PHELMA - GRENOBLE INP

MACHINE LEARNING

5PMBMLD0

Logistic Regression - Sequence 3

Students :
Allan DIZET
Matteo MARENGO

03/10/2022

1 Introduction

This sequence is focused on logistic regression . During the lectures, we studied the cost function and the decision boundaries of the logistic regression. We have seen example for the multiclass classification and regularized classification.
The first exercise was dealing with logistic regression (or binary classification). The second exercise is about multi-class classification and the third one will deal with Non-Linear data Classification.

2 Exercise 2 - Multiclass Model Performances

In this exercise, the target was to classify the 10 class digit dataset of the scikit-learn.datasets module using a logistic regression.

2.1 Dataset analysis

We can look at the documentation of the dataset with the .DESCR method. By looking at the documentation related to the digit dataset, there are 1797 samples, it has 64 numerical features (8x8 pixels) and a 10 class target variable (0-9).

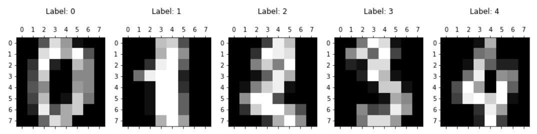


FIGURE 1 – Display of the 5 first digits

After a small script we can calculate how many samples do we have per class ; 178 zeros, 182 one, 177 two, 183 three, 181 four, 182 five, 181 six, 179 seven, 174 eight, 180 nine.

```python
for i in range(0,10):
    tot = 0
    for j in range(len(y)):
        if y[j]==i:
            tot=tot+1
    print(tot)
```

code1.py

2.2 Logistic Regression Model

We split the data into a training set of 80 % and a test set of 20 %. We use the "one versus all" strategy. We do the prediction of the class of the 200 th sample. In our case, we obtained 5 when the good results should have been 1.

```python
# Multiclass Logistic regression: one versus all (parameter
    multi_class= OVR)

# Data splitting

XTrain, XTest, yTrain, yTest = train_test_split(X, y, test_
    size = 0.2, random_state = 0)

# Training the model

clf = LogisticRegression(C=1,max_iter=2900,multi_class='ovr')
clf.fit(XTrain,yTrain)

ypredtrain = clf.predict(XTrain)
ypredtest = clf.predict(XTest)

# Testing the model on the 200th image


print("Prediction of the 200th sample:\n",ypredtest[200])
```

code2.py

2.3 Performances analysis

First, we report the training and testing accuracies. We have a training accuracy of 0,998 and a testing accuracy of 0,947. We then compute the confusion matrix. For class 8, we must look up at the 7th label :

$$TP = 37$$

$$FN = 2$$

$$FP = 0$$

$$TN = 27+31+1+3+34+2+28+1+30+39+1+1+43+3+1+34+1+1+1+1+38 = 321$$

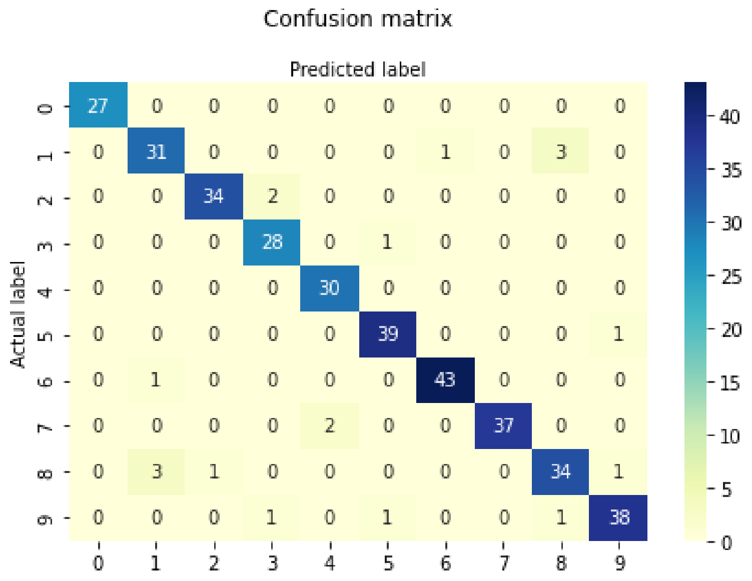
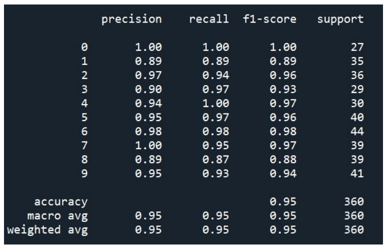


FIGURE 2 – Heatmap of the confusion matrix

Then, we compute the classification report. We have access to three common metrics ; precision is the percentage of correct positive predictions relative to total positive predictions. Recall is the percentage of correct positive predictions relative to total actual positives. F1 score is a weighted harmonic mean of precision and recall.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        27
           1       0.89      0.89      0.89        35
           2       0.97      0.94      0.96        36
           3       0.90      0.97      0.93        29
           4       0.94      1.00      0.97        30
           5       0.95      0.97      0.96        40
           6       0.98      0.98      0.98        44
           7       1.00      0.95      0.97        39
           8       0.89      0.87      0.88        39
           9       0.95      0.93      0.94        41

    accuracy                           0.95       360
   macro avg       0.95      0.95      0.95       360
weighted avg       0.95      0.95      0.95       360
```

FIGURE 3 – Classification report

The class that is the worst predicted is class 1.

3 Exercise 3

In this exercise, we will deal with data that is not linearly separable, which simply means that we cannot separate classes with a line.

3.1 Logistic regression

3.1.1

On the whole dataset, we obtain an accuracy of 0.5. It is really low, so we definitely have to find a new model to do the regression.

3.1.2

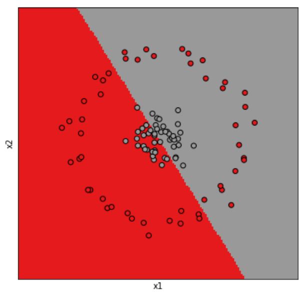


FIGURE 4 – Decision boundary with Logistic Regression

The decision boundary is not good at all, it splits the dataset in two but the two classes are not well separated at all. We have to find a new solution.

3.2 Adding new features

3.2.1

Now, that we have included a new variable, the two classes are linearly separable.

3.2.2

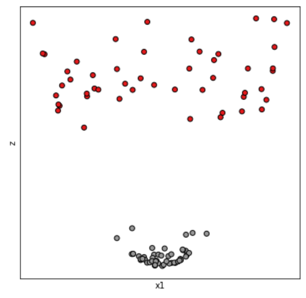


FIGURE 5 – The data with a new variable

```python
clf = LogisticRegression()
clf.fit(X_new,y)
ypred=clf.predict(X_new)
print(accuracy_score(y,ypred))
cnf_matrix=confusion_matrix(y,ypred)
```

code3.py

After fitting the logistic regression on the new dataset we have an accuracy of 1. The confusion matrix is perfect.

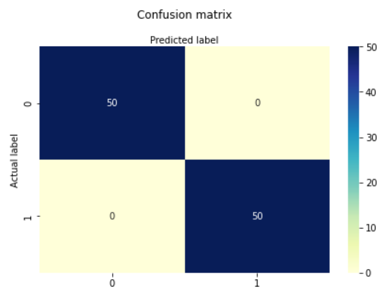


FIGURE 6 – Heatmap of the confusion matrix

4 Conclusion

The logistic regression is a powerful tool for classification and it is easy to implement. However, we have to be aware that in some cases it needs some adaptation to be efficient such as with non-linear data.