# Report

## Logistic Regression

### Iris

I use the logistic regression model to train the dataset. I tested the errors with C from 0 to 100, and I divided the part from 0 to 1 and the part from 1 to 100 respectively into 20 parts. (The logistic() function is defined to train the dataset and return the training and testing error rates respectively into two lists.)

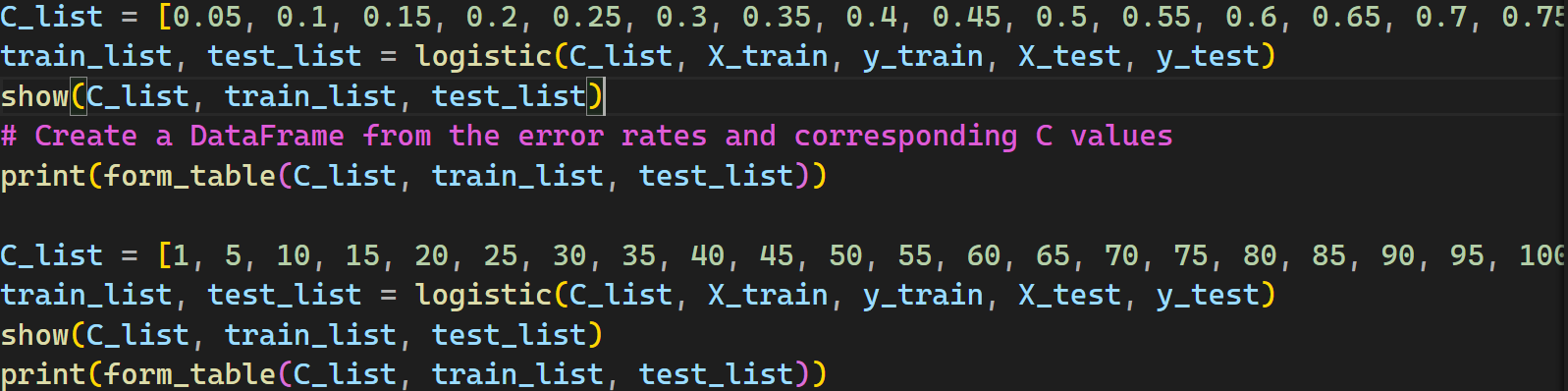


Figure 1 Code for Logistic Regression Model

The following pictures are what we got through the show() function and form\_table() function, which are used to plot the error rate of the training and testing dataset in the form of a line-chart and in the form of a table.

#### Underfitting

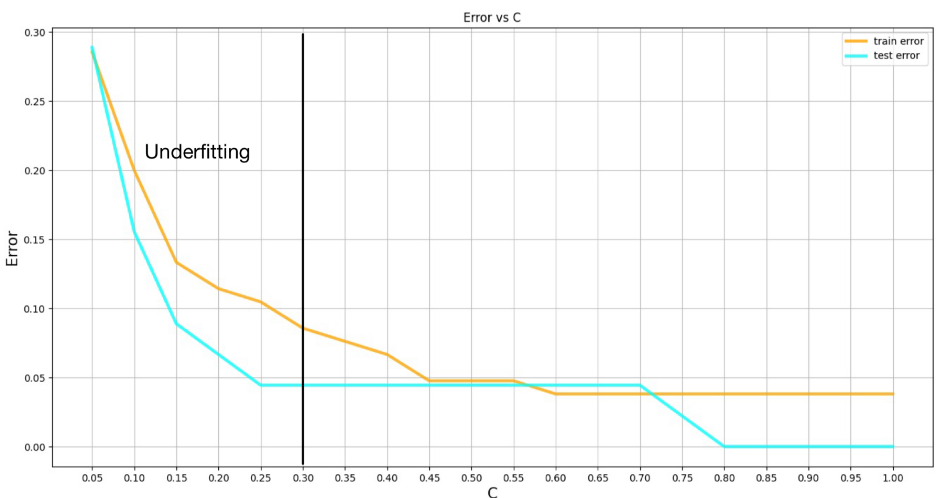


Figure 2 Error vs C Graph for 0 < C ≤ 1

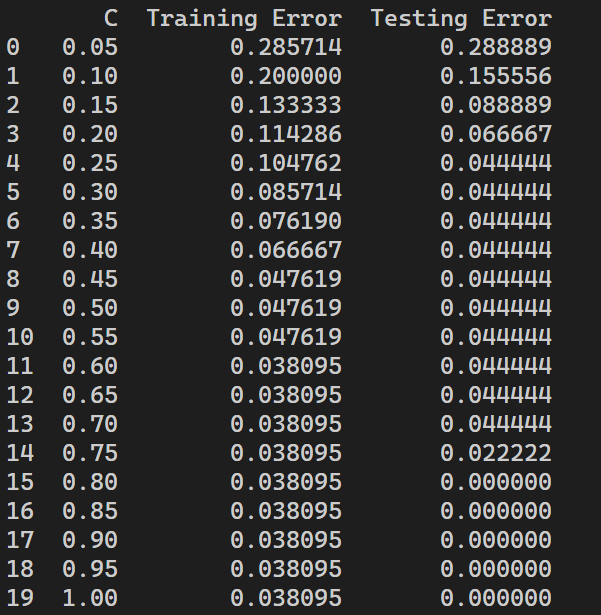


Figure 3 Training and Testing Error for Corresponding C (0 < C ≤ 1)

From the line chart and the form table, we can see that **when C is smaller than 0.3, both error rates will be relatively high, resulting in the underfitting phenomenon.**

The reasons are:

When using logistic regression (or some other machine learning model), the regularization parameter C controls the degree of constraint on the model complexity and is actually the inverse of the regularization strength. A smaller C value means a stronger regularization. Regularization is a technique used to prevent the overfitting of a model by imposing penalties on the parameters of the model, such as the size of the weights. When the C value is small, this penalty is severe, causing the model to be so simplified that it does not adequately capture the complexity and variability in the data - a phenomenon known as underfitting.

#### Overfitting

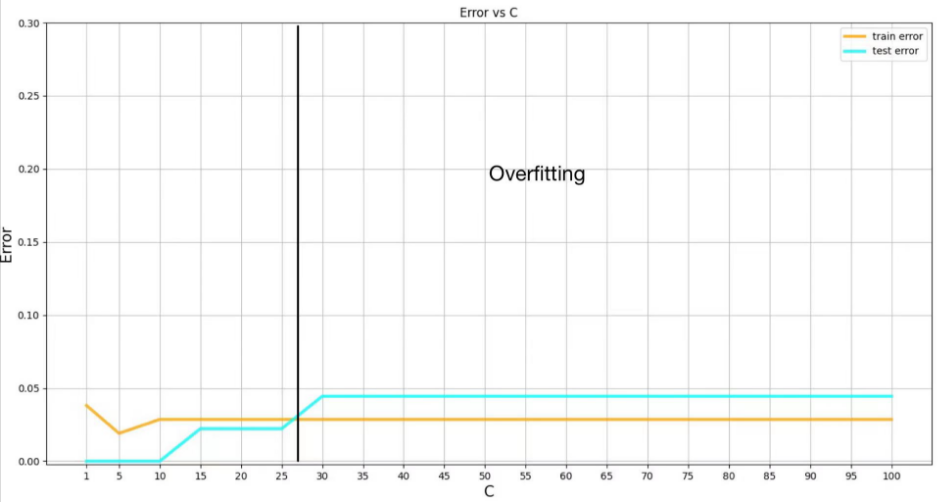


Figure 4 Error vs C Graph for 1 < C ≤ 100

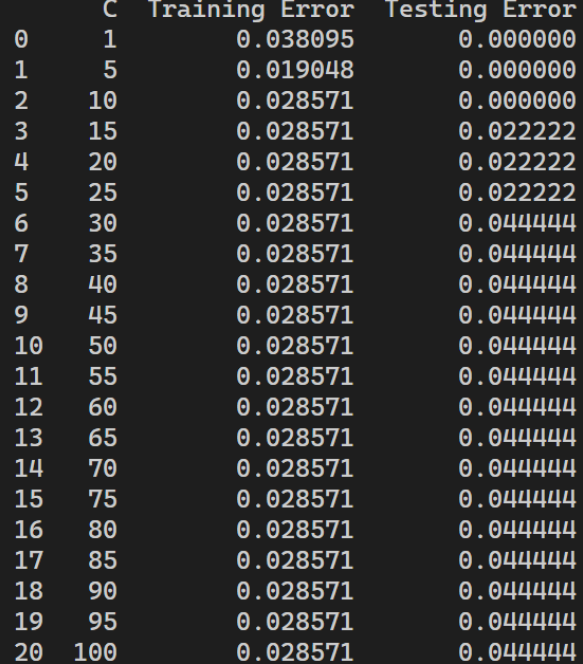
0

Figure 5 Training and Testing Error for Corresponding C (1 < C ≤ 100)

From the line chart and the form table, we can see that when C is **larger than 27 (about 27), the test error is obviously greater than the training error, resulting in the overfitting phenomenon.**

The reasons are:

When using logistic regression (or some other machine learning model), the regularization parameter C controls the model's tolerance for complexity and is actually the inverse of the regularization strength. A larger C value implies a weaker regularization. Regularization aims to prevent the model from overfitting - that is, the model is so complex that it learns not only the valid signals in the data, but also the noise. When the C value is large, the regularization penalty for the model is almost negligible, allowing the model to become too complex, which can lead to overfitting.

The following pictures are what I made to see how the C would cause to underfitting and overfitting and the normal situation.

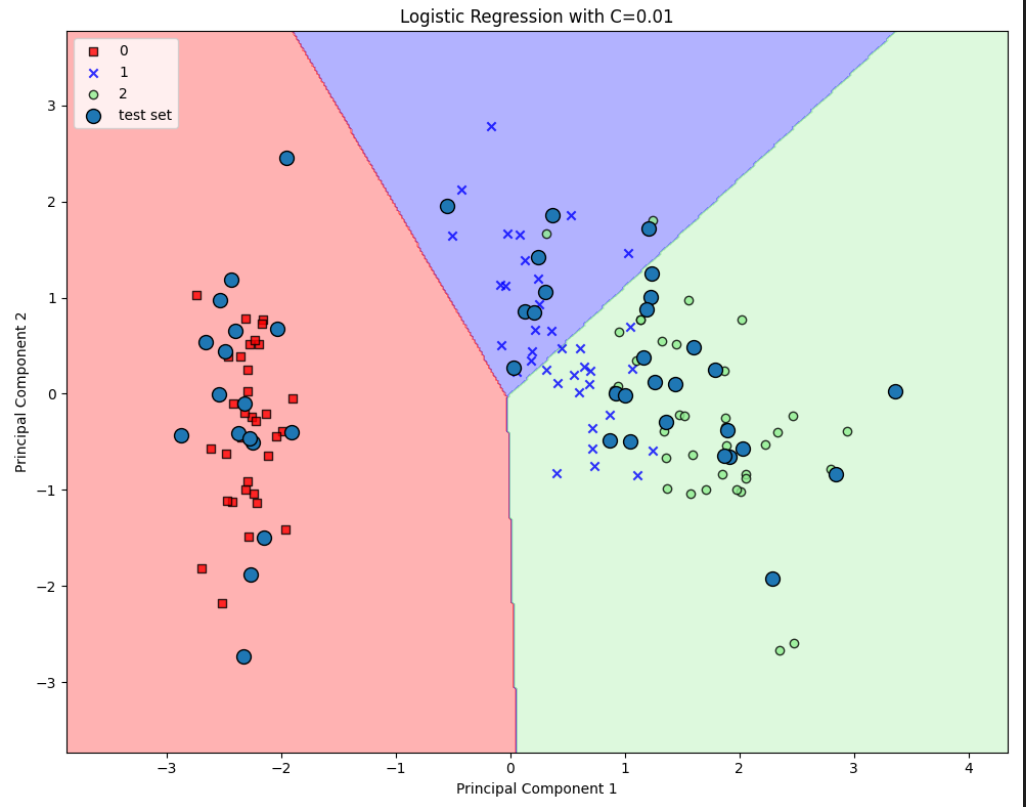


Figure 6 Underfitting for Logistic Regression Model Iris Dataset

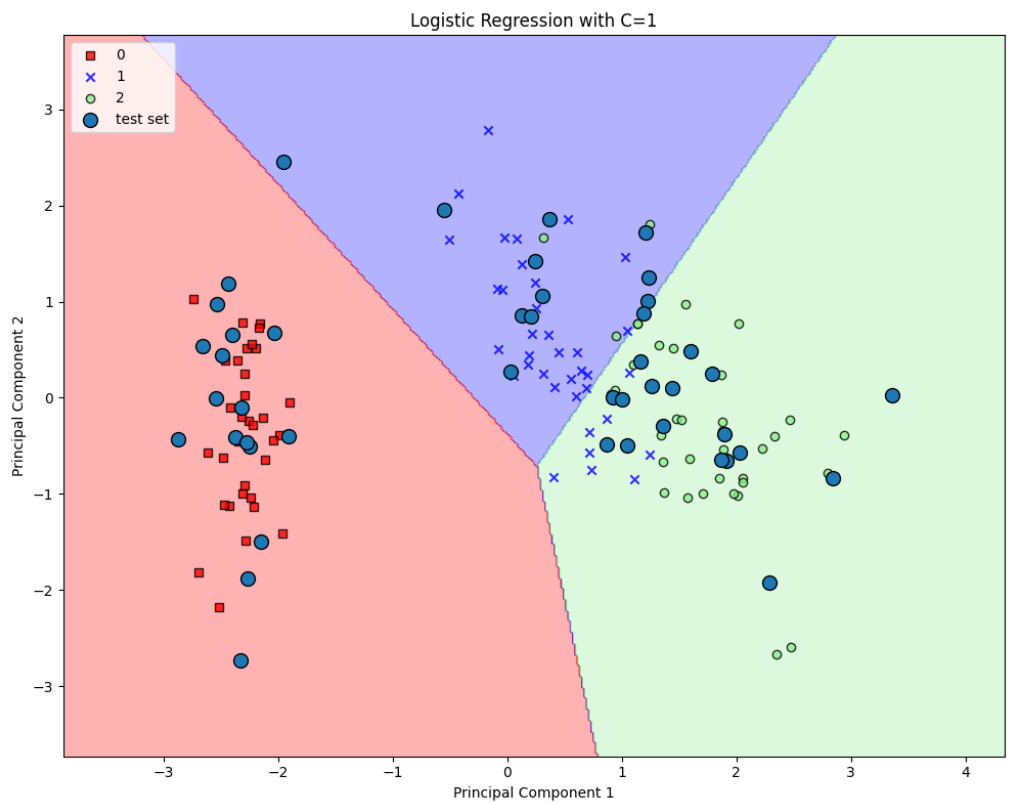


Figure 7 Appropriate C for Logistic Regression Model Iris Dataset

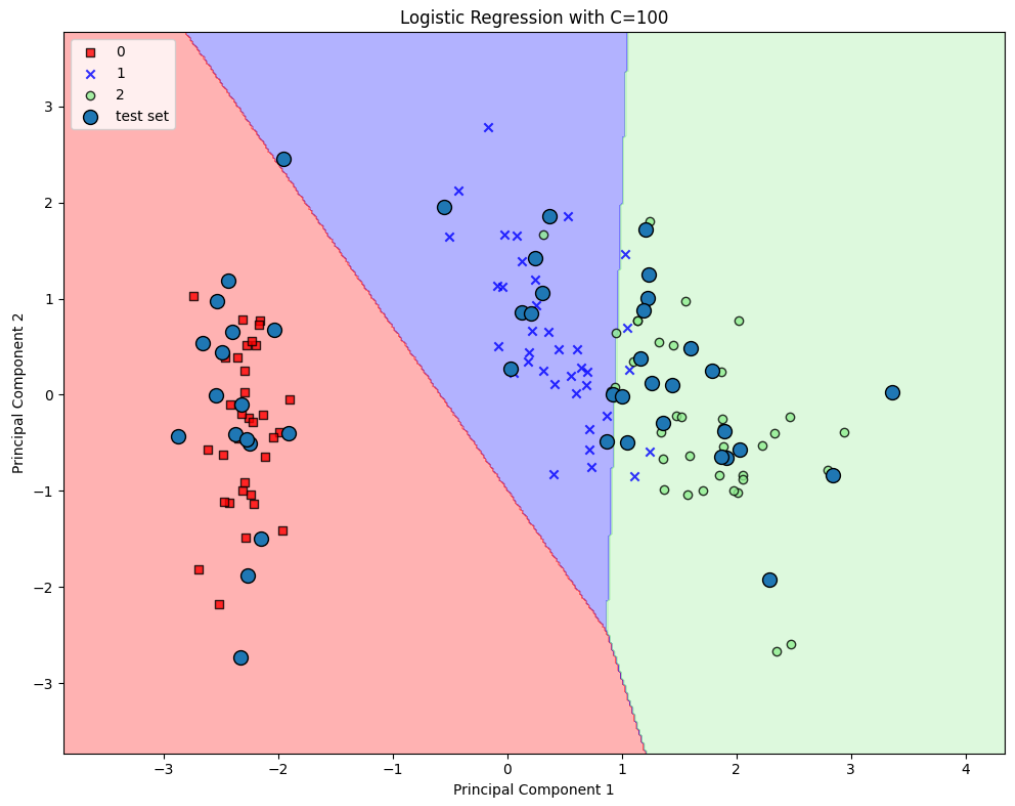


Figure 8 Overfitting for Logistic Regression Model Iris Dataset

### Wine

This time I added one more function predict\_instance() to output predicted and actual categories of the instances. Other functions remain the same as above.

手机屏幕截图

描述已自动生成

Figure 9 Code to Classify each Instance in the Dataset into 3 Types of Wines

文本

描述已自动生成

Figure 10 Code the Apply the predict\_instance() Function

I tested the errors with C from 0 to 100, and I divided the part from 0 to 0.1, the part of 0.1 to 1 and the part from 1 to 100 respectively into 20 parts.

The following pictures show what we got.

#### Underfitting

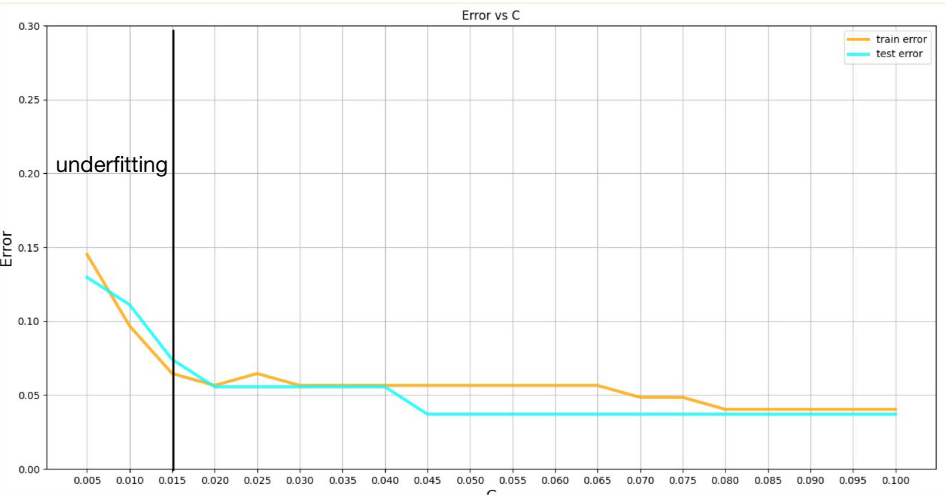


Figure 11 Error vs C Graph for 0 < C ≤ 0.1

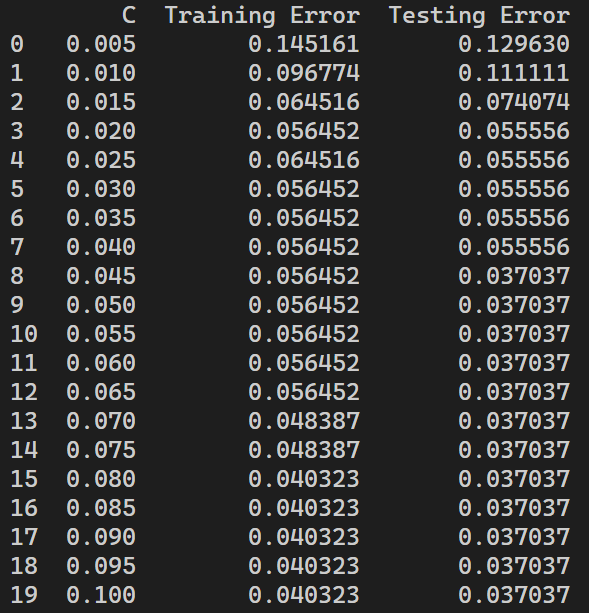


Figure 12 Training and Testing Error for Corresponding C (0 < C ≤ 0.1)

From the line chart and the form table, we can see that **when C is smaller than 0.015, the error rate will be relatively high, resulting in the underfitting phenomenon.**

The reasons are:

When using logistic regression (or some other machine learning model), the regularization parameter C controls the degree of constraint on the model complexity and is actually the inverse of the regularization strength. A smaller C value means a stronger regularization. Regularization is a technique used to prevent the overfitting of a model by imposing penalties on the parameters of the model, such as the size of the weights. When the C value is small, this penalty is severe, causing the model to be so simplified that it does not adequately capture the complexity and variability in the data - a phenomenon known as underfitting. In this question, when c is smaller than 0.015, the penalty is strong enough to make the error rate bigger.

#### Appropriate C

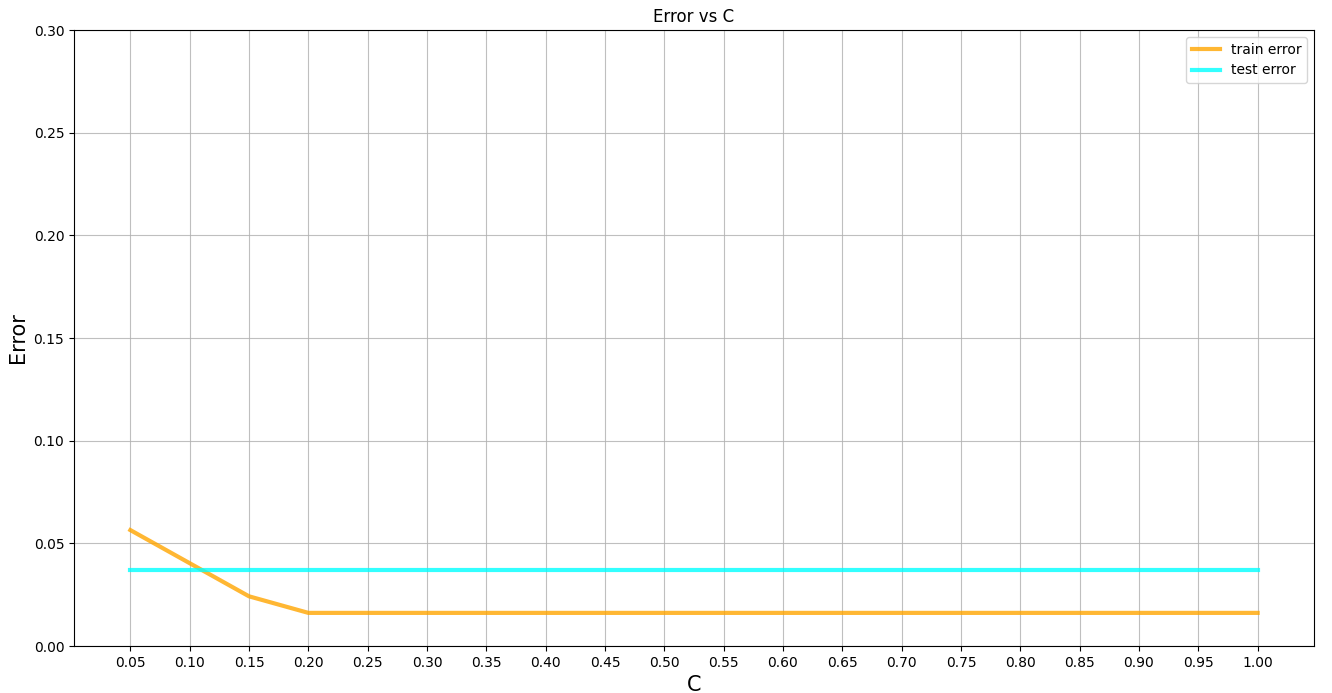


Figure 13 Error vs C Graph for 0.1 < C ≤ 1

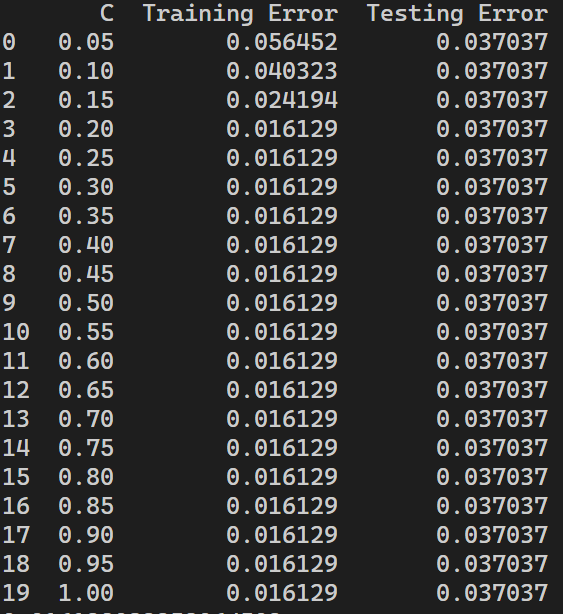


Figure 14 Training and Testing Error for Corresponding C (0.1 < C ≤ 1)

When C is between 0.1 and 1, the testing error is higher than the training error, but the high degree is not very large, within the acceptable range.

#### Overfitting

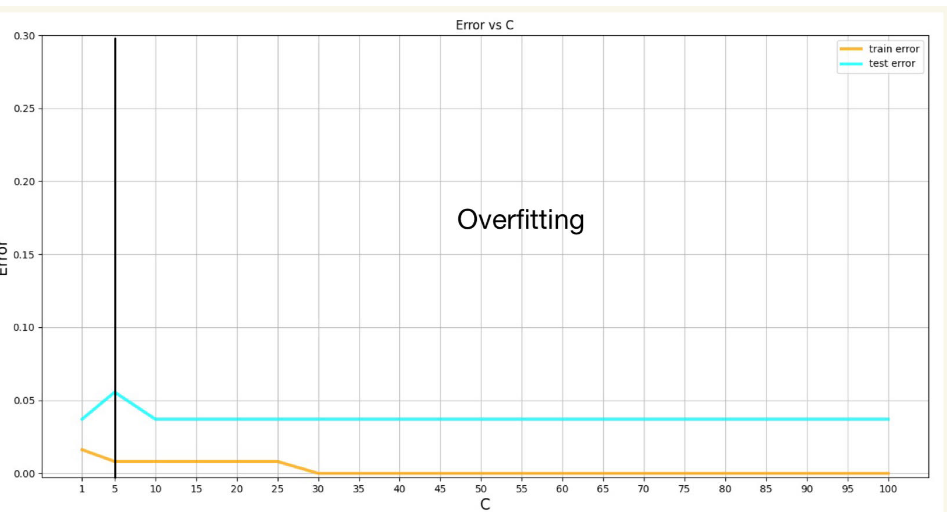


Figure 15 Error vs C Graph for 1 < C ≤ 100

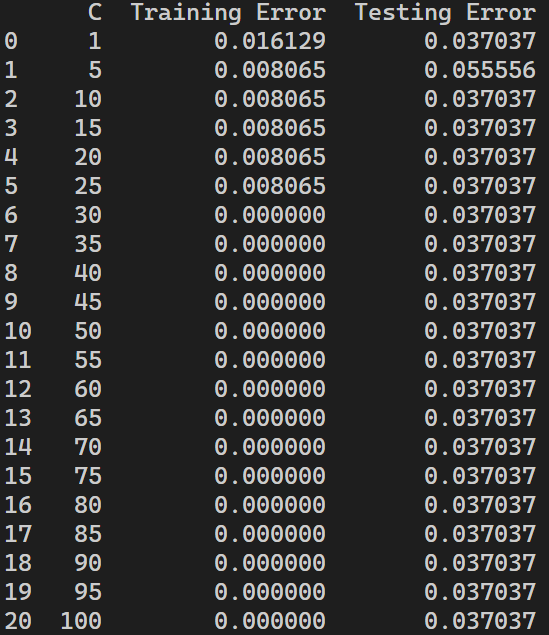


Figure 16 Training and Testing Error for Corresponding C (1 < C ≤ 100)

From the line chart and the form table, we can see that when C is **larger than 5, the test error is obviously greater than the training error, resulting in the overfitting phenomenon.** And when C is larger than 10, the training error even became 0 while the testing error still exists, which is an obvious overfitting phenomenon.

The reasons are:

When using logistic regression (or some other machine learning model), the regularization parameter C controls the model's tolerance for complexity and is actually the inverse of the regularization strength. A larger C value implies a weaker regularization. Regularization aims to prevent the model from overfitting - that is, the model is so complex that it learns not only the valid signals in the data, but also the noise. When the C value is large, the regularization penalty for the model is almost negligible, allowing the model to become too complex, which can lead to overfitting. In this situation, the C is not small enough so that the penalty strength is too small to make the model easier. Thus it got overfitting finally.

#### Instance Classification

Also, we use predict\_instance() to show the classification of the entire wine dataset. We chose C = 0.5, which results in neither overfitting nor underfitting, thus being appropriate to train the model. It has the following training and testing error rate.

图形用户界面, 文本

描述已自动生成

Figure 17 Training and Testing Error Rate with C = 0.5

Below is the screenshot of the classification.

日历

中度可信度描述已自动生成 文本

中度可信度描述已自动生成

文本

中度可信度描述已自动生成文本

描述已自动生成

电脑萤幕

低可信度描述已自动生成 文本

描述已自动生成

文本

描述已自动生成 文本

描述已自动生成

**文本

描述已自动生成文本

描述已自动生成**

Figure 18 Classification of each Instance in the Dataset into 3 Types of Wines

## Random Forest Classifier

### Iris

Now, I use the random forest classifier model to train the dataset. The show() function for the logistic regression model to plot the line-chart is slightly modified into two different functions, show\_depth() and show\_samples(), to draw the line-charts when respectively the parameters max\_depth (the maximum depth of a tree in the forest) and max\_samples (number of samples drawn from the dataset which forms the bootstrap sample to build a tree in the forest) are changed. The form\_table() function is also slightly modified into form\_table\_dep() and form\_table\_sam() to plot the table of the training and testing error rates. First, I tested the errors when max\_depth is changed to every integer from 1 to 20 and max\_samples remains unchanged. Then, I tested the errors when max\_depth remains unchanged. In the meantime, max\_samples is changed from 1 to 100 (every integer from 1 to 20 and every multiple of 5 from 20 to 100).

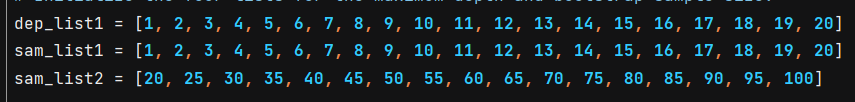


Figure 19 List for Changing max\_depth and max\_samples

文本

描述已自动生成

Figure 20 Code for Random Forest Classifier Iris Dataset

#### Changing max\_depth

The following pictures are what I got when executing the functions show\_depth() and form\_table\_dep(), which are used to plot the error rate of the training and testing dataset in the form of a line-chart and in the form of a table when max\_depth is changed and max\_samples remains unchanged.

##### Underfitting and Overfitting

图表, 折线图

描述已自动生成

Figure 21 Error vs Maximum Depth Graph 1 ≤ (Maximum Depth) ≤ 20

电脑屏幕截图

中度可信度描述已自动生成

Figure 22 Training and Testing Error for Corresponding 1 ≤ Maximum Depth ≤ 20

From the line chart and the table above, we can conclude that **when maximum depth of a tree in the forest is 1, the error rate of both training and testing will be relatively high, resulting in the underfitting phenomenon.** We can also see that **when maximum depth is larger than 4, the test error is evidently greater than the training error, which is the overfitting phenomenon.**

The reasons are:

When we use the random forest classifier to train models, if the trees are too shallow, trees have limited capacity to capture complex patterns in the data. They may fail to capture important interactions, dependencies and nuances among features, leading to a limited ability to model the training data. This leads to underfitting.

On the other hand, if individual trees are very deep, they have the ability to create complex decision boundaries by splitting the feature space into finer partitions. This helps to capture complex features in the training data and reduce the error rate in the training data. However, the trees capture all available information in the training data, which makes them sensitive to noise and outliers and they may not generalize well to unseen data, resulting in higher error rate in the testing data. Thus, they adapt to the randomness rather than learning the underlying patterns and this leads to overfitting.

Therefore, we have to find an appropriate maximum depth in between for the trees to train a good model.

#### Changing max\_samples

The following pictures are what I got when executing the functions show\_samples() and form\_table\_sam(), which are used to plot the error rate of the training and testing dataset in the form of a line-chart and in the form of a table when max\_samples is changed and max\_depth remains unchanged.

##### Underfitting

图表, 条形图, 直方图

描述已自动生成

Figure 23 Error vs Bootstrap Sample Graph 1 ≤ (Bootstrap Sample Size) ≤ 20

图片包含 文本

描述已自动生成

Figure 24 Training and Testing Error for Corresponding 1 ≤ Bootstrap Sample Size ≤ 20

From the line chart and the table above, we can conclude that **when the size of a bootstrap sample of a tree in the forest is less than 4, the error rate of both training and testing will be relatively high, resulting in the underfitting phenomenon.**

The reasons are:

When the size of the bootstrap samples is very small, it increases the diversity among individual trees of the forest, since the probability of a particular training sample being the bootstrap sample is very low. Thus, the randomness of individual trees is very high and many trees predict a large number of different results. The correct result cannot be found by majority vote and the error rate is very high, leading to underfitting.

Apart from underfitting, we can also see that a small bootstrap sample results in a very high error rate, i.e. a low overall performance, but there is little gap between the training and testing performance.

##### Overfitting

图表, 折线图

描述已自动生成

Figure 25 Error vs Bootstrap Sample Graph 20 ≤ (Bootstrap Sample Size) ≤ 100

图片包含 表格

描述已自动生成

Figure 26 Training and Testing Error for 20 ≤ Corresponding Bootstrap Sample Size ≤ 100

From the line chart and the table above, we can conclude that **when maximum depth is larger than 55, the test error is evidently greater than the training error, which is the overfitting phenomenon.**

The reasons are:

If the bootstrap sample size is very large, it is very likely that many decision trees in the forest contain the same samples of the training data. The decision trees are very similar to each other and learn to fit the training data very closely. However, the model is sensitive to noise and outliers in the training data and they may not generalize well to unseen data, resulting in higher error rate in the testing data. This results in overfitting.

### Breast Cancer

The functions are very much the same as when dealing with the Iris dataset. I still apply the functions, show\_depth(), show\_samples(), form\_table\_dep() and form\_table\_sam() to plot the line chart and the table. The list for changing max\_depth and max\_samples are the same as above.

文本

描述已自动生成

Figure 27 Code for Random Forest Classifier Breast Cancer Dataset

#### Changing max\_depth

The following pictures are what I got when executing the functions show\_depth() and form\_table\_dep().

##### Underfitting and Overfitting

图表, 折线图

描述已自动生成

Figure 28 Error vs Maximum Depth Graph 1 ≤ (Maximum Depth) ≤ 20

图片包含 文本

描述已自动生成

Figure 29 Training and Testing Error for Corresponding 1 ≤ Maximum Depth ≤ 20

From the line chart and the table above, we can conclude that **when maximum depth of a tree in the forest is 1, the error rate of both training and testing will be relatively high, resulting in the underfitting phenomenon.** We can also see that **when maximum depth is larger than 4, the test error is obviously greater than the training error, which is the overfitting phenomenon.**

The reasons are:

When we use the random forest classifier to train models, if the trees are too shallow, trees have limited capacity to capture complex patterns in the data. They may fail to capture important dependencies and nuances among features, leading to a limited ability to model the training data. As a result, the model is underfitted.

On the other hand, if individual trees are very deep, they capture complex features in the training data and reduce the error rate in the training data. However, the trees capture all available information in the training data, which makes them sensitive to noise and outliers, resulting in higher error rate in the testing data. Thus, they adapt to the randomness rather than learning the underlying patterns and this leads to overfitting.

Therefore, we have to find an appropriate maximum depth in between for the trees to train a good model.

#### Changing max\_samples

The following pictures are what I got when executing the functions show\_samples() and form\_table\_sam().

##### Underfitting

图表, 折线图

描述已自动生成

Figure 30 Error vs Bootstrap Sample Graph 1 ≤ (Bootstrap Sample Size) ≤ 20

图片包含 日历

描述已自动生成

Figure 31 Training and Testing Error for Corresponding 1 ≤ Bootstrap Sample Size ≤ 20

From the line chart and the table above, we can conclude that **when the size of a bootstrap sample of a tree in the forest is less than 4, the error rate of both training and testing will be relatively high, resulting in the underfitting phenomenon.**

The reasons are:

When the size of the bootstrap samples is very small, it increases the diversity among individual trees of the forest, since the probability of a particular training sample being the bootstrap sample is very low. Thus, the randomness of individual trees is very high and many trees predict a large number of different results. The correct result cannot be found by majority vote and the error rate is very high, leading to underfitting.

Apart from underfitting, we can also see that a small bootstrap sample results in a very high error rate, i.e. a low overall performance, but there is little gap between the training and testing performance.

##### Overfitting

图表, 折线图

描述已自动生成

Figure 32 Error vs Bootstrap Sample Graph 20 ≤ (Bootstrap Sample Size) ≤ 100

图片包含 文本

描述已自动生成

Figure 33 Training and Testing Error for 20 ≤ Corresponding Bootstrap Sample Size ≤ 100

From the line chart and the table above, we can conclude that **when maximum depth is larger than 55, the test error is obviously greater than the training error, which is the overfitting phenomenon.**

The reasons are:

If the bootstrap sample size is very large, it is very likely that many decision trees in the forest contain the same samples of the training data. The decision trees are very similar to each other and learn to fit the training data very closely. However, the model is sensitive to noise and outliers in the training data and they may not generalize well to unseen data, resulting in higher error rate in the testing data. This results in overfitting.

## Conclusion

When applying different methods of machine learning, a significant task to do is to choose the appropriate parameters for the training of models. Choosing the inappropriate parameters could easily result in overfitting and underfitting, which results in the failure of training the model.

When the model is underfitted, it has limited capacity to capture complex patterns in the data. On the other hand, overfitting leads to complex decision boundaries, which makes the model sensitive to noise and outliers and they may not generalize well to unseen data. Therefore, choosing the right parameters is a must to train the best-performing model.