Crop Classification based on Soil Information

Krish L Sharma, Samarth Meghani, Sweta Singh, Ankur Jain[a]

[a]*VIT, Bhopal, , Bhopal, 466114, Madhya Pradesh, India*

Abstract

India is known for its rich Agricultural land. Farmers have a great knowledge of farming, climate, irrigation, etc. But as we can see the world is changing with respect to climate and soil content we as human beings are unable to cope up with this change. Hence, helping the farmers and the ones who are going to start farming we have made a classification model which takes the soil information as input and gives you the classification of which type of crop is best suited for the given soil. This will not only help in knowing which crop can be grown but also what all changes are required to grow another type of crops.

Keywords: Soil Information, Crop recommendation, Artificial Neural Network

Contents

1 Introduction **1**

2 Problem Formulation **1**

3 Methodology **1**

4 Results **3**

5 Conclusions **5**

1. Introduction

Just like humans are heavily dependent on food, water and climate, crops are dependent and highly sensitive to the above parameters. In order to grow a healthy and high yield crop we need to take care of the soil content(Macro nutrients content: Nitrogen, Phosphorus, Potassium; soil temperature and environment temperature; pH level of soil and the humidity level of the soil).

Soil information is a very crucial piece of information for farming. Plants macro nutrients such as Nitrogen, Phosphorus and Potassium should be well balanced by the farmers by adding manure or fertilizers. We need to know how much amount of fertilizers is to be added in order to maintain a balanced soil for the plants. If a proper balance is not maintained in the soil then due to high concentration in the soil of the nutrients the water will be retracted from the plant also known as reverse osmosis. This will suck the plant dry which will hence lead to death of the crop.

2. Problem Formulation

Nowadays due to increase in population there is an acute shortage of food supply and more than that farmers are facing very tough times because of financial issues. Many experienced farmers are aged and are unable to work properly and on the other hand the youngsters in villages are moving out to the city to work. Information about soil and weather are the key source for farmers to make a good plan. If the initial planning is done properly and all the counter measures are taken care of then the yield percentage will increase exponentially.

Moreover, we need to make a system which will help both experienced and novice farmers. This will help the obsolete knowledge of farmers to stay up-to date and the novice farmers will gain more knowledge with some additional information.

Our data set that we have picked is from the **kaggle** (the link is given in the reference). It contains soil nutrient (Nitrogen, Phosphorus and Potassium content), soil temperature, solid humidity, soil pH level, amount of rainfall and finally the crop label.

Soil nutrient is an important parameter about the soil as it tells us the amount of macro nutrients required for the plant, which are Nitrogen, Phosphorus and Potassium. This information will give a heads up that how much fertile the land is and at the same time it will give a rough idea of how much fertilizer is required for the crop to grow.

Soil humidity tells us how much water is present in the soil and moreover it will give us an idea how much irrigation is required for the plant. Also the temperature will tell us is it feasible for the plant to grow. The temperature also tells us the bacteria growth rate and hence be aware of the future diseases that might arrive. The pH level tells us the amount of acidity of the soil.

Finally our objective is to built a classification system which will take the soil information and then classify which type of crop will be best for farming for the particular soil.

3. Methodology

After reviewing all the previous work on recommendation system on farming, it has been seen that the data are based on either solar or soil moisture and its type. These are very good

point with which we can give a recommendation but our research puts more emphasis on soil content information rather than the above given content.

Moreover, our ANN model is more about classification whereas the other research paper are more about regression model. This model is basically recommending what type of crop can be grown based on the information about the soil content.

Let us discuss about the ANN model we have implemented. We have 8 models being implemented using ANN single-layer feed-forward network. We have used tanh, Rectified Linear Unification(ReLU), Sigmoid and Softmax function in each of the models with a combination of two functions on each model. As for the loss function, we are using 'Categorical Cross Entropy' and 'Categorical Hinge' which gives a better loss value for multi-classification problems.

Tanh and ReLU are used to simplify the numerical data so that the classification activation functions(sigmoid and siftmax) can be used to classify the data properly.

Design: As we have discussed that there are 8 models to compare each other's performance and get the best out of all the models. Model1 and Model2 contains tanh as the hidden layer activation function and sigmoid as the output function. The only difference between these two model is the loss function. Model1 contains categorical cross entropy and Model2 contains categorical hinge as loss function.

Similarly, Model3 and Model4 are made up of ReLU function as the activation function of hidden layer and Sigmoid function as the output layer function. Model3 has categorical cross entropy as loss function and Model4 has categorical hinge as loss function.

Now the rest 4 models (Model5, Model6, Model7, Model8) are going to be the same as the above mentioned models but the only difference is that all the output layer function contains Softmax function.

The basic idea of using this combinations of activation functions and loss functions is to see which combination gives the best performance by doing trial and error.

Analysis: Before implementing the models, lets first analyse the data we have.

First, we need to understand what we are dealing with by looking at the data set.

```
In [3]: crop_dataset.head()
```

Out[3]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Figure 1: Data set of the soil information

As we can see in *Figure 1*, there is Nitrogen, Phosphorus, Potassium, Humidity, pH, rainfall and label in the data set.

Since we got the meaning of the features of the data set, now we need should know the data type of each feature for 2 reasons:

1. we need to see what sort of calculations we are dealing with.
2. we need to also see what kind of data type changes we need to perform.

```
crop_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            2200 non-null   int64
 1   P            2200 non-null   int64
 2   K            2200 non-null   int64
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

Figure 2: Data type of each feature belonging to the data set

From *Figure 2*, we get to know that the data set is consistent as there is no null records present in it. Moreover, all the features are numerical data except label which is a categorical data. Now we need to understand that Artificial Neural Network is based on mathematical computation of the data set which will at then end give a numeric data. Since label is the data which we want to predict, we need to convert it into numerical values. Lets first look how many unique values does label feature contains.

```
crop_dataset['label'].unique()
```

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

Figure 3: Unique values containing in Label column

In *Figure 3*, we can see that we have 22 unique values in label, which means that we have to classify the data in one of 22 labels. Now we need to convert the labels in such a format that the calculations wont bother the output in what so ever format. We can use One Hot Encoding or Label Encoder. One Hot Encoding is much better for this situation as the label contains a nominal type data, which means the data doesn't have any order whereas Label Encoder will just give us the numeric data in an order which will disrupt the calculation as each level will have different weights. No doubt both of them will work but it will be much more computationally feasible. We will discuss how we have implemented the One Hot Encoding in the *Implementation* section of the Methodology.

We have finished our analysis of the data set. Lets implement the neural network based on the information we have.

Implementation: From the above analysis we have the following information:

1. We have 8 features in the data set out of which 7(Nitrogen, Phosphorus, Potassium, temperature, humidity, rainfall and pH) are independent and 1(Label) is dependent.

2. The dependent feature, which is called Label in data set, is a categorical nominal data. Categorical nominal data are those string value data type which does not have any order but it is just to classify from a set of entities.
3. Since the dependent feature is a string value, we need to convert it into numerical value so that we can calculate it as an output value in ANN. For that we are going to use One Hot Encoding as discussed above.

We have understood the architecture of the models we are going to use and also we have analysed the data set using pandas. Now let us make the neural network for each model. Below will be the code implementation of each model. We have used 22 nodes in the hidden layer and 22 nodes in output layer. 22 nodes in the output layer is justified by the reason that there are 22 classes in the data set that we need to predict so we are going to have 22 such nodes in the output layer.

Before making the neural network, first we need to pre process the data. We need to convert the *Label* feature to One Hot Encoding. In the below figure, *Figure 4* we can see how we can convert the feature into One Hot Encoding using sci-kit package.

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

ohe = OneHotEncoder()

crop_dataset_encoded = pd.DataFrame(ohe.fit_transform(crop_dataset[['label']]).toarray())

crop_dataset_encoded.info()
```

Figure 4: One Hot Encoding of *Label* feature of the data set

Now, everything is ready for our neural network. We now have to create the neural network using tensorflow package. We have discussed about the architecture of all the models that we are going to use. Let us see the coding perspective of each model.

```
#creating neural network

model1 = tf.keras.models.Sequential()
model1.add(tf.keras.layers.Dense(units = 22, activation = 'tanh'))
model1.add(tf.keras.layers.Dense(units = 22, activation = 'sigmoid'))
model1.compile(optimizer =  'adam', loss = 'categorical_crossentropy', metrics = ["accuracy"])

model1.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 5: *Model1* Code Implementation

```
#creating neural network

model2 = tf.keras.models.Sequential()
model2.add(tf.keras.layers.Dense(units = 22, activation = 'tanh'))
model2.add(tf.keras.layers.Dense(units = 22, activation = 'sigmoid'))
model2.compile(optimizer =  'adam', loss = 'categorical_hinge', metrics = ["accuracy"])

model2.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 250, batch_size = 10)
```

Figure 6: *Model2* Code Implementation

These are the models that have been made. As mentioned, we have used all the possible combination of the activation function and the loss function trying to find the best combination possible.

4. Results

After making all the models to solve the problem, now is the time to compare performances of all the models with each other

```
#creating neural network

model3 = tf.keras.models.Sequential()
model3.add(tf.keras.layers.Dense(units = 22, activation = 'relu'))
model3.add(tf.keras.layers.Dense(units = 22, activation = 'sigmoid'))
model3.compile(optimizer =  'adam', loss = 'categorical_crossentropy', metrics = ["accuracy"])

model3.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 7: *Model3* Code Implementation

```
#creating neural network

model4 = tf.keras.models.Sequential()
model4.add(tf.keras.layers.Dense(units = 22, activation = 'relu'))
model4.add(tf.keras.layers.Dense(units = 22, activation = 'sigmoid'))
model4.compile(optimizer =  'adam', loss = 'categorical_hinge', metrics = ["accuracy"])

model4.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 8: *Model4* Code Implementation

```
#creating neural network

model5 = tf.keras.models.Sequential()
model5.add(tf.keras.layers.Dense(units = 22, activation = 'tanh'))
model5.add(tf.keras.layers.Dense(units = 22, activation = 'softmax'))
model5.compile(optimizer =  'adam', loss = 'categorical_crossentropy', metrics = ["accuracy"])

model5.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 9: *Model5* Code Implementation

```
#creating neural network

model6 = tf.keras.models.Sequential()
model6.add(tf.keras.layers.Dense(units = 22, activation = 'tanh'))
model6.add(tf.keras.layers.Dense(units = 22, activation = 'softmax'))
model6.compile(optimizer =  'adam', loss = 'categorical_hinge', metrics = ["accuracy"])

model6.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 10: *Model6* Code Implementation

```
#creating neural network

model7 = tf.keras.models.Sequential()
model7.add(tf.keras.layers.Dense(units = 22, activation = 'relu'))
model7.add(tf.keras.layers.Dense(units = 22, activation = 'softmax'))
model7.compile(optimizer =  'adam', loss = 'categorical_crossentropy', metrics = ["accuracy"])

model7.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 11: *Model7* Code Implementation

```
#creating neural network

model8 = tf.keras.models.Sequential()
model8.add(tf.keras.layers.Dense(units = 22, activation = 'relu'))
model8.add(tf.keras.layers.Dense(units = 22, activation = 'softmax'))
model8.compile(optimizer =  'adam', loss = 'categorical_hinge', metrics = ["accuracy"])

model8.fit(X_train, y_train, validation_data=(X_test,y_test), epochs = 200, batch_size = 10)
```

Figure 12: *Model8* Code Implementation

and find out the best model. The criteria for good performance over the other is:

1. Less over fitting or under fitting.
2. Good training accuracy and testing accuracy.
3. getting towards accuracy at a faster rate, i.e, with less epoch coming to a good accuracy.

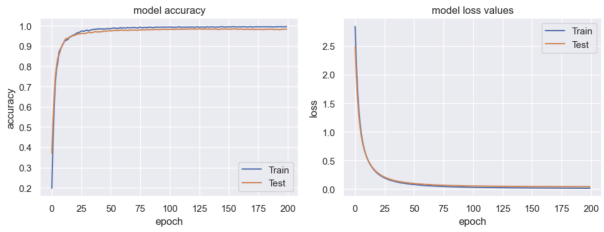

Figure 13: Model 1 Performance

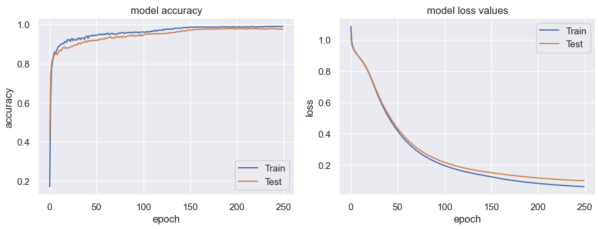

Figure 14: Model 2 Performance

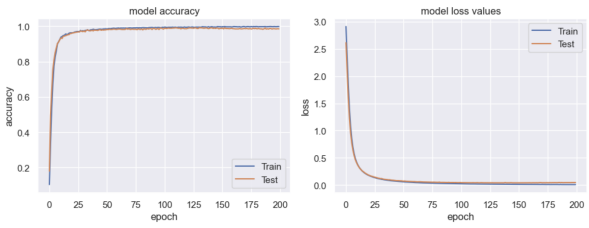

Figure 15: Model 3 Performance

From the above graphs(*Figure 13, Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, Figure 20*), as we can see that:

1. Model1 has **99.772%** as training accuracy and **98.181%** as testing accuracy.
2. Model2 has **98.712%** as training accuracy and **97.272%** as testing accuracy.
3. Model3 has **99.848%** as training accuracy and **98.522%** as testing accuracy.
4. Model4 has **99.091%** as training accuracy and **97.5%** as testing accuracy.
5. Model5 has **99.621%** as training accuracy and **98.522%** as testing accuracy.
6. Model6 has **99.621%** as training accuracy and **98.182%** as testing accuracy.
7. Model7 has **99.848%** as training accuracy and **98.977%** as testing accuracy.
8. Model8 has **99.545%** as training accuracy and **98.522%** as testing accuracy.

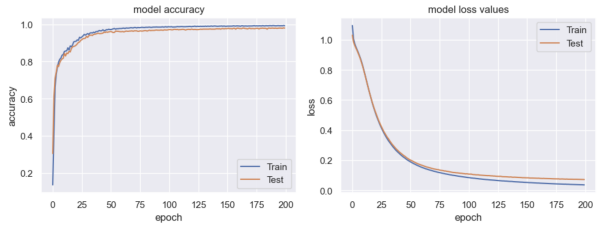

Figure 16: Model 4 Performance

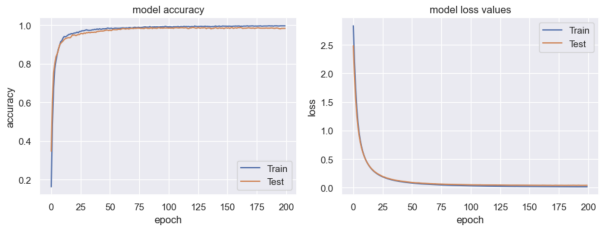

Figure 17: Model 5 Performance

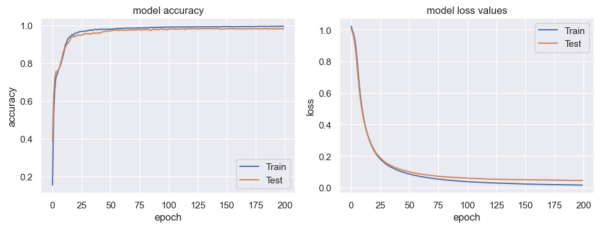

Figure 18: Model 6 Performance

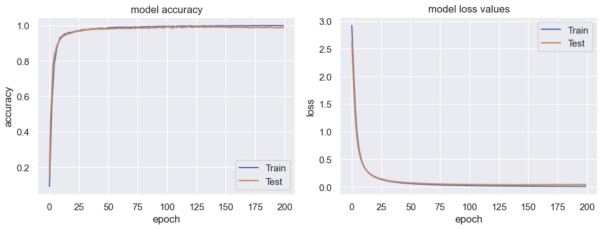

Figure 19: Model 7 Performance

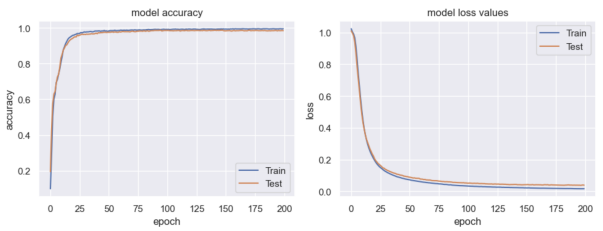

Figure 20: Model 8 Performance

5. Conclusions

From the above results, based on the accuracy of each model and the parameters of optimum model selection, we can conclude that Model7 is the most optimal model for the classification problem. Also, Model8 and Model5 also show some promising results based on the training.

Acknowledgements

Thanks to Professor Ankur Jain, who has given us the opportunity to let us do this research on Artificial Neural Networks used for farming purposes.

References

[Zahraa E. Mohamed(2019)]Using the artificial neural networks for prediction and validating solar radiation

[Naim Karasekreter, Fatih Basciftci, and Ug ur Fidan(2013)]A new suggestion for an irrigation schedule with an artificial neural network