Using Multilayer Perceptron (MLP) to predict crop yields

Thomas Donnelly
Winona State University
tommymdonnelly@gmail.com

ABSTRACT
This study aims to develop a Multilayer Perceptron (MLP) that accurately predicts crop yields within 10% of ground truth in 80% of cases using weather data, region, soil type, temperature, fertilizer, irrigation, days taken to harvest, and rail fall. The dataset has 1 million unique data points and 10 columns. The model will be trained using a random selection of 80% of the data for training and 10% for testing. The effectiveness of the model will be derived from the accuracy and the Mean Square Error (MSE) of the model.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.* This is just an example, please use the correct category and subject descriptors for your submission.

General Terms
Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification.

Keywords
Keywords are your own designated keywords.

1. INTRODUCTION
As technology advances, it is important to explore as many ways it can be used to improve the world and the lives of people around us. One area where this could have a great impact is farming. Farming is notoriously difficult to predict. On average, crop yields in the US have increased since 1995, with 2023 yielding 179.3 bushels of corn per acre, compared to 113.5 in 1995 (USDA Crop, 1). This, however, can vary year to year depending on how much fertilizer is used, yearly rainfall, and the general region and climate. To help with this prediction, this study plans to create a neural network to assist in the prediction. This study plans to create a multilayer perceptron (MLP) to achieve this goal.

2. Hypothesis
Multilayer Perceptron (MLP) accurately predicts crop yields within 10% of ground truth in 80% of cases using weather data, region, soil type, temperature, fertilizer, irrigation, days taken to harvest, and rail fall.

3. Methods
3.1 Model Basics
This project will focus on developing software. This study plans to use Python, a coding language with a large library of neural network creation tools, to create the model using the Pandas library. The model I will be creating is a Multilayer Perceptron (MLP). An MLP is made up of three layers. The first layer receives all input data, including weather data, region, etc. The second layer is all the fillings or the hidden layers. These processes take the input data and try to make predictions based on it. While additional hidden layers can improve performance, having more does not necessarily result in a better model. Lastly, the output layer takes all the info the hidden layers give it and makes a final prediction or guess on the correct answer. See Figure 1 for reference.

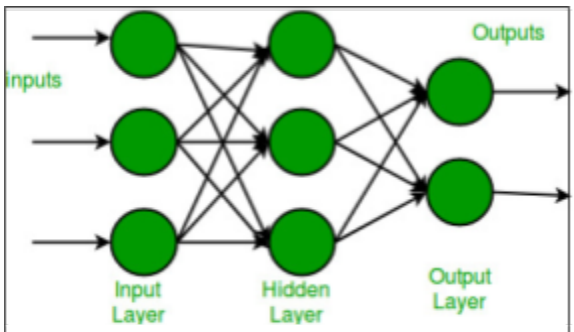

Figure 1. Deep Learning Model Visualization (Taken from Multi-Layer Perceptron Learning in TensorFlow.)

The basic structure of the model using TensorFlow is as follows:

1. Input(shape=(X_train.shape[1],)). Input Layer that just ensures the model fits the data.
2. Dense(x). X is the number of neurons that one layer can have to process data. This can be thought of as the thinking layer.
3. BatchNormalization(). This makes it so multiple Dense layers can read each other's output better.
4. ReLU() helps the model learn complex patterns in the data.
5. Repeat 2-4 with X either increasing or decreasing between layers.
6. Dense(1). This layer is the final output and prediction.

3.2 Geometric Pyrimind Formula
A formula was used to help narrow down the size of the model for training. This is called the Geometric Pyramid Formula. In short, the formula helps prevent creating a model that is too large for the data set given. The version of the formula used goes as follows: # of parameters / 10 > (layer size * previous layer size) + …… (layer(i) * layer(i - 1)).

3.3 Dataset
The dataset that will be used is taken from Kaggle, a site where you can get free datasets to work with. The dataset is titled "Agriculture Crop Yield Dataset" and is a 1 million non-null unique data point set (Agriculture Crop Yield Dataset). From my analysis of the data, it appears to be a generated set, which means it is not taken from the real world but instead created using similar data or trends in the field of interest. This is important to keep in mind as it may affect how the model can predict real-world data. The dataset has ten columns, which are as follows, with date types: Region object, Soil_Type object, Crop object, Railfall_mm float64, Temperature_Celsius float64, Fertilizer_Used boolean, Irrigation_used boolean, Weather_Condition object,

Days_to_Harvest int64, Yield_tons_per_hectare float64 (Agriculture Crop Yield Dataset). The first nine columns will be used to predict the ground truth or actual value of the yield per hectare. Figures 2.1-2.5 are visualizations of the data.

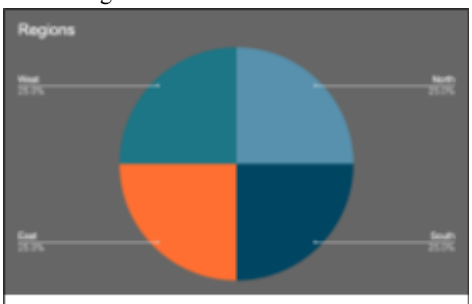

Figure 2.1 Visualization of Regions in the Dataset (Created using Matplotlib)

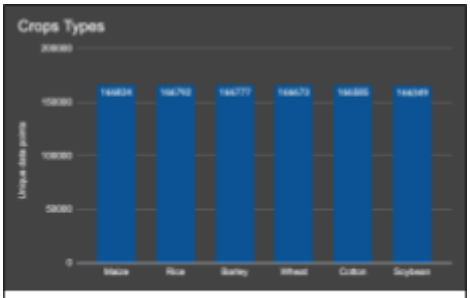

Figure 2.2 Number of Crop Types in the Dataset (Created using Matplotlib)

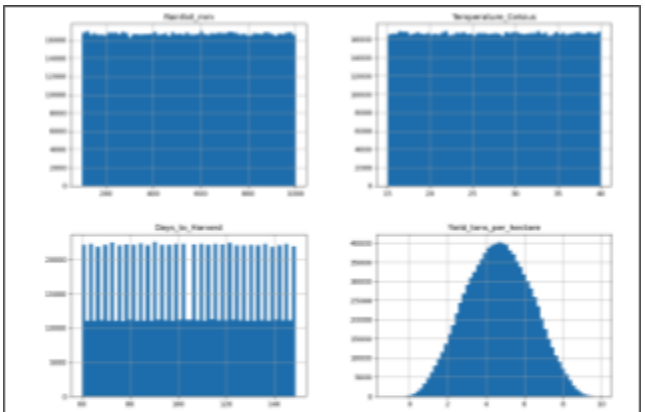

Figure 2.3 Display of the spread of Rainfall, Temperature, Days to Harvest, and Yield values. (Created using Matplotlib)

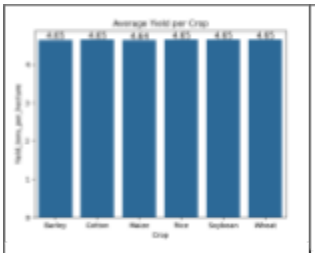

Figure 2.4 The average yield per crop type (Created using Matplotlib)

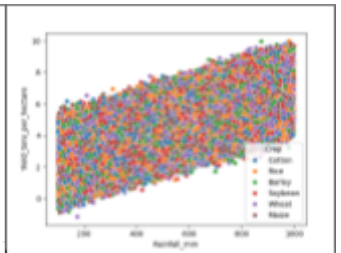

Figure 2.5 How rainfall affects crop yield (Created using Matplotlib)

3.4 Training

The method I will use for the training of the MLP is to divide the dataset into two parts. For the first part, 90% of the data will be available to the model to train on and evaluate its answers. Meanwhile, a random 10% will be set aside for after the model is done training. This data will be used to test how good the model is on previously unseen data. If the model was tested on data it had already seen, the chances would be high that it would recognize the data and learn the answer, and not how to get the answer. This is also assisted by having so many points of data, with 1 million being a large set, making it hard for the model to learn the answer and not the how.

3.5 Evaluation

Two methods will be used to show how effective the model is during and after it is created. The first is accuracy, the goal is for the model to at least 80% of the time be either 10% over or under the testing data. This can be implemented using the following logic:: accuracy = average((predicted >= actual * 0.9) OR (predicted <= actual * 1.1)) >= 0.8. The other method is Mean Square Error (MSE), this formula helps determine how accurate a model is over data, with a lower value meaning the more accurate it is (Mean Squared Error). See Figure 3 below for a similar research using MSE to display the accuracy of models. Their study is a good reference point for the project, as they tested models using only select columns to see which helped get to the ground truth the best. Meanwhile, this study will be using all the columns.

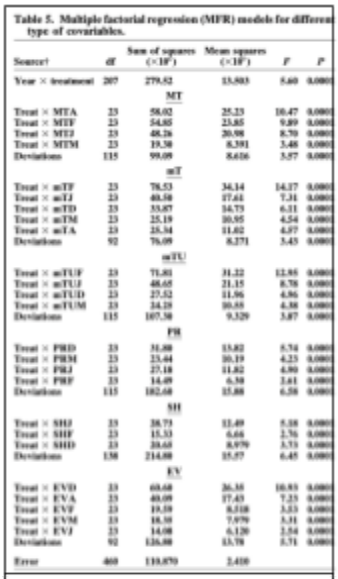

Figure 3. A similar study that compared multiple models and select data with Mean Square Error being one of the evaluations (Vargas 949)

4. EXPERIMENT

4.1 Data Normalization

The first problem encountered in creating a model was that a third of the data could not be run due to being in an unsupported

format. Figure 4 below shows the columns divided by their data formats.

Categorical	Numerical	Boolean
Region	Rainfall_mm	Fertilizer_Used
Soil_Type	Temperature_Celsius	Irrigation_Used
Crop	Days_to_Harvest	
Weather_Condition		

Figure 4. Training columns sorted by data type

Sklearn was used to create a pipeline to separate and manage the data in the format it's in. Since Region, Soil_Type, Crop, and Weather_Condition were strings, they could be run as is. The boolean or yes-no columns Fertilizser_Used and Irrigation_Used had to be transformed to be processed. The method used to achieve this was to convert the data to numbers, with True becoming 1 while False became 0. All the numerical values were able to be run, but a normalizer was applied to help the model read the data, and if missing values were found, the average score of the set was put in their place. After all the data was modified, it was all put back together, and the model was ready to start running.

4.2 Hardware

The expectation was to run the model on a dedicated NVIDIA GPU for its training. This would allow for quicker training, a larger possible model size, and a larger batch size. A larger model size would allow for more neurons; this does not always mean more performance out of a model but would allow the ability to see if a bigger model would be better. The batch size is how much data in a run-over data is stored in memory at a time. A larger batch size generally means higher performance of a model. It was found, however, that the versions of Nvidia GPU driver (Nvidia-smi) and Compute Unified Device Architecture (NVCC) that were installed on the machine were too new for Tensorflow, a library used to create the model, and a different machine was chosen. In the end, a MacBook M2 was used to train the model. This meant that the model had to be trained on a CPU, but given the time constraints, it was good enough.

4.3 Learning Rate

The learning rate, or the rate at which the model can learn the was chosen to be 0.001. A rate of 0.01 was tested but was not used due to the model learning the training data and learning how to predict using the data too quickly. Figure 5 is a quick example of running the model on a 0.01 learning rate. As the epochs increase or the number of times the model goes over the training data, the rate of loss on the testing data starts to reverse. Loss is a measure of how confident the model is in its answer. The reverse in loss in the testing data, but not a similar reverse in training data, shows that the model started to learn the data and not from the data.

Figure 5. Comparing the model using different learning rates

5. RESULTS

See Figure 6 for the best run as it was trained over its 10 Epochs with a 0.001 learning rate and a best of 0.3979 Mean Squared Error. Figure 7 shows how the best result is structured. It has 10 Epochs, 4 hidden layers with their size in order being 256, 128, 64, and 32 neurons.

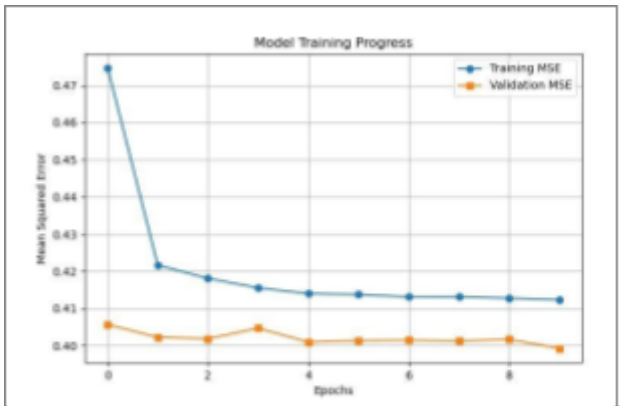

Figure 6. The best model over its epochs showing the Training MSE vs the Testing Validation MSE (Created using Matplotlib)

6. ANALYSIS

Using the best model, the study was able to achieve a total of 62.46% of the predictions being within 10% of the ground truth yield and 87.98 % of the predictions being within 20% of the ground truth yield. This has not met the goal of 80% within 10% of the ground. Training the model with the best parameters over longer epochs would not yield better results, as the model would start to learn the set and could lose accuracy. If you see Figure 8, you can see how little the MAE decreases as the epochs increase. The parameters do, however, have room to increase due to the model being well under the budget of 100,000. The model is 45,825 using the Geometric Pyramid Formula. Though when tested, it was found to slightly decrease the accuracy.

Figure 7. Visualization of the structure of the best model (Created using Netron.app)

Epoch	Training Loss	Training MAE	Validation Loss	Validation MAE
1	0.9848	0.6191	0.2589	0.4057
2	0.2812	0.423	0.2542	0.4023
3	0.2754	0.4185	0.2538	0.4019
4	0.272	0.4161	0.2575	0.4047
5	0.2693	0.4138	0.2528	0.401
6	0.269	0.4137	0.2535	0.4014
7	0.2682	0.4129	0.2536	0.4015
8	0.2693	0.414	0.2533	0.4013
9	0.2682	0.4132	0.2536	0.4017
10	0.268	0.4129	0.2507	0.3993
Final Evaluation			0.2485	0.3978

Figure 8. Table showing the best model's training data

7. CONCLUSION

In conclusion, the study did not achieve the goal it set out to achieve. If the goal post was moved to just 20% within the ground truth, then it would have achieved such. Meanwhile, 62.26% is not nothing. I believe this study has achieved its goal using a network to predict crop yield given the data.

8. REFERENCES

[1] Kaggle. "Agriculture Crop Yield Dataset." Kaggle, 2021, https://www.kaggle.com/datasets/samuelotiattakorah/agriculture-crop-yield/data.

[2] Jha, Dinesh, et al. "Application of Artificial Neural Networks for Predicting Crop Yield: A Review." Frontiers in Plant Science, vol. 10, 2019, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2019.00621/full.

[3] "Corn Yield." USDA National Agricultural Statistics Service, https://www.nass.usda.gov/Charts_and_Maps/graphics/cornyld.pdf.

[4] "Crop Progress & Condition." USDA National Agricultural Statistics Service, 2024, https://www.nass.usda.gov/Charts_and_Maps/Crop_Progress_&_Condition/2024/index.php.

[5] Subramanian, S., et al. "Early Crop Yield Prediction for Agricultural Decision Making Using Remote Sensing Data." Journal of Water and Climate Change, vol. 14, no. 12, 2023, pp. 4729-4741, https://iwaponline.com/jwcc/article/14/12/4729/99202/Early-crop-yield-prediction-for-agricultural.

[6] Vargas, Mateo. "Interpreting Treatment × Environment Interaction in Agronomy Trials." Agronomy Journal, vol. 93, no. 4, 2001, pp. 949–960.

[7] "Multi-Layer Perceptron Learning in TensorFlow." GeeksforGeeks, https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/.

[8] "Mean Squared Error." GeeksforGeeks, https://www.geeksforgeeks.org/mean-squared-error/.

[9]