Project 5 & Project 9 report

Xiaolin Zhang 2019xxxxxx xxxxxxx
Ziyang Xu 2019xxxxxx xxxxxxx
Wangzhihui Mei 2019124044 6603385
Xiaohuan Pei 2019124022 6603592
CCNU-UOW JI

1 Project 5

1.1 Dataset: Artificial Characters Learning Problem

This database has been artificially generated by using a first order theory which describes the structure of ten capital letters of the English alphabet and a random choice theorem prover which accounts for etherogeneity in the instances. The capital letters represented are the following: A, C, D, E, F, G, H, L, P, R. Each instance is structured and is described by a set of segments (lines) which resemble the way an automatic program would segment an image.

Each instance is stored in a separate file whose format is the following:

```
CLASS  OBJNUM  TYPE  XX1  YY1  XX2  YY2  SIZE   DIAG
1      0       line  0    0    0    13   13.00  45.28
1      1       line  20   0    22   15   15.13  45.28
1      2       line  0    13   22   15   22.09  45.28
1      3       line  0    13   0    27   14.00  45.28
1      4       line  22   15   23   39   24.02  45.28
1      5       line  0    27   23   39   25.94  45.28
```

where CLASS is an integer number indicating the class as described below, OBJNUM is an integer identifier of a segment (starting from 0) in the instance and the remaining columns represent attribute values. Schapire and Singer (1999).

The generated character image is like Figure 1

1.2 Data pre-process

The character described by segments is represented as the vertex pair, we transform them to binary grid (12×8), like Figure 2.

Then the problem is transformed into one like the MNIST classification problem. The feature is the flattened 0-1 pixel, whose dimension is 96×1, the label is "A", "C", "D", "E", "F", "G", "H", "L", "P", "R", which correspond to the numbers 0 through 9.

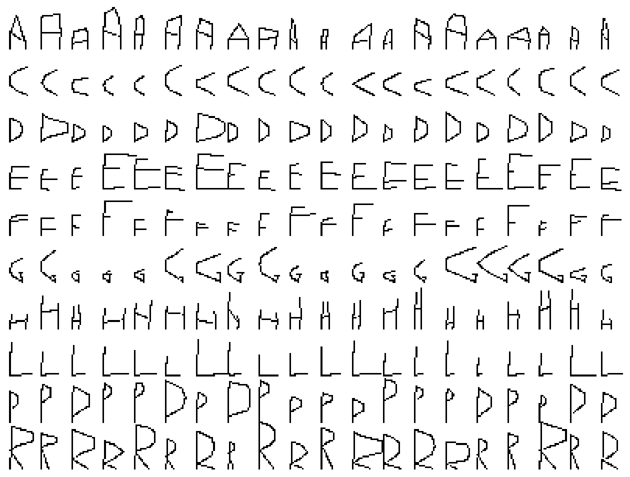


Figure 1: The generated characters

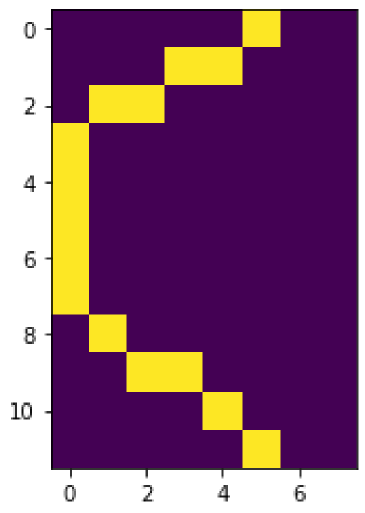


Figure 2: The representation of "C"

We got 6000 training data from the primitive dataset. We shuffled them and use 70% of the data as training data, 30% as test data. The label distribution of training data is like Figure 3

1.3 Model construction

1.3.1 Ensembling Model

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble

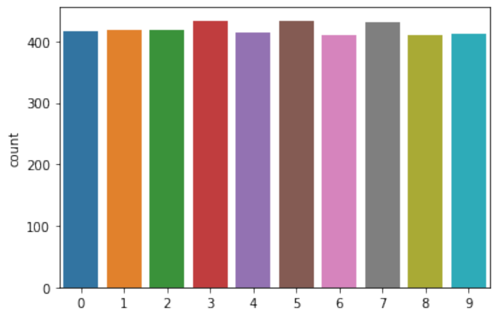


Figure 3: The label distribution of training data

approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model. Most of the practical data mining solutions utilize ensemble modeling techniques. Chapter 4 Classification covers the approaches of different ensemble modeling techniques and their implementation in detail.

In this task, We applied Voting method. Voting is a combination strategy for classification problems in integrated learning. The basic idea is to select the most output class among all machine learning algorithms.

There are two types of output of classification machine learning algorithms: one is to directly output class labels, and the other is to output class probabilities. Using the former to vote is called hard voting (majority/hard voting), and using it to classify is called soft voting (Soft voting). VotingClassifier in sklearn is the implementation of voting method.

In this project, We compared hard voting classifier and soft voting classifier, and got better performance in hard voting classifier, which ensemble Logistic Regression, SGD, SVM, Decision Tree, Random Forest, Extra Tree, MLP as one classifier. The structure is like Figure 4.

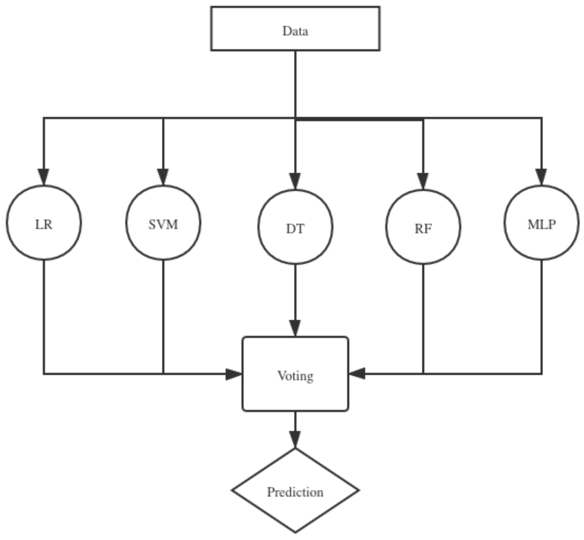
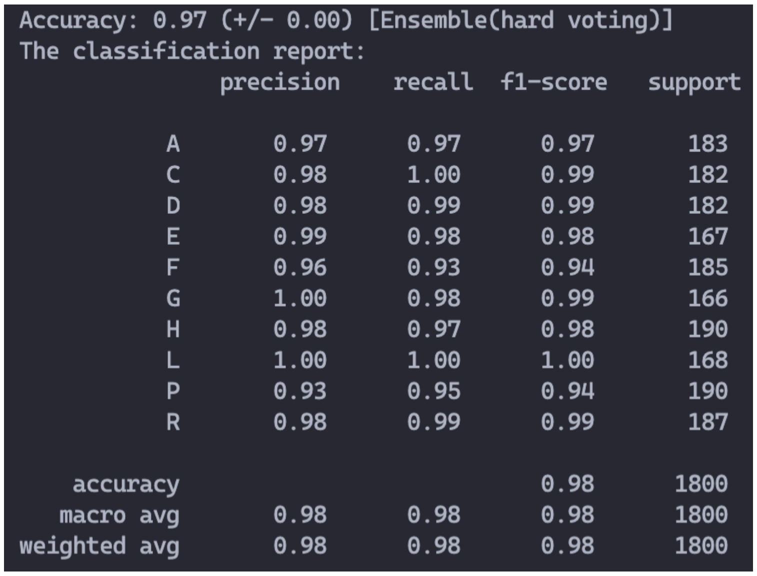


Figure 4: The Ensembled Voting classifier

Performance

We calculated the Recall, Precision, Accuracy and F1-score of the model. As this is a multi-class classification, so each label correspond to one micro value, we can calculate the mean of micro value and get marco value.



```
Accuracy: 0.97 (+/- 0.00) [Ensemble(hard voting)]
The classification report:
              precision    recall  f1-score   support

           A       0.97      0.97      0.97       183
           C       0.98      1.00      0.99       182
           D       0.98      0.99      0.99       182
           E       0.99      0.98      0.98       167
           F       0.96      0.93      0.94       185
           G       1.00      0.98      0.99       166
           H       0.98      0.97      0.98       190
           L       1.00      1.00      1.00       168
           P       0.93      0.95      0.94       190
           R       0.98      0.99      0.99       187

    accuracy                           0.98      1800
   macro avg       0.98      0.98      0.98      1800
weighted avg       0.98      0.98      0.98      1800
```

Figure 5: The Ensembling (Hard voting) performance

So, we can draw the conclusion that Ensemble Model perform well on both micro and marco f1-score.

1.4 Gradient Boosting Decision Tree

1.4.1 model

Gbdt(Gradient Boosting Decision Tree) is an algorithm that classifies or regresses data by using the additive model (i.e. linear combination of basis functions) and reducing the residual generated in the training process.

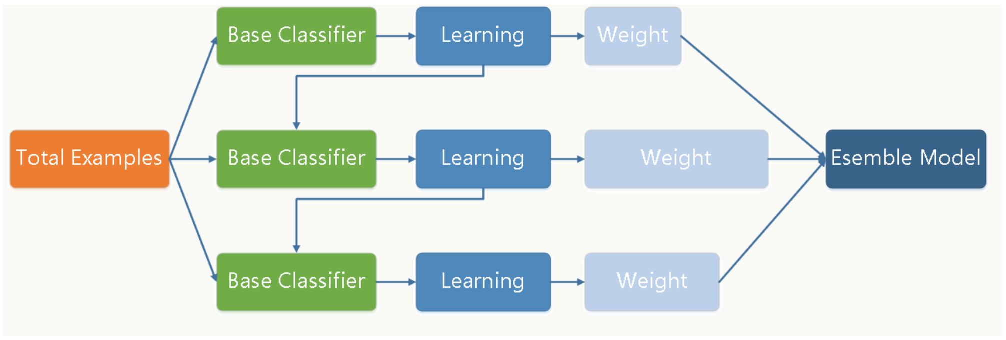


Figure 6: Training process of gbdt

Gbdt generates a weak classifier through multiple iterations, each of which is trained on the basis of the residual of the previous one. The requirements for weak classifiers are generally simple enough, with low variance and high deviation. Because the training process is to continuously improve the accuracy of the final classifier by reducing the deviation (which can be proved here). In general, the weak classifier will choose cart tree. Because of the above high deviation and simple requirements, the depth of each classification regression tree will not be very deep. The final total classifier is the weighted sum of the weak classifiers (that is, the addition model).

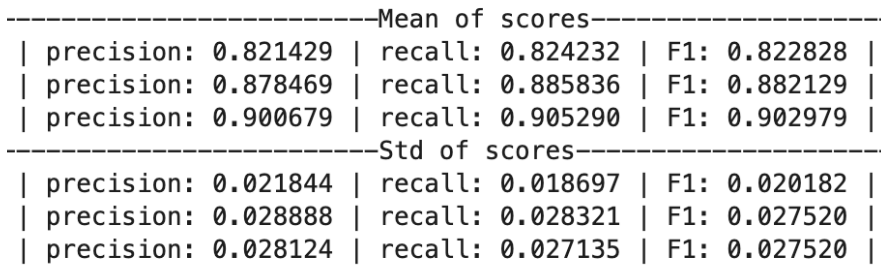
Assume that the dataset is represented by $T = \{(x_i, y_i)\}$, our goal is to get estimate $f(\hat{x})$.The detailed algorithm is as follows:

Algorithm 1: Algorithm of gbdt

```
**Result:** $f(\hat{x})$
**Input:** dataset $T = \{(x_i, y_i)\}, i = 1, 2, ..., n$ , loss function $L(y, f(x))$
Initialize $f_0(x) = argmin_c \sum_{i=1}^{N} L(y_i, x)$
**for** $m = 1, 2, ..., M$ **do**
    **for** $n = 1, 2, ..., N$ **do**
        compute $r_{mi} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]$
        for each $r_{mi}$, fit a CART to get $R_{mj}$ of $m$th tree
        for each j=1,2,...,J compute $c_{mj} = argmin_c \sum L(y_i, f_{m-1}(x_i) + c)$
        update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj} T(x \in R_{mj})$
    **end**
    Then compute $f(\hat{x}) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$
**end**
```

1.4.2 performance

The performance of gbdt is measured by Recall, Precision, Accuracy and F1-score.



```
-------------------------Mean of scores--------------------
 | precision: 0.821429 | recall: 0.824232 | F1: 0.822828 |
 | precision: 0.878469 | recall: 0.885836 | F1: 0.882129 |
 | precision: 0.900679 | recall: 0.905290 | F1: 0.902979 |
-------------------------Std of scores---------------------
 | precision: 0.021844 | recall: 0.018697 | F1: 0.020182 |
 | precision: 0.028888 | recall: 0.028321 | F1: 0.027520 |
 | precision: 0.028124 | recall: 0.027135 | F1: 0.027520 |
```

Figure 7: Performance of gbdt

References

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.