CS573 - Lab4 Report

Phuong Nguyen

April 2020

1 Overview

We used python and the Scikit-Learn library for this lab.

For ensemble classifier using weighted majority, we consider two types: hard voting and soft voting.

2 Tasks

2.1 Task 1

Train ensemble models Random Forest (RF) and AdaBoost.M1 with decision stumps as base classifiers (Note you may use other versions of AdaBoost if the machine learning package you choose to use does not have this particular version). Experiment with different values of hyper-parameters such as number of base classifiers etc.

- Ensemble model Random Forest (RF)
  - The corresponding hyper-parameters:
    - min_samples_leaf = 9
    - min_samples_split = 2
    - n_estimators = 10
  - The overall classification accuracy of RF on testing data: 82.06%
  - The confusion matrix of RF is shown in figure 1.
- Ensemble model Ada Boost (ADA) with decision stumps as base classifiers
  - The corresponding hyper-parameters:
    - learning_rate = 0.5
    - n_estimators = 100
  - The overall classification accuracy of ADA on testing data: 81.06%

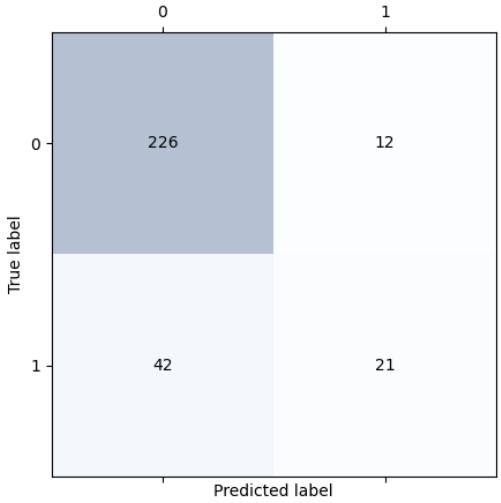

Figure 1: Confusion matrix of Random Forest (RF).

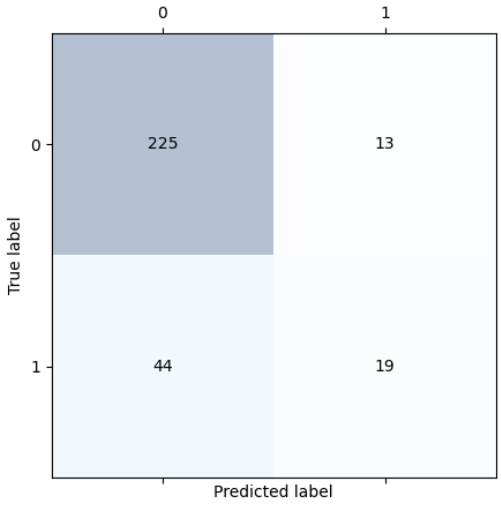

Figure 2: Confusion matrix of Ada Boost (ADA).

  - The confusion matrix of ADA is shown in figure 2.

- Discuss the results

The accuracy of RF is a bit better than that of ADA. Their results are pretty good considering this is a unbalanced data set (class distribution is not balanced).

2.2 Task 2

Train 4 individual models: Neural Network (NN), Logistic Regression (LR), Naive Bayes (NB), Decision Tree (DT). Report the confusion matrix and classification accuracy on the test data for each of them. When training the 4 models, slightly tune the hyper-parameters (extensive grid search is not required). Report the experiments you have done.

- Neural Network (NN)
  - The corresponding hyper-parameters:
    - hidden_layer_sizes = (50, 50)
  - The overall classification accuracy of NN on testing data: 78.41%
  - The confusion matrix of NN is shown in figure 3.

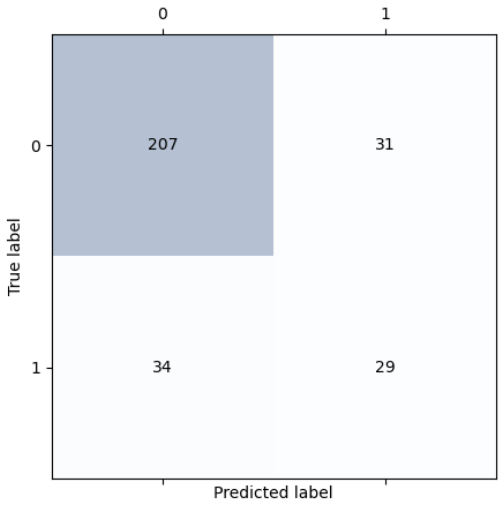

Figure 3: Confusion matrix of Neural Network (NN).

- Logistic Regression (LR)
  - The corresponding hyper-parameters:
    - Cs=1

  - The overall classification accuracy of LR on testing data: 81.40%
  - The confusion matrix of LR is shown in figure 4.

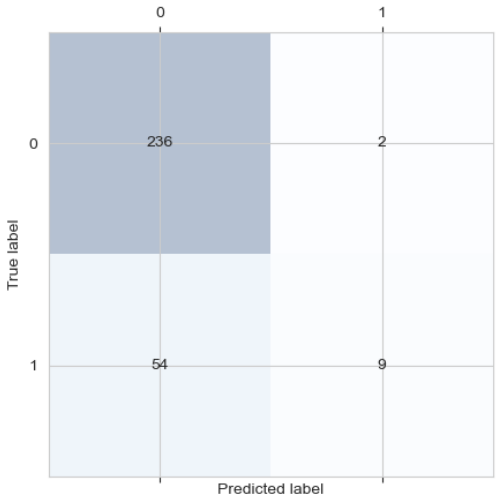

Figure 4: Confusion matrix of Logistic Regression (LR).

- Naive Bayes (NB)
  - There is no corresponding hyper-parameters.
  - The overall classification accuracy of NB on testing data: 80.07%
  - The confusion matrix of NB is shown in figure 5.
- Decision Tree (DT)
  - The corresponding hyper-parameters:
    - max_depth = 3
  - The overall classification accuracy of DT on testing data: 81.40%
  - The confusion matrix of DT is shown in figure 6.
- ensemble classifier using unweighted majority vote
  - The overall classification accuracy on testing data: 81.06%
- ensemble classifier using weighted majority vote (hard voting)
  - The corresponding hyper-parameters:

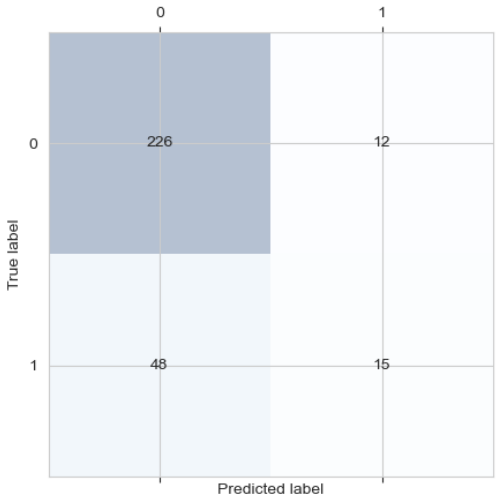

Figure 5: Confusion matrix of Naive Bayes (NB).

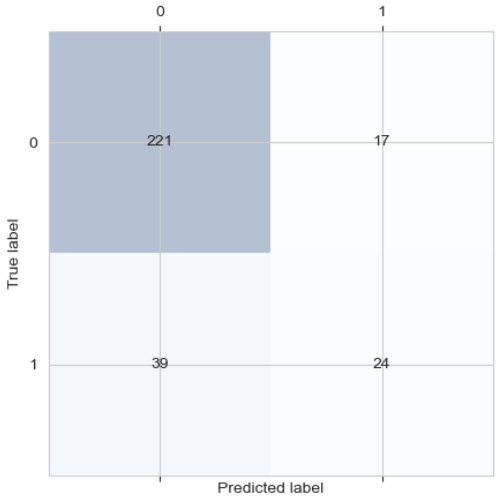

Figure 6: Confusion matrix of Decision Tree (DT).

∗ weights = [1, 3 ,2 ,3] (we chose weights proportional to the classification accuracy)

  - The overall classification accuracy on testing data: 81.39%

- ensemble classifier using weighted majority vote (soft voting)
  - The corresponding hyper-parameters:
    - weights = [1, 3 ,2 ,3] (we chose weights proportional to the classification accuracy)
  - The overall classification accuracy on testing data: 82.06%

- Discuss the results

  The four models NN, LR, NB and DT generally predicts with less accuracy on testing data than RF and ADA. Out of them, LR and DT have the highest accuracy while NN is the worst.

  The unweighted voting classifier is simply not good enough when its accuracy is even lower than that of LR and DT. This is probably because it just simply averages out all four classifiers.

  Meanwhile, the weighted voting classifier (hard voting method) performs as well as LR and DT on testing data. Usually, the weighted voting classifier will slightly outperforms all individual classifiers but in this case, the unbalanced data may affect the general performance.

  Finally, if all classifiers are able to estimate class probabilities (which they all are in this case), then we can tell Scikit-Learn to predict the class with the highest class probability, averaged over all the individual classifiers. This is called "soft voting". It often achieves higher performance than hard voting because it gives more weight to highly confident votes. As expected, the weighted soft voting classifier achieves 82.06% accuracy.