Advanced Data Analysis Methods Final Paper

Zhan Gong

Machine Learning using Wine Data Set

Introduction

I used the same data set, the Wine Data Set, throughout the semester. This data set was obtained from the UCI Machine Learning Repository. I chose this dataset partly because I am interested in the wine culture, partly because it is a high-quality dataset. It contains 13 features and 177 samples, which are large enough to predict a satisfactory result. These data are the results of a chemical analysis of wines grown in Italy but derived from three different cultivars. The goal of this study is to determine the origin of wines using the quantities of 13 constituents found in each of the three types of wines.

Data Cleaning

The dataset from the UCI Machine Learning Repository was pretty clean. However, I still needed to change a little bit to let the data fit my module. I first added a name for each column in my data frame to understand what these variables are. I then took out the "class" column and made it my predict target. Before building a machine learning model, I could see what these features are: alcohol, malic acid, ash, and so on.

To begin with, I had to test if the dataset works. Thus, I used the "train test split" from scikit-learn to divide my data (75% training and 25% test). I have 133 samples in the training set and 45 in the testing set. From the scatterplot, I saw the same colored dots clustered together (Figure 1),

which meant that a machine learning model would be able to learn to separate them.

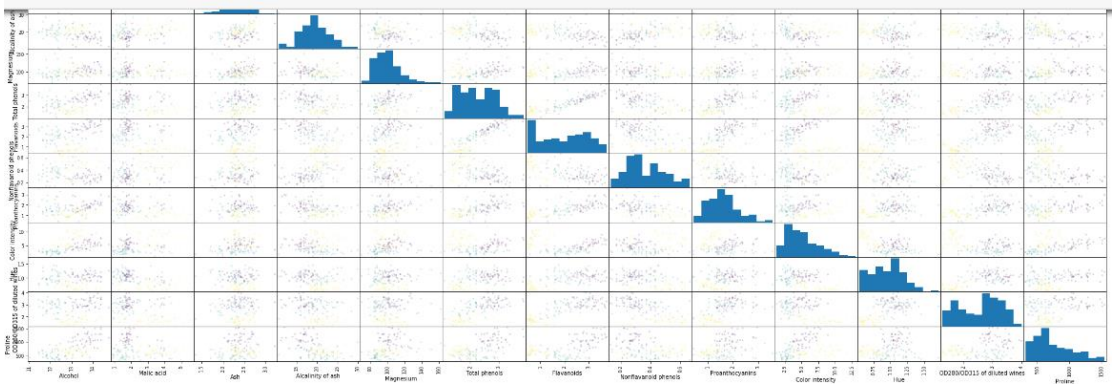

Figure 1.

Supervised Learning

I chose the K-nearest-neighbor and the Decision tree algorithms for my supervised learning project. From my understanding, the K-nearest neighbor algorithm functions this way: first, we store a lot of samples in the training set, just like storing a lot of dots in one chart. When we want to predict a new sample, the algorithm puts the new dot in the chart, finds the closest sample dot to this new dot, and assigns the label of the nearest dot to the new dot. For example, suppose we want to identify K closest neighbors. When we put the new dot in the chart, we find three closest sample dots, two of them are labeled 1 and one is labeled 2. The algorithm will choose to assign the new dot to the side which is more frequent: 1.

What I have done in my project is I first stored the samples by using the training set and predicted the new sample by using my test set and tried to figure out the accuracy, and I got an accuracy value of 0.73. Then I gave a range of how many neighbors I wanted, according to the plot (see Figure 2), when neighbors were equal to 4 or 7, I got a higher accuracy.

```
Out[28]: <matplotlib.legend.Legend at 0x1d0707b2c08>
```

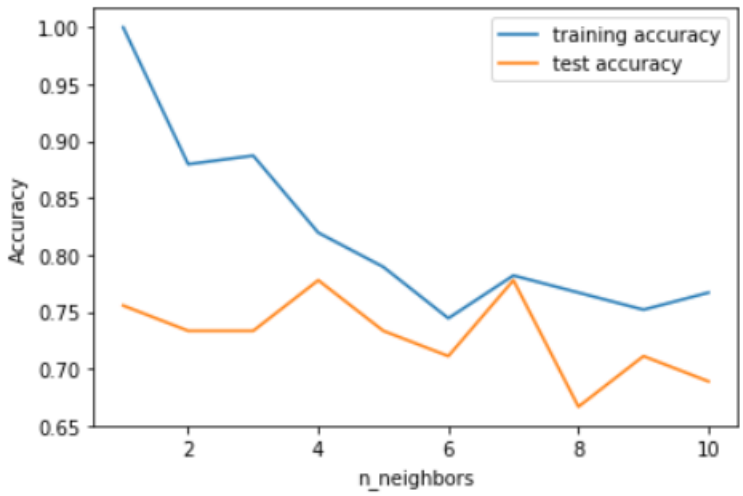

Figure 2.

The second algorithm I chose is the Decision tree. This algorithm is like a red-black tree in the data structure, trying to question the data "if" or "else." For example, if the value of alcohol is greater than 14, then the sample is class 1. If the value is less than 14, I would ask if it is greater than 13.5; if so, the sample is class 2; if not, the sample is class3. We can get better accuracy by asking more times. The number of questions we ask is called "depth." In my Decision tree, I set the depth to 4, and the accuracy on the test set was 0.933. When I set the depth to 2, the accuracy on the test set went down to 0.844. Unfortunately, I did not try to set a range of depth to test more options of depth, but I think it would be nice to see that plot.

After calculating the RMSE, I got my K-nearest-neighbors classification's RMSE is 0.77, and the Decision tree's RMSE is 0.26. In general, lower values of RMSE indicate better fit. Higher values of RMSE indicate that some of my classifications are pretty accurate, while some of the predictions are pretty inaccurate. As the Decision tree's RMSE value is lower than that of the K-nearest-neighbors, we can say that the Decision tree classification is more stable than K-nearest-

neighbors classification in my dataset.

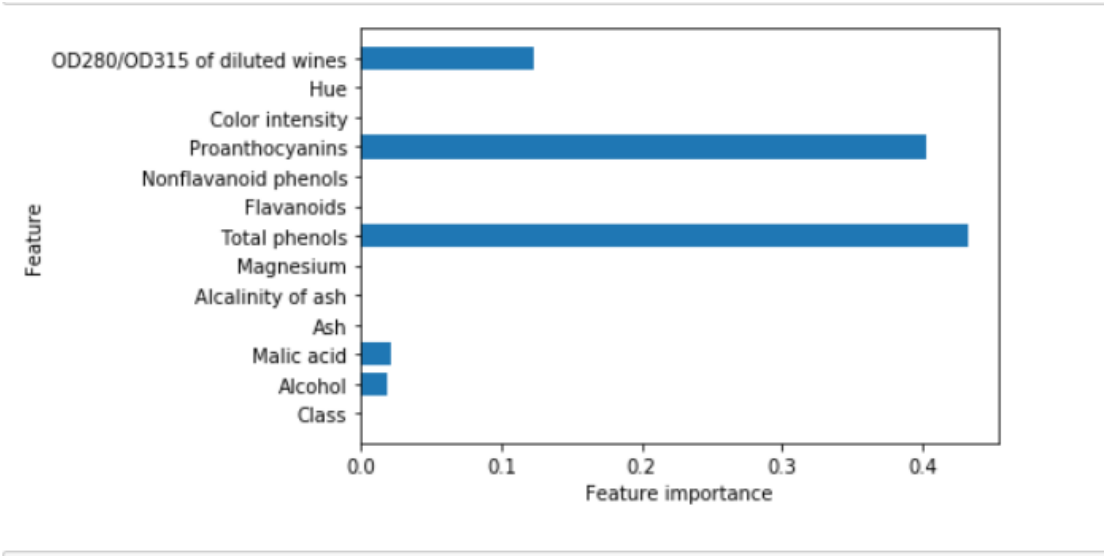

Figure 3.

Unsupervised Learning

I used the wine dataset because I've been using this data set for all my projects. Initially, I did not know it was such a popular dataset. From my understanding, PCA is an algorithm for deducting dimensions. The Wine data set has 13 features; what PCA is doing is trying to have all the features together. The algorithm attempts to find the orthogonal axis ordinarily. The first new axis uses the direction of maximum variance in the original dataset. The second axis gets the direction of the maximum variance in the orthogonal between the plane of the first axis and the plane of the second axis. We can get an N axis like this, but we keep the axis that contains most of the maximum variance. In a word, the goal of this algorithm is dimensionality reduction.

In K-means classification, regardless of n_cluster is equal to 1 or larger than 1, K-means with PCA always gives a better result than without PCA. Especially in the scatter plot, K-means without PCA is messier (see Figure 5).

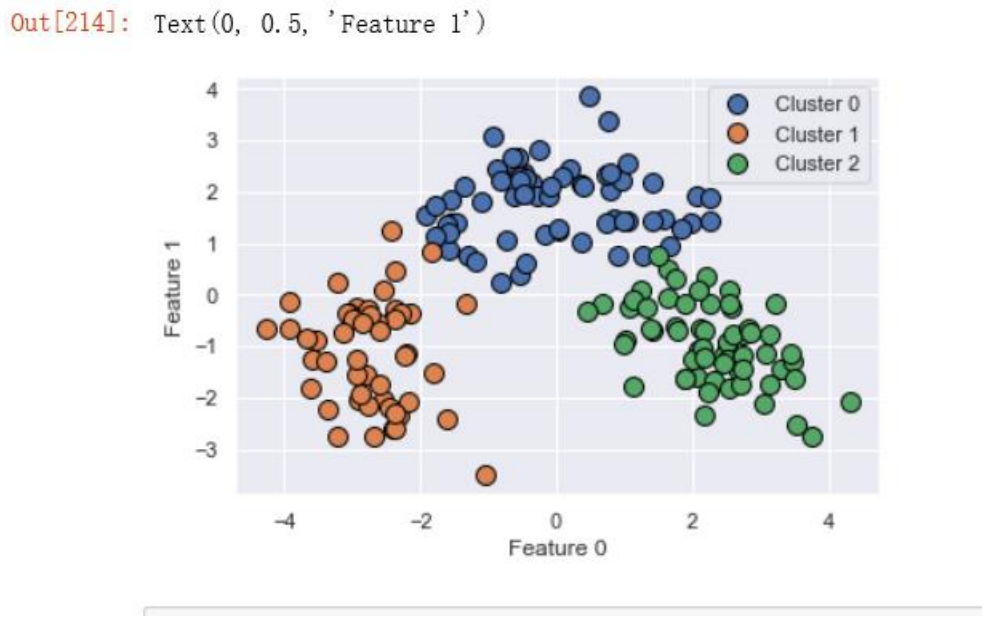

Figure 4. K-means with PCA.

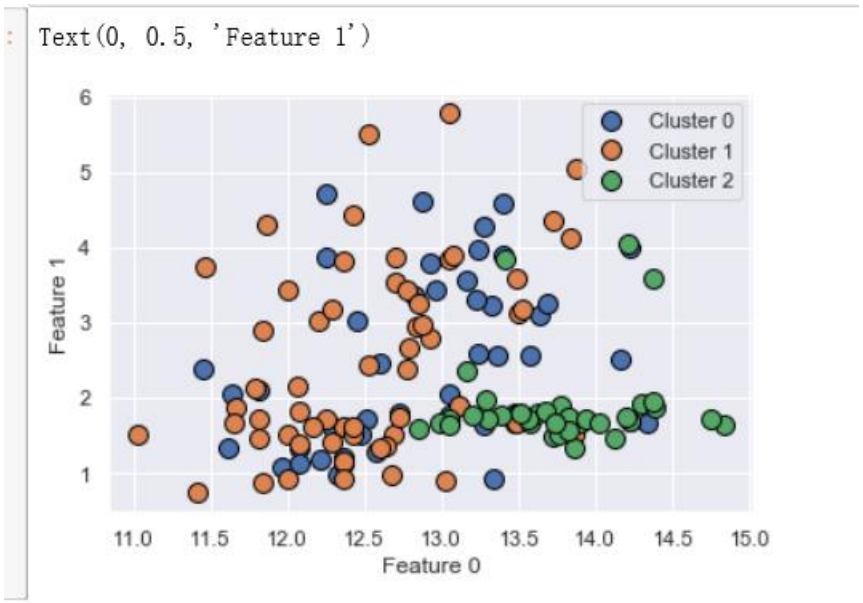

Figure 5. K-means without PCA.

Agglomerative clustering is clustering the other similar point together starting from each point itself. In other words, this is a "bottom-up" approach. I'm using ward in agglomerative clustering; this means the algorithm picks two clusters to merge and maintains the variance within all clusters to increase the least. Agglomerative clustering with PCA produces a more separate distribution

than without PCA (Figure 6), though agglomerative clustering without PCA is also highly classified (Figure 7).

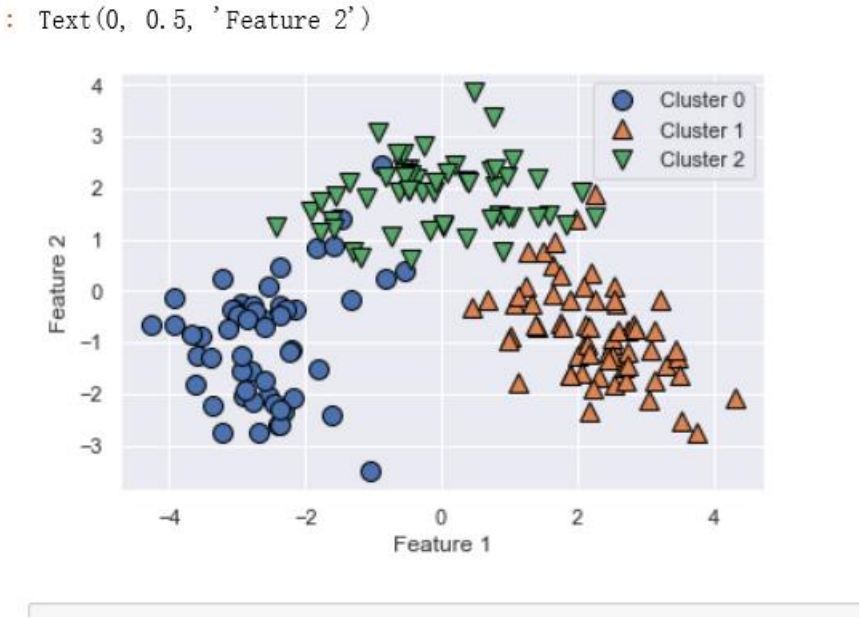

Figure 6. Agglomerative clustering with PCA

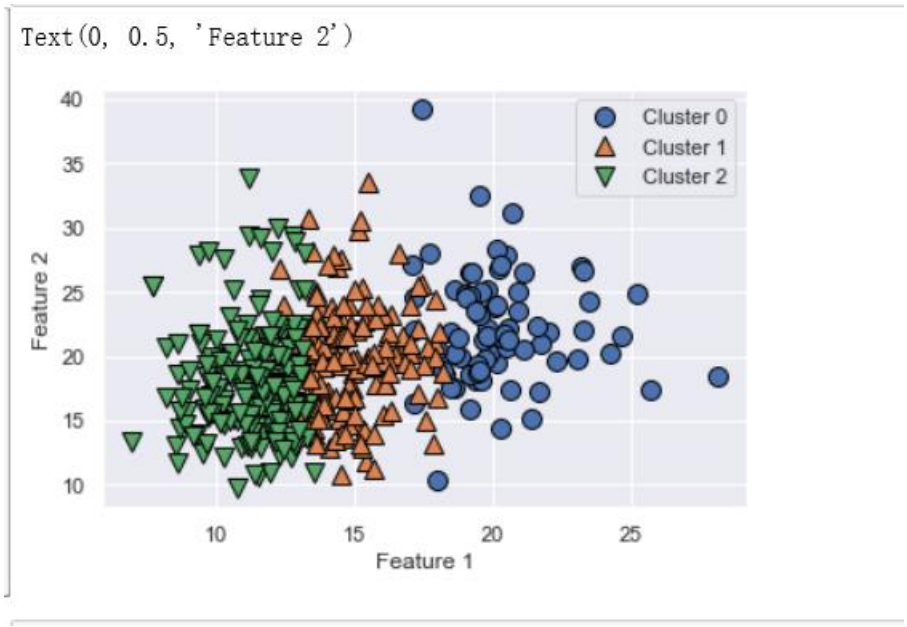

Figure 7. Agglomerative clustering without PCA

DBSCAN is similar to agglomerative clustering but needs to be set eps. It's like giving a range in agglomerative clustering. According to Figure 8 and 9, it's obvious to see the difference between

with PCA or without PCA.

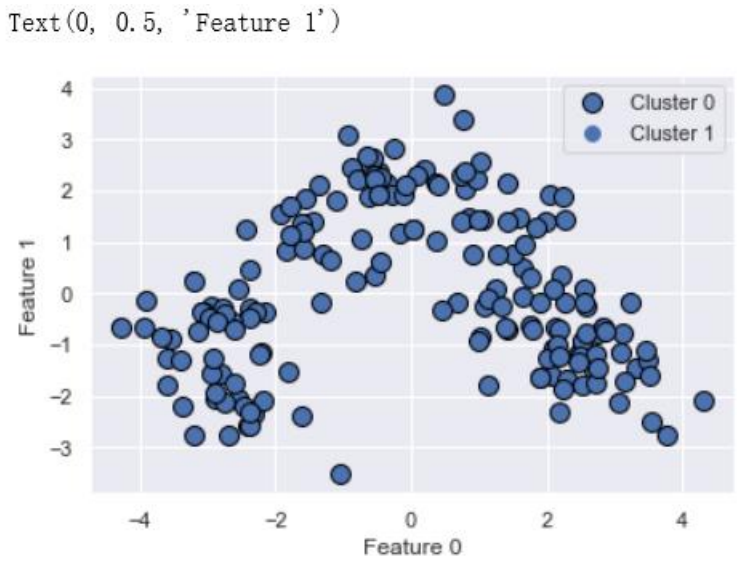

Figure 8. DBSCAN with PCA

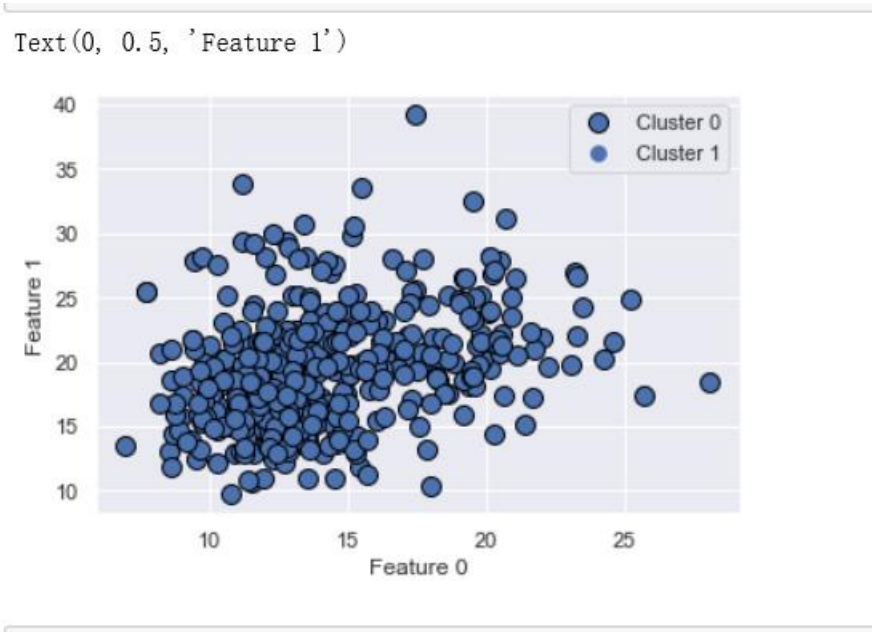

Figure 9. DBSCAN without PCA

In general, supervised learning and unsupervised learning are useful in different ways. While supervised learning is suitable for classification problems and regression problems, unsupervised learning is helpful when researchers are asking the algorithm questions they don't know the answer

to. In my case, it seems supervised learning works better since there are already three pre-labeled cultivars. In the future, I would love to learn about other deep learning models, such as semi-supervised learning or reinforcement learning.