**Assignment3**

Name:Dew

ID:18406040245

# Ensemble learning notebook

## (1)Bagging

%matplotlib inline

import itertools

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from mlxtend.plotting import plot\_learning\_curves

from mlxtend.plotting import plot\_decision\_regions

np.random.seed(0)

label = ['Decision Tree', 'K-NN', 'Bagging Tree', 'Bagging K-NN']

clf\_list = [clf1, clf2, bagging1, bagging2]

fig = plt.figure(figsize=(10, 8))

gs = gridspec.GridSpec(2, 2)

grid = itertools.product([0,1],repeat=2)

for clf, label, grd in zip(clf\_list, label, grid):

scores = cross\_val\_score(clf, X, y, cv=3, scoring='accuracy')

print "Accuracy: %.2f (+/- %.2f) [%s]" %(scores.mean(), scores.std(), label)

clf.fit(X, y)

ax = plt.subplot(gs[grd[0], grd[1]])

fig = plot\_decision\_regions(X=X, y=y, clf=clf, legend=2)

plt.title(label)

plt.show()

# 截屏2020-12-26 下午8.04.38

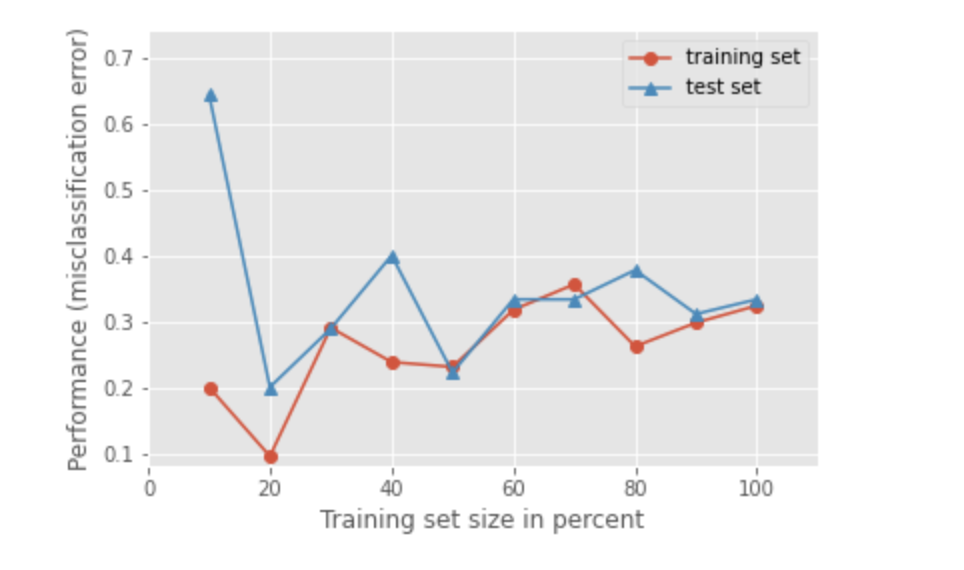
#plot learning curves

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

plt.figure()

plot\_learning\_curves(X\_train, y\_train, X\_test, y\_test, bagging1, print\_model=False, style='ggplot')

plt.show()



#Ensemble Size

num\_est = np.linspace(1,100,20).astype(int)

bg\_clf\_cv\_mean = []

bg\_clf\_cv\_std = []

for n\_est in num\_est:

bg\_clf = BaggingClassifier(base\_estimator=clf1, n\_estimators=n\_est, max\_samples=0.8, max\_features=0.8)

scores = cross\_val\_score(bg\_clf, X, y, cv=3, scoring='accuracy')

bg\_clf\_cv\_mean.append(scores.mean())

bg\_clf\_cv\_std.append(scores.std())

plt.figure()

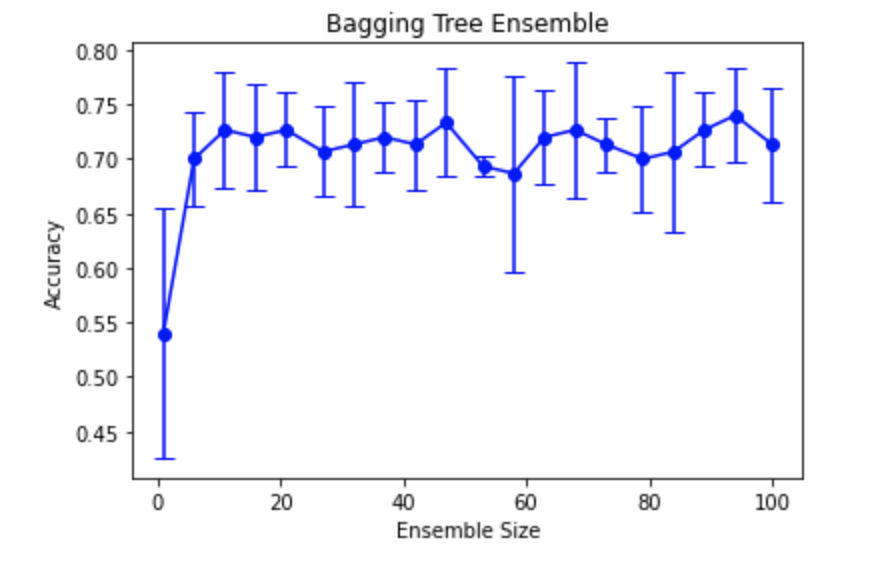
(\_, caps, \_) = plt.errorbar(num\_est, bg\_clf\_cv\_mean, yerr=bg\_clf\_cv\_std, c='blue', fmt='-o', capsize=5)

for cap in caps:

cap.set\_markeredgewidth(1)

plt.ylabel('Accuracy'); plt.xlabel('Ensemble Size'); plt.title('Bagging Tree Ensemble');

plt.show()



## Boosting

import itertools

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import AdaBoostClassifier

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from mlxtend.plotting import plot\_learning\_curves

from mlxtend.plotting import plot\_decision\_regions

iris = datasets.load\_iris()

X, y = iris.data[:, 0:2], iris.target

#XOR dataset

#X = np.random.randn(200, 2)

#y = np.array(map(int,np.logical\_xor(X[:, 0] > 0, X[:, 1] > 0)))

clf = DecisionTreeClassifier(criterion='entropy', max\_depth=1)

num\_est = [1, 2, 3, 10]

label = ['AdaBoost (n\_est=1)', 'AdaBoost (n\_est=2)', 'AdaBoost (n\_est=3)', 'AdaBoost (n\_est=10)']

fig = plt.figure(figsize=(10, 8))

gs = gridspec.GridSpec(2, 2)

grid = itertools.product([0,1],repeat=2)

for n\_est, label, grd in zip(num\_est, label, grid):

boosting = AdaBoostClassifier(base\_estimator=clf, n\_estimators=n\_est)

boosting.fit(X, y)

ax = plt.subplot(gs[grd[0], grd[1]])

fig = plot\_decision\_regions(X=X, y=y, clf=boosting, legend=2)

plt.title(label)

plt.show()

## 截屏2020-12-26 下午8.10.09

#plot learning curves

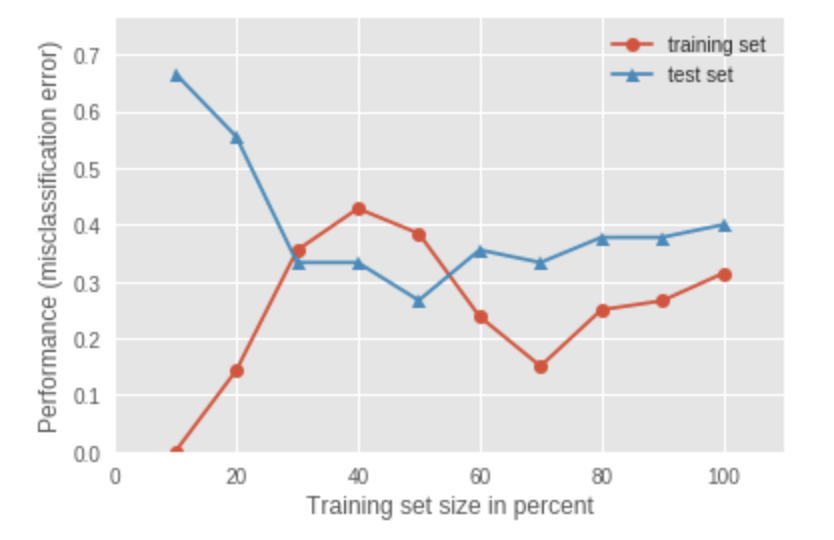
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

boosting = AdaBoostClassifier(base\_estimator=clf, n\_estimators=10)

plt.figure()

plot\_learning\_curves(X\_train, y\_train, X\_test, y\_test, boosting, print\_model=False, style='ggplot')

plt.show()



#Ensemble Size

num\_est = np.linspace(1,100,20).astype(int)

bg\_clf\_cv\_mean = []

bg\_clf\_cv\_std = []

for n\_est in num\_est:

ada\_clf = AdaBoostClassifier(base\_estimator=clf, n\_estimators=n\_est)

scores = cross\_val\_score(ada\_clf, X, y, cv=3, scoring='accuracy')

bg\_clf\_cv\_mean.append(scores.mean())

bg\_clf\_cv\_std.append(scores.std())

plt.figure()

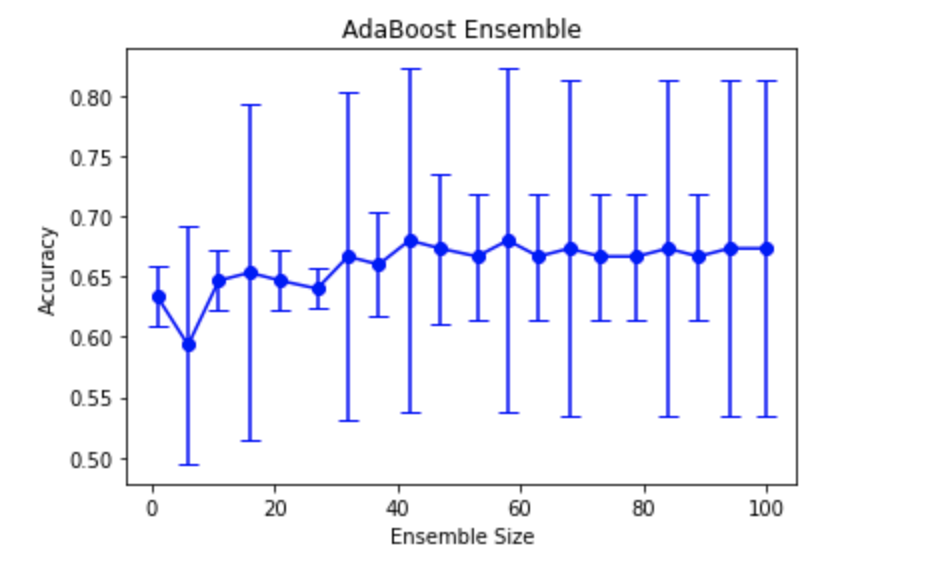
(\_, caps, \_) = plt.errorbar(num\_est, bg\_clf\_cv\_mean, yerr=bg\_clf\_cv\_std, c='blue', fmt='-o', capsize=5)

for cap in caps:

cap.set\_markeredgewidth(1)

plt.ylabel('Accuracy'); plt.xlabel('Ensemble Size'); plt.title('AdaBoost Ensemble');

plt.show()



## Stacking

import itertools

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

from sklearn import datasets

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from mlxtend.classifier import StackingClassifier

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from mlxtend.plotting import plot\_learning\_curves

from mlxtend.plotting import plot\_decision\_regions

iris = datasets.load\_iris()

X, y = iris.data[:, 1:3], iris.target

clf1 = KNeighborsClassifier(n\_neighbors=1)

clf2 = RandomForestClassifier(random\_state=1)

clf3 = GaussianNB()

lr = LogisticRegression()

sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],

meta\_classifier=lr)

label = ['KNN', 'Random Forest', 'Naive Bayes', 'Stacking Classifier']

clf\_list = [clf1, clf2, clf3, sclf]

fig = plt.figure(figsize=(10,8))

gs = gridspec.GridSpec(2, 2)

grid = itertools.product([0,1],repeat=2)

clf\_cv\_mean = []

clf\_cv\_std = []

for clf, label, grd in zip(clf\_list, label, grid):

scores = cross\_val\_score(clf, X, y, cv=3, scoring='accuracy')

print ("Accuracy: %.2f (+/- %.2f) [%s]" %(scores.mean(), scores.std(), label))

clf\_cv\_mean.append(scores.mean())

clf\_cv\_std.append(scores.std())

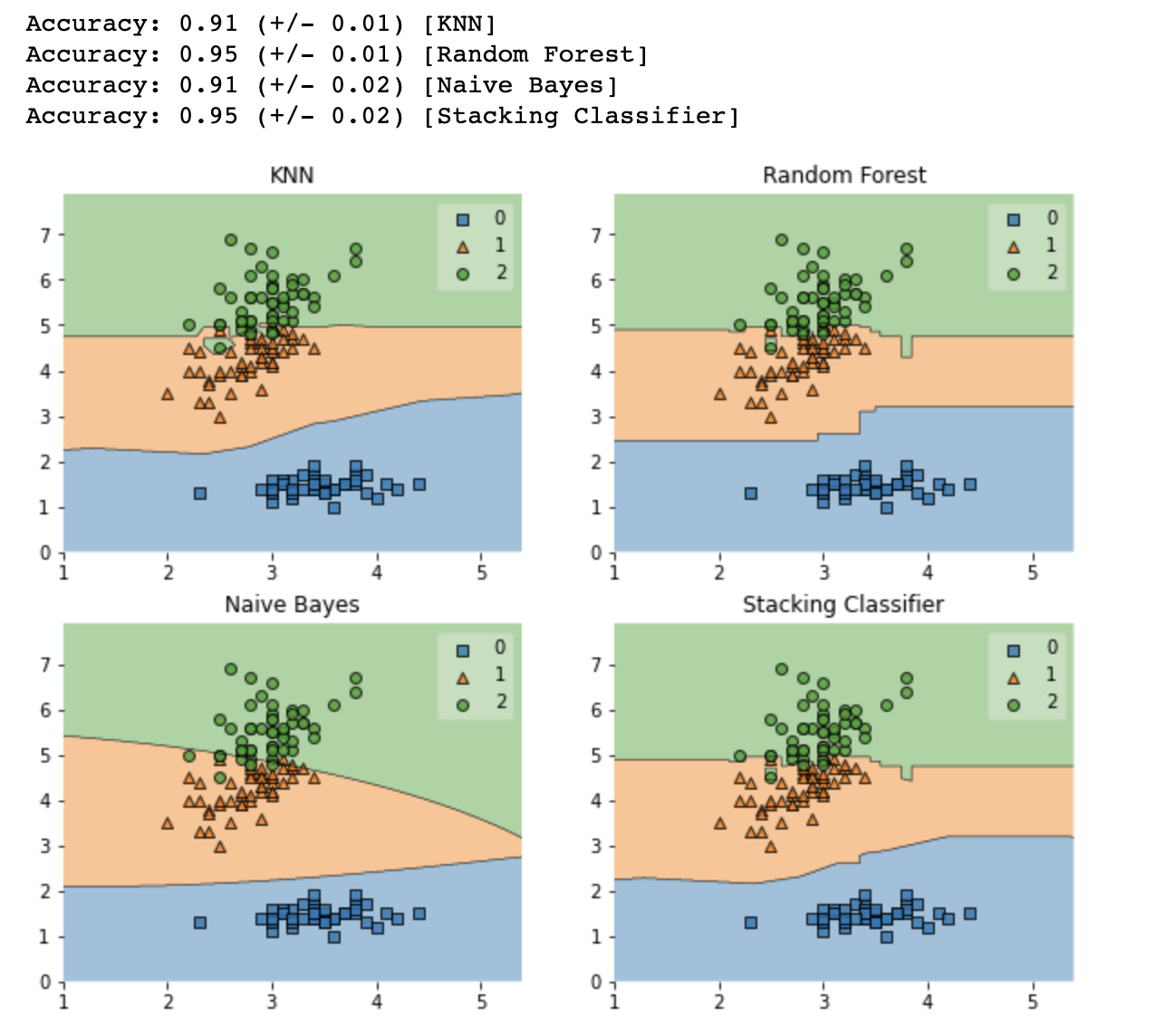
clf.fit(X, y)

ax = plt.subplot(gs[grd[0], grd[1]])

fig = plot\_decision\_regions(X=X, y=y, clf=clf)

plt.title(label)

plt.show()



plt.figure()

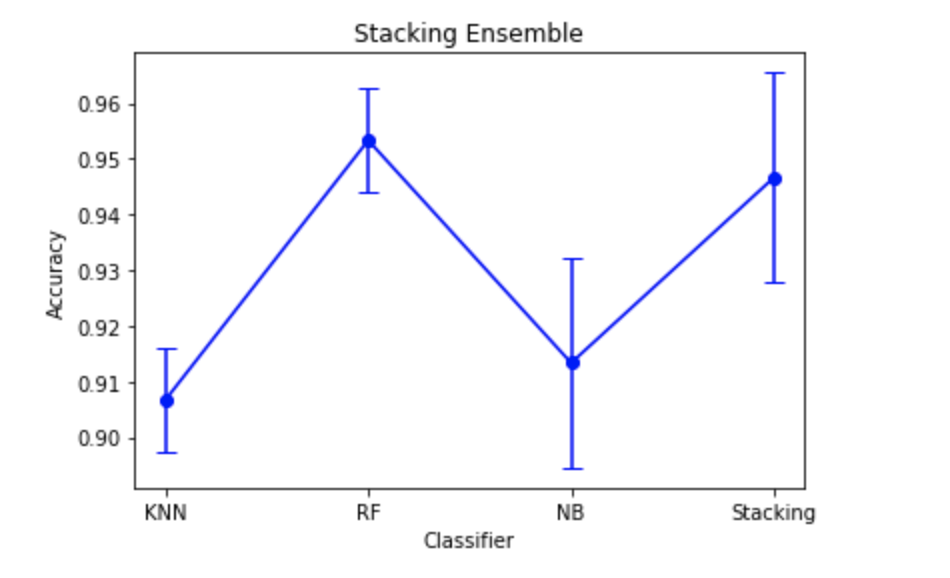
(\_, caps, \_) = plt.errorbar(range(4), clf\_cv\_mean, yerr=clf\_cv\_std, c='blue', fmt='-o', capsize=5)

for cap in caps:

cap.set\_markeredgewidth(1)

plt.xticks(range(4), ['KNN', 'RF', 'NB', 'Stacking'])

plt.ylabel('Accuracy'); plt.xlabel('Classifier'); plt.title('Stacking Ensemble');

plt.show()

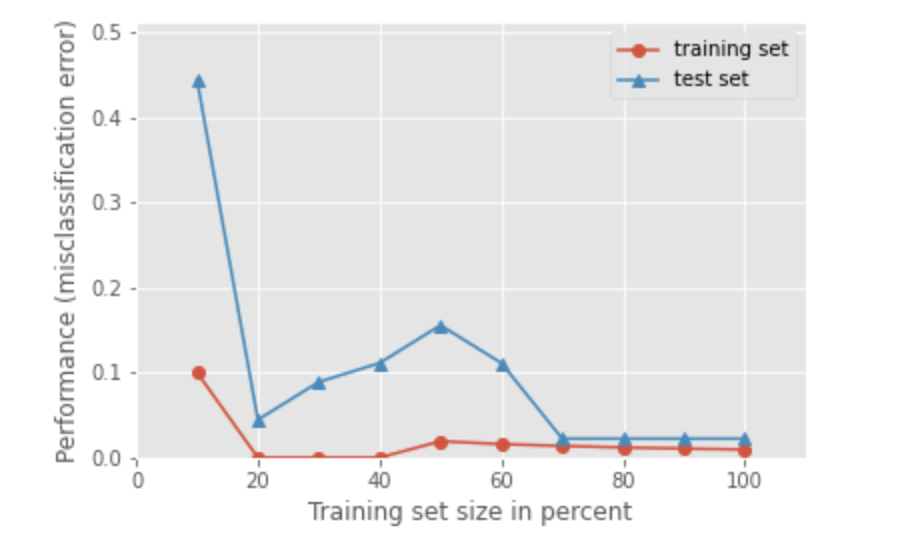
#plot learning curves

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

plt.figure()

plot\_learning\_curves(X\_train, y\_train, X\_test, y\_test, sclf, print\_model=False, style='ggplot')

plt.show()



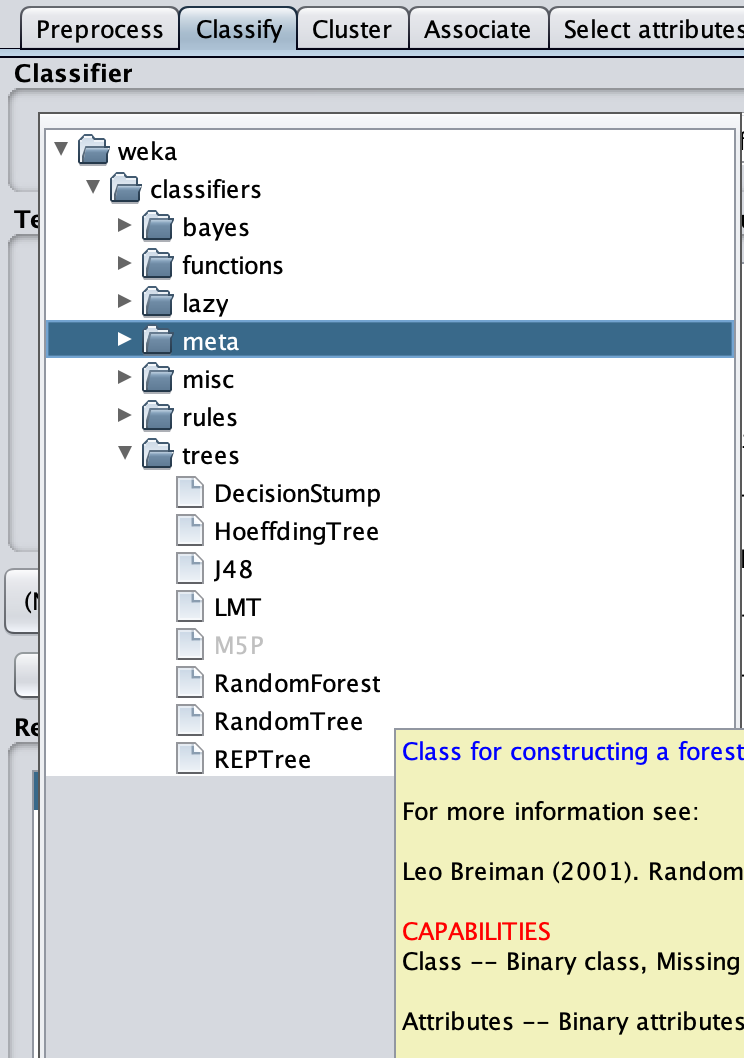
# Work on the glass.arff using RandomForest and AdaBoost algorithms in Weka.

## （1）RandomForest

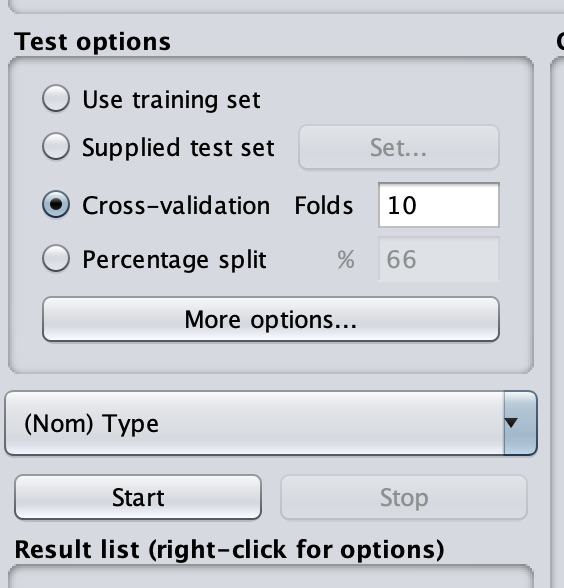
First, open the data file

## 截屏2020-12-26 下午8.33.02

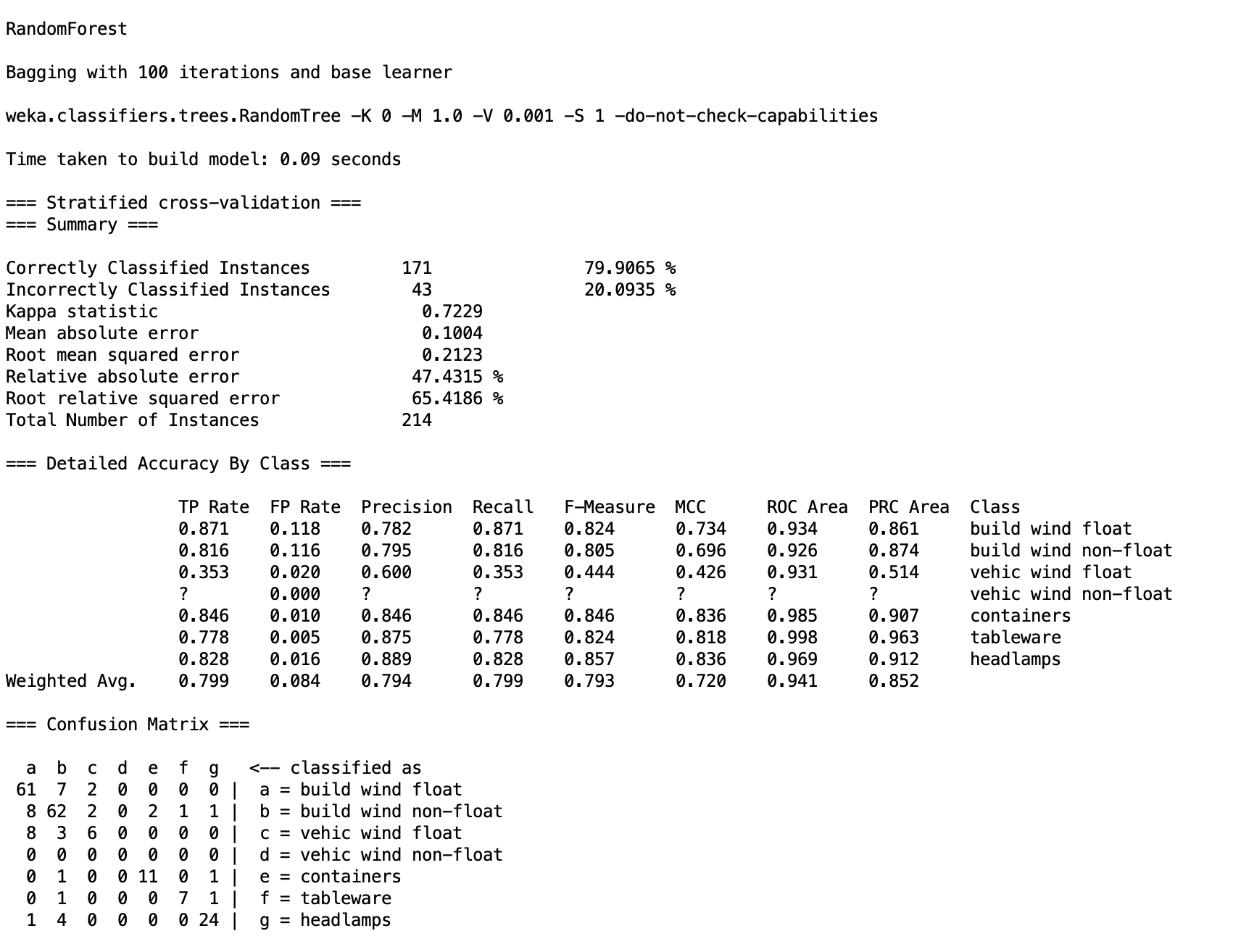
Then,choose RandomForest classifier in classify.



Click start

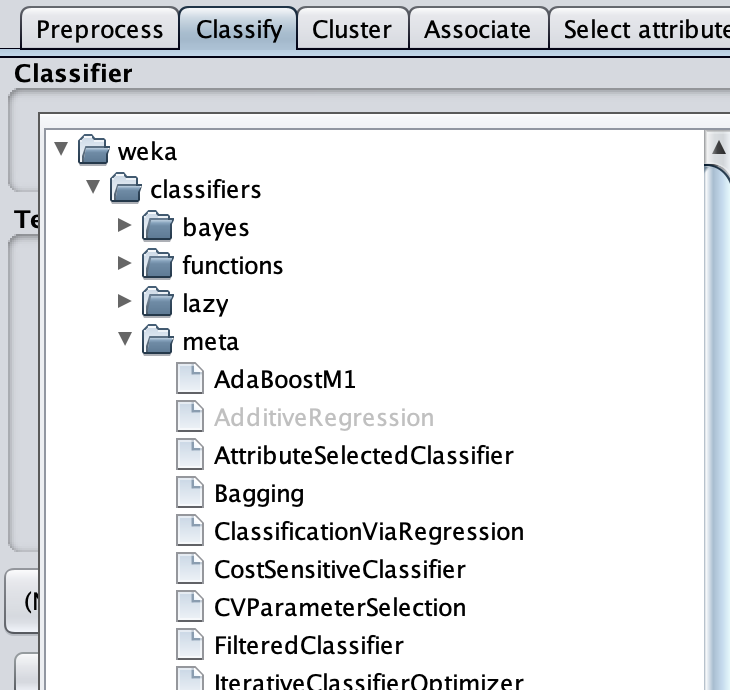


At last,we get the result.

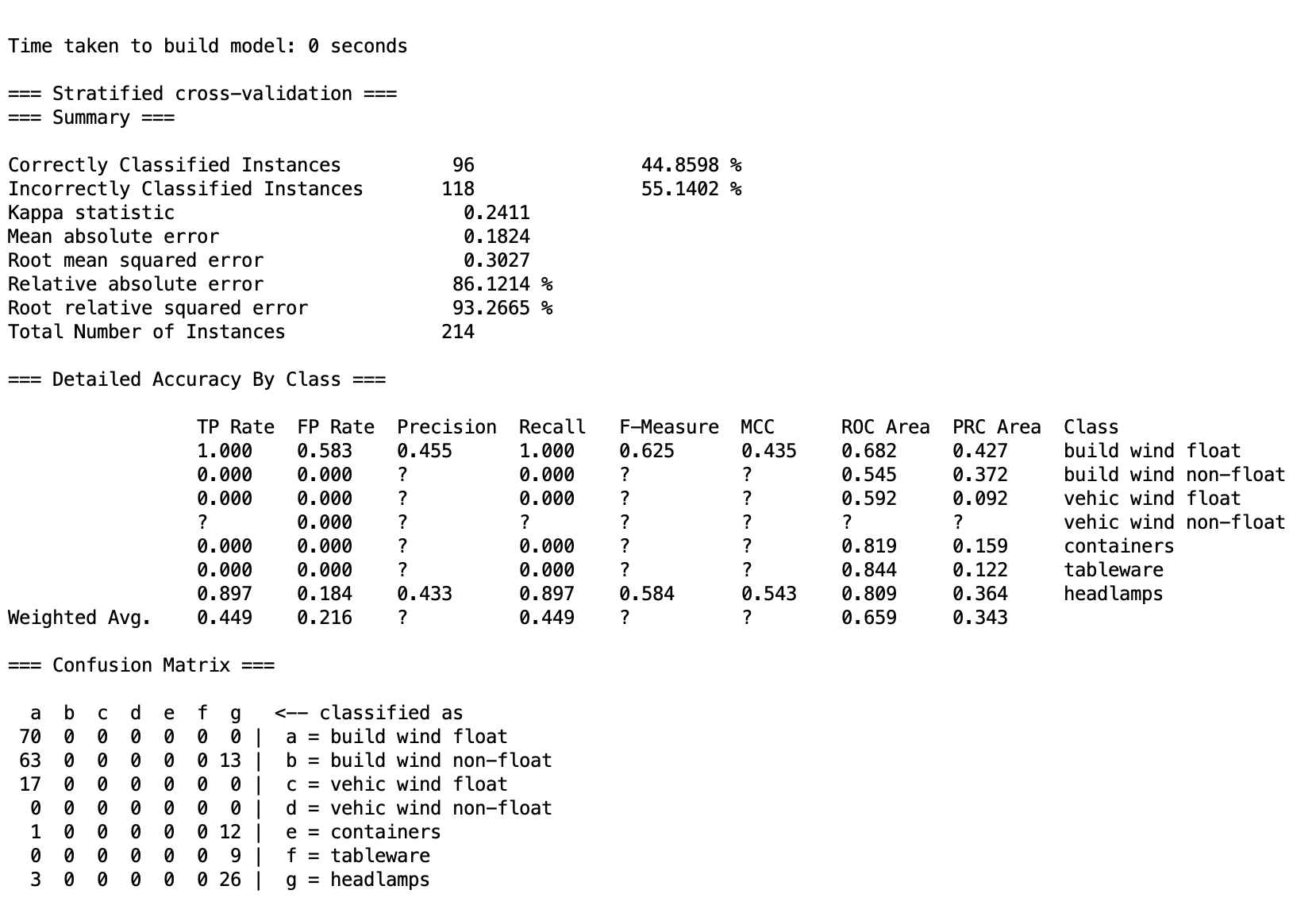


## AdaBoost

The specific step is same as RandomForest,but we should choose AdaBoostM1 classifier in classify.



At last, we get result.

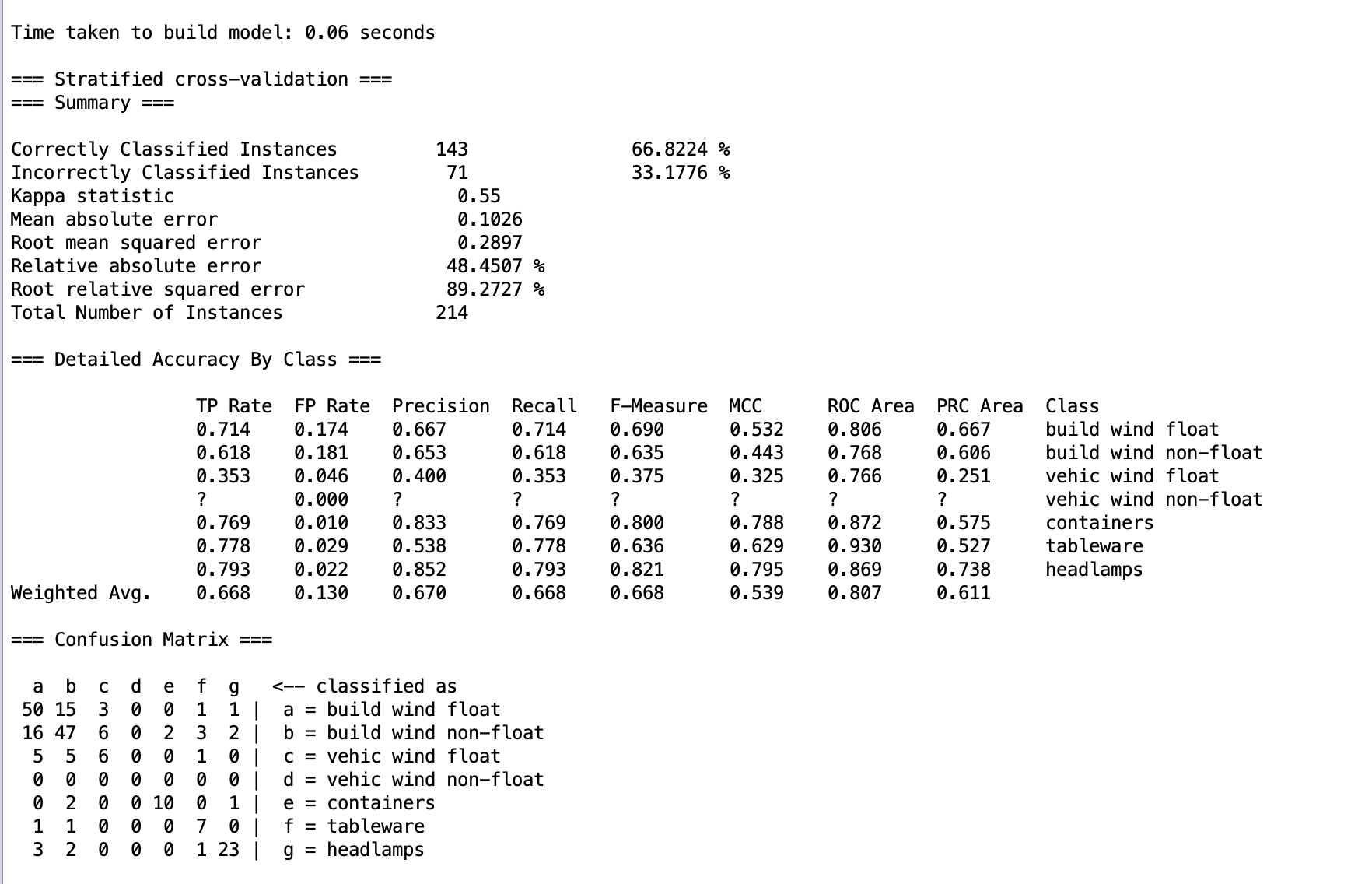


# Compare your accuracy of results with that of J48 (Decision Tree) algorithm

For J48 (Decision Tree) algorithm:

# 截屏2020-12-26 下午9.18.12

Click start,we get result.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **algorithm** | **Correctly Classified rate** | **Kappa statistic** | **Mean absolute error** | **Root mean squared error** | **Relative absolute error** | **Root relative squared error** |
| **RandomForest** | 79.9065 % | 0.7229 | 0.1004 | 0.2123 | 47.4315 % | 65.4186 % |
| **Adaboost** | 44.8598 % | 0.2411 | 0.1824 | 0.3027 | 86.1214 % | 93.2665 % |
| **J48** | 66.8224 % | 0.55 | 0.1026 | 0.2897 | 48.4507 % | 89.2727 % |

In conclusion,The accuracy rank of these algorithm here is

**RandomForest>J48>Adaboost**