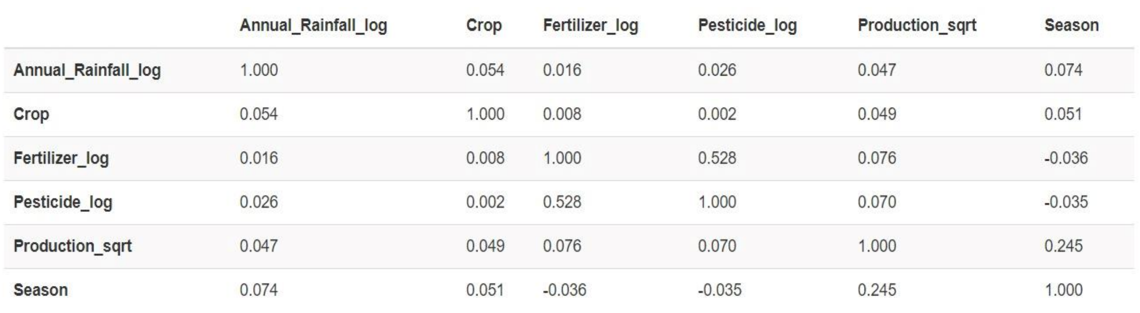


```
df['Production']=df['Production']/df['Area']
df['Pesticide']=df['Pesticide']/df['Area']
df['Fertilizer']=df['Fertilizer']/df['Area']
✓ 0.3s
```

Here we selected important independent features that had more correlation with yield.



```
X = df.drop(columns=['Yield','Fertilizer','Pesticide','Area','Production','Production_log','Annual_Rainfall',
                     'yield_log','yield_sqrt','Crop_Year','Area','State','Area_log','Crop_Year_log'])
Y = df['yield_log']
✓ 0.0s
```



	Annual_Rainfall_log	Crop	Fertilizer_log	Pesticide_log	Production_sqrt	Season
Annual_Rainfall_log	1.000	0.054	0.016	0.026	0.047	0.074
Crop	0.054	1.000	0.008	0.002	0.049	0.051
Fertilizer_log	0.016	0.008	1.000	0.528	0.076	-0.036
Pesticide_log	0.026	0.002	0.528	1.000	0.070	-0.035
Production_sqrt	0.047	0.049	0.076	0.070	1.000	0.245
Season	0.074	0.051	-0.036	-0.035	0.245	1.000

Figure-11 Correlation Matrix

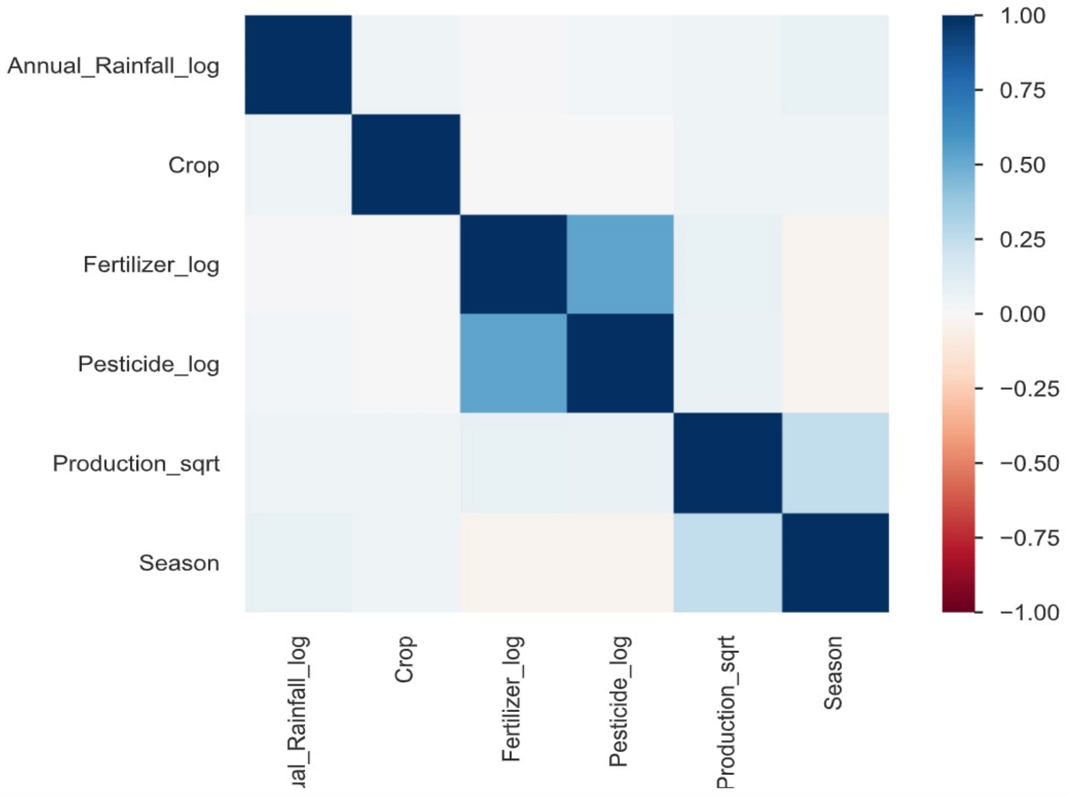


Figure-12 Correlation Histogram

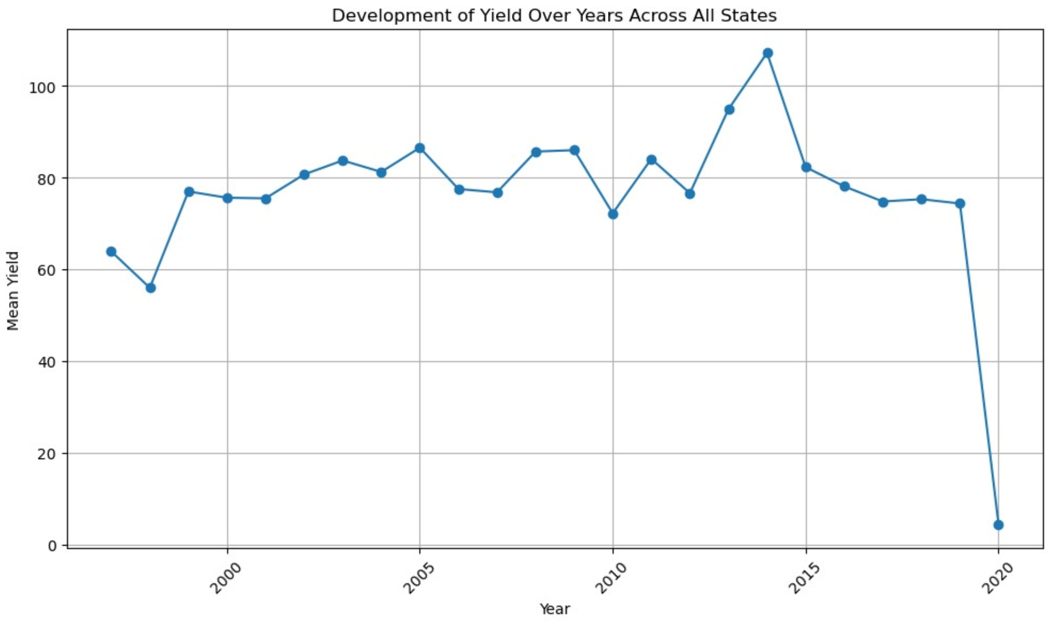


Figure-13 Year-wise Yield across All States

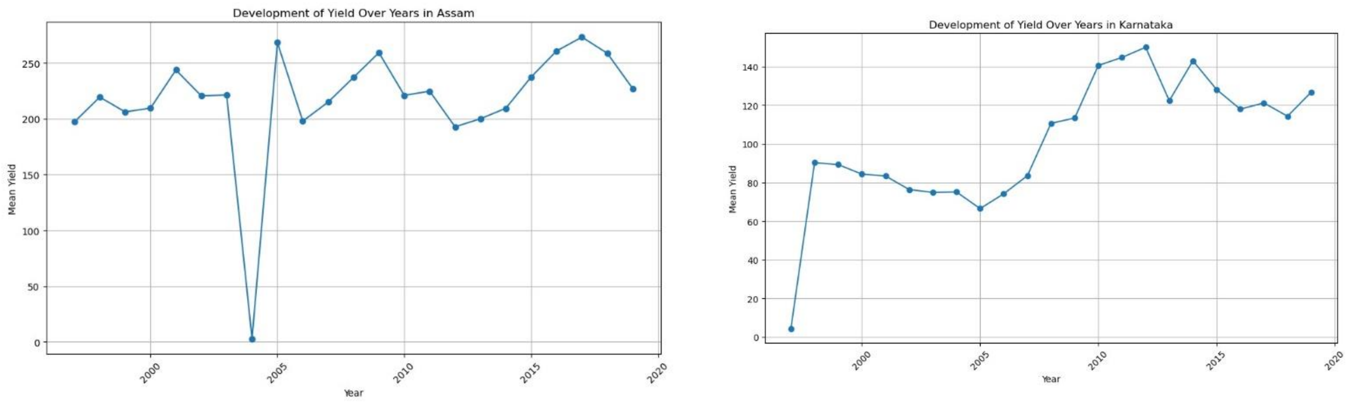
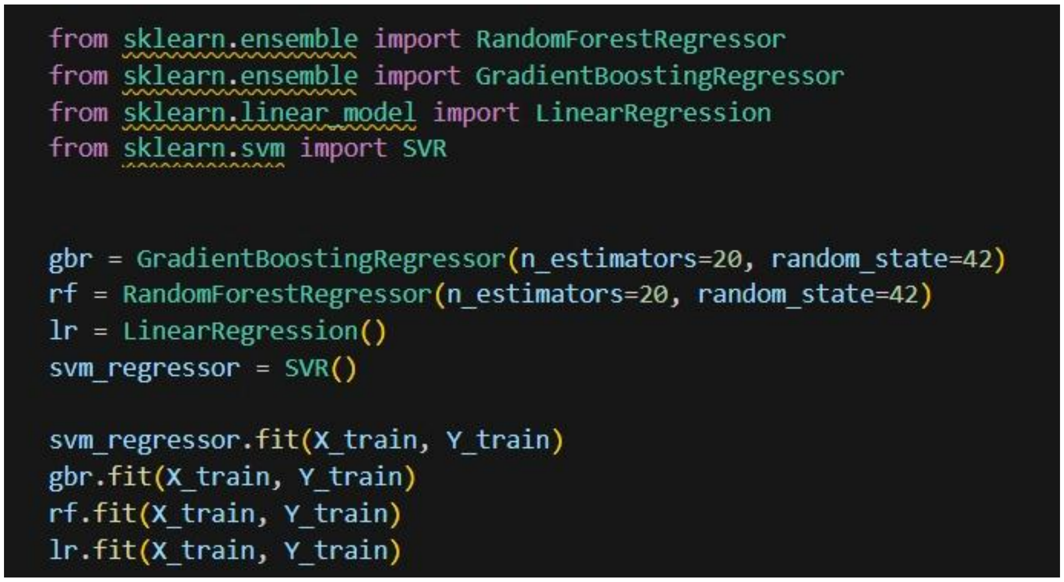


Figure-14 Year-wise Yield in Assam and Karnataka

3.3. Model Training

Here we implemented Gradient Boosting , Random Forest, Linear Regression, Support Vector Machine.

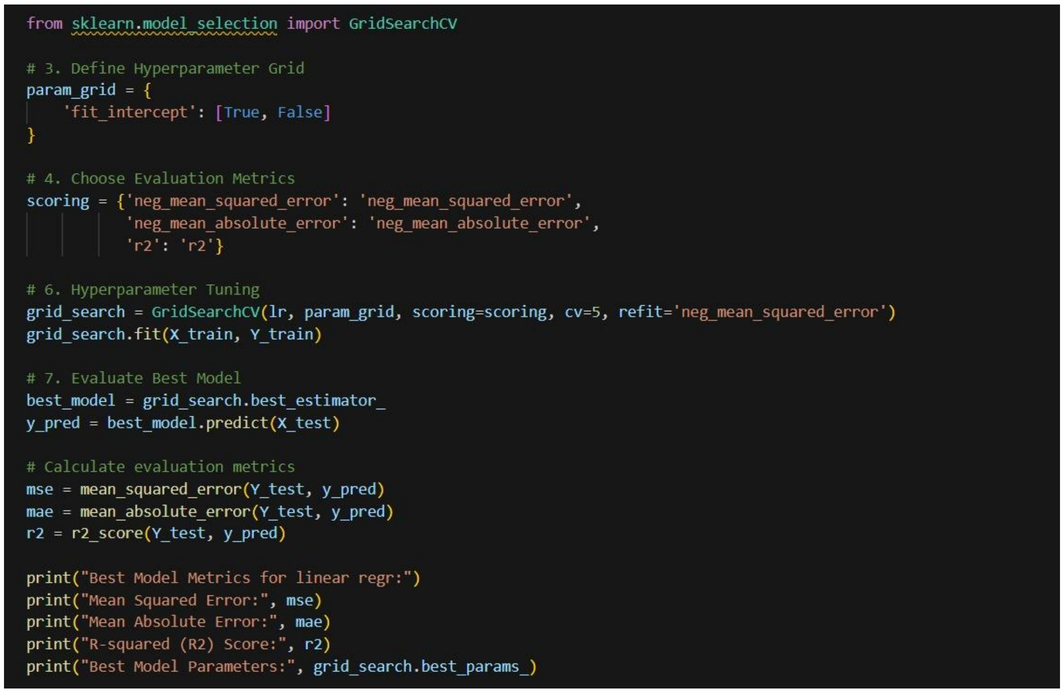


```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR


gbr = GradientBoostingRegressor(n_estimators=20, random_state=42)
rf = RandomForestRegressor(n_estimators=20, random_state=42)
lr = LinearRegression()
svm_regressor = SVR()

svm_regressor.fit(X_train, Y_train)
gbr.fit(X_train, Y_train)
rf.fit(X_train, Y_train)
lr.fit(X_train, Y_train)
```

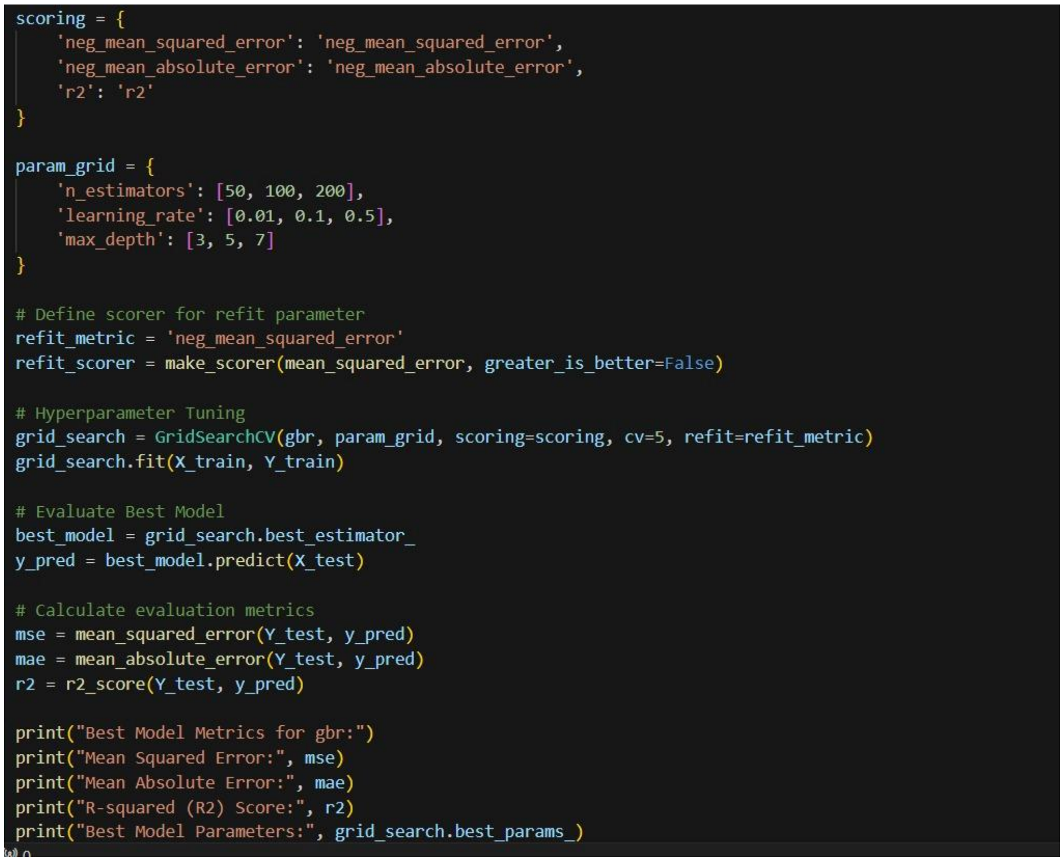
3.4. Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing the hyperparameters of a machine learning model to improve its performance. Hyperparameters are settings that control the learning process, such as learning rate or the number of trees in a random forest. Techniques like grid search or random search systematically explore the hyperparameter space to find the combination that maximizes model performance metrics, such as accuracy or F1 score, leading to better predictive performance on unseen data.

Below we applied Grid Search CV.



```
from sklearn.model_selection import GridSearchCV

# 3. Define Hyperparameter Grid
param_grid = {
    'fit_intercept': [True, False]
}

# 4. Choose Evaluation Metrics
scoring = {'neg_mean_squared_error': 'neg_mean_squared_error',
           'neg_mean_absolute_error': 'neg_mean_absolute_error',
           'r2': 'r2'}

# 6. Hyperparameter Tuning
grid_search = GridSearchCV(lr, param_grid, scoring=scoring, cv=5, refit='neg_mean_squared_error')
grid_search.fit(X_train, Y_train)

# 7. Evaluate Best Model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(Y_test, y_pred)
mae = mean_absolute_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)

print("Best Model Metrics for linear regr:")
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared (R2) Score:", r2)
print("Best Model Parameters:", grid_search.best_params_)
```



```
scoring = {
    'neg_mean_squared_error': 'neg_mean_squared_error',
    'neg_mean_absolute_error': 'neg_mean_absolute_error',
    'r2': 'r2'
}

param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.5],
    'max_depth': [3, 5, 7]
}

# Define scorer for refit parameter
refit_metric = 'neg_mean_squared_error'
refit_scorer = make_scorer(mean_squared_error, greater_is_better=False)

# Hyperparameter Tuning
grid_search = GridSearchCV(gbr, param_grid, scoring=scoring, cv=5, refit=refit_metric)
grid_search.fit(X_train, Y_train)

# Evaluate Best Model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(Y_test, y_pred)
mae = mean_absolute_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)

print("Best Model Metrics for gbr:")
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared (R2) Score:", r2)
print("Best Model Parameters:", grid_search.best_params_)
```

3.5. Validation

1. Mean Squared Error (MSE): MSE measures the average squared difference between the predicted and actual spending scores. MSE can be calculated as in Eq. (1).[7]

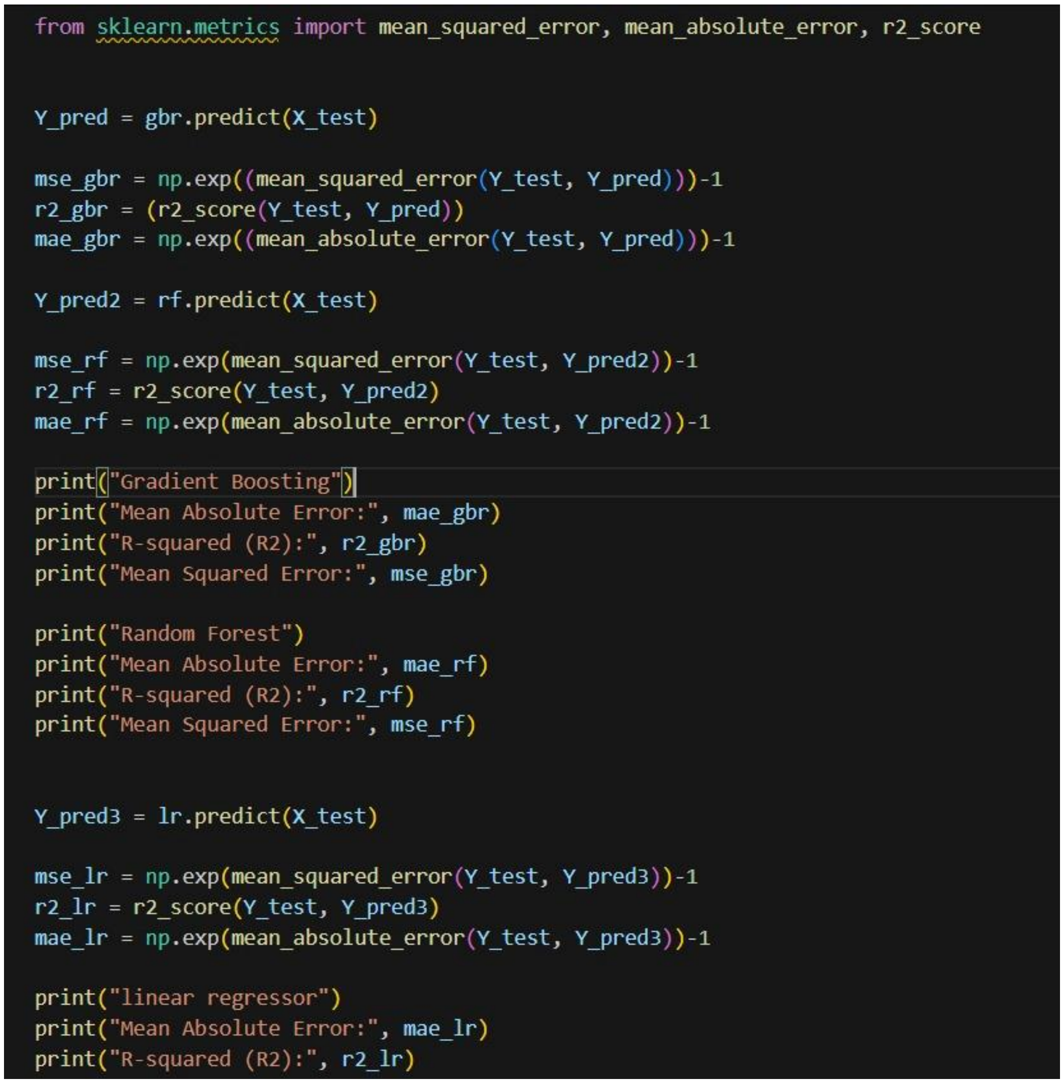
$$\mathrm{MSE} = \left(\frac{1}{n}\right) * \mathrm{sum}\left(\left(y_{\mathrm{pred}} - y_{\mathrm{actual}}\right)^{2}\right) \quad \text{Eq. (1)}$$

2. Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted and actual spending scores. MAE can be calculated as in Eq. (2).[7]

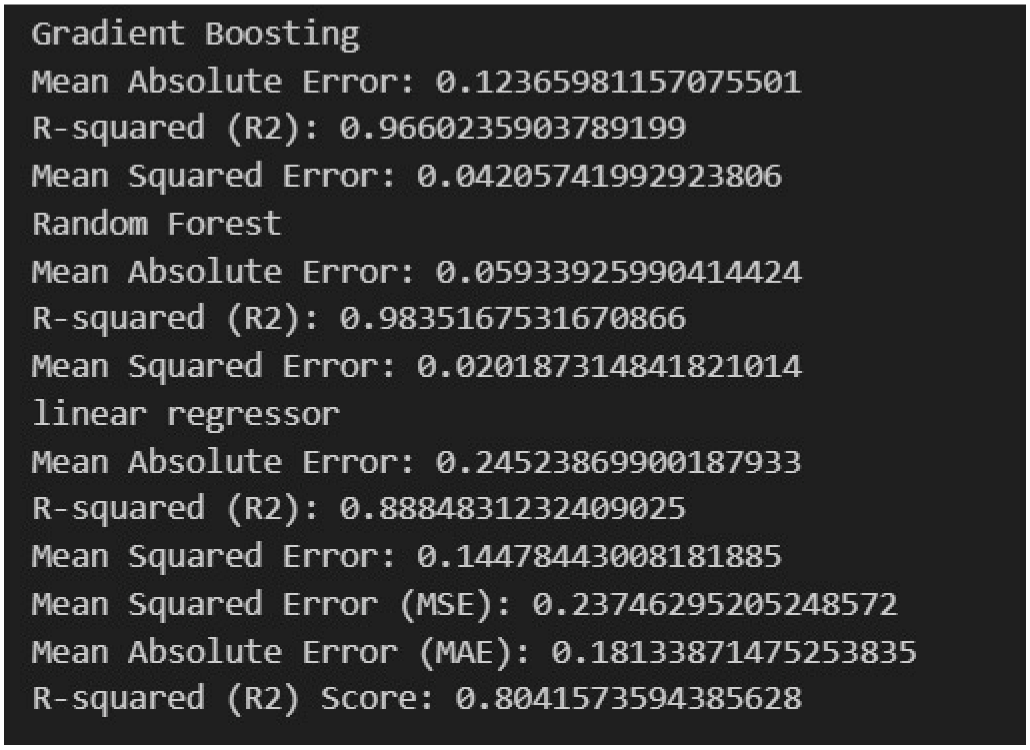
$$\mathrm{MAE} = (1/n) * \mathrm{sum}\left(\mathrm{abs}\left(y_{\mathrm{pred}} - y_{\mathrm{actual}}\right)\right) \quad \text{Eq. (2)}$$

3. R-squared: R-squared is a statistical measure representing the proportion of variance in the dependent variable explained by the independent variables. R-squared can be calculated as in Eq. (3).[7]

$$R^{2} = 1 - \left(\frac{\mathrm{sum}\left(\left(y_{\mathrm{actual}} - y_{\mathrm{pred}}\right)^{2}\right)}{\mathrm{sum}\left(\left(y_{\mathrm{actual}} - y_{\mathrm{mean}}\right)^{2}\right)}\right) \quad \text{Eq. (3).}$$



```python
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


Y_pred = gbr.predict(X_test)

mse_gbr = np.exp((mean_squared_error(Y_test, Y_pred)))-1
r2_gbr = (r2_score(Y_test, Y_pred))
mae_gbr = np.exp((mean_absolute_error(Y_test, Y_pred)))-1

Y_pred2 = rf.predict(X_test)

mse_rf = np.exp(mean_squared_error(Y_test, Y_pred2))-1
r2_rf = r2_score(Y_test, Y_pred2)
mae_rf = np.exp(mean_absolute_error(Y_test, Y_pred2))-1

print("Gradient Boosting")
print("Mean Absolute Error:", mae_gbr)
print("R-squared (R2):", r2_gbr)
print("Mean Squared Error:", mse_gbr)

print("Random Forest")
print("Mean Absolute Error:", mae_rf)
print("R-squared (R2):", r2_rf)
print("Mean Squared Error:", mse_rf)


Y_pred3 = lr.predict(X_test)

mse_lr = np.exp(mean_squared_error(Y_test, Y_pred3))-1
r2_lr = r2_score(Y_test, Y_pred3)
mae_lr = np.exp(mean_absolute_error(Y_test, Y_pred3))-1

print("linear regressor")
print("Mean Absolute Error:", mae_lr)
print("R-squared (R2):", r2_lr)
```



```
Gradient Boosting
Mean Absolute Error: 0.12365981157075501
R-squared (R2): 0.9660235903789199
Mean Squared Error: 0.04205741992923806
Random Forest
Mean Absolute Error: 0.05933925990414424
R-squared (R2): 0.9835167531670866
Mean Squared Error: 0.020187314841821014
linear regressor
Mean Absolute Error: 0.24523869900187933
R-squared (R2): 0.8884831232409025
Mean Squared Error: 0.14478443008181885
Mean Squared Error (MSE): 0.23746295205248572
Mean Absolute Error (MAE): 0.18133871475253835
R-squared (R2) Score: 0.8041573594385628
```

Figure-15 MSE, MAE, R2-Score across Applied Models

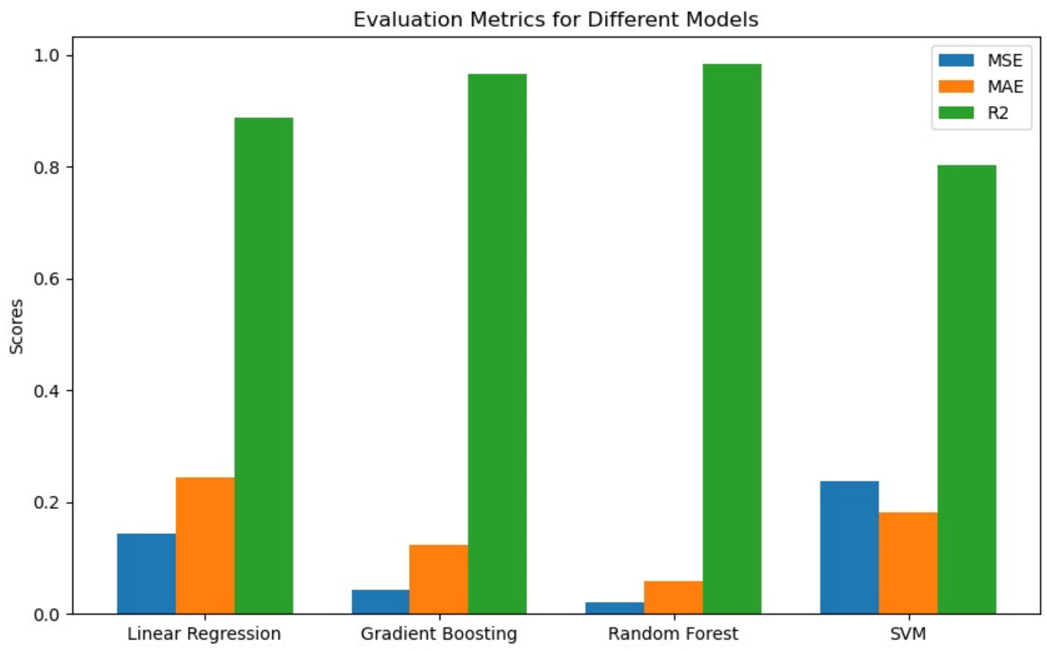
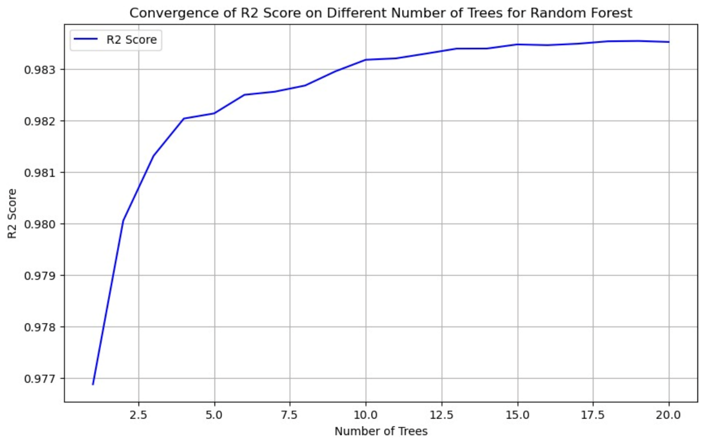


Figure-16 Evaluation Metrics



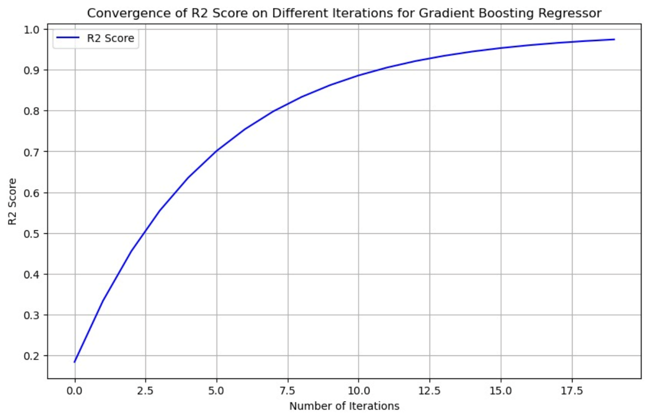
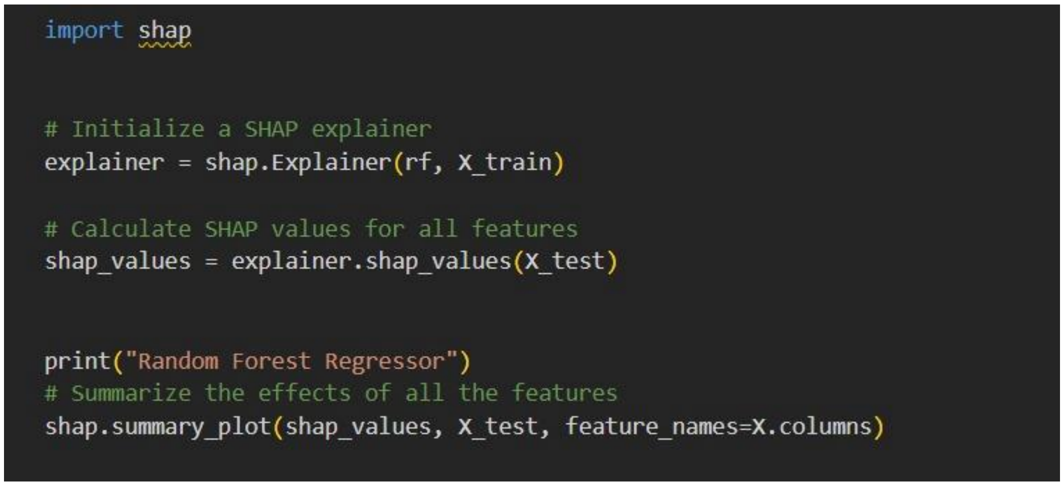
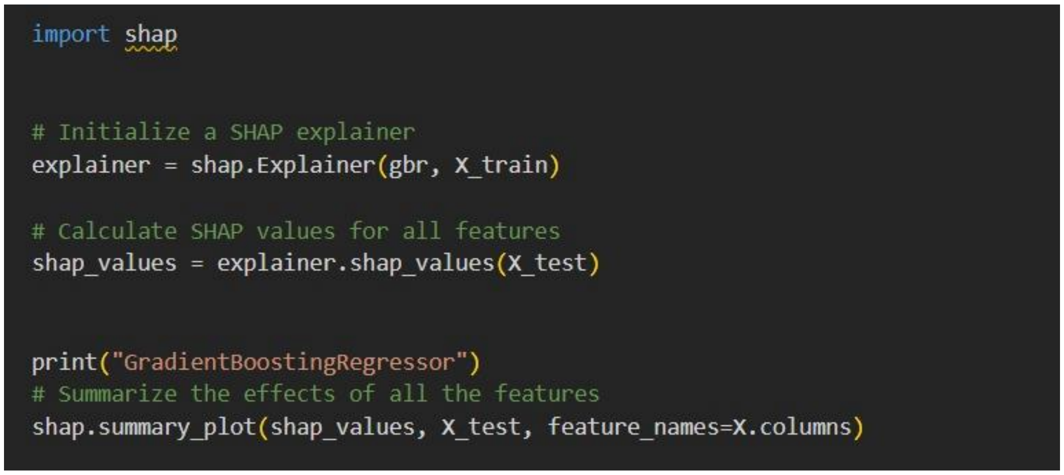


Figure-17 Convergence of r2 score for GBR and RF

3.6. XAI Integration



```
import shap


# Initialize a SHAP explainer
explainer = shap.Explainer(rf, X_train)

# Calculate SHAP values for all features
shap_values = explainer.shap_values(X_test)


print("Random Forest Regressor")
# Summarize the effects of all the features
shap.summary_plot(shap_values, X_test, feature_names=X.columns)
```



```
import shap


# Initialize a SHAP explainer
explainer = shap.Explainer(gbr, X_train)

# Calculate SHAP values for all features
shap_values = explainer.shap_values(X_test)


print("GradientBoostingRegressor")
# Summarize the effects of all the features
shap.summary_plot(shap_values, X_test, feature_names=X.columns)
```

Random Forest Regressor

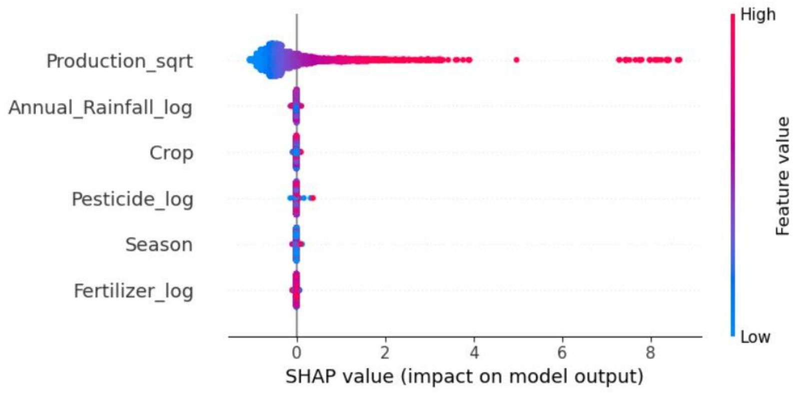


Figure-18 SHAP output for Random Forest

GradientBoostingRegressor

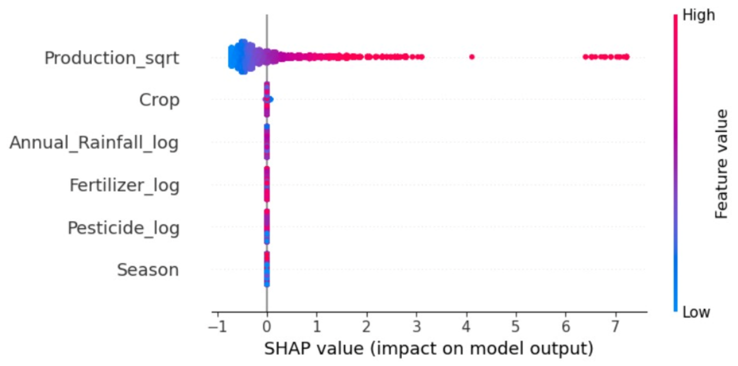
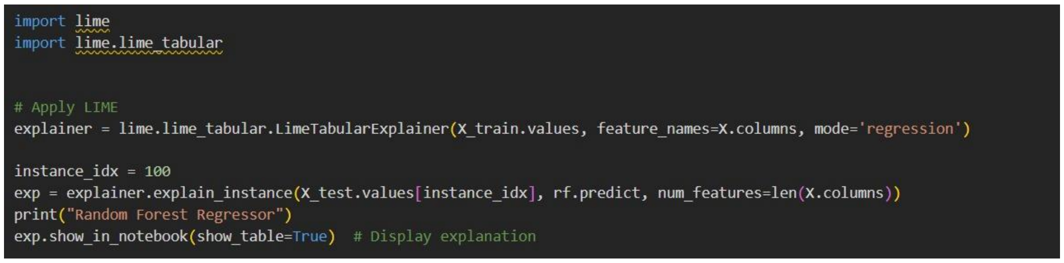
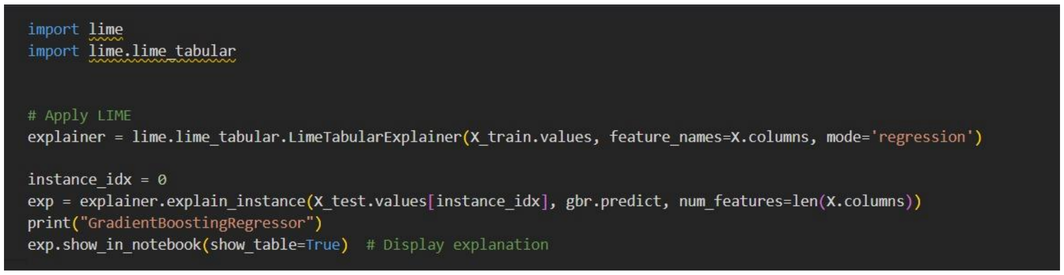


Figure-19 SHAP output for Gradient Boosting

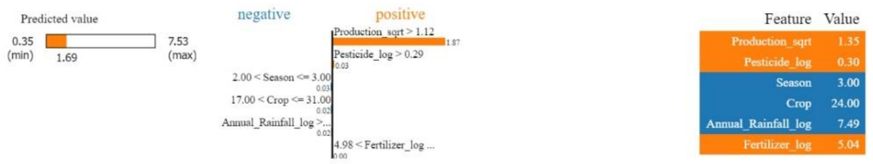


```
import lime
import lime.lime_tabular


# Apply LIME
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X.columns, mode='regression')

instance_idx = 100
exp = explainer.explain_instance(X_test.values[instance_idx], rf.predict, num_features=len(X.columns))
print("Random Forest Regressor")
exp.show_in_notebook(show_table=True)  # Display explanation
```



```
import lime
import lime.lime_tabular


# Apply LIME
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X.columns, mode='regression')

instance_idx = 0
exp = explainer.explain_instance(X_test.values[instance_idx], gbr.predict, num_features=len(X.columns))
print("GradientBoostingRegressor")
exp.show_in_notebook(show_table=True)  # Display explanation
```

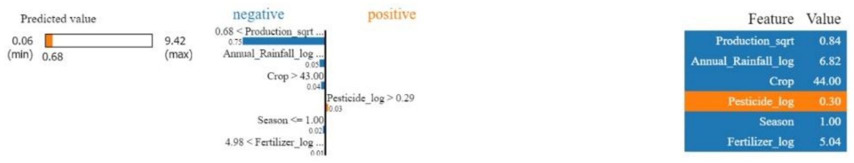
```
GradientBoostingRegressor
c:\Users\bhhav\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but GradientBoostingRegressor was fitted with feature names
  warnings.warn(
```



Feature	Value
Production_sqrt	1.35
Pesticide_log	0.30
Season	3.00
Crop	24.00
Annual_Rainfall_log	7.49
Fertilizer_log	5.04

Figure-20 LIME output for Gradient Boosting

```
Random Forest Regressor
c:\Users\bhhav\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

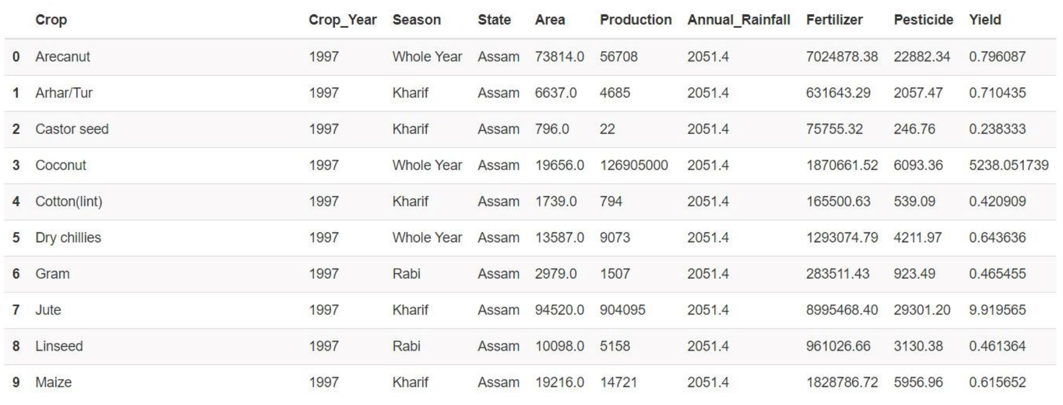


Feature	Value
Production_sqrt	0.84
Annual_Rainfall_log	6.82
Crop	44.00
Pesticide_log	0.30
Season	1.00
Fertilizer_log	5.04

Figure-21 LIME output for Random Forest

4. Experimental Setup and Results Analysis

4.1. Dataset



	Crop	Crop_Year	Season	State	Area	Production	Annual_Rainfall	Fertilizer	Pesticide	Yield
0	Arecanut	1997	Whole Year	Assam	73814.0	56708	2051.4	7024878.38	22882.34	0.796087
1	Arhar/Tur	1997	Kharif	Assam	6637.0	4685	2051.4	631643.29	2057.47	0.710435
2	Castor seed	1997	Kharif	Assam	796.0	22	2051.4	75755.32	246.76	0.238333
3	Coconut	1997	Whole Year	Assam	19656.0	126905000	2051.4	1870661.52	6093.36	5238.051739
4	Cotton(lint)	1997	Kharif	Assam	1739.0	794	2051.4	165500.63	539.09	0.420909
5	Dry chillies	1997	Whole Year	Assam	13587.0	9073	2051.4	1293074.79	4211.97	0.643636
6	Gram	1997	Rabi	Assam	2979.0	1507	2051.4	283511.43	923.49	0.465455
7	Jute	1997	Kharif	Assam	94520.0	904095	2051.4	8995468.40	29301.20	9.919565
8	Linseed	1997	Rabi	Assam	10098.0	5158	2051.4	961026.66	3130.38	0.461364
9	Maize	1997	Kharif	Assam	19216.0	14721	2051.4	1828786.72	5956.96	0.615652

Figure-22 First ten rows of Dataset[5]

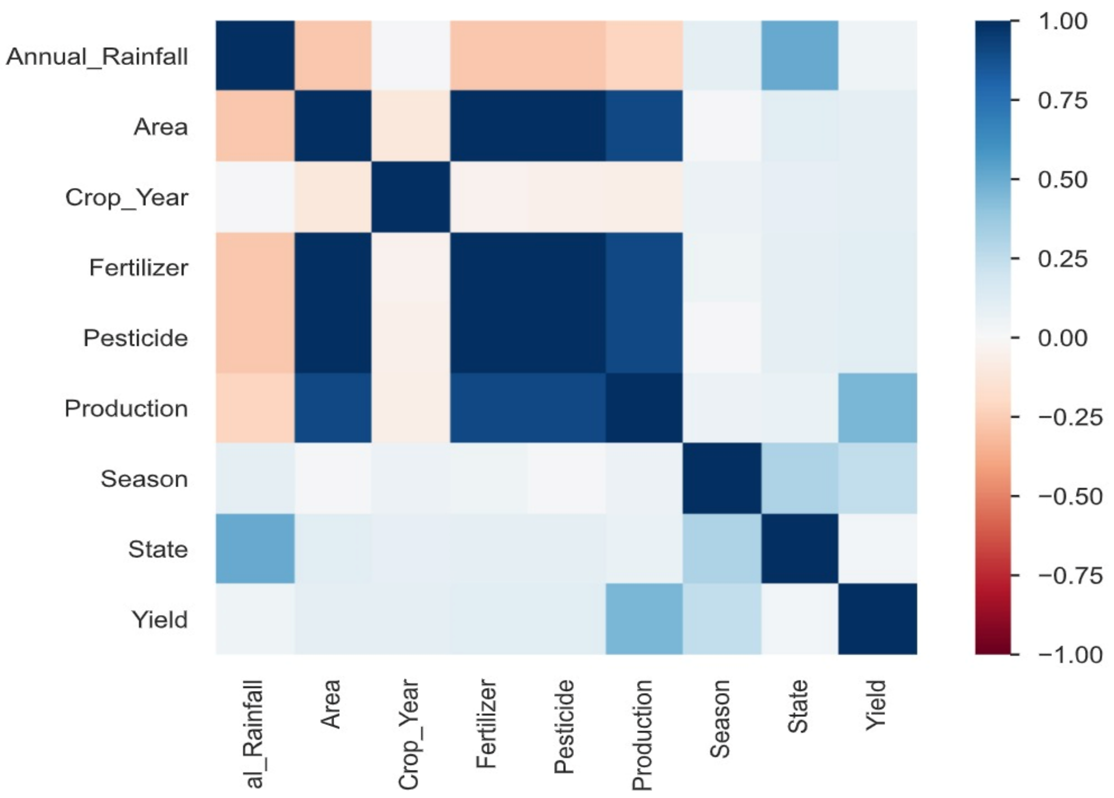
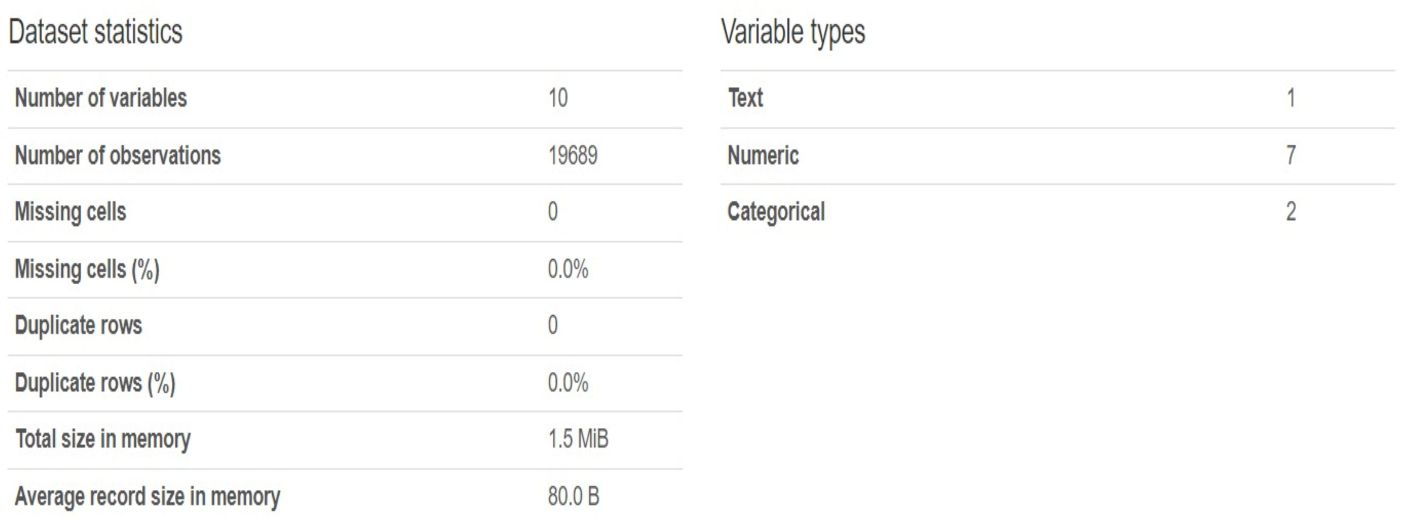


Figure-23 Correlation Histogram



Dataset statistics

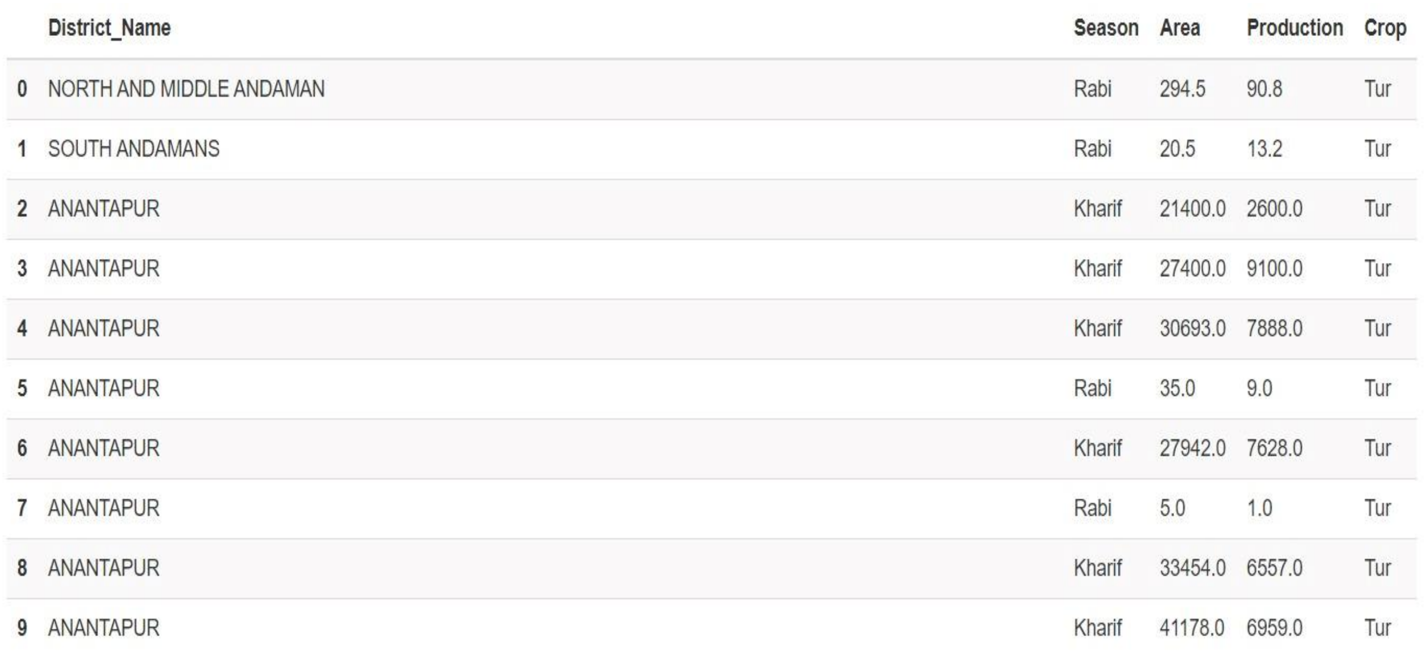
Number of variables	10
Number of observations	19689
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.5 MiB
Average record size in memory	80.0 B

Variable types

Text	1
Numeric	7
Categorical	2

Figure-24 Attribute Info.

4.2 Alternate Dataset



	District_Name	Season	Area	Production	Crop
0	NORTH AND MIDDLE ANDAMAN	Rabi	294.5	90.8	Tur
1	SOUTH ANDAMANS	Rabi	20.5	13.2	Tur
2	ANANTAPUR	Kharif	21400.0	2600.0	Tur
3	ANANTAPUR	Kharif	27400.0	9100.0	Tur
4	ANANTAPUR	Kharif	30693.0	7888.0	Tur
5	ANANTAPUR	Rabi	35.0	9.0	Tur
6	ANANTAPUR	Kharif	27942.0	7628.0	Tur
7	ANANTAPUR	Rabi	5.0	1.0	Tur
8	ANANTAPUR	Kharif	33454.0	6557.0	Tur
9	ANANTAPUR	Kharif	41178.0	6959.0	Tur

Figure-25 First ten rows of Dataset[8]

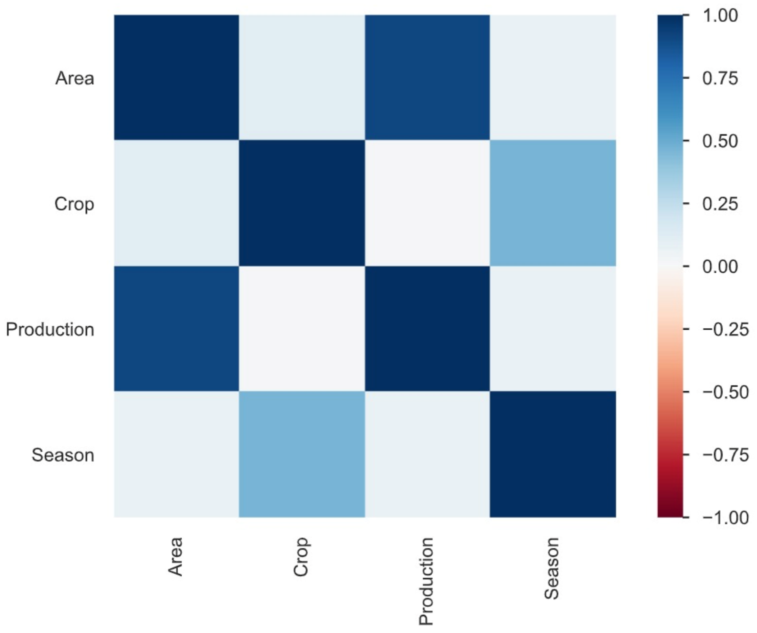
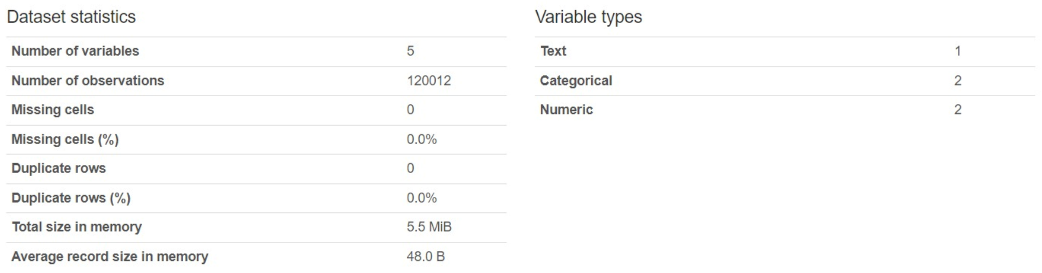


Figure-26 Correlation Histogram



Dataset statistics

Number of variables	5
Number of observations	120012
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	5.5 MiB
Average record size in memory	48.0 B

Variable types

Text	1
Categorical	2
Numeric	2

Figure-27 Attribute Info.

5. Conclusion and Future Work

5.1. Conclusion

In conclusion, our project delved into the realm of explainable artificial intelligence (XAI) within the agricultural domain, specifically focusing on predicting crop yield. By employing advanced techniques like LIME and SHAP alongside robust machine learning models such as Random Forest, Gradient Boosting, Support Vector Machines, and Decision Trees, we aimed to elucidate the intricate relationships between various agricultural attributes and crop production.

Through meticulous evaluation using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R2), we gained valuable insights into the performance and interpretability of our models. Our findings underscore the significance of transparency and interpretability in agricultural decision-making processes.

Moreover, leveraging a comprehensive dataset encompassing attributes like annual rainfall, fertilizer usage, pesticide application, and geographical factors including state and year, fortified our analyses with real-world relevance and depth.

Ultimately, our project not only contributes to the advancement of XAI techniques in agriculture but also offers practical solutions to optimize crop yield prediction, thereby potentially revolutionizing agricultural practices for enhanced productivity and sustainability in the future.

5.2. Future Works

Future endeavors in this field could explore several avenues to further enhance the efficacy and applicability of our findings. Firstly, incorporating more granular and diverse datasets could enrich model performance and generalization. Additionally, exploring ensemble techniques to amalgamate the strengths of various models could yield even more robust predictions. Further investigation into optimizing feature selection methods tailored specifically for agricultural data could streamline model interpretability and efficiency. Moreover, delving into the development of user-friendly interfaces or decision support systems could facilitate the practical implementation of our predictive models by farmers and agricultural stakeholders. Lastly, exploring the integration of real-time or IoT data streams could enable dynamic and adaptive crop management strategies, thereby fostering resilience and sustainability in agricultural practices.

References

1. Schlenker W, Roberts MJ (2009) Nonlinear temperature effects indicate severe damages to US crop yields under climate change. Proc Natl Acad Sci 106(37):15594–15598.

2. Asif D, Bibi M, Arif MS, Mukheimer A (2023) Enhancing heart disease prediction through ensemble learning techniques with hyperparameter optimization. Algorithms 16(6):6. https://doi.org/10.3390/a16060308

3. Linear regression https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.spiceworks.com%2Ftech%2Fartificial-intelligence%2Farticles%2Fwhat-is-linear-regression%2F&psig=AOvVaw1DPEhDegSiI_IMAQ8ZkpVZ&ust=1715254979321000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCMDg8u_8_YUDFQAAAAAdAAAAABAI

4. SVM Regressor https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm.png

5. Original Dataset https://www.kaggle.com/code/vaibhavsingh2161/crop-prediction/input

6. Random Forest https://miro.medium.com/v2/resize:fit:1010/1*R3oJiyaQwyLUyLZL-scDpw.png

7. IBM Docs https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=forecasting-statistical-details

8. Alternate Dataset https://www.kaggle.com/datasets/ananysharma/crop-yield