과목명	인공지능
담당교수	황두성 교수님
학과	소프트웨어학과
학번	32153180 / 32152057 / 32171652
이름	이 상 민 / 방 승 환 / 박 세 지
제출일자	2019.12.9

Problem 1

Answer the following question on a 2-class classification problem. The training set is given in Figure 1. Check out the accuracy of a training and testing sets when 80% data are used for training and the rest for testing. The file name is *dstest.txt.*

(a) Build and evaluate decision trees in terms of max_depth, min_samples_split, and min_samples_leaf.

```
from sklearn import tree
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
import os

report1 = pd.read_csv("C:/Users/sangmin/Desktop/dstest.csv")
X = np.array(pd.DataFrame(report1, columns=['X','y']))
y = np.array(pd.DataFrame(report1, columns=['C']))
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8)

dt_clf = DecisionTreeClassifier(criterion='entropy')
dt_clf = dt_clf.fit(X_train, y_train)
y_train_pred = dt_clf.predict(X_train)
y_test_pred = dt_clf.predict(X_test)
tree_train = accuracy_score(y_train, y_train_pred)
tree_test = accuracy_score(y_test, y_test_pred)
print('[Default] \t train accuracy %.3f \n\t\t test accuracy %.3f' % (tree_train,
tree_test))
print('\n')

# max_depth
num = 2;
for i in range(5):
    dt_clf = DecisionTreeClassifier(criterion='entropy', max_depth = num)
    dt_clf = dt_clf.fit(X_train, y_train)
    y_train_pred = dt_clf.predict(X_train)
    y_test_pred = dt_clf.predict(X_test)
    tree_train = accuracy_score(y_train, y_train_pred)
```

```
    tree_test = accuracy_score(y_test, y_test_pred)
    print('[max_depth=%d]        train accuracy %.3f \n\t\t test accuracy %.3f' %
(num, tree_train, tree_test))
    num += 1
print('\n')

# min_samples_split
num = 250
for i in range(10):
    dt_clf = DecisionTreeClassifier(criterion='entropy', min_samples_split = num)
    dt_clf = dt_clf.fit(X_train, y_train)
    y_train_pred = dt_clf.predict(X_train)
    y_test_pred = dt_clf.predict(X_test)
    tree_train = accuracy_score(y_train, y_train_pred)
    tree_test = accuracy_score(y_test, y_test_pred)
    print('[min_split=%d]      train accuracy %.3f \n\t\t test accuracy %.3f' % (num,
tree_train, tree_test))
    num += 10
print('\n')

# min_samples_leaf
num = 10
for i in range(10):
    dt_clf = DecisionTreeClassifier(criterion='entropy', min_samples_leaf = num)
    dt_clf = dt_clf.fit(X_train, y_train)
    y_train_pred = dt_clf.predict(X_train)
    y_test_pred = dt_clf.predict(X_test)
    tree_train = accuracy_score(y_train, y_train_pred)
    tree_test = accuracy_score(y_test, y_test_pred)
    print('[min_leaf=%d]      train accuracy %.3f \n\t\t test accuracy %.3f' % (num,
tree_train, tree_test))
    num += 5
```

```
========== RESTART: C:₩Python37₩Decision_tree.py ==========
[Default]         train accuracy 1.000
                  test accuracy 1.000


[max_depth=2]     train accuracy 0.854
                  test accuracy 0.800
[max_depth=3]     train accuracy 0.944
                  test accuracy 0.883
[max_depth=4]     train accuracy 1.000
                  test accuracy 1.000
[max_depth=5]     train accuracy 1.000
                  test accuracy 1.000
[max_depth=6]     train accuracy 1.000
                  test accuracy 1.000


[min_split=250]     train accuracy 1.000
                  test accuracy 1.000
[min_split=260]     train accuracy 0.944
                  test accuracy 0.883
[min_split=270]     train accuracy 0.944
                  test accuracy 0.883
[min_split=280]     train accuracy 0.944
                  test accuracy 0.883
[min_split=290]     train accuracy 0.944
                  test accuracy 0.883
[min_split=300]     train accuracy 0.854
                  test accuracy 0.800
[min_split=310]     train accuracy 0.854
                  test accuracy 0.800
[min_split=320]     train accuracy 0.854
                  test accuracy 0.800
[min_split=330]     train accuracy 0.854
                  test accuracy 0.800
[min_split=340]     train accuracy 0.854
                  test accuracy 0.800


[min_leaf=10]     train accuracy 1.000
                  test accuracy 1.000
[min_leaf=15]     train accuracy 1.000
                  test accuracy 1.000
[min_leaf=20]     train accuracy 1.000
                  test accuracy 1.000
[min_leaf=25]     train accuracy 1.000
                  test accuracy 1.000
[min_leaf=30]     train accuracy 0.994
                  test accuracy 1.000
[min_leaf=35]     train accuracy 0.983
                  test accuracy 0.992
[min_leaf=40]     train accuracy 0.969
                  test accuracy 0.983
[min_leaf=45]     train accuracy 0.956
                  test accuracy 0.983
[min_leaf=50]     train accuracy 0.938
                  test accuracy 0.967
[min_leaf=55]     train accuracy 0.917
                  test accuracy 0.858
>>>
```

(b) Build and evaluate an AdaBoost ensemble where max_depth is 2 and n_estimators varies from 10 to 50 in 5 increments.

```
from sklearn import tree
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

report1 = pd.read_csv("C:/Users/sangmin/Desktop/dstest.csv")
X = np.array(pd.DataFrame(report1, columns=['X','y']))
y = np.array(pd.DataFrame(report1, columns=['C']))
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size = 0.8)

est = 10;
for i in range(9):
    ada = AdaBoostClassifier(n_estimators=est, random_state = 5)
    ada.fit(X_train, y_train)
    y_train_pred = ada.predict(X_train)
    y_test_pred = ada.predict(X_test)
    ada_train = accuracy_score(y_train, y_train_pred)
    ada_test = accuracy_score(y_test, y_test_pred)
    print('%d est AdaBoost train/test accuracies %.3f/%.3f' % (est, ada_train,
ada_test))
    est = est +5;
```

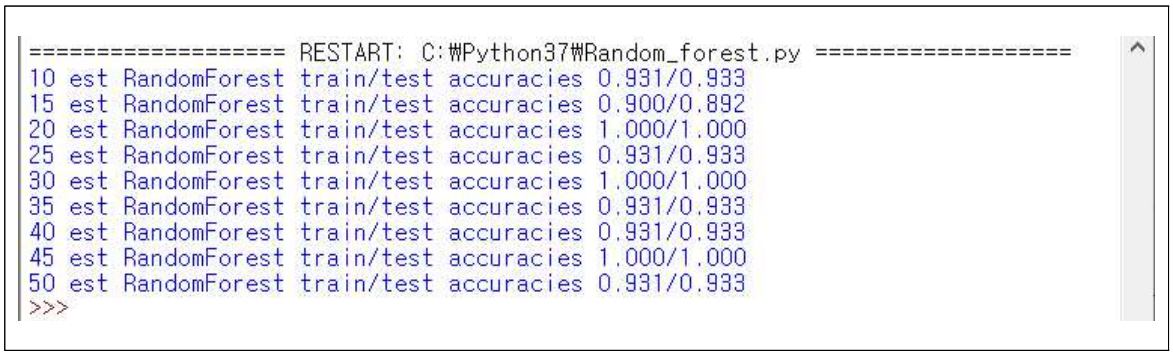
```
====================== RESTART: C:₩Python37₩Adaboost.py ======================
10 est AdaBoost train/test accuracies 1.000/1.000
15 est AdaBoost train/test accuracies 1.000/1.000
20 est AdaBoost train/test accuracies 1.000/1.000
25 est AdaBoost train/test accuracies 1.000/1.000
30 est AdaBoost train/test accuracies 1.000/1.000
35 est AdaBoost train/test accuracies 1.000/1.000
40 est AdaBoost train/test accuracies 1.000/1.000
45 est AdaBoost train/test accuracies 1.000/1.000
50 est AdaBoost train/test accuracies 1.000/1.000
>>>
```

(c) Build and evaluate a random forest ensemble where max_depth is 2 and n_estimators varies from 10 to 50 in 5 increments.

```
from sklearn import tree
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import pylab as plt
import os
import warnings
warnings.filterwarnings('ignore')

report1 = pd.read_csv("C:/Users/sangmin/Desktop/dstest.csv")
X = np.array(pd.DataFrame(report1, columns=['X','y']))
y = np.array(pd.DataFrame(report1, columns=['C']))
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size = 0.8)

est = 10;
for i in range(9):
    ran = RandomForestClassifier(n_estimators=est, criterion='entropy',
                                 max_depth=2, min_samples_split=10,
                                 min_samples_leaf=10, max_leaf_nodes=100,
                                 bootstrap=True)
    ran.fit(X_train, y_train)
    y_train_pred = ran.predict(X_train)
    y_test_pred = ran.predict(X_test)
    ran_train = accuracy_score(y_train, y_train_pred)
    ran_test = accuracy_score(y_test, y_test_pred)
    print('%d est RandomForest train/test accuracies %.3f/%.3f' % (est,ran_train,
ran_test))
    est = est+5;
```



(d) Discuss the results between decision tree and ensemble model.

주어진 데이터는 class와 attribute가 각각 2개로 비교적 단순하고 잘 분류되는 데이터이다. 그래서 decision tree와 ensemble model 모두 특정한 제한을 주지 않아도 매우 높은 정확도를 보였다. max_depth, min_samples_split, and min_samples_leaf처럼 early stopping 통해 simpler tree를 형성했다.

하지만 대부분의 데이터는 이 문제처럼 잘 분류되지 않는다. 많은 attribute에 대한 분류를 진행하는 경우 depth가 길어지고 node도 상당히 많아진다.

그에 따라 overfitting 가능성도 높아진다. 따라서 이러한 overfitting이 발생하지 않도록 하는 것이 중요하다.

The data given is relatively simple and well classified with two classes and two attributes. So both decision tree and ensemble model showed very high accuracy without any specific restrictions. This formed a simpler tree through early stopping, like max_depth, min_samples_split, and min_samples_leaf.

But most data is not as well classified as this one. When classifying many attributes, the depth becomes long and the node becomes quite large. This increases the chance of overfitting.

Therefore, it is important that such overfitting does not occur.

Problem 2

Answer the questions.

(a) Write a generic procedure for building a decision tree.

-> S1 Start at the top of the tree

S2 Grow it by splitting attributes one by one to determine which attribute to split, look at node impurity

S3 Assign leaf nodes the majority vote in the leaf

S4 When getting to the bottom, prune the tree to prevent overfitting

(b) Discuss the advantages and drawbacks of decision tree applications for solving real-world problems.

-> 장점 : interpretable or intuitive reasoning, model discrete outcomes nicely
단점 : overfitting, Learning the optimal decision tree is NP-complete

Problem 3

Answer the questions on information gain.

(a) Define information $I(p)$ from observing the occurrence of an event with probability p.

-> Number of bits needed to encode the probability of the event

$no.\ of\ bits = -\log_2 p, \quad 0 \le p \le 1$

(b) What is the meaning of information $I(p)$?

-> The expected number of bits needed to encode p

$I(p) = -\log_2 p$

(c) Prove the following subproblems with probability p and integer m and n.

(1) $I(p^n) = n \times I(p)$

-> $I(p^n) = -\log_2 p^n = n \times (-\log_2 p) = n \times I(p)$

(2) $I(p) = I(p^{(1/m)^m}) = mI(p^{1/m})$

-> $I(p^{(1/m)^m}) = -\log_2 p^{(1/m)^m} = m \times (-\log_2 p^{1/m}) = mI(p^{1/m})$

(3) $I(p^{m/n}) = \frac{m}{n} I(p)$

-> $I(p^{m/n}) = -\log_2 p^{m/n} = \frac{m}{n} \times (-\log_2 p) = \frac{m}{n} I(p)$

Table 1: A toy problem

X_1	X_2	X_3	Y
T	T	F	T
T	T	T	F
T	F	T	F
T	F	F	T
F	T	F	T
F	T	T	F
F	F	F	F
F	F	T	T

Problem 4

Draw a decision tree using information gain $IG(X)$ from Table 1. The decision tree is based on the pseudo code.

S1 Start from empty decision tree

S2 Split on the best attribute selected by computing

$$\arg\max_i IG(X_i) = \arg\max_i (H(Y) - H(Y|X_i))$$

S3 Do S2 until the termination condition is met

Problem 5

Describe and discuss an overfitting in decision tree practice.

-> As the training accuracy continues to increase, the test accuracy also increases initially. When the number of decision tree branches increase, test accuracy becomes decrease and this is called overfitting

So, muse use tricks to find simpler trees (fixed depth, early stopping, pruning) and use ensembles of different trees

Problem 6

The ensemble approach has become a cutting edge model in artificial intelligence. Below are some questions on ensemble model.

(a) Why do we use an ensemble model instead of a single model?

-> Decisions made by groups are often better than decisions made by a single member of a group

- Diversity of opinion, Independence, Decentralization, Aggregation

=> minimize errors, decrease overfitting

(b) Discuss bagging and boosting strategies to construct an ensemble model.

-> bagging : minimize variance, randomly drawn training set,
low computing, based on the majority voting,
training set with replacement

boosting : decrease bias, training set is weight misclassified data,
high computing,
increasing the weights of the incorrectly predicted examples

비교	Bagging	Boosting
특징	병렬 앙상블 모델 (각 모델은 서로 독립적)	연속 앙상블 (이전 모델의 오류를 고려)
목적	Variance 감소	Bias 감소
적합한 상황	복잡한 모델 (High variance, Low bias)	Low variance, High bias 모델
대표 알고리즘	Random Forest	Gradient Boosting, AdaBoost
Sampling	Random Sampling	Random Sampling with weight on error

(c) Point out the drawback of ensemble models.

Harder to tune than a single model
Lack of interpretability, compared to linear classifiers
Need high computation time

(d) Give model examples using bagging or boosting strategy for an ensemble model.

-> bagging : random forest / boosting : Adaboost, gradient boosting