Iris Flower Classification Problem Using Various Different Classifier Methods

Michael Reilly

Professor Wang

CS-559-A

12/15/20

I pledge my honor that I have abided by the Stevens Honor System.

What is the problem to be solved?

The purpose of this project is to use various classification methods that have been learned throughout the semester and see which are most effective with this common dataset from the UCI ML repository. This code can be executed in the command line using:
python3 Reilly_FinalProject.py.

What is being classified and what are the classes?

The dataset that was used is the iris dataset, iris.data. In this data there are three different classifications possible for each of the 150 iris flowers in the dataset, iris-setosa, iris-versicolor, iris-virginica. The goal is using the classifiers to train the machine to correctly classify each of the 150 flowers in the dataset with their according iris classification. Ideally one of the classifiers may even perfectly be able to determine which type of iris any new data-type. The classifications are based on the parameters for deciding the classification which are sepal-length, sepal-width, petal-length, and petal-width as these are different for each different iris-type.

Train/test splits:

In the code there is a train test split where 75% of the data is used to train the classifier and the other 25% is used to test each classifier for accuracy. This split is done through the train_test_split function in python imported from the sklearn.model_selection python library. This creates the x_train, x_test, y_train, and y_test variables.

Results:

Classifier:	Mean:	Standard Deviation:
KNN	0.947	1.17x10^-16

Logistic Regression	0.921	2.34x10^-16
SVM	0.974	1.17x10^-16

Based on these results it can be determined that SVM is the most effective classifier on this dataset in this test case. If an individual tests again, due to how the train and test split it is possible for it to also change so that another classifier, be it KNN or Logistic Regression, may work better for this relatively small dataset. The KNN classifier may also work better if given a different number of nearest neighbors as with the current implementation it has n_neighbors=5. For example, when n_neighbors=1 the mean accuracy of KNN instead became 0.921, however when n_neighbors=7 the mean accuracy of KNN then became 0.974. So changing the number of neighbors in KNN to a different number than 5 may make it more accurate, such as in the case of n_neighbors=7, but also could make it less accurate, such as in the case of n_neighbors=1. This effect is however minimal due to most of the data being very clustered together. This can be seen by the figures below:

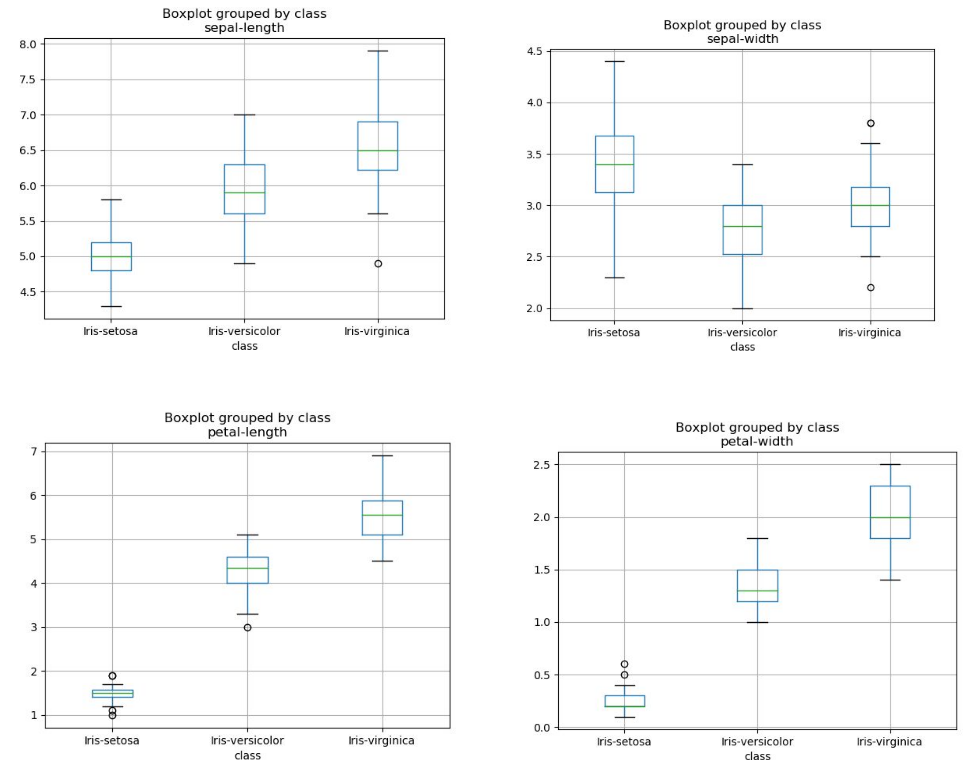

Figure 1: Boxplots of all parameters

As can be seen, the data is split into clusters, such as iris-setosa having much smaller petal-length and petal-width, iris-virginica having the largest petal-length and petal-width on average and iris-versicolor being in the middle for both variables. However, based on the classifier results after repeated tests, this does not appear to greatly affect any of the classifiers in this case, likely due to the dataset only having 150 items.

These models are all effective for various reasons.

The KNN classifier works effectively, as due to variables such as petal-length and petal-width having such extreme differences in values between the three sets it is very likely that the neighbors to those variables will be close, making their euclidean distance smaller than it will be to those from other iris types.

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

The Logistic Regression classifier works effectively for fundamentally similar reasons, except instead it uses a sigmoid function which also is able to calculate an estimate of the data point being in a certain class, which again when the data is able to separate the way this data is, it is extremely effective.

$$S(x) = \frac{1}{1 + e^{-x}}$$

The SVM classifier works effectively on this dataset as it is a small set with effective parameters for classification, making the SVM classifier extremely accurate. The hyperplane made with the SVM classifier is a good model of this data as a result of those previously mentioned factors.

Based on these deductions it appears that the classifiers are all generally good models for representing this small dataset. After running the code 10 times, and each classifier being run 100 times on the data on each run of the code there was no consistently better performing classifier out of the three. Only once did any of the classifiers fall below 90%, that being Logistic Regression, however that appeared to be an outlier.