MSC-BDT5002, Spring 2020

Knowledge Discovery and Data Mining

Assignment 2 Report

HUANG XIN XIAN 20630198

1. Comparison of Classifier

Decision Tree

The performance of decision trees with different parameters are shown in the following table. The metrics including precision, recall and F1 use the **macro** average.

Classifier	Accuracy	Precision	Recall	F1	Training Time
gini, depth = 5	0.3694	0.392422	0.368504	0.327031	0.0965425
gini, depth = 10	0.7142	0.764549	0.717519	0.727371	0.0881393
gini, depth = 15	0.8322	0.842009	0.832842	0.83492	0.102693
gini, depth = 20	0.8656	0.867507	0.865704	0.865984	0.100667
gini, depth = 25	0.874	0.874684	0.874163	0.874173	0.10457
entropy, depth = 5	0.5014	0.560991	0.501367	0.498582	0.0697527
entropy, depth = 10	0.7976	0.802208	0.797646	0.798261	0.0919123
entropy, depth = 15	0.8686	0.868946	0.869505	0.868792	0.105164
entropy, depth = 20	0.8718	0.872123	0.872152	0.87189	0.10348
entropy, depth = 25	0.8718	0.872123	0.872152	0.87189	0.102728

From the table above, we can know that:

1. The performance of Decision Tree classifiers gets better for both gini and entropy criterions as the trees grow deeper. The reason may be that there are 26 classes in the dataset and deeper trees can fit the complicated data better. The performance of decision trees with entropy criterion are the same for depth 20 and 25. Actually the depth is the "max_depth" attribute in sklearn Decision Tree, and by using "get_depth()", we can find that the real depths are both 20. Therefore, they have the same performance.
   Generally, for the same depth, classifier with entropy criterion perform better than that with gini criterion, especially when the depth is small.
   Intuitively, the training time also increase with deeper trees, and the training time for trees with the same depth under different criterion are roughly close.

2. For each classifier, accuracy, precision, recall and f1 are close. This is a multi-class classification problem and the distribution of training dataset and testing dataset is balanced, so the metrics will have close values.

```
# the distribution of the training data
train_y.value_counts()
```

```
13    607
21    606
14    605
25    602
20    599
4     598
16    597
1     594
17    594
24    583
22    581
23    576
10    575
15    574
11    572
18    568
2     567
9     567
5     565
6     565
19    563
12    560
3     554
7     547
26    543
8     538
Name: 0, dtype: int64
```

```
# the distribution of the test data
test_y.value_counts()
```

```
7     226
6     210
4     207
21    207
16    206
24    204
5     203
12    201
2     199
20    197
8     196
1     195
26    191
18    190
17    189
9     188
19    185
13    185
25    184
22    183
3     182
15    179
14    178
23    176
10    172
11    167
Name: 0, dtype: int64
```

KNN, Random Forest

Classifier	Accuracy	Precision	Recall	F1	Training Time
KNN, n_neighbors=1	0.957	0.956873	0.957202	0.956972	0.0973889
KNN, n_neighbors=3	0.9516	0.951401	0.952159	0.951601	0.0912945
KNN, n_neighbors=5	0.9524	0.952433	0.952713	0.952311	0.0923076
Random Forest, max_depth=10	0.8178	0.832969	0.818894	0.822028	0.174276
Random Forest, max_depth=15	0.9112	0.914094	0.911998	0.912351	0.180936
Random Forest, max_depth=20	0.9306	0.931697	0.931407	0.931154	0.187442

From the table above, we can know that:

1. KNN classifiers take much less training time than Random Forest classifier. It is because KNN classifiers just store the dataset during training without calculation, and Random Forest classifiers have to do a lot calculation in the training phrase.

2. KNN classifiers outperform the Random Forest on this dataset, and KNN is usually suitable for multi-class classification problems. When n_neighbors=1, the KNN model has the best performance among the three ones because the closest point is very likely to share the same class label. But KNN classifier for 5 neighbors perform better slightly than classifier with 3 neighbors. It may because when we consider more neighbors in a reasonable range, we can consider more points and do better estimation.
3. For Random Forest, classifiers with larger max_depth can have better performance, because deeper trees can fit the complicated dataset better. Intuitively, classifier with larger max_depth also takes more time to train.
4. For each classifier, the metrics have similar values because of the balance of the dataset.

2. Implementation of Adaboost

Using five weak classifiers, the final expression and details of each weak classifier given by my program are shown as below.

```
final classifier:
C*(x) = sign[1.738C1(x) + 1.5906C2(x) + 1.4904C3(x) + 1.5938C4(x) + 1.5963C5(x)]
-------- basic classifiers details --------
classifier 1:
x < 2.5,    1
x >= 2.5,  -1
-------------------------------------------
classifier 2:
x < 7.5,    1
x >= 7.5,  -1
-------------------------------------------
classifier 3:
x > 4.0,    1
x <= 4.0,  -1
-------------------------------------------
classifier 4:
x < 1.5,    1
x >= 1.5,  -1
-------------------------------------------
classifier 5:
x > 5.0,    1
x <= 5.0,  -1
-------------------------------------------
predict:
[ 1.   1. -1. -1. -1. -1.   1.   1. -1. -1.]
accuracy = 0.8
```