Comparison of Different Machine Learning Algorithms for Crop Classification

Mary Wang
Department of Mathematics
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada N2L 3G1
m426wang@uwaterloo.ca

Abstract

This report investigates the classification of crops in India using six machine learning models: K-Nearest Neighbors, Logistic Regression, Support Vector Machine, Decision Tree, Gaussian Naive Bayes, and Multi-layer Perceptron. The study involves exploring various parameters and hyperparameters of each model and using Exhaustive Grid Search technique to determine the optimal combination of hyperparameters. The study then evaluates each model's performance using various metrics such as accuracy, precision, recall, F1 score, and confusion matrix. The results show that all six models performed relatively well on the dataset, with the MLP and Gaussian NB models achieving the highest accuracy. However, it's important to acknowledge the limitations of the study, such as the limited number of parameters that were considered due to time and efficiency constraints. Future research could explore more reasonable options for hyperparameters and additional models to train the data. It is important to consider the socio-economic and cultural factors that may impact crop choice and yield in addition to the environmental and geographical factors that we have considered in this study. Overall, this study highlights the potential of machine learning models to accurately classify crops and improve crop yields for farmers in India.

1 Introduction

Crop yield prediction is an essential task in agriculture that enables farmers, policy-makers, and other stakeholders to make informed decisions related to crop planning, resource allocation, and food security. The availability of historical crop yield data, weather, soil, and other relevant data has enabled researchers to develop machine learning models that can effectively predict crop yields for future seasons or locations.

Machine learning models have become increasingly popular for crop yield prediction due to their ability to handle large datasets, complex patterns, and high-dimensional data. A wide range of machine learning techniques, including regression models, decision trees, support vector machines, and neural networks, have been applied to crop yield prediction tasks.

However, developing accurate and reliable crop yield prediction models using machine learning techniques is challenging due to various factors such as the quality and availability of data, selection of appropriate features and variables, and accuracy and interpretability of the models.

In this project, we aim to develop a machine learning-based crop yield prediction model using historical crop yield data, weather, soil, and other relevant data for 22 different crops in India. We will apply a variety of machine learning techniques and feature engineering approaches to develop and

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

compare different models' performance. The accuracy and robustness of the model will be evaluated using cross-validation and other metrics.

The project's outcomes will provide insights into the potential of machine learning-based crop yield prediction models for Indian agriculture and inform decision-makers about crop planning and resource allocation strategies for maximizing crop yield and food security.

2 Dataset

In this project, we aim to apply classification methods to analyze the Crop Prediction dataset. The dataset used is obtained from Kaggle. The dataset consists of 2,200 observations, with 8 features describing the soil, meteorological, environmental conditions, and 22 crops indicating the most recommended crop for each field. The dataset consists of the following features:

- N_SOIL: amount of nitrogen (N); integer between 0 and 140.
- P_SOIL: amount of phosphorus (P); integer between 5 and 145.
- K_SOIL: amount of potassium (K); integer between 5 and 205.
- TEMPERATURE: temperature in degree Celsius; float between 8.83 and 43.68.
- HUMIDITY: relative humidity; percentage number between 14.3% and 100%.
- PH: acidity of soil; float between 3.5 and 9.94.
- RAINFALL: rainfall in mm; float number between 20.21 and 298.56.
- STATE: state name where the field locates; Uttar Pradesh, Kerala, Tamil Nadu, Punjab, Maharashtra.

3 Methodology

3.1 Data preprocessing

Before proceeding with the classification analysis, we preprocess the dataset to ensure its suitability. The following changes are made:

- **From categorical to numeric:** The package 'Scikit-learn' requires categorical features and labels to be encoded as numbers, rather than strings. There are two types of encoding for categorical data: numeric encoding and "one-hot" encoding. Generally numeric encoding is used when there is an inherent order of the categories, and one-hot encoding otherwise. In our case, we do not have any inherent order in the 'STATE'. Hence, we applied 'one-hot' encoding to it. In addition, we will
- **Prepare training and test data:** We shuffled the data and divided it into 80% for training and 20% for testing where training set is used for model training and testing set is used for model evaluation for result reporting.

3.2 Models

We applied six classic classification models, namely K-Nearest Neighbors, Logistic Regression, Support Vector Machine, Decision Tree, Gaussian Naive Bayes and Multi-layer Preceptron. For each model, we explored various parameters and hyperparameters, and used the Exhaustive Grid Search technique to determine the optimal model for hyperparameter optimization. This involves testing every possible combination of the hyperparameters and determining the best values through cross-validation(k = 10, scoring="accuracy").The following are the parameters we will tune for each model:

- **KNearest Neighbors:** n_neighbors: [1, 2, 3, ... 30]; weights: Check whether adding weights to the data points is beneficial to the model or not ["uniform", "distance"].
- **Logistic Regression:** penalty: ["l1", "l2", None]; C: inverse of regularization strength. smaller values specify stronger regularization: [1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03];

- **Support Vector Machine:** penalty: [0.1,1,10, 100, 1000]; kernel: ["rbf", "poly", "sigmoid"]; gamma: kernel coefficient for "rbf" ["scale", "auto"]; degree: degree of the polynomial kernel function [2, 3].
- **Decision Tree:** criterion: the function to measure the quality of a split: ["gini", "entropy"]; max_depth: [4,6,8,10,12]; min_samples_leaf: [2, 3, 4]; min_samples_split: [2, 3, 4].
- **Gaussian Naive Bayes:** var_smoothing: np.logspace(0,-9, num=50)
- **Multi-layer Perceptron:** hidden_layer_sizes: [(50,50,50), (50,100,50), (100,)], activation: ["tanh", "relu"], solver: ["sgd", "adam"], alpha: [0.0001, 0.05], learning_rate: ["constant","adaptive"]

After obtaining the optimal combination of hyperparameters, we will train the model with this optimal combination of hyperparameters. Once we have built models using each method, we will predict the test data using these models. To evaluate the model performace, we will evaluate various metrics such as accuracy, precision, recall, F1 score and confusion matrix. Finally, we will end up suggesting the most accurate model for the classification of crops in India.

4 Results

4.1 KNN

The hyperparameter "k" represents the number of nearest neighbors to consider when making a prediction. A small value of "k" may result in overfitting, while a large value of "k" may result in underfitting. The optimal value of "k" depends on the characteristics of the dataset, such as the number of samples, number of features, and distribution of the classes.

The hyperparameter "weight" determines how the contributions of the neighbors are weighted in the prediction. The two options are "uniform" and "distance". When "uniform" is used, all neighbors are weighted equally, while when "distance" is used, the closer neighbors have more influence on the prediction.

In the grid search, the optimal values of "k" and "weight" were found to be 7 and "distance", respectively. This suggests that the KNN model performs better when the closer neighbors are given more weight in the prediction, rather than treating all neighbors equally.

The choice of "k" as 7 may have been influenced by several factors, such as the size and complexity of the dataset, the distribution of the classes, and the nature of the features. It's possible that for this specific dataset and feature set, a value of 7 is optimal in terms of balancing bias and variance and achieving the best performance.

4.2 Logistic Regression

In logistic regression, the hyperparameters C, penalty, and solver play a crucial role in determining the performance of the model.

The hyperparameter C represents the inverse of the regularization strength. A smaller value of C will increase the regularization strength, resulting in simpler models that may be less prone to overfitting. On the other hand, a larger value of C will decrease the regularization strength, resulting in more complex models that may be more prone to overfitting.

The penalty hyperparameter determines the type of regularization used in the model. The two options are "l1" and "l2" regularization. L1 regularization adds a penalty term to the loss function that is proportional to the absolute value of the coefficients, while L2 regularization adds a penalty term that is proportional to the square of the coefficients.

The solver hyperparameter determines the algorithm used to optimize the logistic regression model. The available options we provided are "lbfgs", "liblinear", and "sag". These solvers use different optimization algorithms, and the choice of solver can have an impact on the convergence speed and accuracy of the model.

In the grid search, the optimal hyperparameters were found to be C=1000.0, penalty="l2", and solver="liblinear". This suggests that the logistic regression model performs better when the regularization strength is relatively low and L2 regularization is used. Additionally, the "liblinear" solver

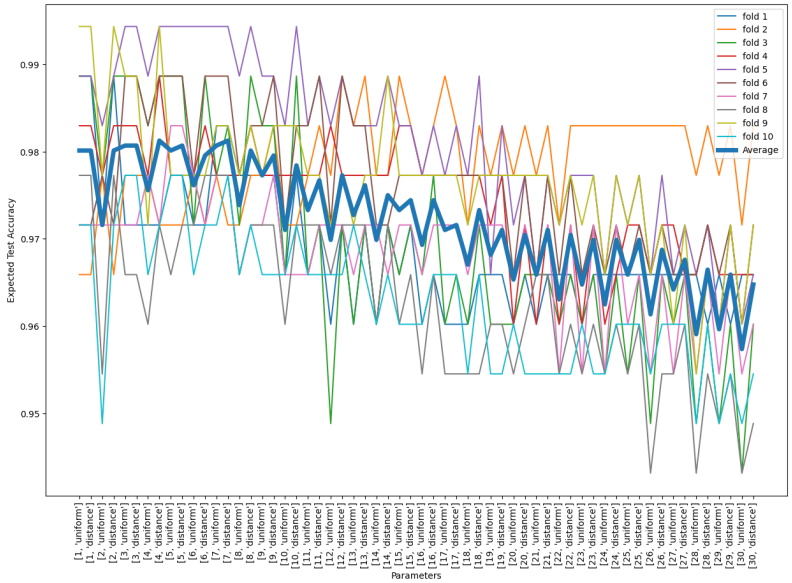


Figure 1: KNN GridSearch Result.

was found to be optimal, which may indicate that it was the most efficient or effective optimization algorithm for this specific problem.

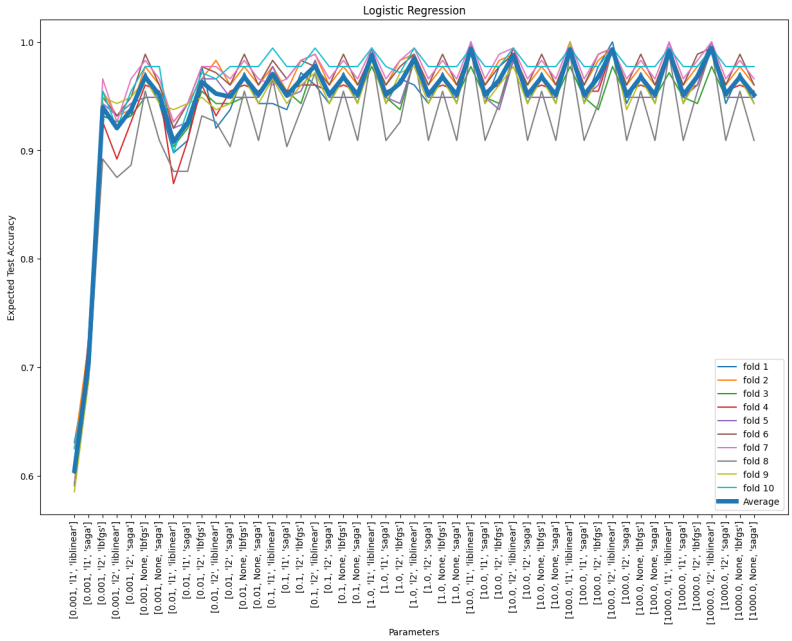


Figure 2: Logistic Regression GridSearch Result.

4.3 SVM

The hyperparameter C represents the penalty parameter of the error term in the SVM optimization problem. A small value of C will create a wider margin hyperplane at the cost of more margin violations, which may result in higher bias but lower variance. Conversely, a large value of C will create a narrow margin hyperplane at the cost of fewer margin violations, which may result in lower bias but higher variance.

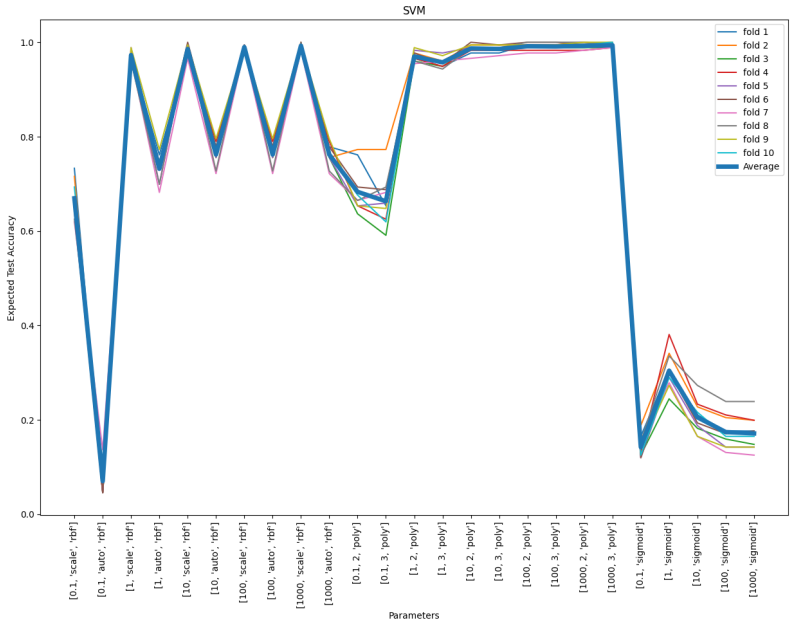


Figure 3: SVM GridSearch Result.

The kernel hyperparameter determines the type of kernel function used to transform the input data into a higher-dimensional space, where the decision boundary can be more easily defined. The three most commonly used kernels are linear, polynomial, and radial basis function (RBF). The polynomial kernel has an additional hyperparameter, the degree, which determines the degree of the polynomial function used in the transformation.

In the grid search, the optimal hyperparameters were found to be C=1000, kernel="poly", and degree=3. This suggests that the SVM model performs better with a relatively high value of C, indicating a preference for a narrower margin hyperplane with fewer margin violations. The polynomial kernel was found to be optimal, which may indicate that the decision boundary of the data is non-linear and can be better modeled using a polynomial function. The degree of the polynomial kernel was found to be 3, which suggests that a cubic transformation of the input data was the best choice for this specific problem.

4.4 Decision Tree

The CV plot indicates that the accuracy of our models varies widely across different folds. However, some models consistently outperform others, suggesting that certain hyperparameters have a greater impact on the model's performance. Interestingly, we observed that the choice of the criterion hyperparameter, which determines the function used to measure the quality of a split, did not have a significant effect on the results. Both gini and entropy produced similar results in terms of accuracy, suggesting that the choice of criterion may not be crucial for our dataset and model.

We have observed that the max_depth hyperparameter in our decision tree model has a significant impact on its performance, with a larger value leading to lower accuracy. This is because "max_depth" determines the balance between bias and variance. A shallow tree with a small "max_depth" value will have high bias and low variance, meaning that it is more likely to underfit the data. On the other hand, a deep tree with a large "max_depth" value will have low bias and high variance, meaning that it is more likely to overfit the data.

The best model we found is the one with criterion="entropy", max_depth=8, min_samples_leaf=2, min_samples_split=4. This suggests that the entropy criterion was the best measure for evaluating the quality of the splits. The optimal max_depth was found to be 8, indicating that a tree of moderate complexity was optimal for this problem. The optimal values of min_samples_leaf and

min_samples_split suggest that these hyperparameters can be used to control the complexity of the tree and prevent overfitting.

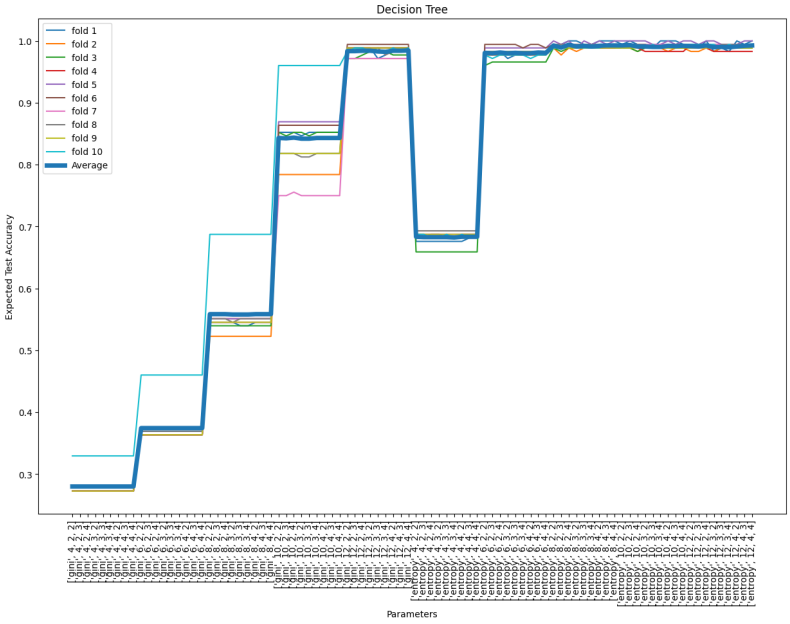


Figure 4: Decision Tree GridSearch Result.

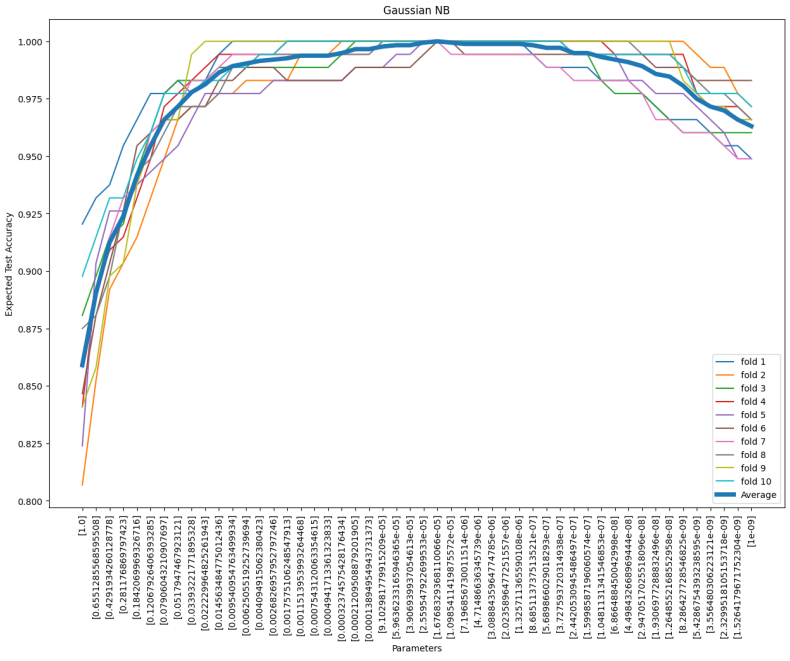


Figure 5: Gaussian Naive Bayes GridSearch Result.

4.5 Gaussian Naive Bayes

The "var_smoothing" hyperparameter is a smoothing parameter that is added to the variance of each feature in the dataset. It is used to avoid problems with zero variance features, which can cause numerical errors during the calculation of the class likelihoods. In general, the optimal value for "var_smoothing" should strike a balance between underfitting and overfitting. A very small value of

"var_smoothing" may lead to overfitting, where the model becomes too complex and fits the noise in the data, while a very large value of "var_smoothing" may lead to underfitting, where the model is too simple and fails to capture the underlying patterns in the data.

Based on the cross-validation result, 1.6768329368110066e-05 is selected as the optimal "var_smoothing" value, suggesting that a small amount of smoothing is beneficial for the dataset and problem at hand.

4.6 Multi-layer Perceptron

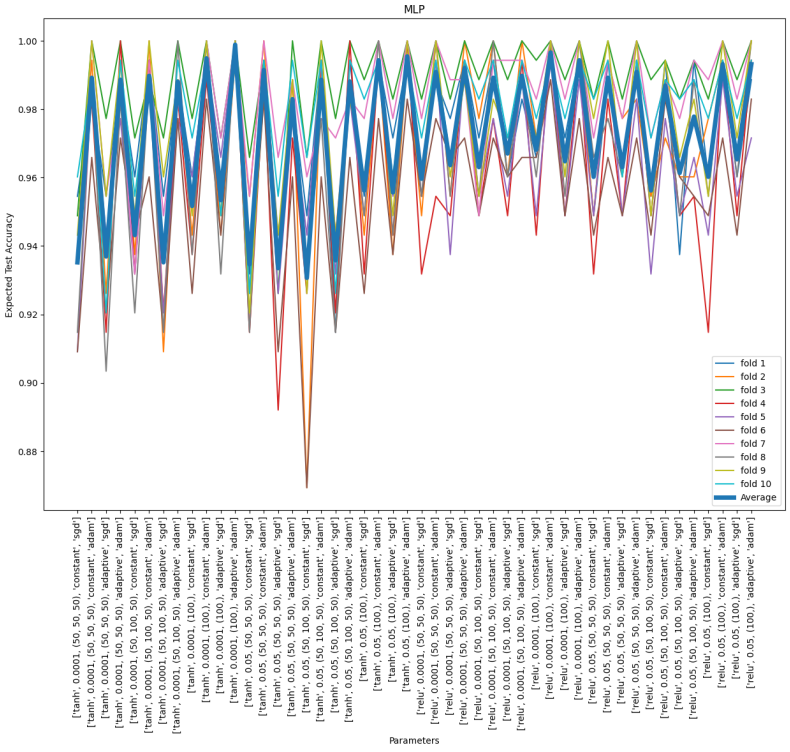


Figure 6: MLP GridSearch Result.

In the cv plot, the first we notice are those periodical ups and downs. This periodicity suggests that the "adam" solver is better than "SGD" because "adam" is an adaptive learning rate optimization algorithm that is well-suited for problems with large datasets and complex models. It can automatically adjust the learning rate and momentum parameters during training and can provide good performance for a wide range of problems.

The "tanh" activation function was chosen as the optimal activation function. The "tanh" function maps the input values to the range [-1, 1] and is often used in MLPs because it is computationally efficient and can prevent the vanishing gradient problem.

The optimal alpha value of 0.0001 suggests that the model benefits from a small amount of L2 regularization, which helps prevent overfitting.

The optimal value for the hidden layer sizes is a single layer with 100 units. This suggests that a relatively simple network architecture is sufficient for the problem at hand. A single layer with 100 units is a common choice for MLPs, and it strikes a balance between model complexity and performance.

The "adaptive" learning rate was chosen as the optimal learning rate. This means that the learning rate is adjusted dynamically during training based on the change in training loss. The adaptive learning rate can help the model converge more quickly and prevent oscillations around the optimal solution.

It is also worthy mentioning that the fluctuation range is very limited (between 0.9 and 1). This susggests that MLP is the model with highest overall performance even without any hyperparamter tuning.

4.7 Model Evaluation

Based on the test accuracies listed in the table, it appears that all six models perform relatively well on the dataset. An accuracy of 0.97 indicates that the KNN model correctly classified 97% of the test instances, while an accuracy of 0.99 indicates that the logistic regression, SVM, and decision tree models correctly classified 99% of the test instances. An accuracy of 1.0 for the MLP and Gaussian NB models indicates that these models correctly classified all of the test instances.

It's important to note that test accuracy is only one measure of model performance. In addition to accuracy, we also incorporate precision, recall, F1 score. Since we are dealing with a multi-class classification problem, we use micro-averaged precision and macro-averaged precision to help us understand the average of the precision scores for each class with and without weighting by the number of instances in each class respectively. We are having similar micro-averaged precision and macro-averaged precision results because the dataset is relatively balanced with the number of instances in each class is similar.

Moreover, a high precision and recall indicates we did a good job minimizing both false negative false positives and false negatives. F1 score is the harmonic mean of precision and recall, and it combines both metrics to provide a balanced measure of the model's performance. A high F1 score suggests that the model has a high level of precision and recall, and is able to correctly identify instances in the dataset with a high degree of accuracy.

Overall, these results suggest that all six models are able to accurately classify instances from the dataset, with the MLP and Gaussian NB models performing the best.

Table 1: Model Evaluation

		Macro-Averaged			Micro-Averaged		
Model	Accuracy	Precision	Recall	F1	Precision	Recall	F1
KNN	0.97	0.97	0.97	0.97	0.97	0.97	0.97
Logistic Regression	0.99	0.99	0.99	0.99	0.99	0.99	0.99
SVM	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Gaussian NB	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MLP	1.00	1.00	1.00	1.00	1.00	1.00	1.00

5 Discussion

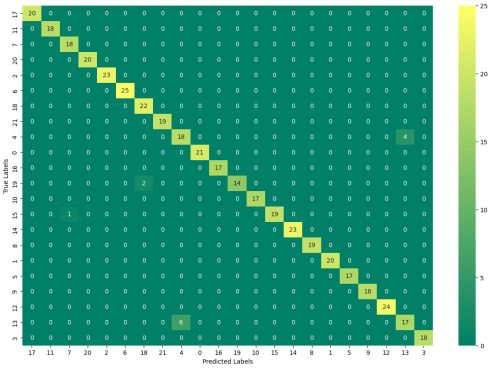
The results show that all six models performed relatively well in predicting the crop type, with the MLP and Gaussian NB models achieving the highest accuracy.

However, It is important to acknowledge that our methodology has some limitations. Since we conducted grid search, we could only consider a limited number of parameters that we deemed reasonable for our predictive model. Due to time and efficiency constraints, we had to select only several options for each hyperparameter. As a result, we cannot guarantee that the model we selected is the optimal one for each method.

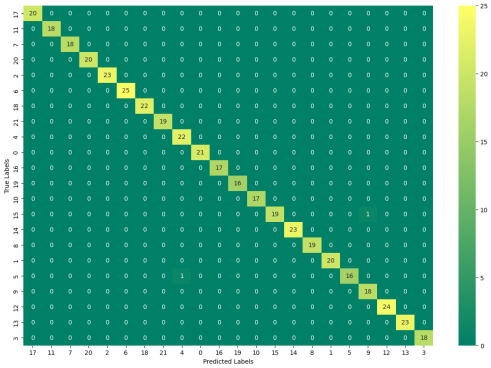
More specifically, we have observed that each model has its own strengths and weaknesses.

The logistic regression model assumes a linear relationship between the features and recommended crop, which can be a limitation when the true relationship is more complex.

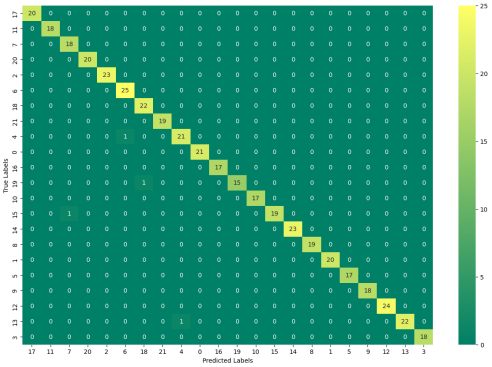
SVM is particularly effective in dealing with high dimensional data. However, it takes longer time to do grid search than other models. SVM is computationally expensive as the time complexity is at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples.



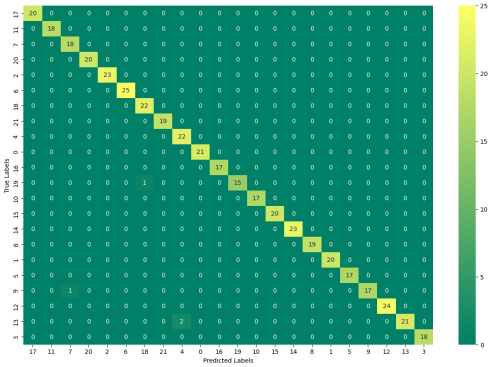
(a) KNN Confusion Matrix



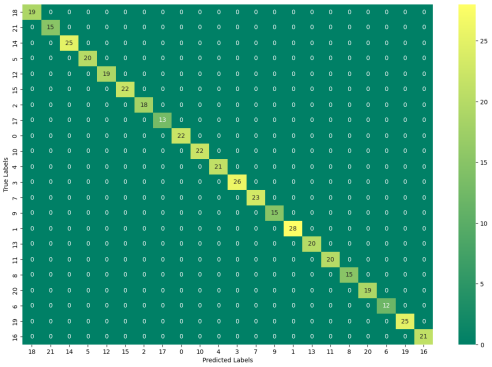
(b) Logistic Regression Confusion Matrix



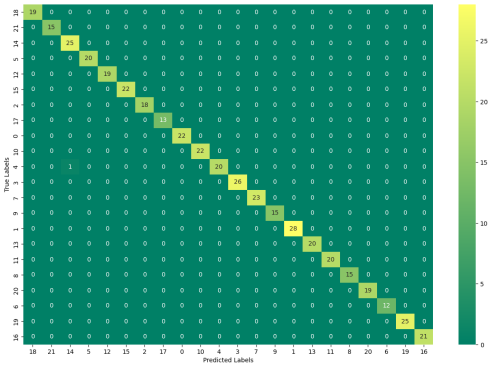
(c) SVM Confusion Matrix



(d) DT Confusion Matrix



(e) Gaussian NB Confusion Matrix



(f) MLP Confusion Matrix

Figure 7: Confusion Matrices

Decision trees are easy to interpret and can capture nonlinear relationships between the features and the outcome variable. Moreover, we can see that many features have a permutation importance of 0, which means the model uses the least number of parameters. If we want a simpler model, decision trees may be the best option. However, it can be prone to over-fitting and may not generalize well to new data, as we can see it has a relatively large misclassification rate among our models.

Gaussian Naive Bayes is a fast algorithm that works well on large datasets with many variables. It is also easy to interpret, making it a useful tool for exploratory data analysis.However, one of the main weaknesses of Gaussian Naive Bayes is its assumption of feature independence, which may not hold true in many real-world scenarios. For example, in general higher humidity levels can increase the likelihood of rainfall, as there is more water vapor in the air that can condense into clouds and eventually fall as precipitation. Thus, the assumption that humidity and rainfall are independent can be problematic.

MLP can model non-linear decision boundaries, making it suitable for complex problems. However, similar to SVM, MLP can be computationally expensive and may require a large amount of computational resources and time to train. In our case, it takes about 40 minutes to do MLP grid search. Moreover, MLP can be difficult to interpret, and it may not always be clear how the model is making predictions.

Additionally, it's important to note that the dataset used for this study is based on crop yield data from India. It may not be representative of crop yield data from other regions or countries. Therefore, the results obtained from this study may not be generalizable to other regions or countries.

Furthermore, it is important to consider the socio-economic and cultural factors that may impact crop choice and yield in addition to the environmental and geographical factors that we have considered in this study. This can include factors such as market demand, government subsidies, and traditional farming practices.

In conclusion, while our study provides valuable insights into the use of machine learning models to predict crop yields in India, it is important to acknowledge the limitations of our methodology and consider the broader socio-economic and cultural factors that may impact crop yield. Further research is necessary to explore more reasonable options for hyperparameters, consider additional models to train the data and validate and extend our findings to other regions and countries.

References

[1] MD, J. (2023). Crop-Prediction-Data. Retrieved from https://www.kaggle.com/datasets/jiteshmd/crop-prediction-data

[2] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.