CS 4641 Final Project Report

By: Michael Rodyushkin and Nathaniel Wert

Introduction (1):

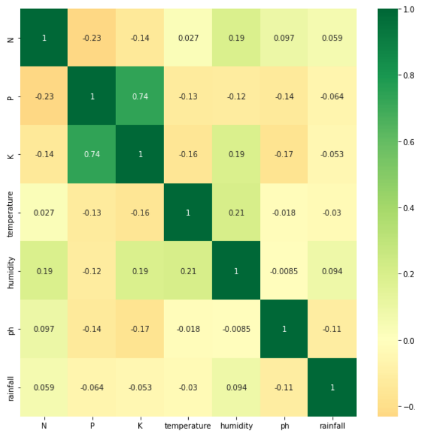
In India, over 60% of its population and roughly 20% relies on agriculture for work and profit respectively [1]. Despite agriculture playing such a key role in Indian society and careful crop diversity selection playing such a key role in long-term agriculture, it seems farmers seem to follow almost an abstract way of choosing which crops to pick for a specific year where "the role of aspirations and culture is central to shaping crop diversity" [2]. Thus, it was natural for us and other researchers to attempt to create a more concrete process for these farmers (and potentially other farmers worldwide) to pick crops based on scientific properties of the soil in a given year. However, many of the past papers that attempted to design machine learning models that picked a crop to use for certain soil conditions simply proposed the concept of an ensemble model that used a basic majority vote approach providing no actual test results [6] or analyzed the accuracy of different models and simply chose one that had the highest accuracy (which many concluded was the ensemble Random Forest Model) [3, 4, 5]. Though their accuracy rates that were between 91% to 98% seem theoretically good, for a farmer whose livelihood revolves around their crops, a roughly 1 in 10 chance every year of the proper crop not being selected by a machine learning model they rely on does not seem too good. Thus, in this report, we set out to design and test a more comprehensive list of models and more specifically test a stacking ensemble model to see if it can consistently better predict the proper crop to grow based on soil conditions for not only Indian farmers that rely on crop selection for their livelihoods but also for other more recreational farmers.

Data Source (2):

It is first vital to discuss the dataset being used to train our model (provided as source 7). After combing through papers that propose or build models about crop recommendations (as referenced above in the introduction), we noticed that many of them referenced a few datasets which could be located on Kaggle. Once we located these data sets, we chose the data that included the most complete and unique data points that had multiple features. The data set we ended up choosing was then easily downloadable through Kaggle as a csv file and read into our notebook using the pandas read csv method. This data compiled different datasets from the Indian Chamber of Food and Agriculture into a singular dataset with 7 features, 22 unique labels, and 2200 complete and unique data points. Each data point represents a unique soil conditions test done in a unique plot of land in India. The soil test results are represented by the 7 features: the ratio between nitrogen, potassium, and phosphorus (each ratio for a specific element is placed in its own column), the temperature in degrees in celsius of the soil, the relative humidity percentage of air near the soil (out of a scale of 0-100), the pH value of the soil, and the average seasonal rainfall in millimeters. Each label represents the best type of crop to grow in this specific soil out of 22 crops (from rice, watermelon, mango, etc.).

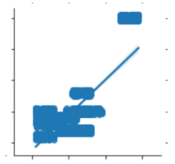
Data Format (3):

With the data contextualized, now we can discuss the data's analysis and preprocessing. Since Kaggle provides information about the data set before downloading, we were able to quickly see that all of the data points were "valid" meaning there were no null values. Thus, we did not prune any data points for the time being (the reason we went no further is discussed in our conclusion). Though our dataset has not that many features, it is still important to check if feature selection and reduction needs to be performed not to improve computational performance (which would be insignificantly changed if 1 or 2 features are removed from a



Heatmap of features correlations

relatively small data set like ours) to prevent overfitting. Specifically, we checked for correlations between the features because we were unsure if the fact the units of the nitrogen, potassium, and phosphorus were ratios of each other would have some correlative effect. If two redundant features were to stay in our dataset the noise each one produces can potentially cause overfitting of our smallish data set. As the heatmap of correlations above shows, only two features are somewhat correlated -- Potassium and Phosphorus. To further investigate the issue, we plotted the data points (pictured below) for every pair of features and found that the high correlation is caused by the small amount of high potassium values that had huge leverage over the correlation value and thus neither feature had to be dropped.



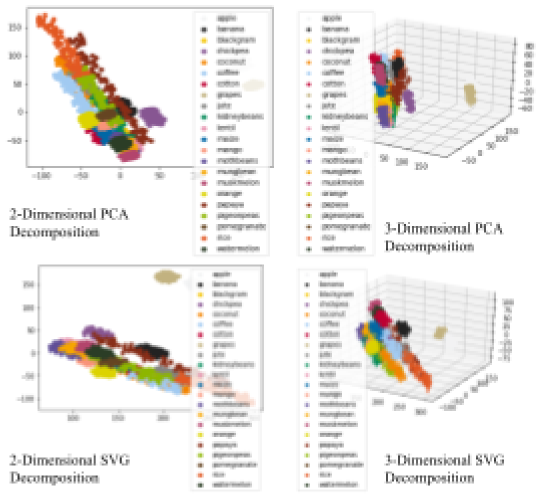
Graph of potassium vs. phosphorus of the data points

Another check we performed was seeing if any of our labels are oversampled or undersampled. However, creating a simple bar graph of the count of each label in the data shows the data already provides exactly 100 data points for each label meaning we do not need to create any synthetic samples.

Data Visualization (4):

Before choosing any models to classify our data into the 22 crop categories, we believed it would be important to understand how the data was distributed. To understand the answer to this question, we plotted the 2 and 3 dimensional pca and svg decompositions of our datasets. Shown in figure 4.1 is the results of this data visualization exercise. The two most notable observations from the visualizations are the isolation of grapes (in both PCA and SVG) and apples (in SVG), in addition to the noticeable clustering of like data points. The isolation of the grapes and apples hints that models will likely have very

Figure 4.1



little difficulty separating these labels and the clustering of like data points will lead to the intuition that a clustering algorithm (like k-nearest neighbor) would perform incredibly well on this dataset.

Method Documentation (5):

In this project, Naive Bayes, Support Vector Machines, K-Nearest Neighbors, Decision Trees, Neural Networks, and Stacked Classifiers were used to create classifiers that worked to correctly classify the correct type of crop for given soil and climate measurements.

For each algorithm (and implementation) the same training and testing methods were conducted. First, a subset of 90% of the total data was selected to be the training subset, while the remaining 10% of the data was to be left as the testing subset. Then, the given algorithm was fitted to the training subset of the data. Next, the fit algorithm was evaluated against the training data to attempt to determine the actual accuracy of the model. At the same time, the models performance in each label was evaluated by having the model predict the labels for all data with each specific label. This process was repeated 10 times for each model (with each time having a different random sample of training and testing data). After each step, the label specific accuracies for the model are averaged and the testing accuracy of the model is recorded. Finally, the maximum, minimum, average, and label-specific average of the model is displayed for these 10 models trained on different randomly sampled data.

The first algorithm we utilized was the Naive Bayes algorithm. Naive Bayes is a probabilistic classifier algorithm that assumes that a specific piece, or group, of data is sampled from a specific distribution [9]. The distributions our team chose to work with were the Gaussian, Bernoulli, Multinomial, and Complement distributions. Each of these concerned models were implemented through the scikit-learn python package [10].

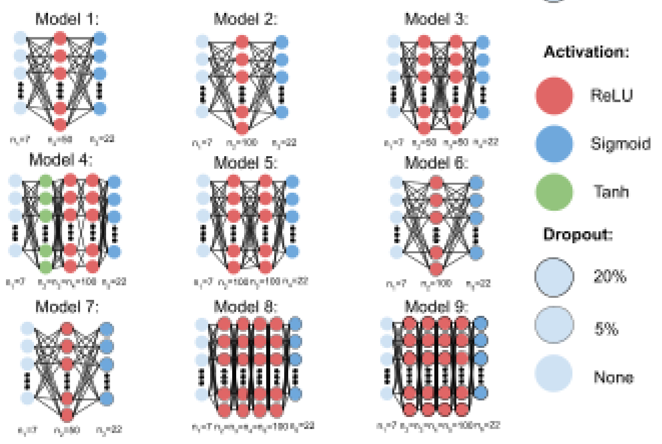
The second algorithm we utilized was support vector machines. Support vector machines work to maximize the boundary between differently labeled data, often by the projection of data into other dimensions to artificially produce linearity in the data [11]. Specifically, we used a linear decision boundary svm, a radial basis function kernel svm, a polynomial of degree 3 kernel svm, and a sigmoid kernel svm. The svm was implemented using the scikit-learn python package [12].

The third algorithm we utilized was the k-nearest neighbors classifier algorithm. K-nearest neighbors classifiers work by "polling" the nearest k data points and assigning a new datapoint a label based on the majority label of surrounding data points [13]. We tested 4 different k values for k-nearest neighbor (1, 3, 5, and 7), to grasp the general trend in increasing the number of neighbors polled for each new datapoint. K-nearest neighbors was implemented using the scikit-learn python package [14].

The fourth algorithm we utilized was decision trees for multi-class classification. From a high level, decision trees work by asking a series of questions that maximally reduce the uncertainty of a certain data point's label [15]. In the project our three different decision trees had varying maximum depths of 3, 5, and 7. The decision tree utilized in our project was from the scikit-learn python package [16].

The fifth algorithm we utilized was variously formatted neural networks. Neural networks are an incredibly prevalent method of function approximation in the world of machine learning. In short, neural networks approximate a specific function (i.e. a decision boundary) by collecting neurons (or nodes) together by weights that are passed through a non-linear activation function [17]. The various setups for our neural networks are given in Figure 5.1. These neural networks were created using the pytorch python library [18]. Each neural network was trained using the cross entropy loss function, the sgd optimizer, a learning rate of 0.5, and a total of 1000 epochs.

Figure 5.1



The sixth, and final, algorithm that we utilized was a bootstrapping of our most performant models in the form of a stacking classifier. After conducting our model evaluations (more detail in section 6), we found that our Gaussian Naive Bayes, Linear Kernel SVM, and K-Nearest Neighbor (k=7) had the highest average accuracy scores. These models were therefore combined into a stacking model using the scikit-learn python package [19].

Method Reasoning (6):

The methods given in the method documentation were chosen as they are the most commonly used techniques for utilizing machine learning on a dataset. The goal of our project was to evaluate the performance of a large variety of machine learning algorithms on our dataset. After we tried a large number of algorithms on the data we also believed it would be valuable to attempt to combine the more performant algorithms to determine whether stacking models can be used to get even better performance on the dataset.

Findings Recording (7):

Findings Table 1 - (*Note: Running the notebook again will likely yield slightly different results)

Name:	Minimum Accuracy:	Maximum Accuracy:	Average Accuracy:
Gaussian Naive Bayes	0.9864	1	0.9964
Bernoulli Naive Bayes	0.0136	0.0273	0.0227
Multinomial Naive Bayes	0.8318	0.9227	0.8923
Complement Naive Bayes	0.3182	0.4273	0.375
linear svm	0.9728	0.9955	0.9869
rbf svm	0.0136	0.0318	0.0209
polynomial svm	0.9682	0.9864	0.9795
sigmoid svm	0.0091	0.0955	0.0564

knn (n=1)	0.9636	0.9909	0.9805
knn (n=3)	0.9591	0.9909	0.9755
knn (n=5)	0.9682	0.9863	0.9795
knn (n=7)	0.9636	0.9909	0.9805
decision tree (max depth = 3)	0.1591	0.2818	0.2132
decision tree (max depth = 5)	0.3091	0.4363	0.3732
decision tree (max depth = 7)	0.5364	0.7591	0.6705
nn 1	0.0136	0.0818	0.0464
nn 2	0.02272	0.0909	0.05
nn 3	0.0273	0.7273	0.2368
nn 4	0.4682	0.6045	0.5255
nn 5	0.0591	0.7773	0.5914
nn 6	0.7091	0.8364	0.7559
nn 7	0.7727	0.8909	0.8505
nn 8	0.8045	0.9045	0.8641
nn 9	0.7455	0.8455	0.7982

Findings Table 2 (External Link - https://docs.google.com/spreadsheets/d/1sBymAbzVQdZxx44V5rg2uuKYaTur5V5wEnbJfWVAGq8/edit?usp=sharing)

Findings Analysis (8):

The models with the highest average accuracy were the Gaussian Naive Bayes, the stacked classifier, the linear svm, and finally the k-nearest neighbor (n=1 and n=7). The high performance of the Gaussian Naive Bayes makes it seem that the data favors a normal (or gaussian distribution), which seems correct when observing a histogram of the data.

On the other hand, the models with the lowest average accuracy were the rbf SVM, Bernoulli Naive Bayes, neural networks 1 and 2, and the sigmoid SVM. Out of these models the most interesting is the terrible performance of the rbf SVM. In table 2, the effectiveness of every algorithm is compared by label and based on this data alone it would seem that the rbf SVM should have performed incredibly well as all of the accuracies are around 90%. However, the testing performance of the rbf SVM is incredibly low, which means that this more complicated model seems to have very poor generalization of the dataset and has a high tendency to overfit. This trend seems to hold true for the sigmoid SVM also.

Another important finding of the project was the underperformance of neural networks. In general, the neural networks in this project were more computation and time expensive to train, yet yielded significantly worse performance than many of the more “simple” machine learning algorithms. In fact, the most high performing model was the Gaussian Naive Bayes model which is (along with k-nearest neighbor) one of the most simple possible supervised learning algorithms.

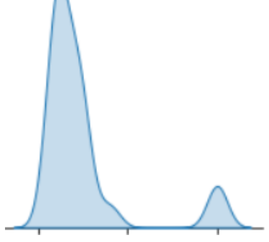
Referring back to the earlier visualizations, the key to why the less complex models were more successful is revealed. In looking at either the PCA or SVG decompositions we see distinct clusters and groups forming (specifically with grapes and apples). Through this visualization, it would seem that the data is quite clustered and that, therefore, more simple machine learning algorithms should have high performance. However, this may not always be the case as there is a possibility of PCA or SVG decomposition to distort the overall picture of the data.

Future Work(9):

For future work, there are a variety of improvements that could be done throughout the process of building the model. In the data cleaning step, we could have worked on detecting outliers through the use of Isolation forests or elliptic envelopes to determine if the high values in potassium were a cause of concern on our models (pictured below) [8]. Furthermore, because our models provided extremely successful results, we did not go through the process of making some feature data into normal distributions that arguably can improve the performance of our models by preventing the distortion of data with long tails. For example, potassium had an unusually long tail with a bump again pictured below. Thus, normalizing might have made the model not overvalue potassium levels that are extremely high. However, it is difficult to determine when the models already performed well.

A next step for this project is to combine this dataset with other crop recommender datasets to attempt to extend the range of application of our model. Currently, all of our data is from India, however there is data available online for a large number of other countries, therefore we could combine these other datasets into our dataset to attempt to train a model to recommend crops regardless of the location in the world

In the future, many real world applications could be considered. One such application would be the creation of an application to assist hobby and more professional subsistence planters with determining the correct crop to plant based on the current nutrients. By utilizing this application, the users would be able to maximize their crop yield per space, while also making sure to not over-utilize specific nutrients in the soil. Another real world application of this project could be the advisory of a board of agriculture of an area on how to best advice farmers in specific conditions as to which vegetables to grow.



Potassium's Variable Distribution found in Features/Target Analysis

1. https://www.ripublication.com/ijafst_spl/ijafstv4n4spl_11.pdf
2. https://www.sciencedirect.com/science/article/pii/S266604902100044X
3. https://ieeexplore.ieee.org/document/9418351
4. https://www.irjet.net/archives/V9/i4/IRJET-V9I4214.pdf
5. https://www.riteshajoodha.co.za/sitepad-data/uploads/2021/10/TakalaniIMITEC.pdf
6. https://www.researchgate.net/publication/331426761_Crop_Recommendation_System_to_
Maximize_Crop_Yield_in_Ramtek_region_using_Machine_Learning
7. https://www.kaggle.com/datasets/siddharthss/crop-recommendation-dataset
8. https://scikit-learn.org/stable/modules/outlier_detection.html#id1
9. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
10. https://scikit-learn.org/stable/modules/naive_bayes.html
11. https://en.wikipedia.org/wiki/Support-vector_machine
12. https://scikit-learn.org/stable/modules/svm.html
13. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
14. https://scikit-learn.org/stable/modules/neighbors.html
15. https://en.wikipedia.org/wiki/Decision_tree
16. https://scikit-learn.org/stable/modules/tree.html
17. https://en.wikipedia.org/wiki/Artificial_neural_network
18. https://pytorch.org/
19. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

Link to Collected Data Spreadsheet:
https://docs.google.com/spreadsheets/d/1sBymAbzVQdZxx44V5rg2uuKYaTur5V5wEnbJfWVAGq8/edit?usp=sharing