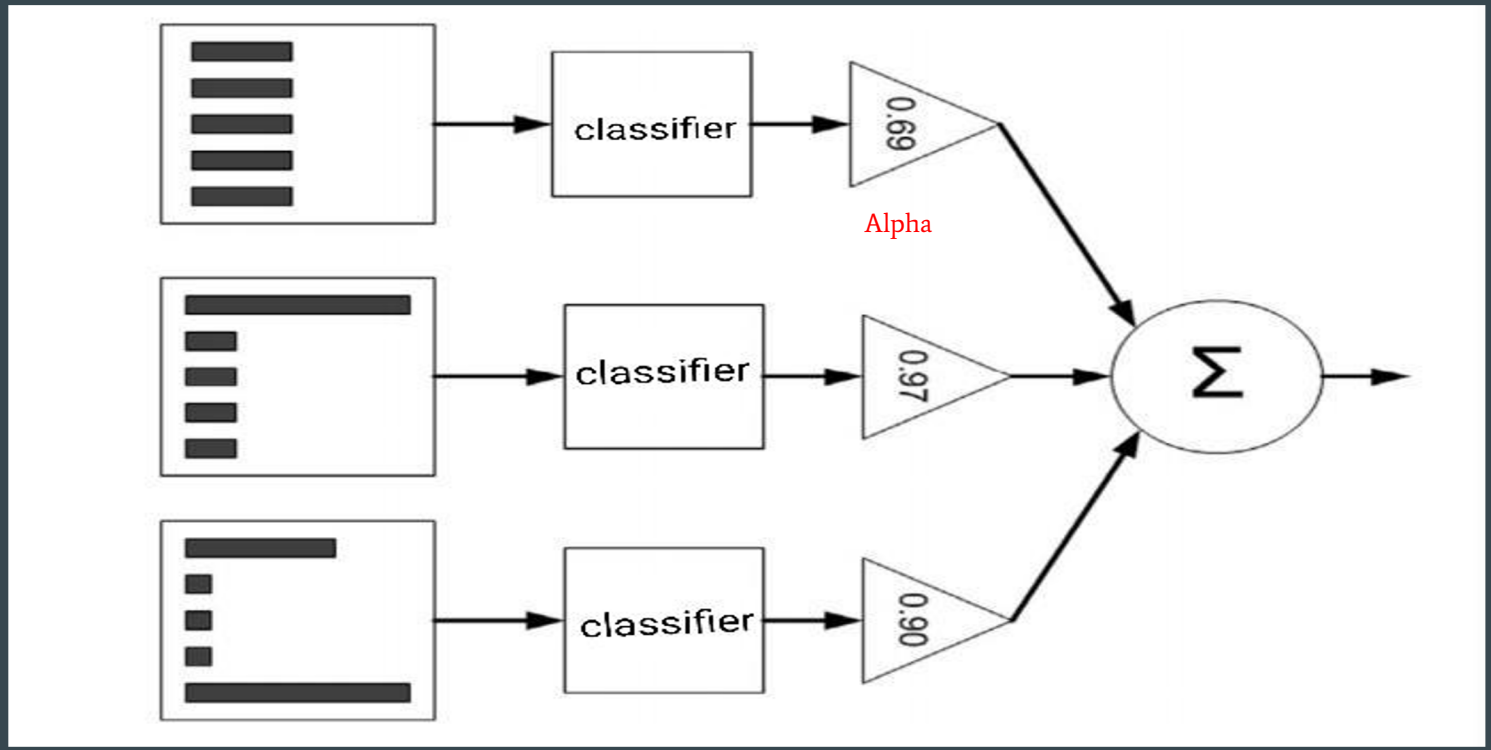
Adaboost Classifier in Python

Machine Learning Final Project
March 26th 2020
Group 4 (Wanzhou Lei, Zhuo Wu, Xinhang Liu)

Introduction of AdaBoost Model



Introduction of our code : Idea of using OOP to build models

Class StumpClassfier(num_steps=20)

Attributes:

self. alpha
self. num_steps
self._splitting_dimen
self._splitting_value
self._threshold_ineq
self. _feature_number

Member Functions:

fit(self,X,y,weights=[],check_data=True)
predict(self,X,check_data=True)
score(self,X,y)
update_alpha(self,learning_rate):
__str__(self):

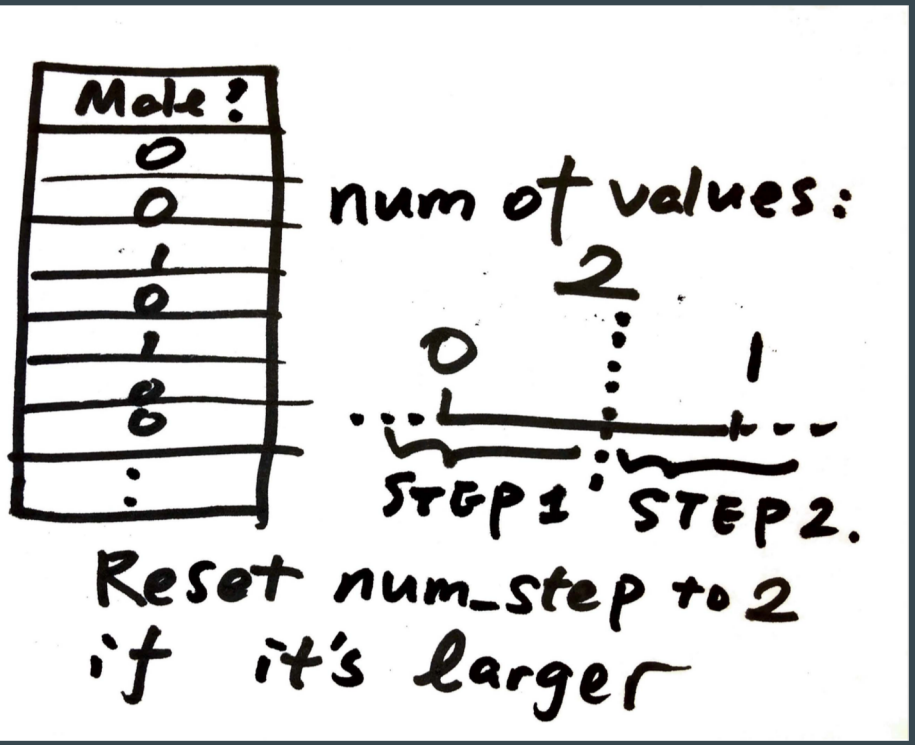
Class AdaBoostClassfier(n_estimators=50, learning_rate=1.0,num_steps=20):

Attributes:

self._n_estimators
self._learning_rate
self._num_steps
self._all_stumps
self._feature_number

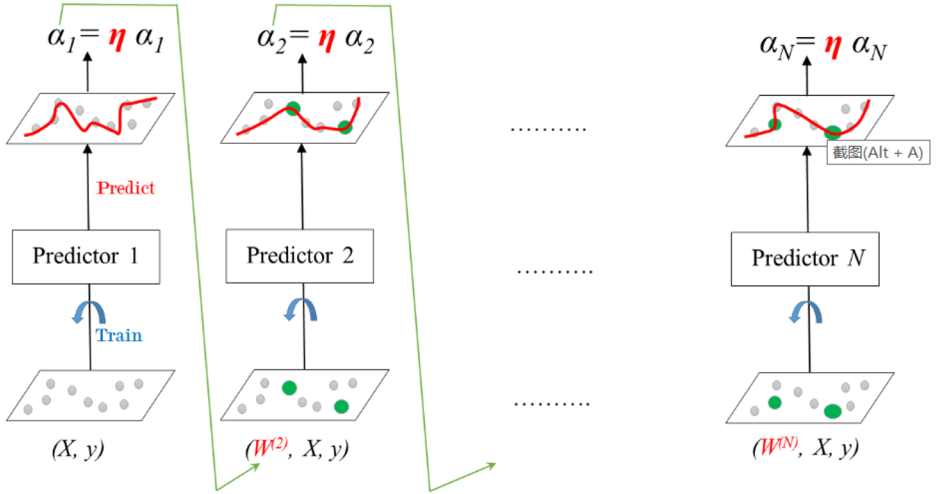
Member Functions:

fit(self,X,y,check_data=True)
predict(self,X,check_data=True)
score(self,X,y):
get_stumps(self)
__str__(self)


Male?
0
0
1
0
1
0
0
num of values:
2
0
1
STEP 1
STEP 2.
Reset num_step to 2
if it's larger

Learning Rate

Learning rate: $0 < \eta \leq 1$



Data Processing and Analyzing: Iris Dataset

4 Features

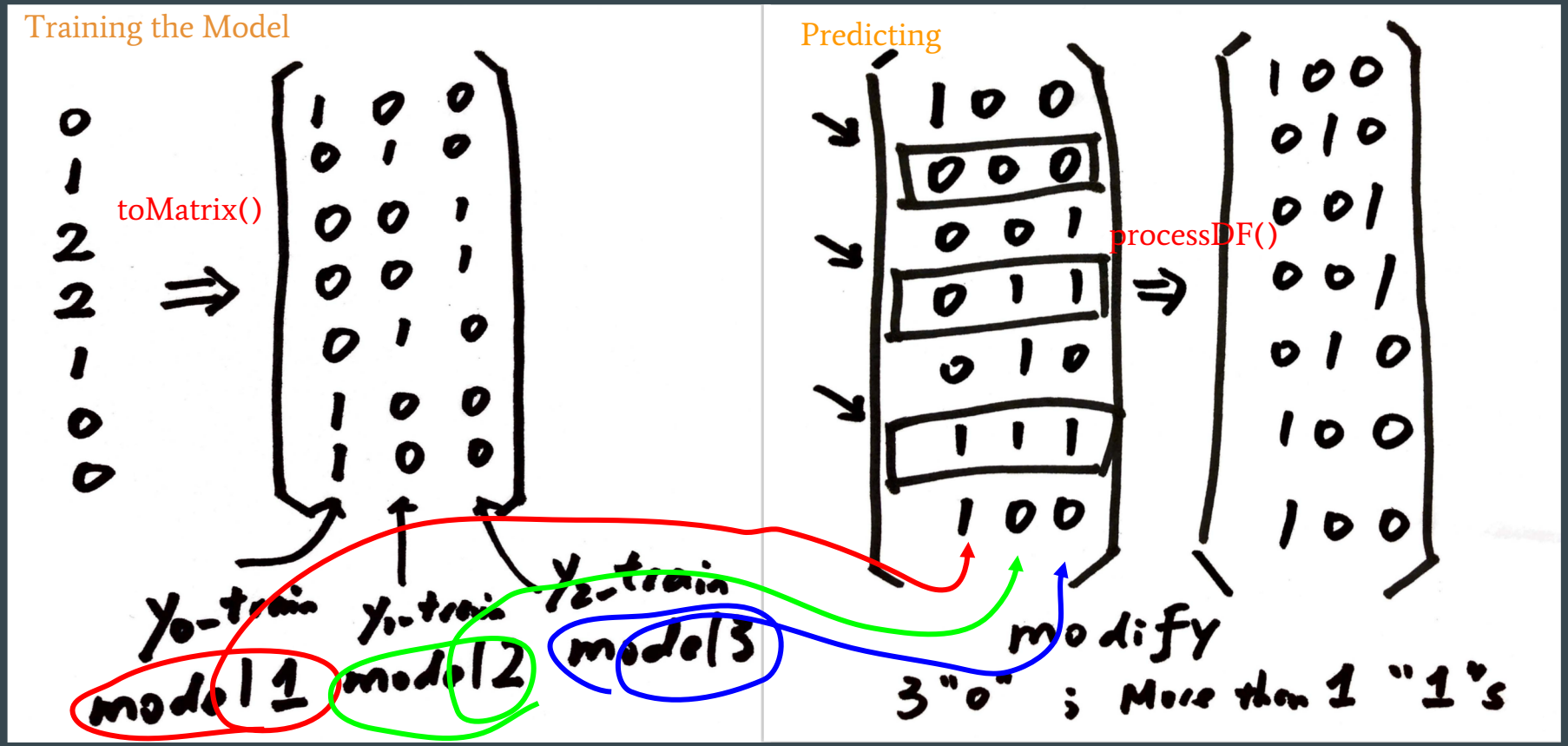
150 Observations
(Small Dataset)

	sepal_length	sepal_width	petal_length	petal_width	Real_type
0	4.5	2.3	1.3	0.3	0
1	5.7	2.8	4.1	1.3	1
2	5.8	2.6	4.0	1.2	1
3	6.5	3.0	5.8	2.2	2
4	6.2	2.9	4.3	1.3	1
5	5.7	4.4	1.5	0.4	0
6	5.6	2.7	4.2	1.3	1
7	6.5	3.2	5.1	2.0	2
8	6.7	3.1	4.4	1.4	1
9	5.4	3.7	1.5	0.2	0
10	7.4	2.8	6.1	1.9	2
11	5.9	3.0	5.1	1.8	2
12	5.1	3.5	1.4	0.2	0
13	4.4	2.9	1.4	0.2	0
14	5.0	2.3	3.3	1.0	1
15	6.1	2.9	4.7	1.4	1

Multi Class

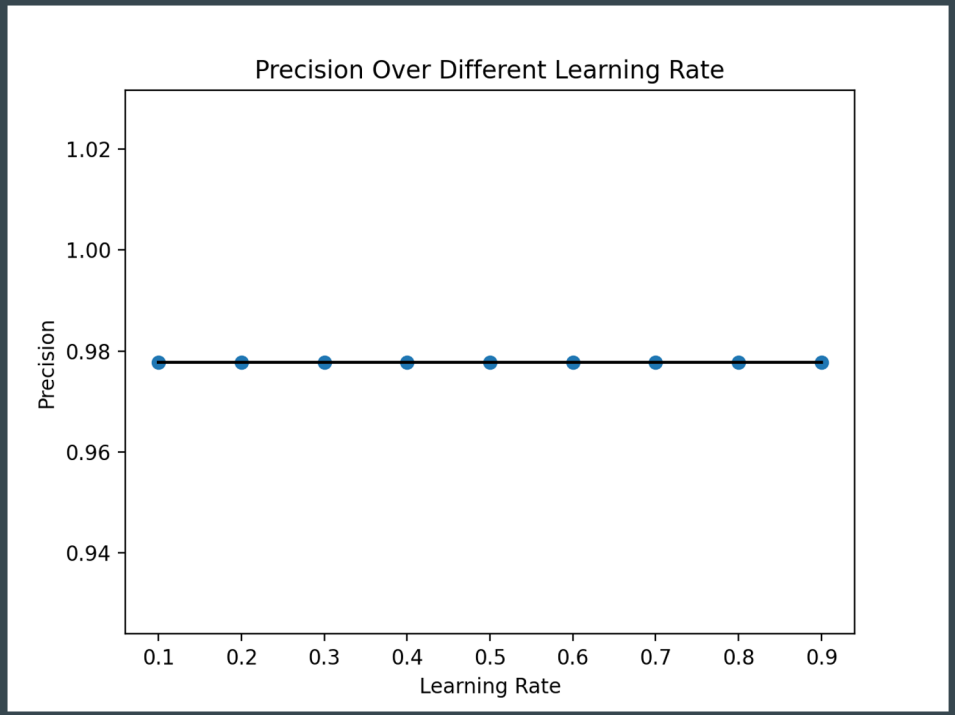
Modify Adaboost to Make Multi-class Classification Possible !!

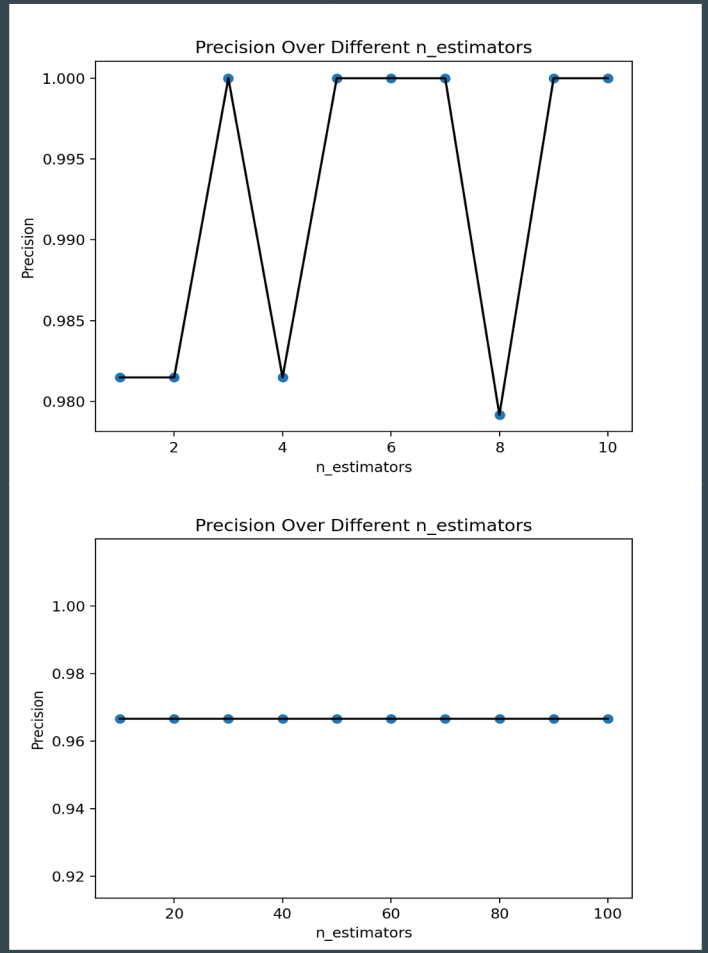
Binary⟶ Multi class Classification: One vs All



Tuning Hyper-Parameter & Result (Iris Dataset)

Hyper-Parameter Visualization





Iris setosa

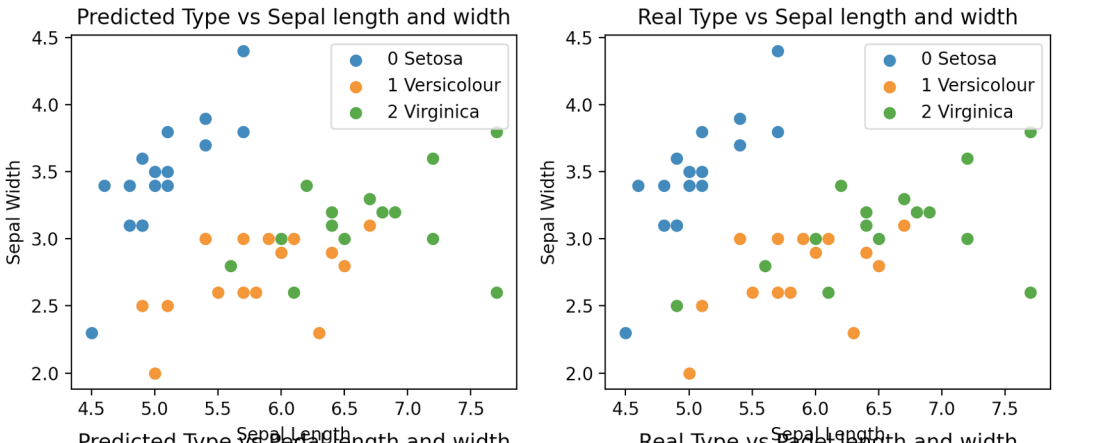
Class 0

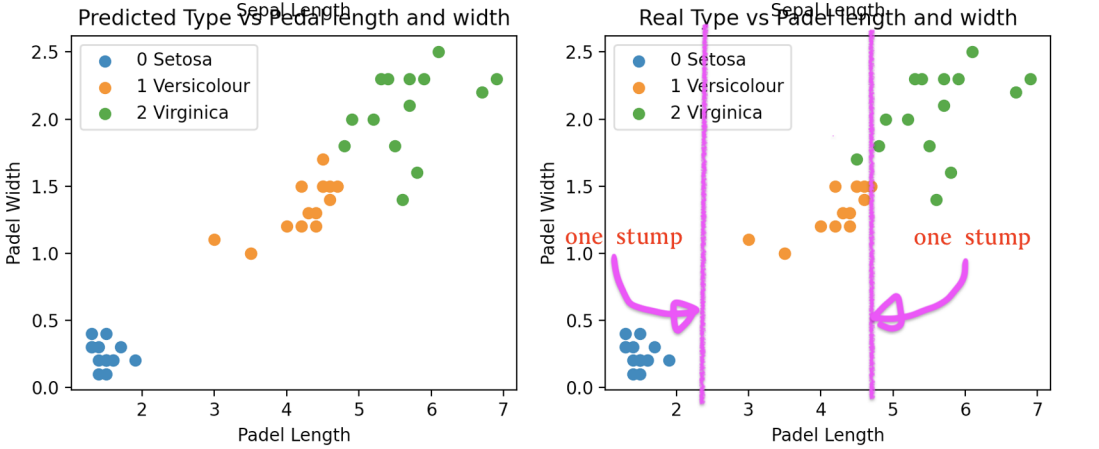
Iris versicolour

Class 1

Iris virginia

Class 2






```
Average Precision: 0.9666666666666667
Average Recall Rate: 0.9487179487179488
```

Titanic Dataset

Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	3	Braund, Mr.	male	22	1	0	A/5 21171	7.25
1	1	Cumings, M	female	38	1	0	PC 17599	71.2833
1	3	Heikkinen, M	female	26	0	0	STON/O2. 3	7.925
1	1	Futrelle, Mrs	female	35	1	0	113803	53.1
0	3	Allen, Mr. W	male	35	0	0	373450	8.05
0	3	Moran, Mr. J	male		0	0	330877	8.4583
0	1	McCarthy, M	male	54	0	0	17463	51.8625
0	3	Palsson, Ma	male	2	3	1	349909	21.075
1	3	Johnson, Mi	female	27	0	2	347742	11.1333
1	2	Nasser, Mrs.	female	14	1	0	237736	30.0708
1	3	Sandstrom,	female	4	1	1	PP 9549	16.7
1	1	Bonnell, Mis	female	58	0	0	113783	26.55
0	3	Saundercock	male	20	0	0	A/5. 2151	8.05
0	3	Andersson,	male	39	1	5	347082	31.275
0	3	Vestrom, Mi	female	14	0	0	350406	7.8542
1	2	Hewlett, Mrs	female	55	0	0	248706	16
0	3	Rice, Master	male	2	4	1	382652	29.125
1	2	Williams, Mi	male		0	0	244373	13
0	3	Vander Plan	female	31	1	0	345763	18
1	3	Masselmani	female		0	0	2649	7.225
0	2	Fynney, Mr.	male	35	0	0	239865	26
1	2	Beesley, Mr.	male	34	0	0	248698	13

6 fetures
892 0bservations

1. Some of the age data are lost.

Calculate the mean age and fill these lost ages with the mean age.

2. Dataset is unbalanced
342 of 892 people(only 38%) survived.

Problem: If use ordinary k fold CV. In each fold the proportion of survived would be different (some higher like 50%, soime lower like 10%)

Fix: Stratified k fold Cross Validation.

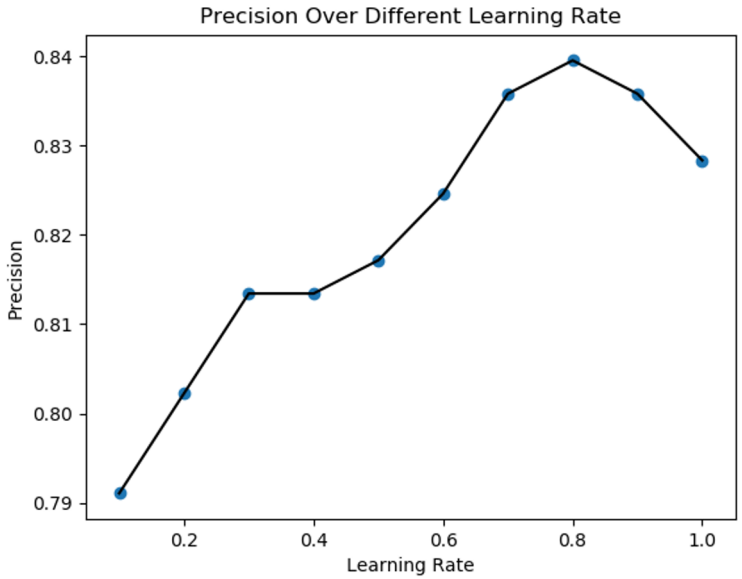
Stratified k-fold cross validation

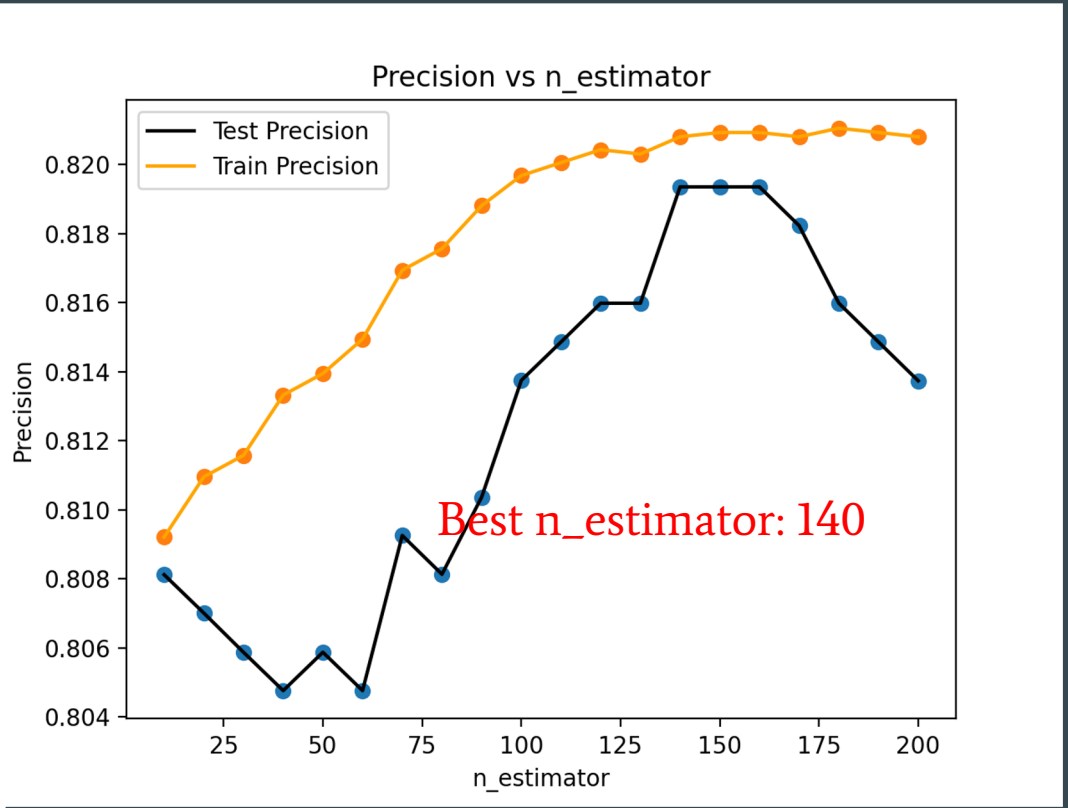
Idea: keep the proportion of survived people in each fold to be the same (38%)

Stratified CV of Titanic Dataset

Note that: two hyper-parameters learning rate and n_estimnator are interrelated. There is offset between decreasing learning rate and increasing n_estimators

For convenience, I only tune the n_estimator hyper-parameter.





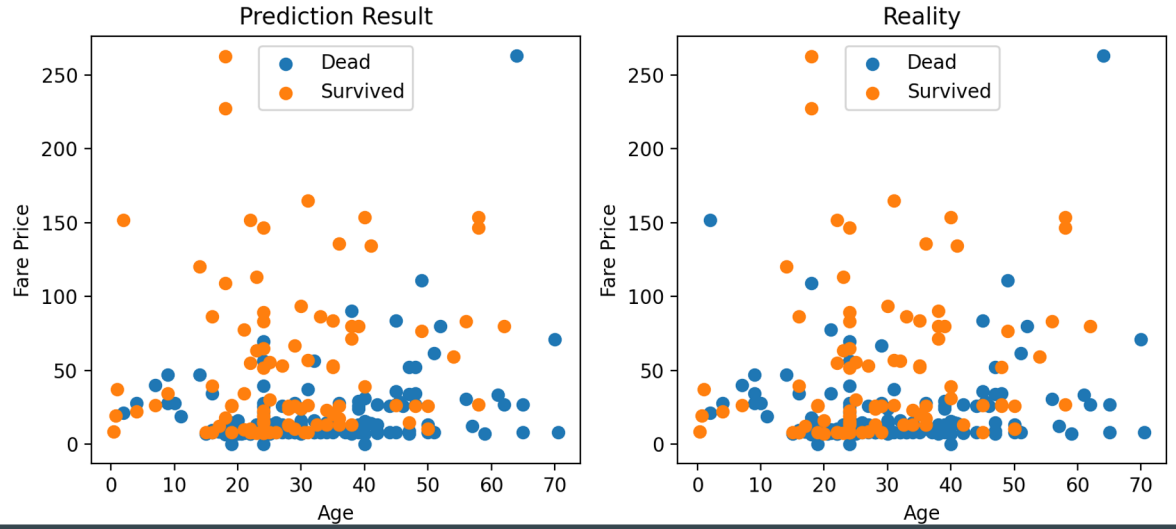
Learning rate=1

Result of Titanic Dataset

Set the learning rate to be 1.
n _estimator to be 140.
Fit the train set and predict on the test set.

	Pclass	Sex	Age	SibSp	Parch	Fare	Survived	Prediction
667	3	1	24.00	0	0	7.7750	0	0.0
425	3	1	24.00	0	0	7.2500	0	0.0
265	2	1	36.00	0	0	10.5000	0	0.0
635	2	0	28.00	0	0	13.0000	1	1.0
803	3	1	0.42	0	1	8.5167	1	1.0
..	...	...	...	...	...	...	...	...
302	3	1	19.00	0	0	0.0000	0	0.0
615	2	0	24.00	1	2	65.0000	1	1.0
259	2	0	50.00	0	1	26.0000	1	1.0
488	3	1	30.00	0	0	8.0500	0	0.0
546	2	0	19.00	1	0	26.0000	1	1.0

[268 rows x 8 columns]
Accuracy: 0.8582089552238806



Pretty decent result:

Accuracy: 0.858

Conclusion

Pros: Adaboost classifier is very useful when processing data that need to do the multi-classifier, and need to fill the lost data. It can make the prediction more accurate by weighting the result with different alpha.

Cons:

1: Hard to do parallel computation (multi-core computation).

2: It is very sensitive to the noise data (decreasing learning rate or decreasing n_estimator can help prevent overfitting).