Huang Hua

Haixu Wang

Homework Assignment 1

September 5, 2018

1 Decision Tree

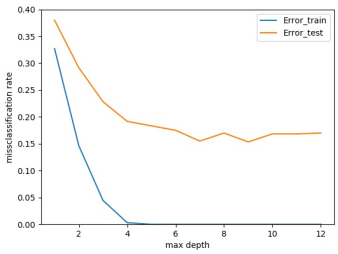
In this assignment, we use xgboost and scikit-learn packages in python to train the datasets separately. Since we don't have labels in test data sets, we have to use valid datasets as test datasets. The scikit-learn DecisionTreeClassifier package has two main splitting criterion: Gini and Entropy. Here, we use entropy as splitting criterion following the lecture notes.

1.1 Madelon Dataset

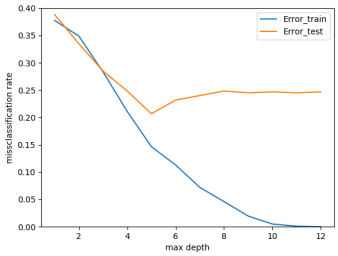
Table 1: Data information

Dataset	Features	**Train_obs**	**Test_obs**
Madelon	500	2000	600

In this dataset, this dataset has 500 features. We have 2000 observations as training data. Figure 1 shows after depth=5, the model from xgboost over fits the training data. And the error of test data converges to 0.15. The optimal choice of depth is 3 0r 4.



(a) Results from xgboost



(b) Results from Scikit-learn

Figure 1: Results for dataset Madelon

Table 2: Test error with depth

Testerror	Tree depth
0.38	1
0.292	2
0.228	3
0.192	4
0.183	5
0.175	6
0.155	7
0.17	8
0.153	9
0.1683	10
0.168	11
0.17	12

1.2 Wilt Dataset

Table 3: Data information

Dataset	Features	**Train_obs**	**Test_obs**
Wilt	5	4339	500

In this dataset, there are only 5 features but 4339 observations. After depth=2, the model from xgboost over fits the training data. And the error of test data converges to 0.18. The optimal choice of depth is 7.

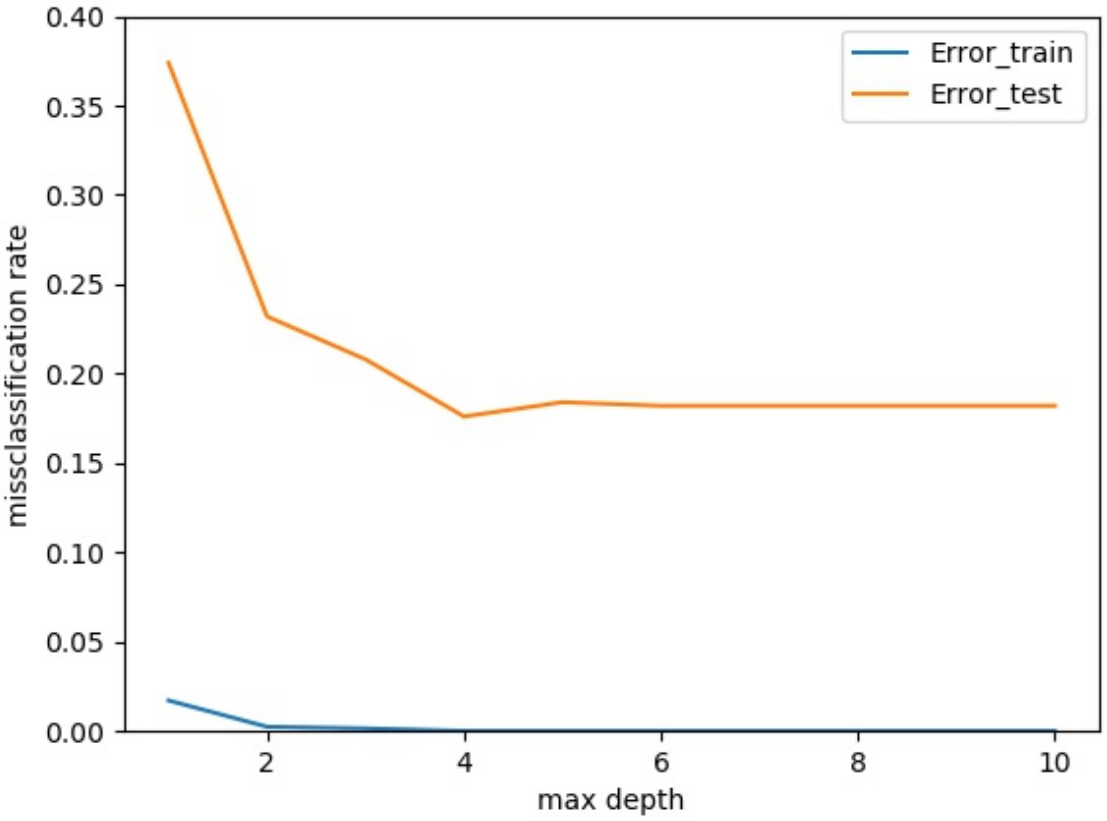


Figure 2

Table 4: Test error with depth

Testerror	Tree depth
0.374	1
0.232	2
0.208	3
0.176	4
0.184	5
0.182	6
0.182	7
0.182	8
0.182	9
0.182	10

1.3 Gisette Dataset

Table 5: Data information

Dataset	Features	**Train_obs**	**Test_obs**
Gisette	5000	6000	1000

First of all, in this dataset, we have too many features and relative small observations. With decision tree algorithm in xgboost, after depth=4, we can see the model over fits the training data. Therefore, the optimal choice of depth is 4.

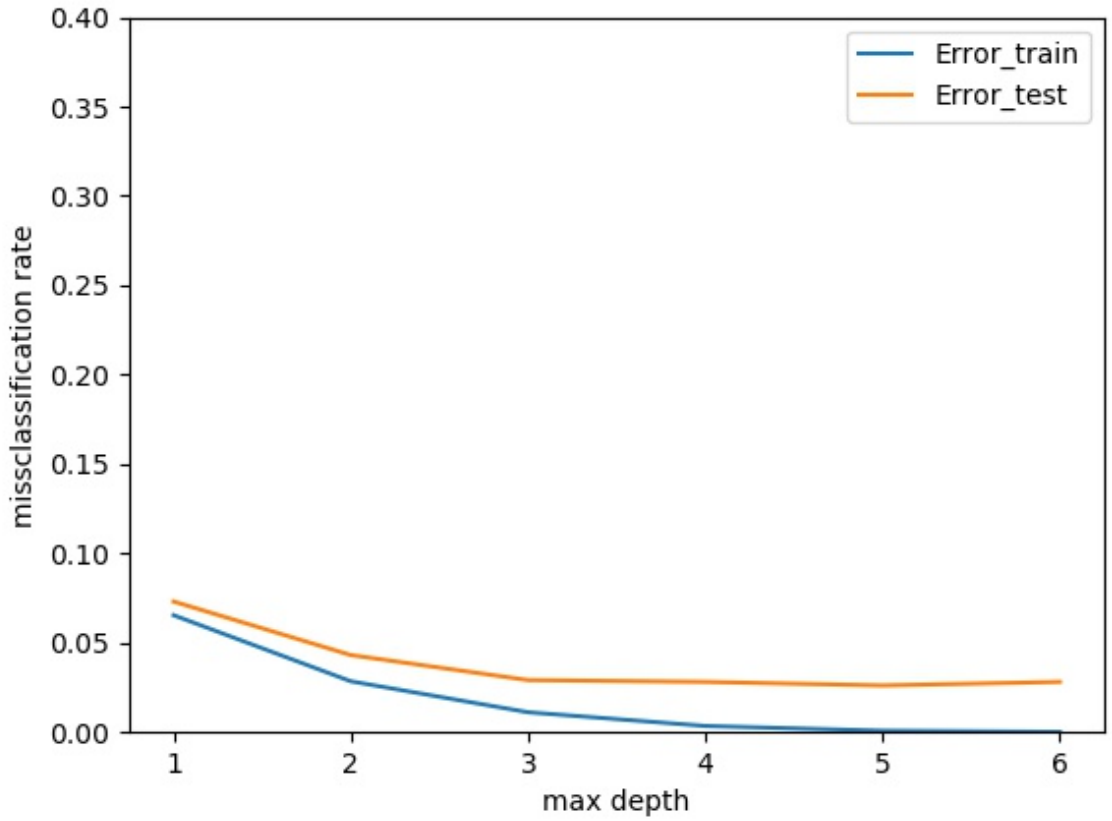


Figure 3

Table 6: Test error with depth

Testerror	Tree depth
0.073	1
0.043	2
0.029	3
0.028	4
0.026	5
0.028	6

2 Code

c def clearall(): all = [var for var in globals() if (var[:2], var[-2:]) != ("$_{","_{")]forvarinall:delglobals()[var]}$
clearall()
import xgboost as xgb import numpy as np from xgboost import XGBClassifier from sklearn import tree from sklearn.model$_{s}electionimporttrain_{t}est_{s}plitfromsklearn.metricsimportaccuracy_{s}coreimportpandasaspdimportmatplotlib.$
def decision$_{t}ree(X, Y, XTest, YTest, maxDepth) : rateTrain = np.zeros(maxDepth.shape[0])rateTest = np.zeros(maxDepth.shape[0])foriinrange(maxDepth.shape[0]) : fitmodelnotrainingdataclf = tree.DecisionTreeClassi$ $entropy', max_{d}epth = maxDepth[i])clf = XGBClassifier(max_{d}epth = maxDepth[i])clf.fit(X, Y)print(clf)y = clf.predict(X)yTest = clf.predict(XTest)py = [round(value)forvalueiny]pyTest = [round(value)forvalueinyTest]rateT$ $1-np.mean(py == Y)rateTest[i] = 1-np.mean(pyTest == YTest)accuracy = accuracy_{s}core(pyTest, YTest)print("Accu$ $print('train, testmisclassificationerror :', rateTrain[i], rateTest[i])returnrateTrain, rateTest$
def problem$_{1}a() : loaddatadmd = np.loadtxt('E : material2018_{F}all2018_{t}rain.txt')dmdl = np.genfromtxt('E : material2018_{F}all2018_{t}rainlabels.txt')dmdv = np.loadtxt('E : material2018_{F}all2018_{v}alid.txt')dmdlv = np.loadtxt('E : material2018_{F}all2018_{v}alidlabels.txt')$
maxDepth=np.arange(12)+1 train, test=decision$_{t}ree(dmd, dmdl, dmdv, dmdlv, maxDepth)$
figureIndex=0 plt.figure(figureIndex) figureIndex += 1 plt.plot(maxDepth, train, label='Error$_{t}rain')plt.plot(maxDepth, test,$ $Error_{t}est')$
plt.xlabel('max depth') plt.ylabel('missclassification rate') plt.ylim([0, 0.4]) plt.legend() plt.show()
def problem$_{1}b() : X = np.genfromtxt('E : material2018_{F}all2018_{t}rain.csv', delimiter =',')Y = np.loadtxt('E : material2018_{F}all2018_{t}rainlabels.txt')XTest = np.genfromtxt('E : material2018_{F}all2018_{t}est.csv', delimiter =' ,')YTest = np.loadtxt('E : material2018_{F}all2018_{t}estlabels.txt')maxDepth = np.arange(10)+1train, test = decision_{t}ree(X, Y, XTest, YTest, maxDepth)$
figureIndex=0 plt.figure(figureIndex) figureIndex += 1 plt.plot(maxDepth, train, label='Error$_{t}rain')plt.plot(maxDepth, test,$ $Error_{t}est')$
plt.xlabel('max depth') plt.ylabel('missclassification rate') plt.ylim([0, 0.4]) plt.legend() plt.show()
def problem$_{1}c() : X = np.loadtxt('E : material2018_{F}all2018_{t}rain.txt')Y = np.loadtxt('E : material2018_{F}all2018_{t}rainlabel$ $np.loadtxt('E : material2018_{F}all2018_{v}alid.txt')YTest = np.loadtxt('E : material2018_{F}all2018_{v}alidlabels.txt')maxDepth =$ $np.arange(6) + 1train, test = decision_{t}ree(X, Y, XTest, YTest, maxDepth)$
figureIndex=0 plt.figure(figureIndex) figureIndex += 1 plt.plot(maxDepth, train, label='Error$_{t}rain')plt.plot(maxDepth, test,$ $Error_{t}est')$
plt.xlabel('max depth') plt.ylabel('missclassification rate') plt.ylim([0, 0.4]) plt.legend() plt.show()
problem$_{1}a()problem_{1}b()problem_{1}c()$