# EE271 Midterm Project

## Solution to the binary classification problem using the logistic regression method and kNN method

1st Hua Yuxiao

*dept. Department of Electronic and Electrical Engineering*

*Southern University of Science and Technology*

Shenzhen, China

1628280289@qq.com

**Abstract**—In this project, we are intended to design two models using the method of LogisticRegression and kNeighbours respectively, and see which one performs better. Moreover, we will try to design suitable hyper-parameters, in order to get a better result.

**Index Terms**—machine learning, logistic regression method, kNN method,binary classification problem

## I. INTRODUCTION

In this project, we are intended to design two models, in order to predict breast cancer with 30 elements, using the method of LogisticRegression and kNeighbours respectively, and see which one performs better. Moreover, we will try to design suitable hyper-parameters, in order to get a better result.

## II. LOGISTICREGRESSION

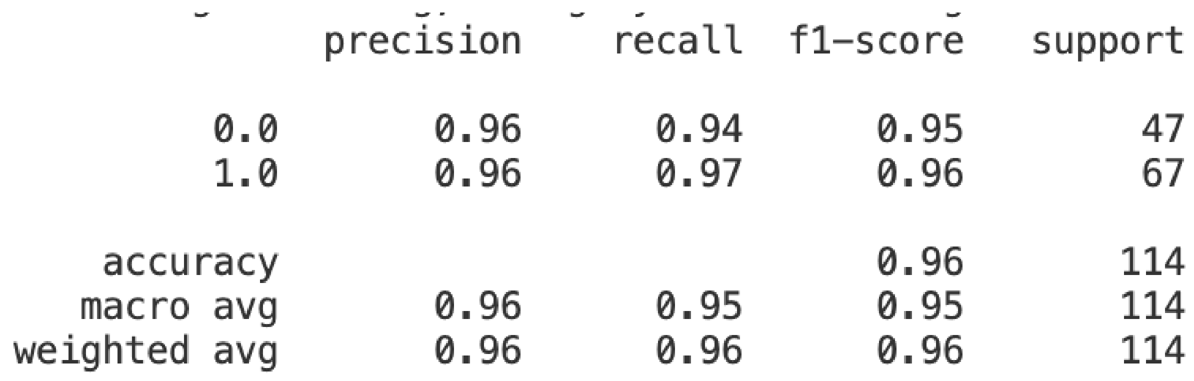
### A. Training result of the model

Here we just post the core codes as following:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
model = LogisticRegression()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
accuracy_score(y_test,y_predict)
```

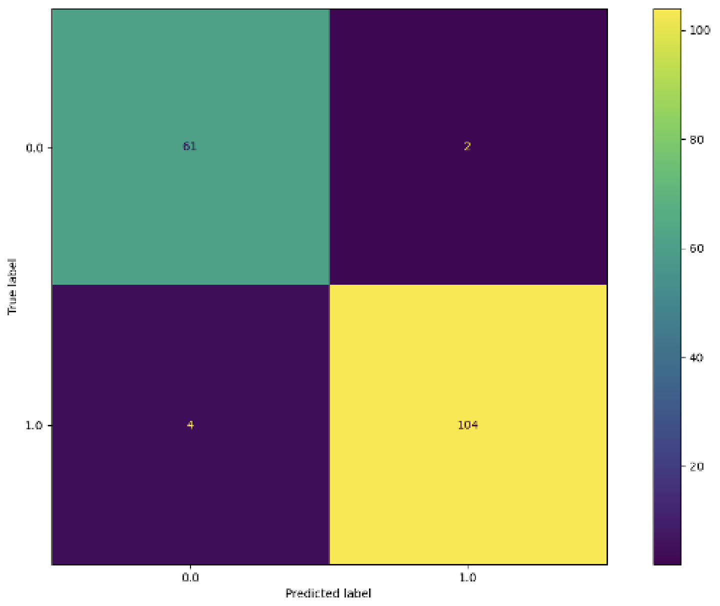
```
plot_confusion_matrix(model,x_test,y_test)
print(classification_report(y_test,y_predict))
xx= np.arange(0, 1)
yy=xx
plt.plot(xx, yy)
plt.title('LogisticRegression')
plt.show()
```

The result of the training of the model is as following:



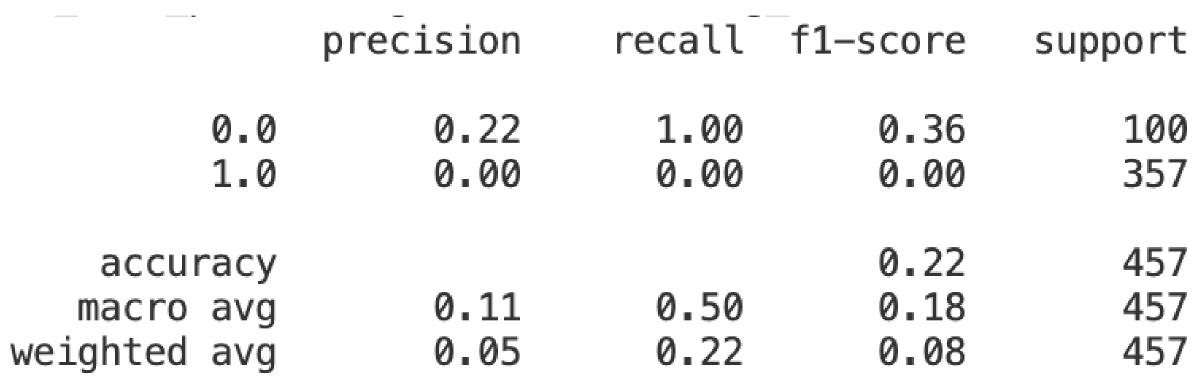
```
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95        47
         1.0       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

Since we randomly separate the dataset each time we train the model, the result differs from each other. Here we just take an example.



## B. Validation result of the model

The result is terrible: all of the test examples are verified as negative.



```
              precision    recall  f1-score   support

         0.0       0.22      1.00      0.36       100
         1.0       0.00      0.00      0.00       357

    accuracy                           0.22       457
   macro avg       0.11      0.50      0.18       457
weighted avg       0.05      0.22      0.08       457
```

After referring to the internet, I find the following information. The category imbalance problem refers to the situation where the number of training samples for different classes in the classification task varies greatly. In general, unbalanced samples cause the trained model to focus on classes with a large number of samples and "trivialize" categories with a small number of samples, so that the model's ability to generalize on the test data will be affected. For example, there are 99 positive samples and 1 negative sample in the training set. In many cases where sample imbalance is not considered, the learning algorithm will make the classifier abandon negative case prediction, because dividing all samples into positives can obtain up to 99% training classification accuracy.

In order to solve the problem, we have these four method:

1. Undersampling: remove some excess samples in the classification to achieve a positive and negative sample balance;
2. Oversampling: adding samples in some categories with a small number of samples to achieve a balance between positive and negative sample numbers;
3. Category uniform sampling method: through some processing of each type of data, the probability of reaching the final sampling of each type of data is the same;
4. Threshold shift: Use the "rescaling" idea: The threshold shift method is a method that uses the original training set to train the classifier and add the idea of rescaling when predicting, which is used to alleviate the imbalance of classes.

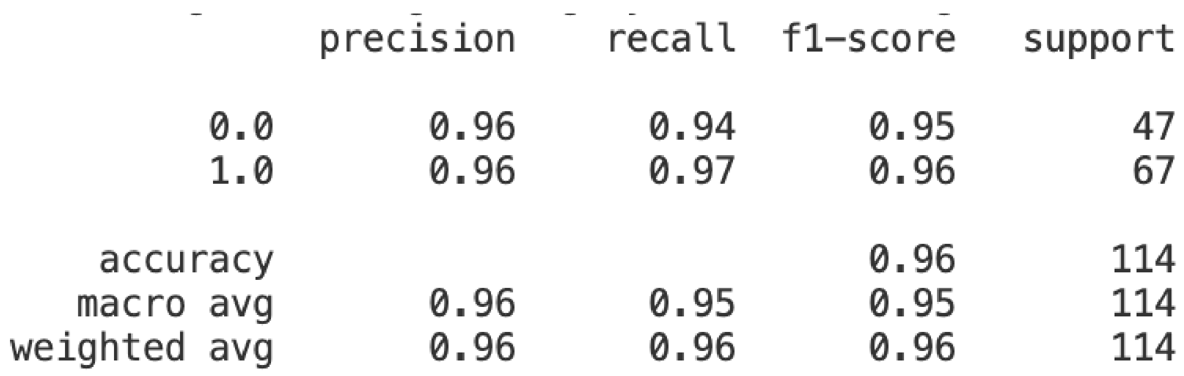
## III. KNEIGHBOURS

### A. Training result of the model

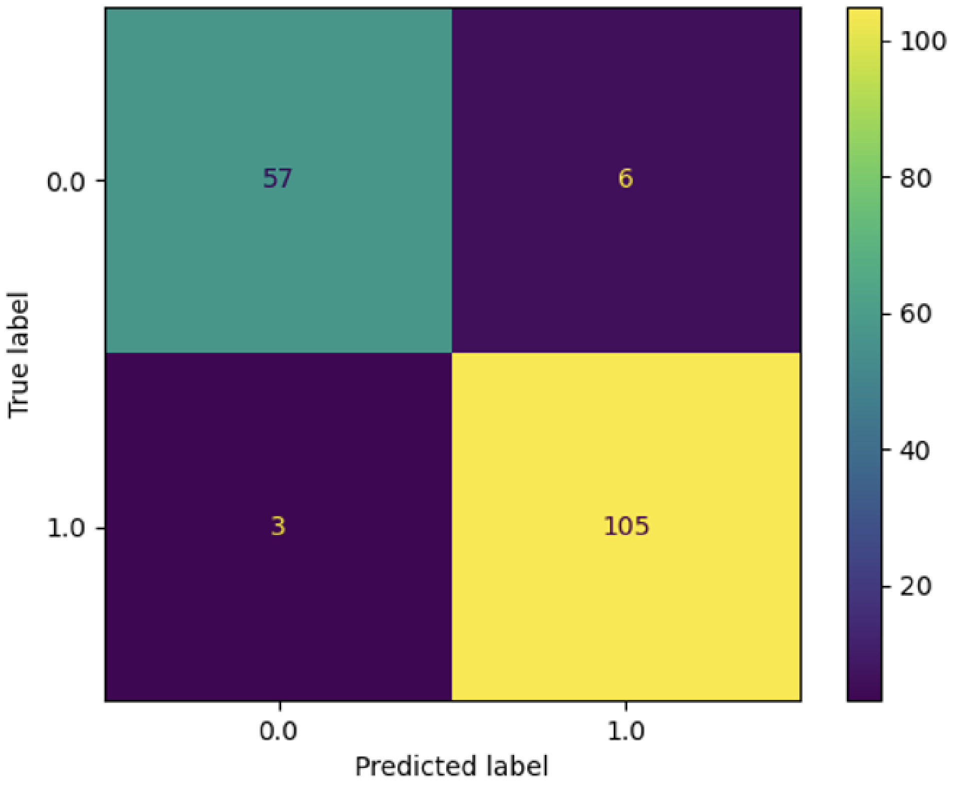
The codes of this method is just like the above one, except for the model. Here the default number for the neighbour is 5. Later we will change it to see which performs best.

```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3,random_state=0)
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
model = KNeighborsClassifier()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
accuracy_score(y_test,y_predict)
plot_confusion_matrix(model,x_test,y_test)
print(classification_report(y_test,y_predict))
xx = np.arange(0, 1)
yy = xx
plt.plot(xx, yy)
plt.show()
```

The result of the training of the model is as following:

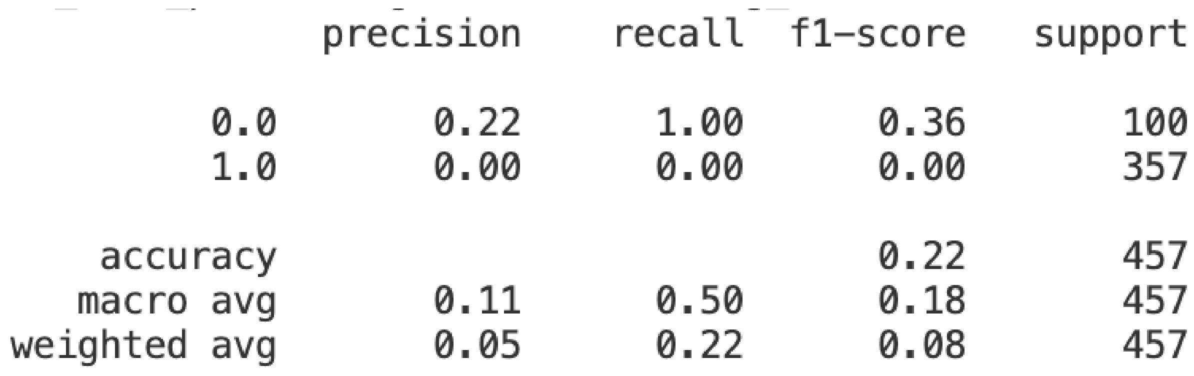


```
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95        47
         1.0       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```



## B. Validation result of the model

The result is also terrible: all of the test examples are verified as negative. However,I do not have enough time to complete the research.



```
              precision    recall  f1-score   support

         0.0       0.22      1.00      0.36       100
         1.0       0.00      0.00      0.00       357

    accuracy                           0.22       457
   macro avg       0.11      0.50      0.18       457
weighted avg       0.05      0.22      0.08       457
```

## III. CONCLUSION

In this project, we learn to solve t a binary classification problem with two kinds of machine learning methods, and we try to find the most suitable hyper parameter.

Since the dataset is not evenly distributed, we take four kinds of methods to revise the validation result.