Final Write-Up

Rishabh Srivastava

December 19, 2019

1 Overview

I decided to work with the iris dataset and try to see how multiple classifiers would perform on this data. For my analysis, I chose to use a Decision Tree Classifier, KNN Classifier, Logistic Regression, and a Neural Network. I went about my analysis by tweaking hyper parameters for my model to see how this would affect performance. For my evaluation, I used 5-fold cross validation to test for accuracy. I was attempting to see which classifier would perform best on this dataset.

2 Iris Dataset

The iris dataset is a multi class dataset of 3 different types of flower species: Virginica, Versicolor, and Setosa. The dataset provides four attributes for each class: Petal Length, Petal Width, Sepal Length, and Sepal width. I imported the data directly into my code through the *sklearn.datasets* module. Scikit-Learn allowed me to directly import my data and labels using the $load_iris$ function. This made it extremely easy for me as I didn't have to worry about any kind of preprocessing.

3 Decision Tree Classifier

A Decision Tree is a great way of classifying multi-class data. It uses a supervised machine learning approach and splits the data according to specific hyper parameters. For my analysis, I decided to play around with the *max depth* hyper parameter and see how this affects accuracy. The *max depth* hyper parameter tells the model when to stop splitting the data once its reached a certain depth. The results for the Decision Tree were quite promising and the model had very high accuracy regardless of depth. The accuracy during hyper parameter tuning oscillated around 96 percent.

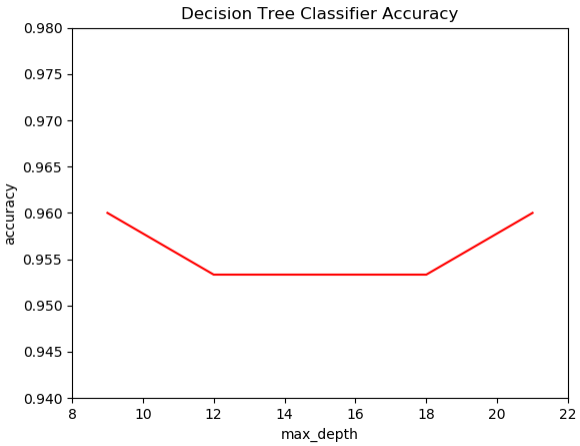

Figure 1: Decision Tree Classifier

The model showed the most promising results while having a max depth of 21.

4 KNN Classifier

KNN is a type of instance based learning which works very well on classification problems. This algorithm is supervised and finds a class for an input based on the closest neighbors from the training data. This model trains quite smoothly because all major computation is only used during the classification step. During my analysis I chose to play around with the K value that I was selecting for my model. My values for K ranged from 3 to 11 inclusively. This model had extremely accurate results and had an oscillating accuracy around 98 percent.

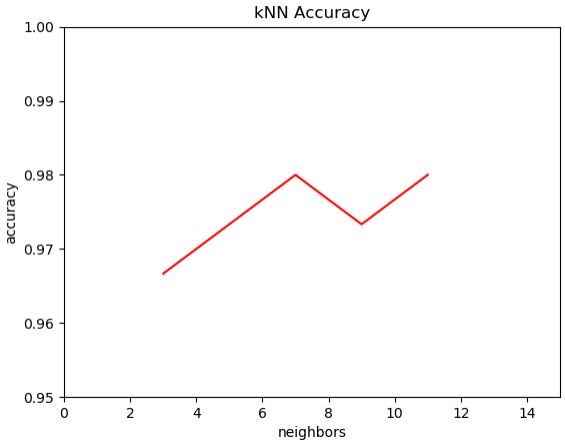

Figure 2: KNN Classifier

The most promising results I received were when I set my k value to 11. The model returned with an accuracy slightly above 98 percent.

5 Logistic Regression

Logistic Regression takes advantage of a logistic function in order to classify binary data. It was not surprising that this form of supervised learning did not perform very well on our data. I used logistic regression more so out of curiosity as there were 3 classes in our Iris dataset, and I was curious to see just how poorly it would perform. It performed the worst out of all the modeling techniques I used with its highest accuracy only being slightly above 0.7.

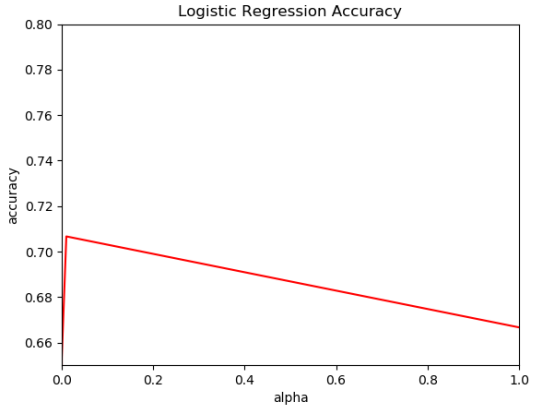

Figure 3: Logistic Regression

This technique by far had the worst accuracy and won't even be in my consideration when I go to choose the best model.

6 Neural Network

This modeling technique was the one I was most excited about by far. I had to read a lot more into how to optimize neural networks as they learn very differently than conventional machine learning models. I had to understand how neurons and the intermediary functions they hold operate and how to implement it in terms of my dataset. Overall the results were very good and the model received an oscillating accuracy between 0.97 and 0.98.

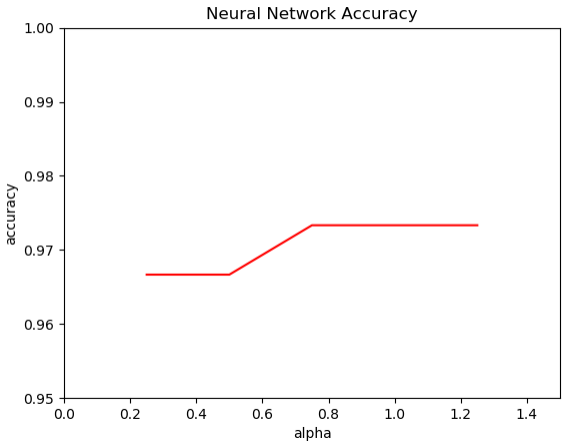

Figure 4: Neural Network

I received my best results for this model while implementing it with higher alpha values.

7 Code

My code was very much built on top of scikit-learn. All the models I created and implemented came from this library. I also used scikit-learn while performing evaluation and leveraged functions such as *cross_val_score*. I even pulled my data directly from scikit-learn with the *load_iris* function. I didn't have any real trouble working with my data because of this form of implementation. I then used matplotlib for all data visualizations and graphs. When looking at my code, you will see two files of importance: *classifers.py* and *visualize.py*. You will find the code for implementing the classification models in *classifers.py* and all the data visualization code in *visualize.py*. You can run both files simply using the python3 compiler. When running *classifiers.py*, you will see a print statement will all the accuracy scores of the models. On the other hand, *visualize.py* will save all important graphs and charts to disk when run.

8 Results

When finding the best model, I took logistic regression out of contention because of how poorly it performed.

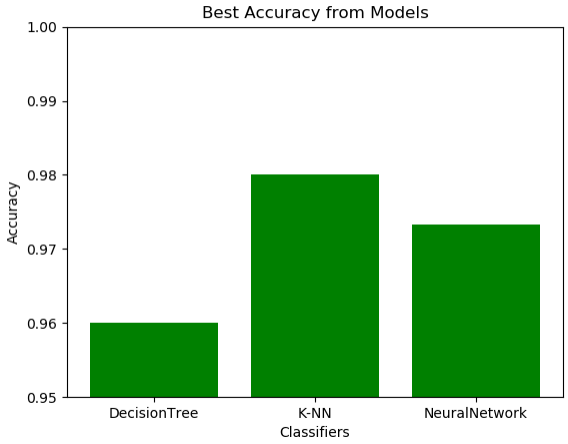

Figure 5: Neural Network

As we can see from the results, the best modeling strategy for this dataset it the KNN classifier. The best performance from KNN comes when you set your K value to 11. Some future exploration would include diving deeper into the hyper parameters of these models and see if there is any increase in accuracy. Specifically, I would like to look more into the hyper parameters of the neural network as there is definitely a lot of optimization that can be done. From my analysis, I was able to understand more about flower attributes and specifically how easy it is to differentiate between flowered through traits such as petal length and sepal length.