Logistic Regression Classifier

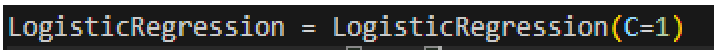
Logistic Regression (LR) is a statistical algorithm that is widely used in supervised learning and for binary classification tasks, where the goal is to predict the probability that an input belongs to one of two possible classes.

In logistic regression, the algorithm models the relationship between the input features and the probability of belonging to a specific class using the logistic function (also known as the sigmoid function). The logistic function maps any input value to a value between 0 and 1, which can be interpreted as a probability.

Parameter C

In LR, the parameter C represents the inverse of regularization strength where regularization is a technique used to prevent overfitting in the model. A smaller value of C increases the regularization strength, making the model simpler and potentially reducing the chances of overfitting. A larger value of C reduces the strength of regularization, allowing the model to fit the training data more closely which might lead to potentially overfitting.

C must be a float number that is bigger than 0. The default value is 1.0 (Figure 5-5-1-1), which means that the ratio of the regularization to the loss function is 1 to 1. In this experiment, students are asked to change the value of C and check how the decision boundary changes.



```
LogisticRegression = LogisticRegression(C=1)
```

Figure 5-5-1-1: C in LR

Sklearn

Scikit-learn, often abbreviated as sklearn, is an open-source machine-learning library for the Python programming language. This library is chosen in this paper due to a variety of algorithms that can be easily implemented for different applications and various datasets that can be applied in machine learning, data science, and other AI areas. Scikit-learn supports a consistent API for different algorithms, making it easier to conduct experiments with different models and techniques.

Iris Dataset

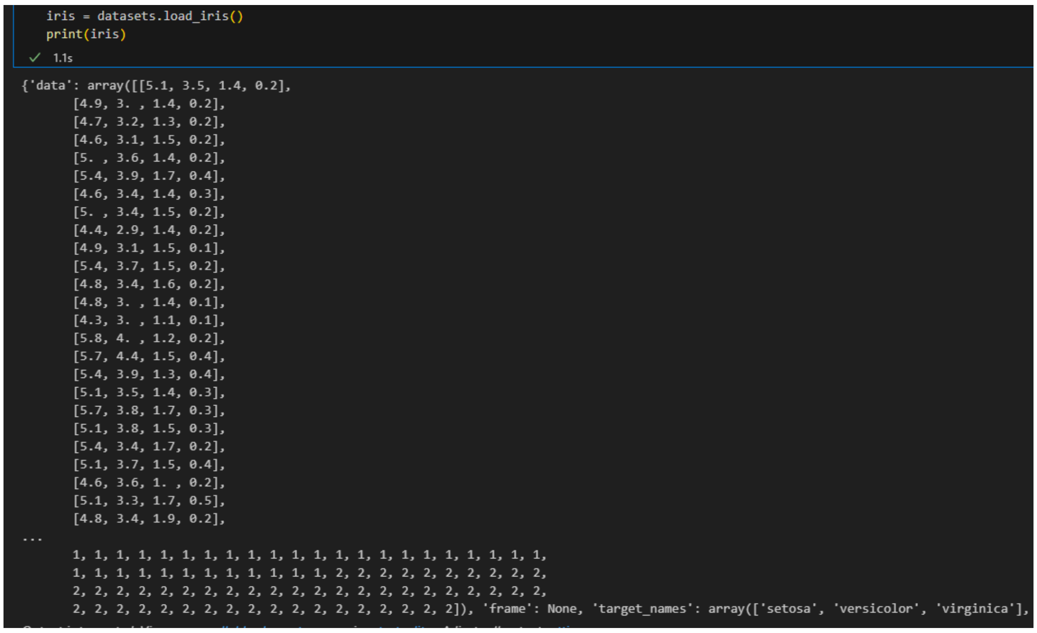
The iris flower has various varieties, and flowers of different varieties have their distinct characteristics. These can be briefly summarized by observing the following four aspects: sepal length, sepal width, petal length, and petal width. We hope that the computer can identify patterns from known sample data so that it can recognize the variety of the flower based on these characteristics.

In Figure 5-5-2-1 we can see that there are 'data', 'target', 'target_names', and 'feature_names', as well as some descriptive information.

'data' is a two-dimensional array with 150 rows and 4 columns. Each row represents a data sample, totaling 150 samples, and the four columns correspond to the four feature names in 'feature_names': 'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'.

'target' is a one-dimensional array with 150 elements, corresponding sequentially to the 150 samples in 'data'. The values 0, 1, and 2 represent the three classes in 'target_names': 'setosa', 'versicolor', and 'virginica', which are the three varieties of iris flowers.

In this experiment, the logistic regression classifier is supposed to be trained with the first two columns of data (sepal length, sepal width) and all the rows from the target. After training, it will be used to classify and determine which target (0,1,2) each row belongs to.



```
iris = datasets.load_iris()
print(iris)
```
✓ 1.1s

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
...
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicolor', 'virginica'],
```

Figure 5-5-2-1: Iris dataset

Decision Boundary Display

A decision boundary is a conceptual boundary that separates different classes or categories in a dataset. It represents the region where the algorithm makes a decision about which class a particular data point belongs to.

In the context of this logistic regression experiment, adjusting the parameter C influences the position and shape of the decision boundary. A larger value of C might result in a more flexible decision boundary that closely fits the training data, potentially leading to overfitting, while a smaller value might result in a simpler decision boundary that is less influenced by individual data points, potentially leading to underfitting. The goal is to find the right balance that generalizes well to unseen data.

To allow users to observe the changes in the decision boundary more intuitively, this project will use the DecisionBoundaryDisplay function of sklearn. inspection to visualize the model results as shown in the following figure.

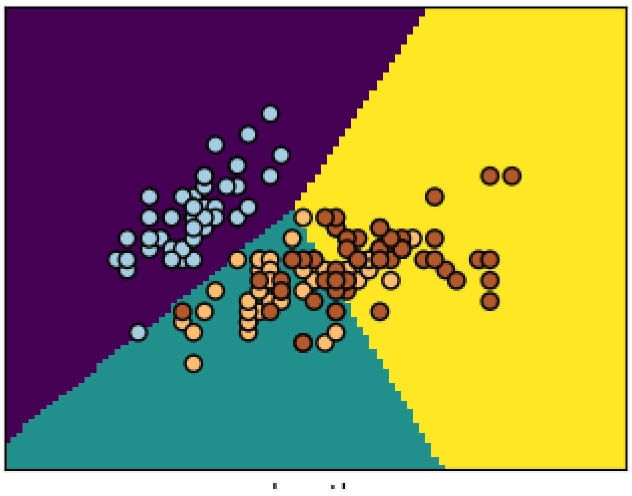


Figure 5-5-2-2: Sample of DecisionBoundaryDisplay

Experiment Procedure

To help developers get a clear and better understanding of how this experiment works, this section presents a workflow chart in Figure 5-5-3 to illustrate the structure of the code file and how the experiment works.

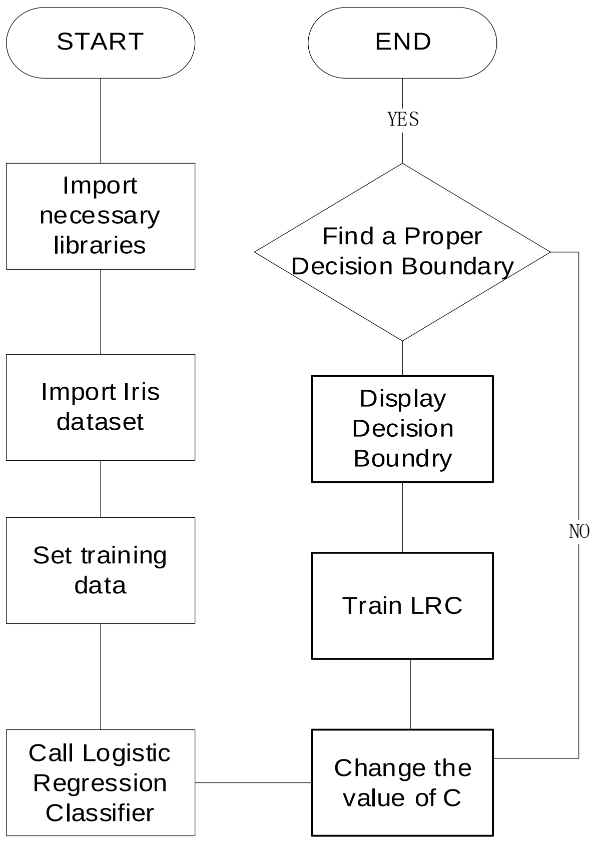


Figure 5-5-3: Experiment Work Flow Chart