Homework 2

Spencer Au

Introduction

The problem I am working on is to classify the "Quality" of apples as either "good" or "bad" based on their Size, Weight, Sweetness, Crunchiness, Juiciness, Ripeness, and Acidity. This is important because it will allow us to sort apples more efficiently and accurately, such as quickly sorting bad apples from good apples in a farm or grocery store. I have chosen to employ both a traditional machine learning model and a deep learning model to compare their performance.

Analysis

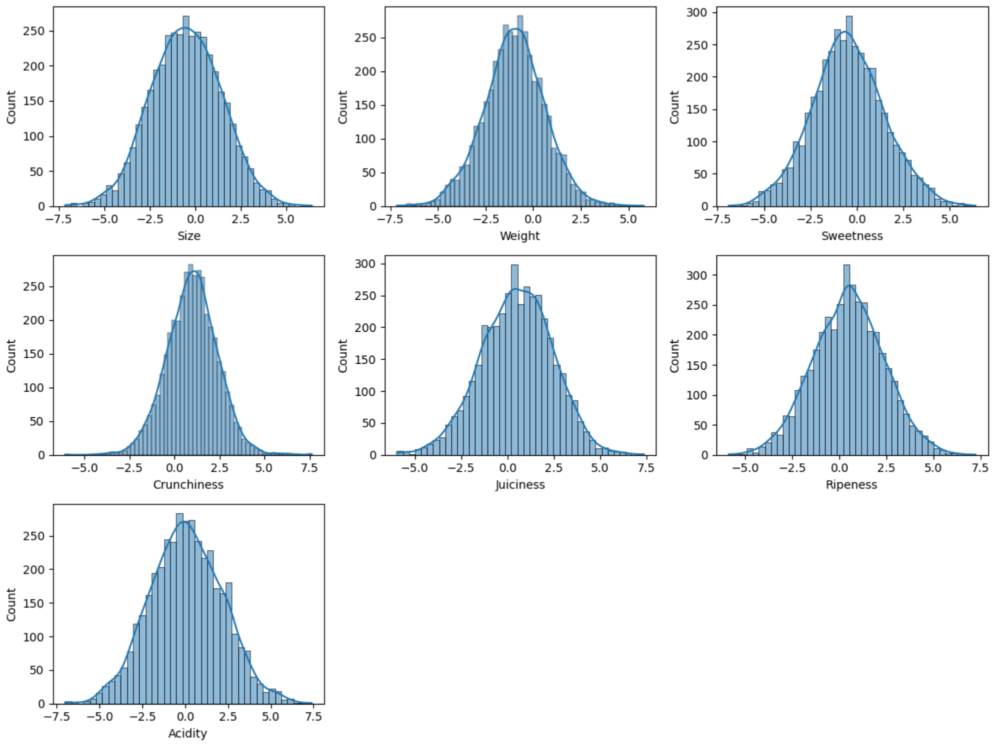
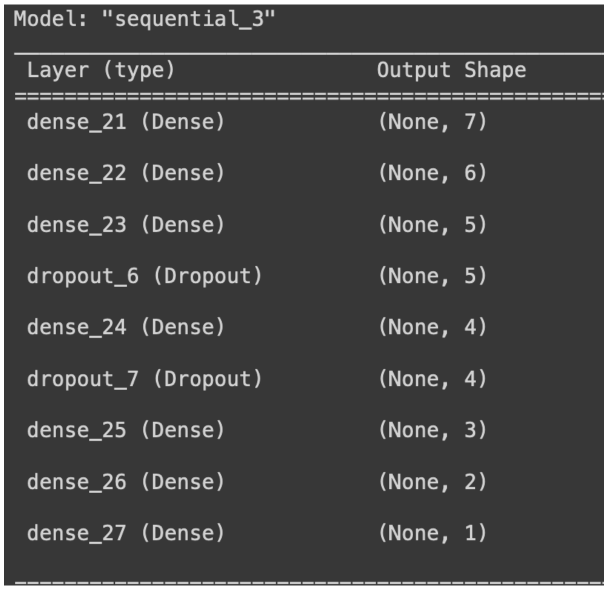


Figure 1: Distribution of Each Variable

The data is made of 7 variables, named Size, Weight, Sweetness, Crunchiness, Juiciness, Ripeness, and Acidity and the predictor variable is named Quality. The predicted variable is a binary variable, with "good" and "bad" as the two possible values. The data is relatively clean, with no missing values. Overall it seems like all the variables are normally distributed.

Methods

We used two models: a traditional machine learning model and a deep learning model. The traditional machine learning model we used was a Random Forest Classifier. The deep learning model we used was a simple feedforward neural network with 7 hidden layers, with 7, 6, 5, 4, 3, 2, and 1 neurons in each layer, respectively. This means that the input layer has 7 neurons, corresponding to the 7 features, and the output layer has 1 neuron, corresponding to the predicted variable. Hidden layers in a neural network are like the middle steps in a decision-making process, where each layer progressively refines the information from the input to help make a final decision or prediction at the output. We used the ReLU activation function for all the hidden layers and the sigmoid activation function for the output layer. ReLU (Rectified Linear Unit) is like a gate that lets positive numbers pass through unchanged but blocks negative numbers, turning them into zeros. Sigmoid, on the other hand, squeezes numbers into a tight range from 0 to 1, making it easier to decide between two options, like yes or no. We also used dropout layers with 0.2 between the layers of 5 and 4 and the layers of 4 and 3 to prevent overfitting. Basically this means that 20% of the neurons in the layer are randomly turned off during training. We also used L2 regularization with a penalty of 0.001 in the hidden layers with 6 and 4 nodes to prevent overfitting as well. L2 regularization is like a tax on the weights of the neurons, which makes the model less likely to rely too much on any one neuron.



```
Model: "sequential_3"
_________________________________________________
 Layer (type)                Output Shape
=================================================
 dense_21 (Dense)            (None, 7)

 dense_22 (Dense)            (None, 6)

 dense_23 (Dense)            (None, 5)

 dropout_6 (Dropout)         (None, 5)

 dense_24 (Dense)            (None, 4)

 dropout_7 (Dropout)         (None, 4)

 dense_25 (Dense)            (None, 3)

 dense_26 (Dense)            (None, 2)

 dense_27 (Dense)            (None, 1)

=================================================
```

Figure 2: Model Architecture

Both models were trained and tested on a 80/20 split of the data, which means that 80% of the data was used to train the model, and 20% was used to test the model. The traditional machine learning model was trained using the Random Forest Classifier from the scikit-learn library. A Random Forest Classifier is like a group of decision trees that work together to make a decision. Each decision tree is like a person making a decision based on a set of rules, and the Random Forest Classifier is like a group of people voting on the best decision. The deep learning model was trained using the Adam optimizer and the binary cross-entropy loss function. The Adam optimizer is like a GPS that helps the model find the best path to the solution, and the binary cross-entropy loss function is like a score that tells the model how well it's doing. The metrics we used to evaluate the models were accuracy and AUC. Accuracy is like the percentage of questions the model got right, and AUC is like the area under the curve of a graph that shows how well the model can distinguish between the two classes.

For the Decision Tree model, we used the following hyperparameters found via GridSearch, which is like hosting an audition to find the best settings for your computer program, by trying out every combination of adjustments and picking the one that performs the best. The optimal hyperparamters were found to be: 'max_depth': 10, 'min_samples_leaf': 10, 'min_samples_split': 2. Max depth is like the maximum number of questions the decision tree can ask, min samples leaf is like the minimum number of apples that can be in a group, and min samples split is like the minimum number of apples that can be in a group before the decision tree can ask a question.

Results

Detailed discussion of how your model performed. Include a discussion about whether or not Deep Learning was necessary in this situation.

Model	Train Accuracy	Train AUC	Test Accuracy	Test AUC
Deep Learning	0.7497	0.8142	0.7437	0.8208
Decision Tree	0.8916	0.9676	0.8125	0.89

The deep learning model had a train accuracy of 0.7497 and a train AUC of 0.8142, and a test accuracy of 0.7437 and a test AUC of 0.8208. The decision tree model had a train accuracy of 0.8916 and a train AUC of 0.9676, and a test accuracy of 0.8125 and a test AUC of 0.89. The decision tree model performed better than the deep learning model in terms of both accuracy and AUC. This means that the decision tree model was better at distinguishing between the two classes and making the right decision. This suggests that deep learning was not necessary in this situation, and that a traditional machine learning model was sufficient and even more effective in order to solve the problem. This is likely because the data was relatively simple and the decision tree model was able to find a simple set of rules to make the decision, whereas the deep learning model was too complex. This is like using a sledgehammer to crack a nut, when a nutcracker would have been sufficient. The Decision Tree did seem to be overfitting on the training data, but it still performed better than the deep learning model on the test data.

Reflection

It does seem to be the case that a deep learning model is unnecessary and arguably a waste of compute resources when trying to predict "simple" data like the quality of an apple based on well normalized and distributed data. I did struggle with the deep learning model, as I had to try out many different combinations of hyperparameters and layers to get the model to perform well. I would approach similar problems in the future by first trying out a simple traditional machine learning model and then only using a deep learning model if the traditional machine learning model does not perform well. I would also try to mess around with the architecture of the deep learning model to see if I could get it to perform better as well as employ more advanced techniques like batch normalization and early stopping. For the decision tree model, I would try to employ regularization techniques to prevent overfitting and try out different hyperparameters to see if I could get the model to perform better.