團隊測驗報告

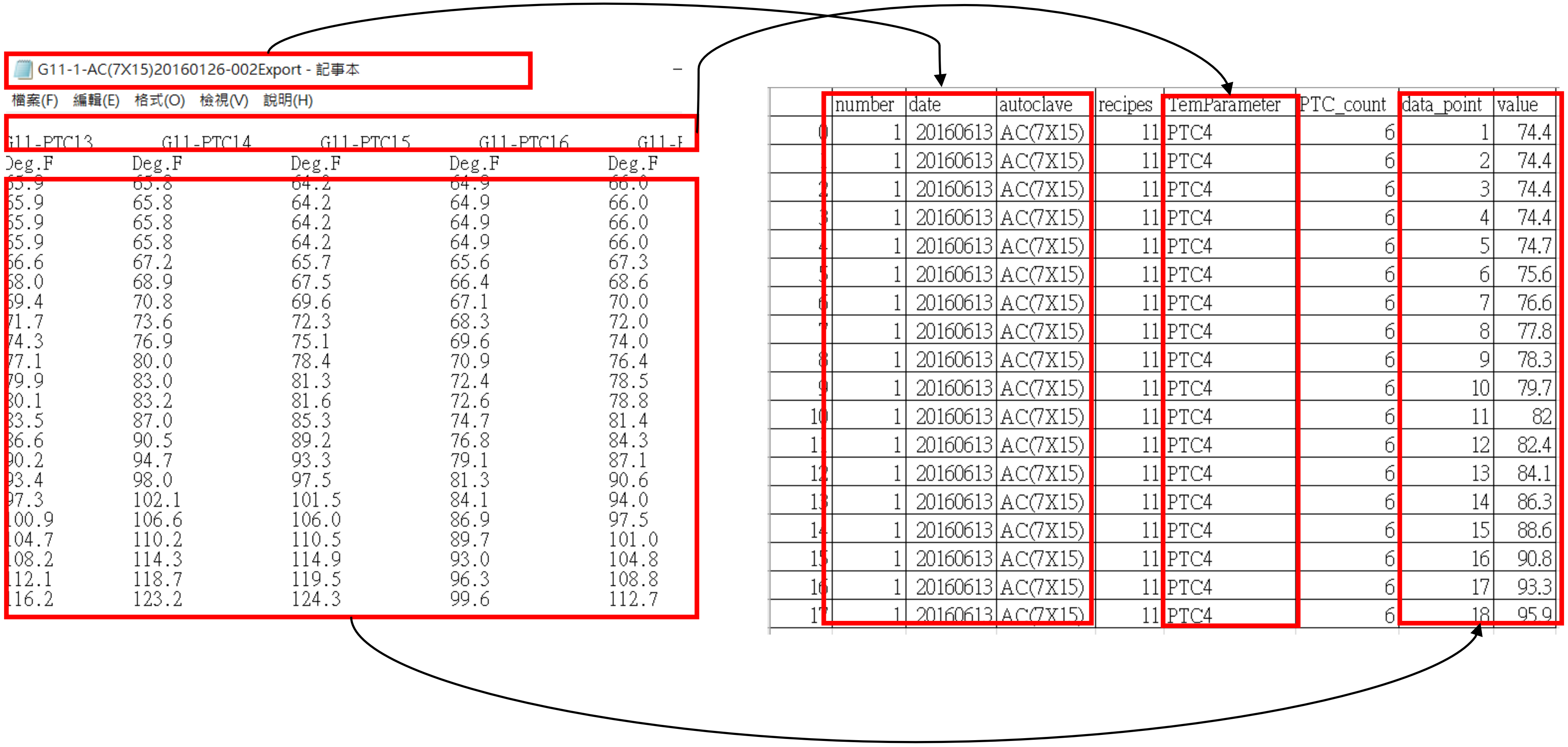
報名序號：108164 (報名序號(格式:108XXX)已寄至隊長email)

團隊名稱：混沌

註1：請用本PowerPoint 文件撰寫團隊程式說明，請轉成PDF檔案繳交。
註2：依據競賽須知第七條，第4項規定：

「測試報告之簡報資料不得出現學校系所標誌、提及學校系所、教授姓名及任何可供辨識參賽團隊組織或個人身分的資料或資訊，違者取消參賽資格或由評審會議決議處理方式。」

一、資料前處理(1)-格式轉化

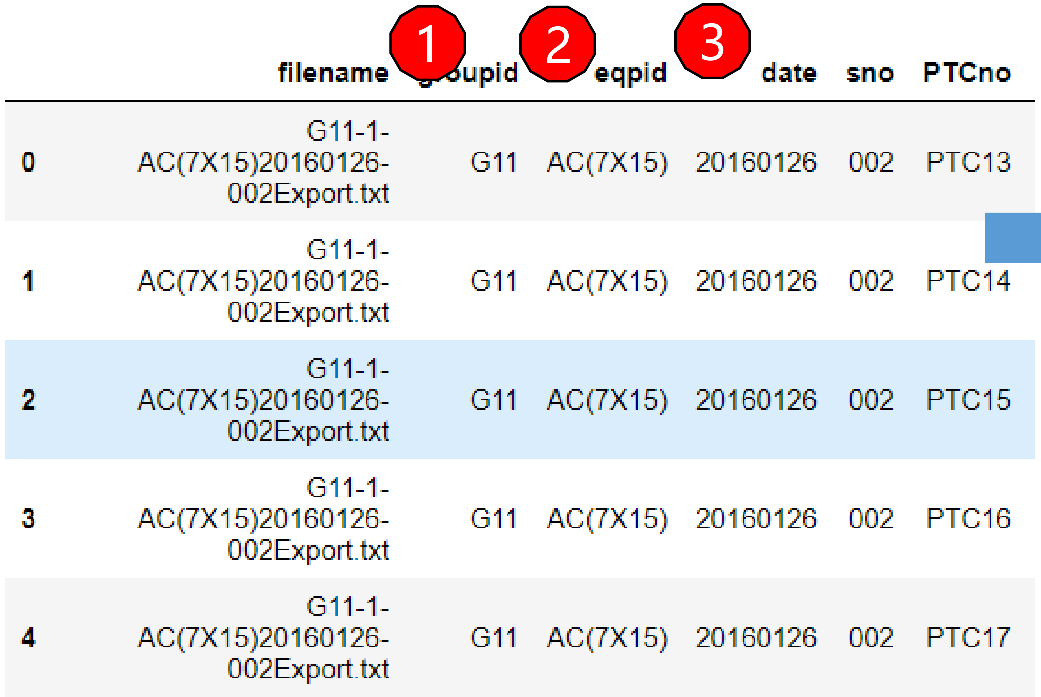


G11-1-AC(7X15)20160126-002Export - 記事本

```
i11-PTC13          G11-PTC14          G11-PTC15          G11-PTC16          G11-F
Deg.F              Deg.F              Deg.F              Deg.F              Deg.F
55.9               65.8               64.2               64.9               66.0
55.9               65.8               64.2               64.9               66.0
55.9               65.8               64.2               64.9               66.0
55.9               65.8               64.2               64.9               66.0
56.6               67.2               65.7               65.6               67.3
58.0               68.9               67.5               66.4               68.6
59.4               70.8               69.6               67.1               70.0
71.7               73.6               72.3               68.3               72.0
74.3               76.9               75.1               69.6               74.0
77.1               80.0               78.4               70.9               76.4
79.9               83.0               81.3               72.4               78.5
30.1               83.2               81.6               72.6               78.8
33.5               87.0               85.3               74.7               81.4
36.6               90.5               89.2               76.8               84.3
90.2               94.7               93.3               79.1               87.1
93.4               98.0               97.5               81.3               90.6
97.3               102.1              101.5              84.1               94.0
100.9              106.6              106.0              86.9               97.5
104.7              110.2              110.5              89.7               101.0
108.2              114.3              114.9              93.0               104.8
112.1              118.7              119.5              96.3               108.8
116.2              123.2              124.3              99.6               112.7
```

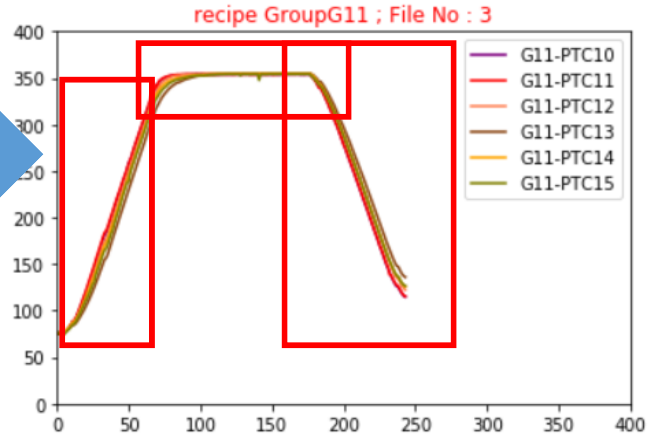
	number	date	autoclave	recipes	TemParameter	PTC_count	data_point	value
0	1	20160613	AC(7X15)	11	PTC4	6	1	74.4
1	1	20160613	AC(7X15)	11	PTC4	6	2	74.4
2	1	20160613	AC(7X15)	11	PTC4	6	3	74.4
3	1	20160613	AC(7X15)	11	PTC4	6	4	74.4
4	1	20160613	AC(7X15)	11	PTC4	6	5	74.7
5	1	20160613	AC(7X15)	11	PTC4	6	6	75.6
6	1	20160613	AC(7X15)	11	PTC4	6	7	76.6
7	1	20160613	AC(7X15)	11	PTC4	6	8	77.8
8	1	20160613	AC(7X15)	11	PTC4	6	9	78.3
9	1	20160613	AC(7X15)	11	PTC4	6	10	79.7
10	1	20160613	AC(7X15)	11	PTC4	6	11	82
11	1	20160613	AC(7X15)	11	PTC4	6	12	82.4
12	1	20160613	AC(7X15)	11	PTC4	6	13	84.1
13	1	20160613	AC(7X15)	11	PTC4	6	14	86.3
14	1	20160613	AC(7X15)	11	PTC4	6	15	88.6
15	1	20160613	AC(7X15)	11	PTC4	6	16	90.8
16	1	20160613	AC(7X15)	11	PTC4	6	17	93.3
17	1	20160613	AC(7X15)	11	PTC4	6	18	95.9

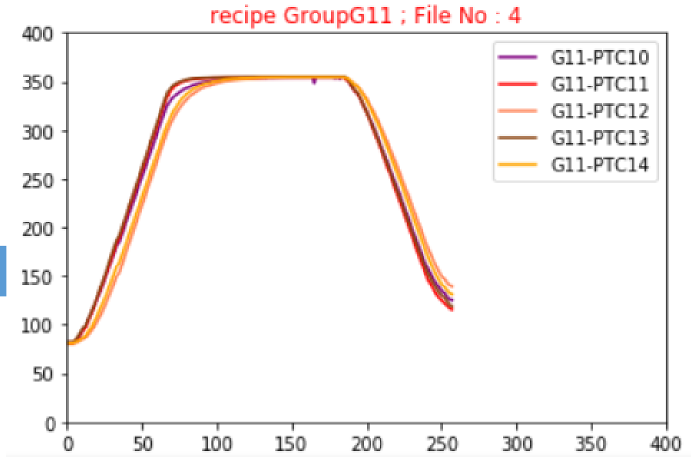
一、資料前處理(2) -特徵提取



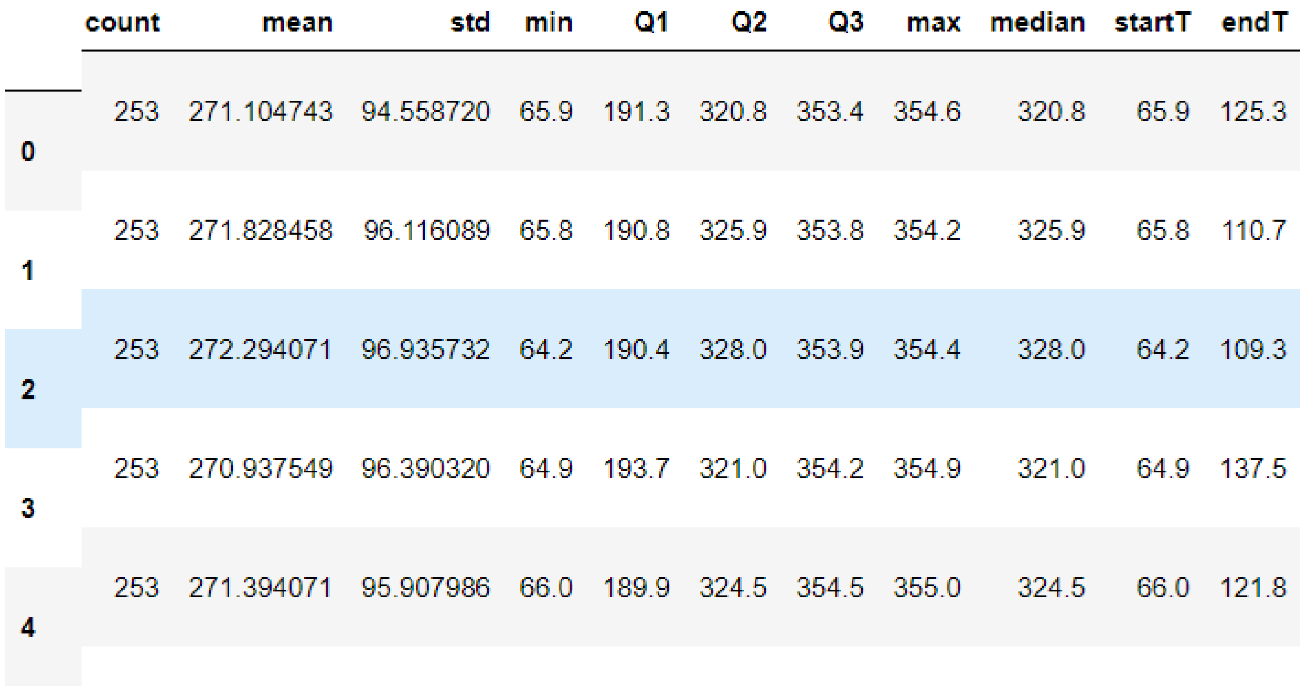
1 2 3

	filename	groupid	eqpid	date	sno	PTCno
0	G11-1-AC(7X15)20160126-002Export.txt	G11	AC(7X15)	20160126	002	PTC13
1	G11-1-AC(7X15)20160126-002Export.txt	G11	AC(7X15)	20160126	002	PTC14
2	G11-1-AC(7X15)20160126-002Export.txt	G11	AC(7X15)	20160126	002	PTC15
3	G11-1-AC(7X15)20160126-002Export.txt	G11	AC(7X15)	20160126	002	PTC16
4	G11-1-AC(7X15)20160126-002Export.txt	G11	AC(7X15)	20160126	002	PTC17



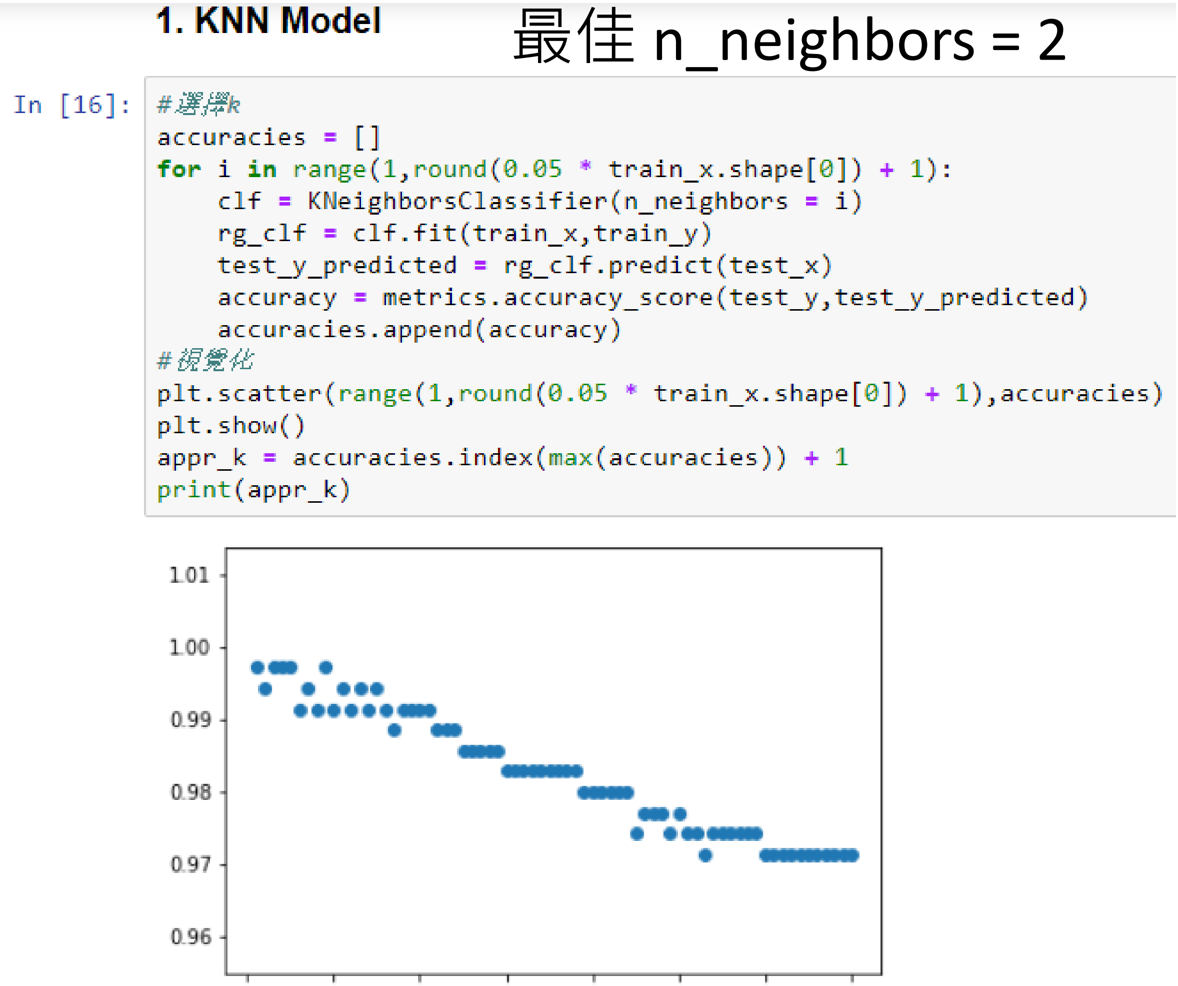


特徵提取



	count	mean	std	min	Q1	Q2	Q3	max	median	startT	endT
0	253	271.104743	94.558720	65.9	191.3	320.8	353.4	354.6	320.8	65.9	125.3
1	253	271.828458	96.116089	65.8	190.8	325.9	353.8	354.2	325.9	65.8	110.7
2	253	272.294071	96.935732	64.2	190.4	328.0	353.9	354.4	328.0	64.2	109.3
3	253	270.937549	96.390320	64.9	193.7	321.0	354.2	354.9	321.0	64.9	137.5
4	253	271.394071	95.907986	66.0	189.9	324.5	354.5	355.0	324.5	66.0	121.8

二、演算法和模型介紹-KNN(1)



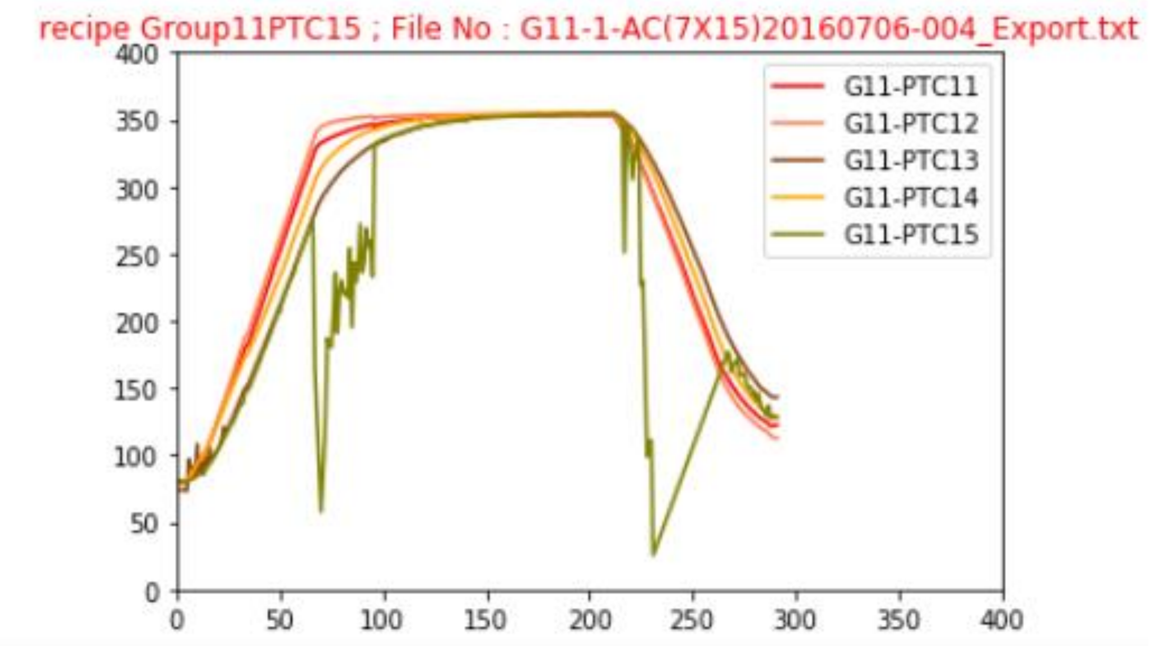
1. KNN Model

最佳 n_neighbors = 2

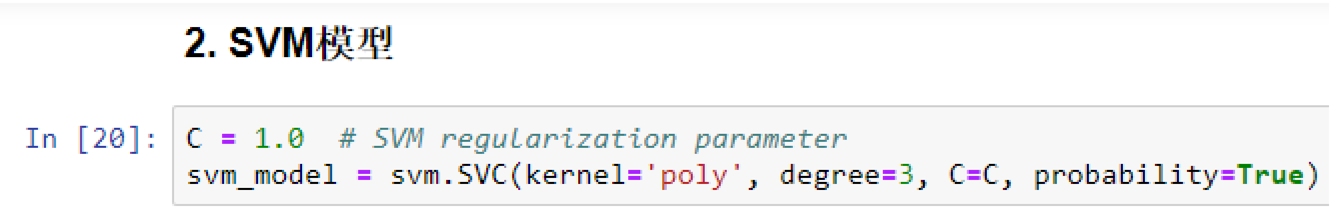
In [16]:

```
#選擇k
accuracies = []
for i in range(1,round(0.05 * train_x.shape[0]) + 1):
    clf = KNeighborsClassifier(n_neighbors = i)
    rg_clf = clf.fit(train_x,train_y)
    test_y_predicted = rg_clf.predict(test_x)
    accuracy = metrics.accuracy_score(test_y,test_y_predicted)
    accuracies.append(accuracy)
#視覺化
plt.scatter(range(1,round(0.05 * train_x.shape[0]) + 1),accuracies)
plt.show()
appr_k = accuracies.index(max(accuracies)) + 1
print(appr_k)
```

分錯一個-將G11分為 G49



二、演算法和模型介紹-SVC(2)

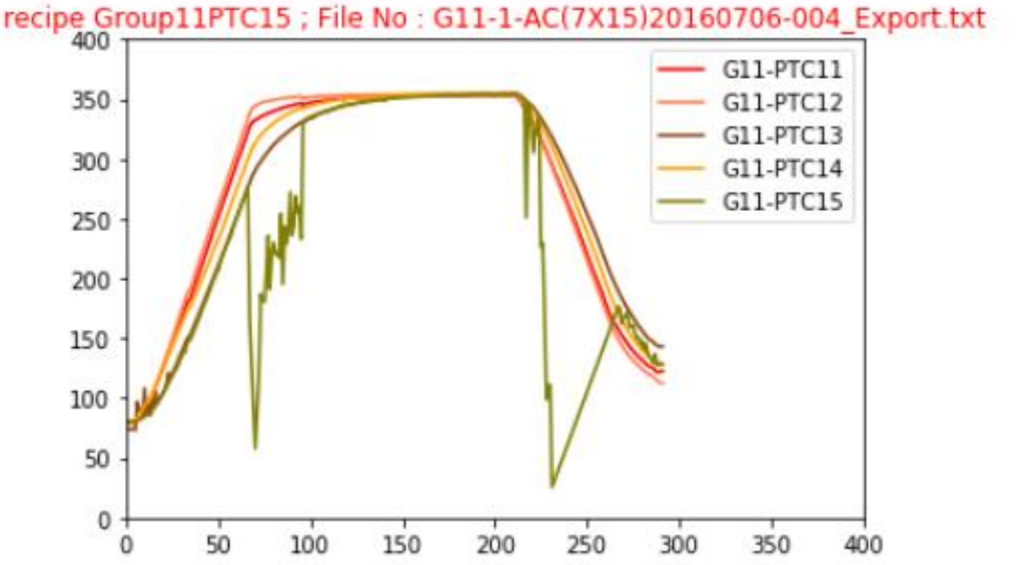


2. SVM模型

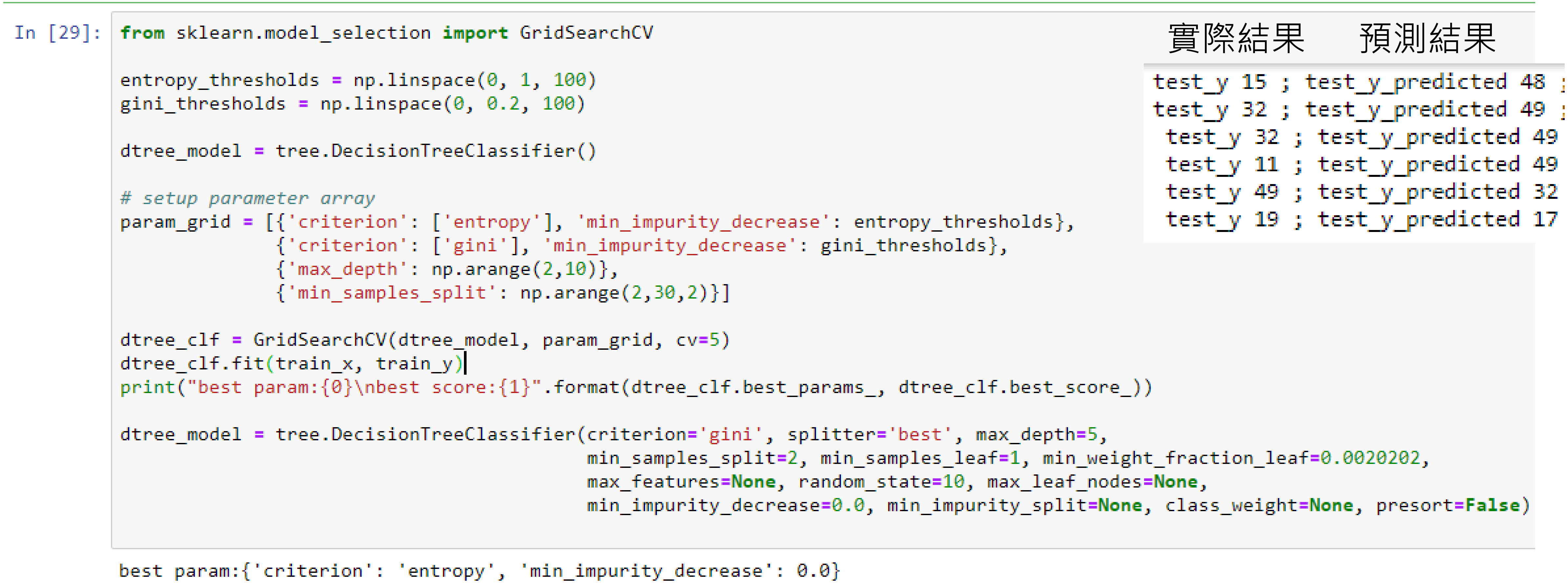
```
In [20]: C = 1.0  # SVM regularization parameter
         svm_model = svm.SVC(kernel='poly', degree=3, C=C, probability=True)
```

參數調優:
最佳Kernel = poly
Degree = 3
C = 1

分錯一個-將G11分為 G32



二、演算法和模型介紹-Decision Tree(3)



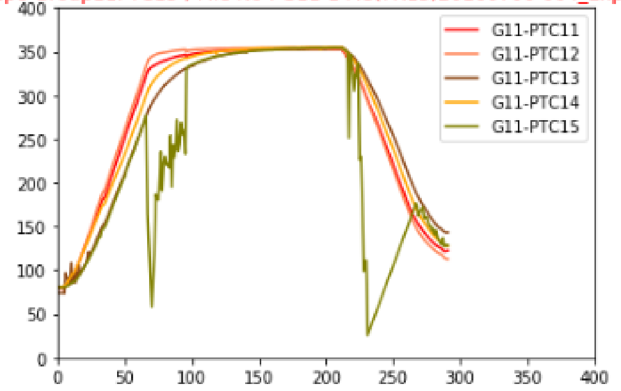
```
In [29]: from sklearn.model_selection import GridSearchCV

         entropy_thresholds = np.linspace(0, 1, 100)
         gini_thresholds = np.linspace(0, 0.2, 100)

         dtree_model = tree.DecisionTreeClassifier()

         # setup parameter array
         param_grid = [{'criterion': ['entropy'], 'min_impurity_decrease': entropy_thresholds},
                       {'criterion': ['gini'], 'min_impurity_decrease': gini_thresholds},
                       {'max_depth': np.arange(2,10)},
                       {'min_samples_split': np.arange(2,30,2)}]

         dtree_clf = GridSearchCV(dtree_model, param_grid, cv=5)
         dtree_clf.fit(train_x, train_y)
         print("best param:{0}\nbest score:{1}".format(dtree_clf.best_params_, dtree_clf.best_score_))

         dtree_model = tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=5,
                                                   min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0020202,
                                                   max_features=None, random_state=10, max_leaf_nodes=None,
                                                   min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)

         best param:{'criterion': 'entropy', 'min_impurity_decrease': 0.0}
```

實際結果　　預測結果

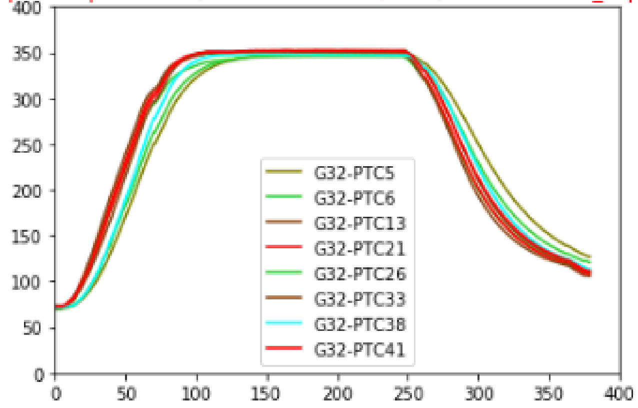
```
test_y 15 ; test_y_predicted 48 ;
test_y 32 ; test_y_predicted 49 ;
 test_y 32 ; test_y_predicted 49
 test_y 11 ; test_y_predicted 49
 test_y 49 ; test_y_predicted 32
 test_y 19 ; test_y_predicted 17
```

二、演算法和模型介紹-Decision Tree(3)

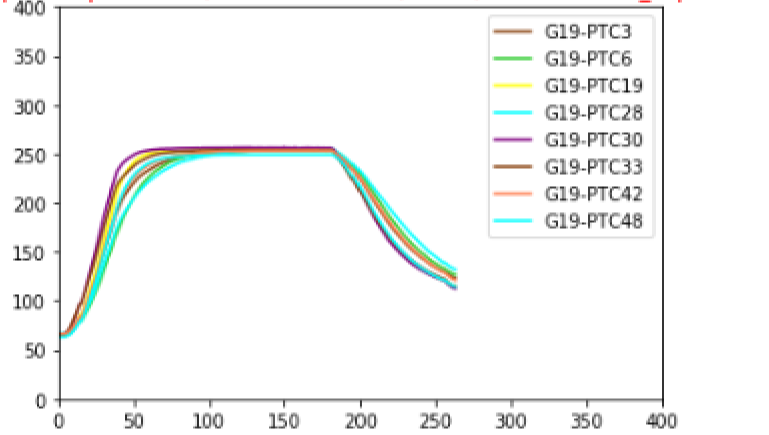
recipe Group11PTC15 ; File No : G11-1-AC(7X15)20160706-004_Export.txt



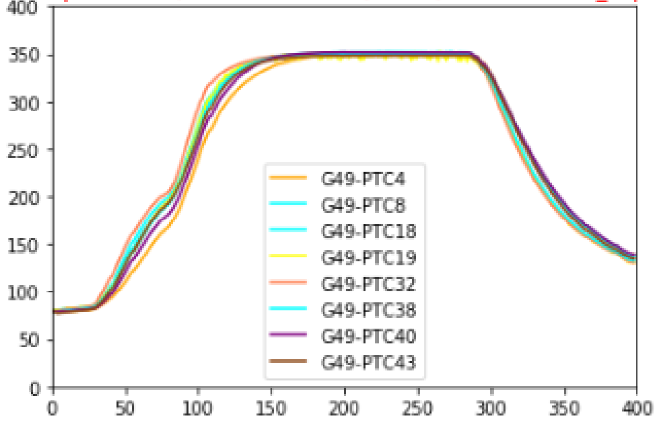
recipe Group32PTC33 ; File No : G32-AC(9X30)20190108-001_Export.txt



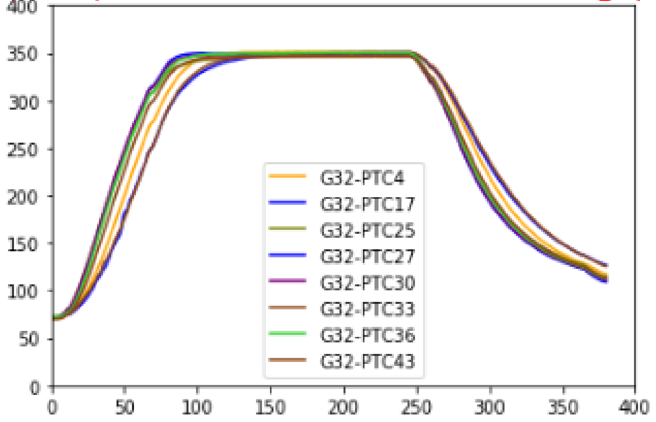
recipe Group19PTC30 ; File No : G19-AC(12X30)20170120-002_Export.txt



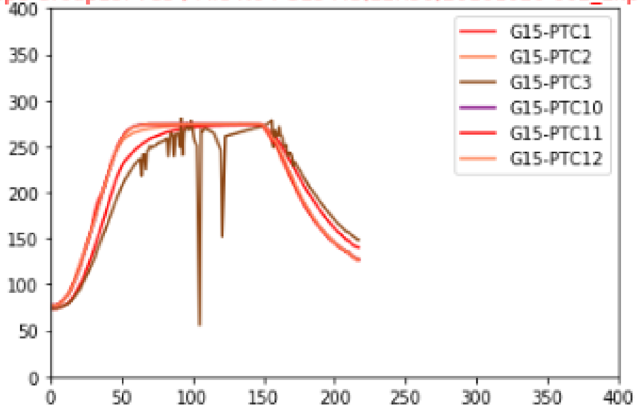
recipe Group49PTC18 ; File No : G49-AC(12x30)20190423-001_Export (2).txt



recipe Group32PTC4 ; File No : G32-AC(9x30)20190404-001_Export.txt

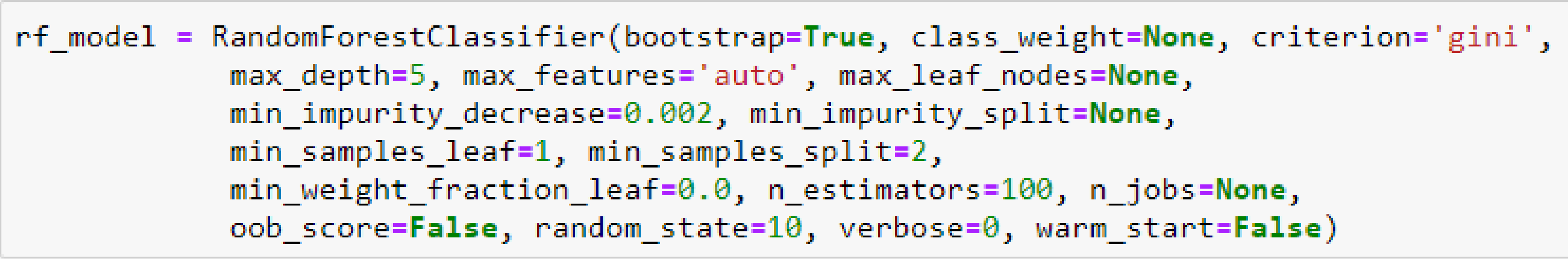


recipe Group15PTC3 ; File No : G15-AC(12X30)20161020-002_Export.txt



二、演算法和模型介紹-隨機森林(4)

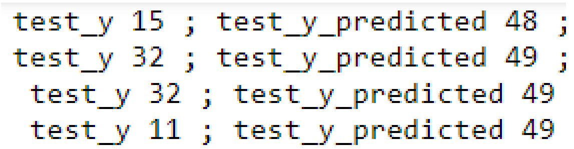
3. RandomForestClassifier 模型



```
rf_model = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.002, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=10, verbose=0, warm_start=False)
```

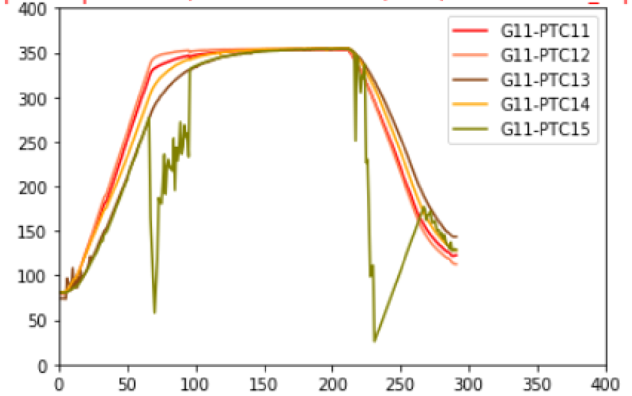
四張曲線圖為分錯圖形

實際結果　　預測結果

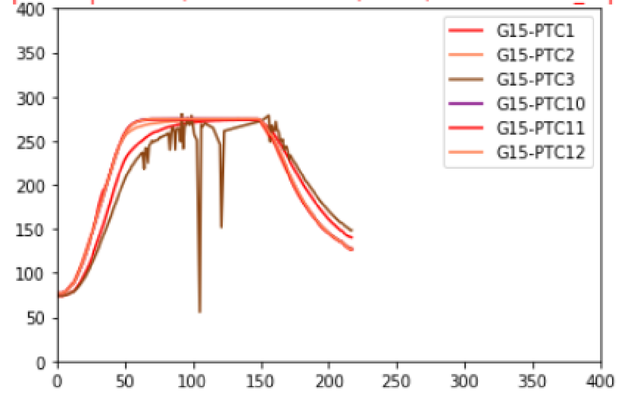


```
test_y 15 ; test_y_predicted 48 ;
test_y 32 ; test_y_predicted 49 ;
 test_y 32 ; test_y_predicted 49
 test_y 11 ; test_y_predicted 49
```

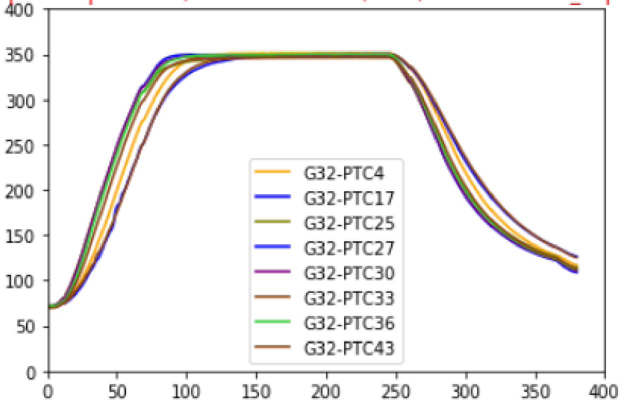
recipe Group11PTC15 ; File No : G11-1-AC(7X15)20160706-004_Export.txt



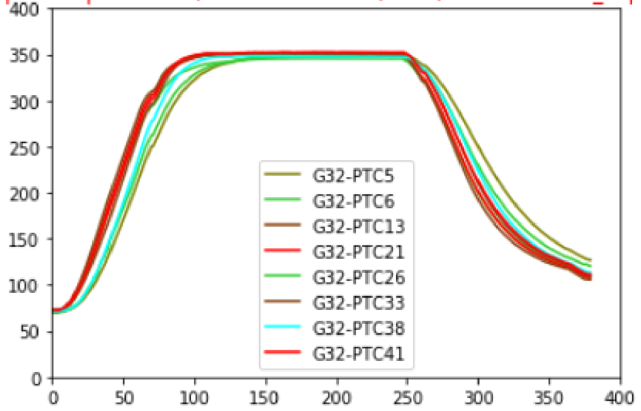
recipe Group15PTC3 ; File No : G15-AC(12X30)20161020-002_Export.txt



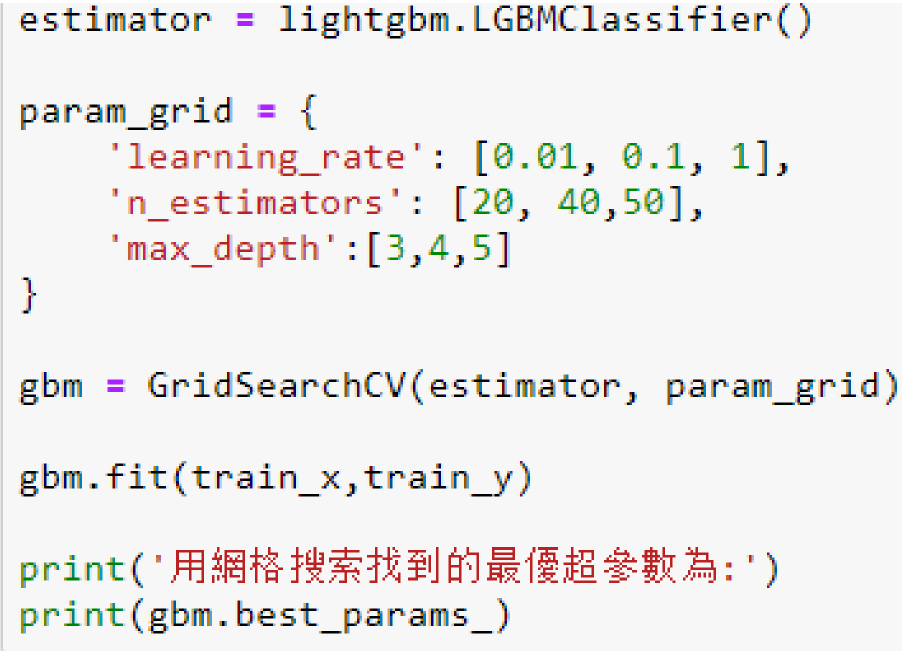
recipe Group32PTC4 ; File No : G32-AC(9x30)20190404-001_Export.txt



recipe Group32PTC33 ; File No : G32-AC(9X30)20190108-001_Export.txt



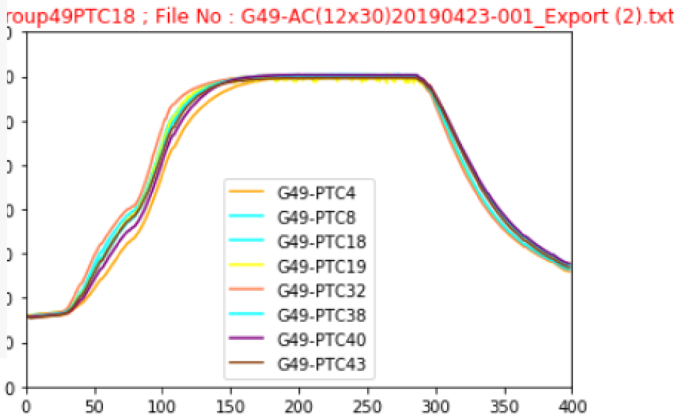
二、演算法和模型介紹- LightGBM(5)

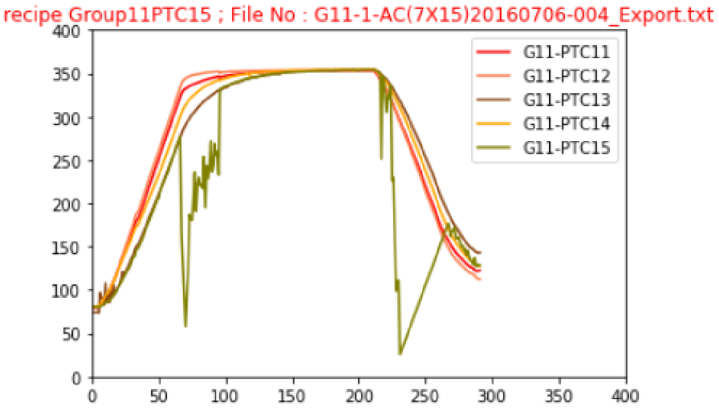


```
estimator = lightgbm.LGBMClassifier()

param_grid = {
    'learning_rate': [0.01, 0.1, 1],
    'n_estimators': [20, 40,50],
    'max_depth':[3,4,5]
}

gbm = GridSearchCV(estimator, param_grid)

gbm.fit(train_x,train_y)

print('用網格搜索找到的最優超參數為:')
print(gbm.best_params_)
```

實際結果　　預測結果

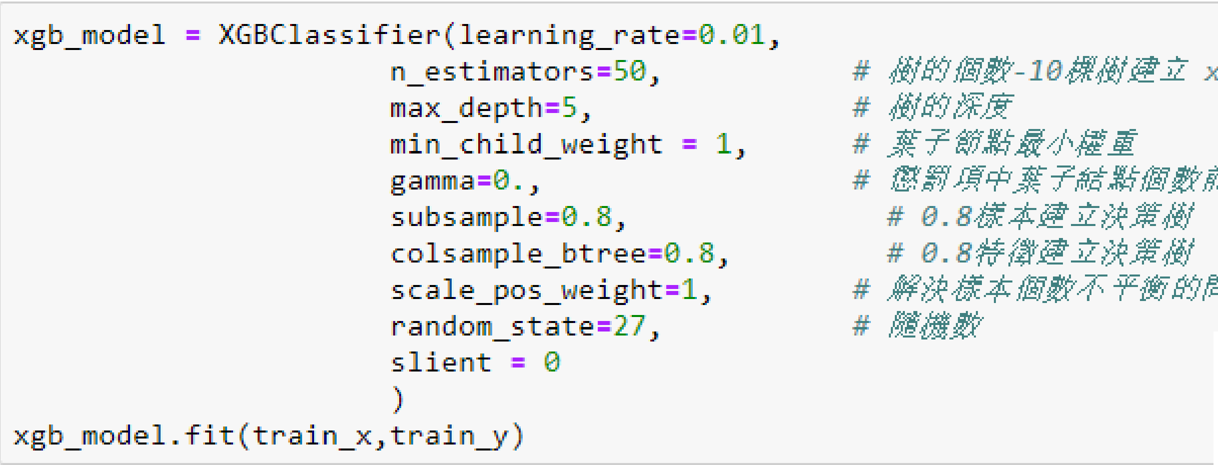
```
test_y 11 ; test_y_predicted 49 ;
test_y 49 ; test_y_predicted 32 ;
```



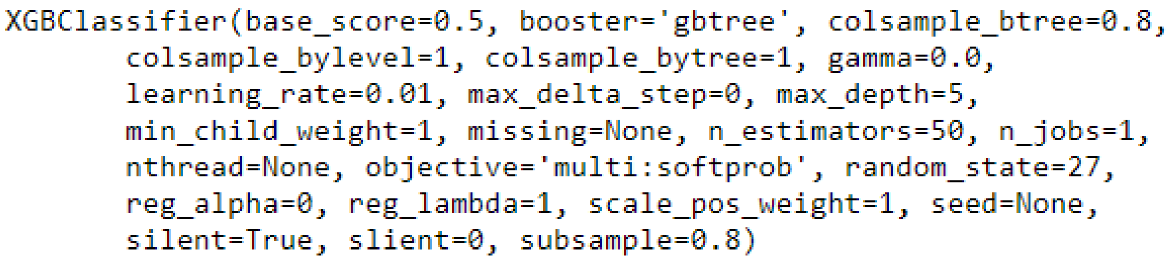


```
用網格搜索找到的最優超參數為:
{'learning_rate': 1, 'max_depth': 3, 'n_estimators': 20}
```

二、演算法和模型介紹-Xgboost(6)

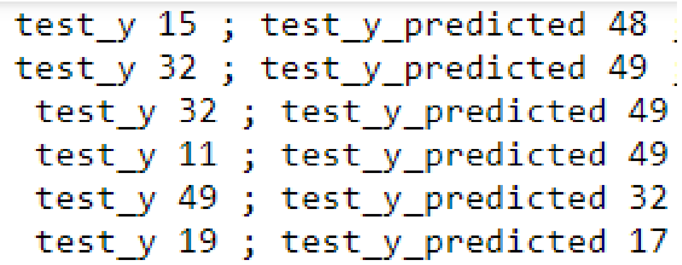


```
xgb_model = XGBClassifier(learning_rate=0.01,
                          n_estimators=50,          # 樹的個數-10棵樹建立 xgboost
                          max_depth=5,              # 樹的深度
                          min_child_weight = 1,     # 葉子節點最小權重
                          gamma=0.,                 # 懲罰項中葉子結點個數前的參數
                          subsample=0.8,              # 0.8樣本建立決策樹
                          colsample_btree=0.8,        # 0.8特徵建立決策樹
                          scale_pos_weight=1,       # 解決樣本個數不平衡的問題
                          random_state=27,          # 隨機數
                          slient = 0
                          )
xgb_model.fit(train_x,train_y)
```

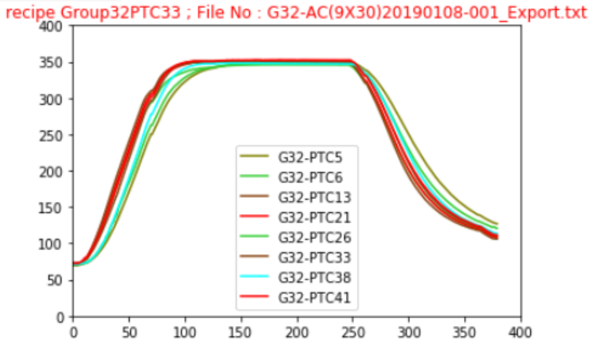


```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_btree=0.8,
        colsample_bylevel=1, colsample_bytree=1, gamma=0.0,
        learning_rate=0.01, max_delta_step=0, max_depth=5,
        min_child_weight=1, missing=None, n_estimators=50, n_jobs=1,
        nthread=None, objective='multi:softprob', random_state=27,
        reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
        silent=True, slient=0, subsample=0.8)
```

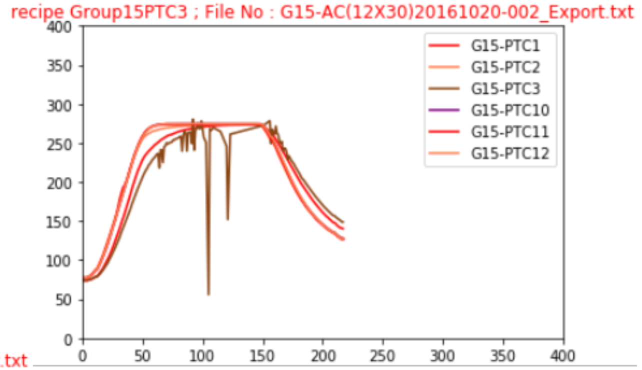
實際結果　預測結果



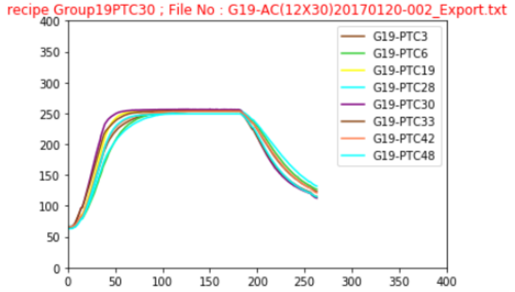
```
test_y 15 ; test_y_predicted 48
test_y 32 ; test_y_predicted 49
 test_y 32 ; test_y_predicted 49
 test_y 11 ; test_y_predicted 49
 test_y 49 ; test_y_predicted 32
 test_y 19 ; test_y_predicted 17
```



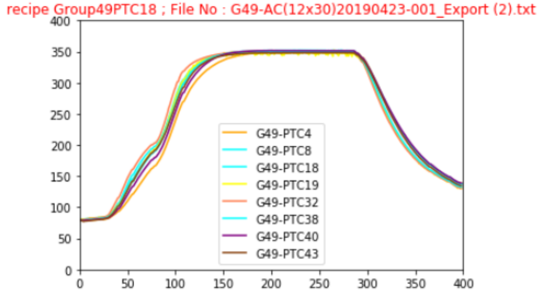
recipe Group32PTC33 ; File No : G32-AC(9X30)20190108-001_Export.txt



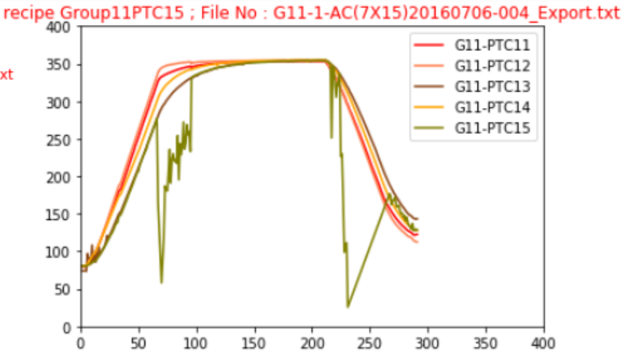
recipe Group15PTC3 ; File No : G15-AC(12X30)20161020-002_Export.txt



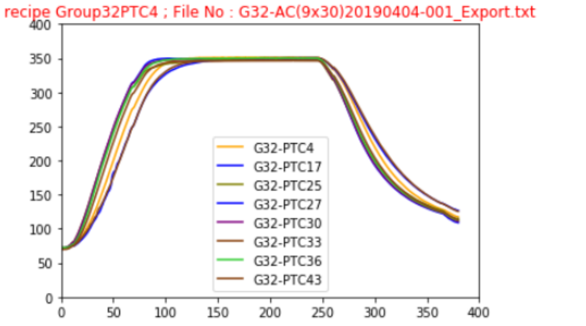
recipe Group19PTC30 ; File No : G19-AC(12X30)20170120-002_Export.txt



recipe Group49PTC18 ; File No : G49-AC(12x30)20190423-001_Export (2).txt



recipe Group11PTC15 ; File No : G11-1-AC(7X15)20160706-004_Export.txt



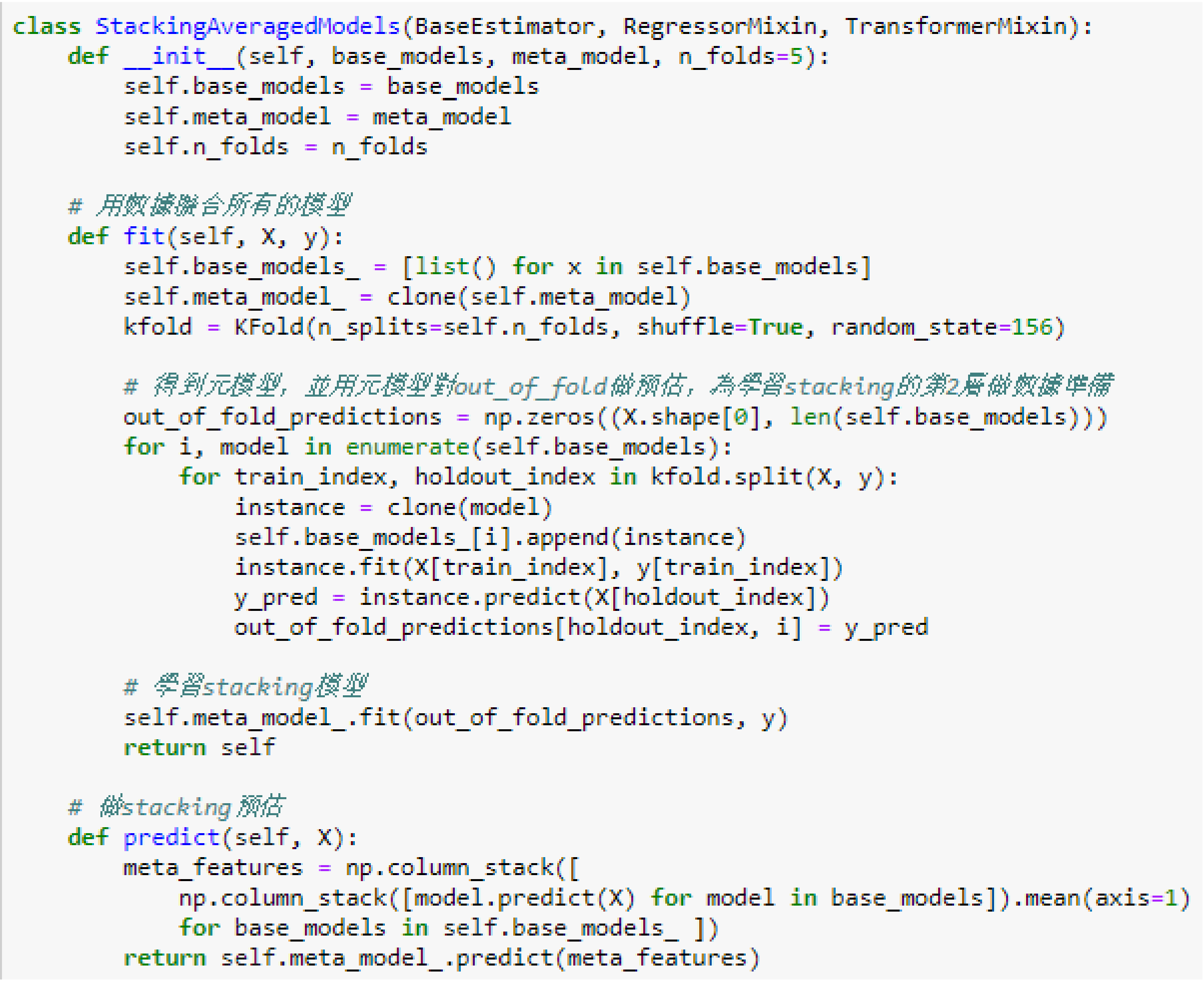
recipe Group32PTC4 ; File No : G32-AC(9x30)20190404-001_Export.txt

二、演算法和模型介紹-Meta-model Stacking(7)

Meta-model Stacking： 在這種方法中，我們在平均基礎模型上添加Meta-model，並使用這些基模型的out-of-folds預測來訓練我們的Meta-model。 訓練部分的步驟如下：

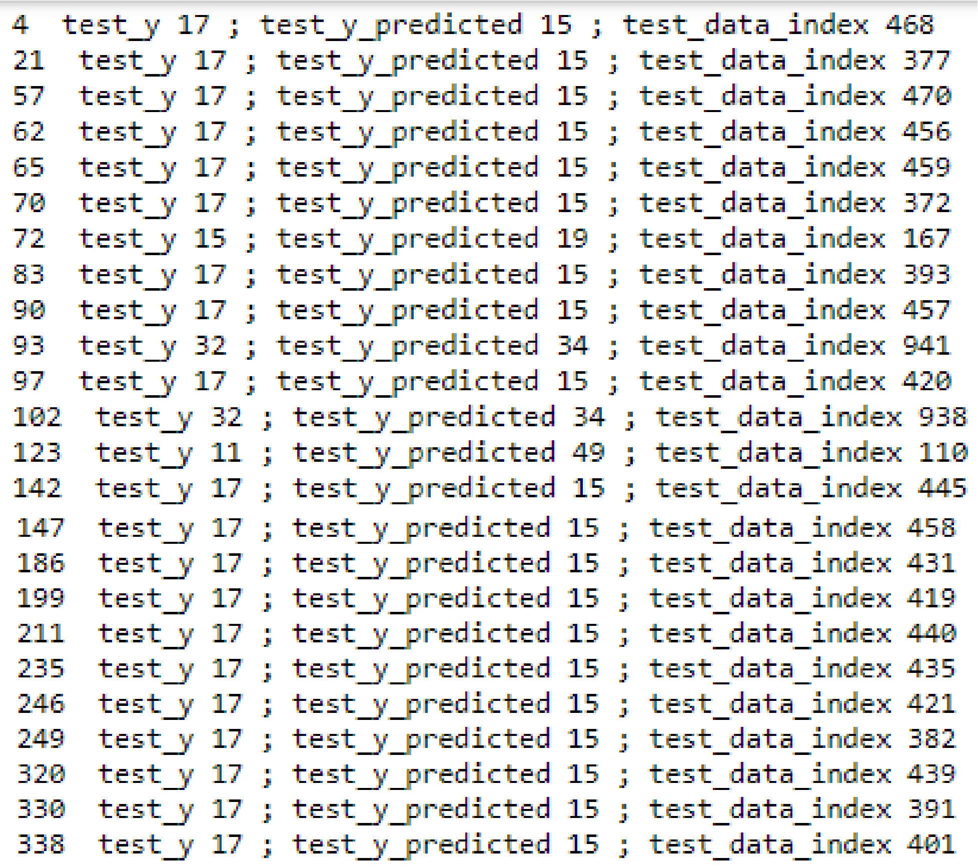
1. 將整個訓練集分解成兩個不相交的集合（這裡是train和.holdout）。
2. 在第一部分（train）上訓練幾個基本模型。
3. 在第二個部分（holdout）上測試這些基本模型。
4. 使用(3)中的預測（稱為 out-of-fold 預測）作為輸入，並將正確的標籤（目標變量）作為輸出來訓練更高層次的學習模型稱為元模型。前三個步驟是迭代完成的。例如，如果我們採取5倍的fold，我們首先將訓練數據分成5次。然後我們會做5次迭代。在每次迭代中，我們訓練每個基礎模型4倍，並預測剩餘的fold（holdout fold）。

二、演算法和模型介紹-Meta-model Stacking(7)



```
class StackingAveragedModels(BaseEstimator, RegressorMixin, TransformerMixin):
    def __init__(self, base_models, meta_model, n_folds=5):
        self.base_models = base_models
        self.meta_model = meta_model
        self.n_folds = n_folds

    # 用数據擬合所有的模型
    def fit(self, X, y):
        self.base_models_ = [list() for x in self.base_models]
        self.meta_model_ = clone(self.meta_model)
        kfold = KFold(n_splits=self.n_folds, shuffle=True, random_state=156)

        # 得到元模型，並用元模型對out_of_fold做预估，為學習stacking的第2層做数據準備
        out_of_fold_predictions = np.zeros((X.shape[0], len(self.base_models)))
        for i, model in enumerate(self.base_models):
            for train_index, holdout_index in kfold.split(X, y):
                instance = clone(model)
                self.base_models_[i].append(instance)
                instance.fit(X[train_index], y[train_index])
                y_pred = instance.predict(X[holdout_index])
                out_of_fold_predictions[holdout_index, i] = y_pred

        # 學習stacking模型
        self.meta_model_.fit(out_of_fold_predictions, y)
        return self

    # 做stacking预估
    def predict(self, X):
        meta_features = np.column_stack([
            np.column_stack([model.predict(X) for model in base_models]).mean(axis=1)
            for base_models in self.base_models_ ])
        return self.meta_model_.predict(meta_features)
```

實際結果　　預測結果　　Index

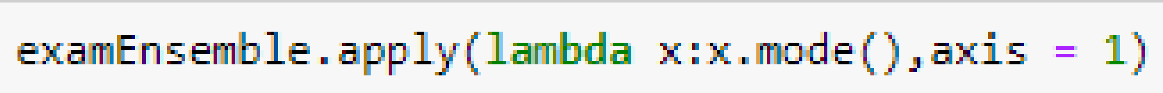


```
4  test_y 17 ; test_y_predicted 15 ; test_data_index 468
21  test_y 17 ; test_y_predicted 15 ; test_data_index 377
57  test_y 17 ; test_y_predicted 15 ; test_data_index 470
62  test_y 17 ; test_y_predicted 15 ; test_data_index 456
65  test_y 17 ; test_y_predicted 15 ; test_data_index 459
70  test_y 17 ; test_y_predicted 15 ; test_data_index 372
72  test_y 15 ; test_y_predicted 19 ; test_data_index 167
83  test_y 17 ; test_y_predicted 15 ; test_data_index 393
90  test_y 17 ; test_y_predicted 15 ; test_data_index 457
93  test_y 32 ; test_y_predicted 34 ; test_data_index 941
97  test_y 17 ; test_y_predicted 15 ; test_data_index 420
102  test_y 32 ; test_y_predicted 34 ; test_data_index 938
123  test_y 11 ; test_y_predicted 49 ; test_data_index 110
142  test_y 17 ; test_y_predicted 15 ; test_data_index 445
147  test_y 17 ; test_y_predicted 15 ; test_data_index 458
186  test_y 17 ; test_y_predicted 15 ; test_data_index 431
199  test_y 17 ; test_y_predicted 15 ; test_data_index 419
211  test_y 17 ; test_y_predicted 15 ; test_data_index 440
235  test_y 17 ; test_y_predicted 15 ; test_data_index 435
246  test_y 17 ; test_y_predicted 15 ; test_data_index 421
249  test_y 17 ; test_y_predicted 15 ; test_data_index 382
320  test_y 17 ; test_y_predicted 15 ; test_data_index 439
330  test_y 17 ; test_y_predicted 15 ; test_data_index 391
338  test_y 17 ; test_y_predicted 15 ; test_data_index 401
```

三、演算法排名

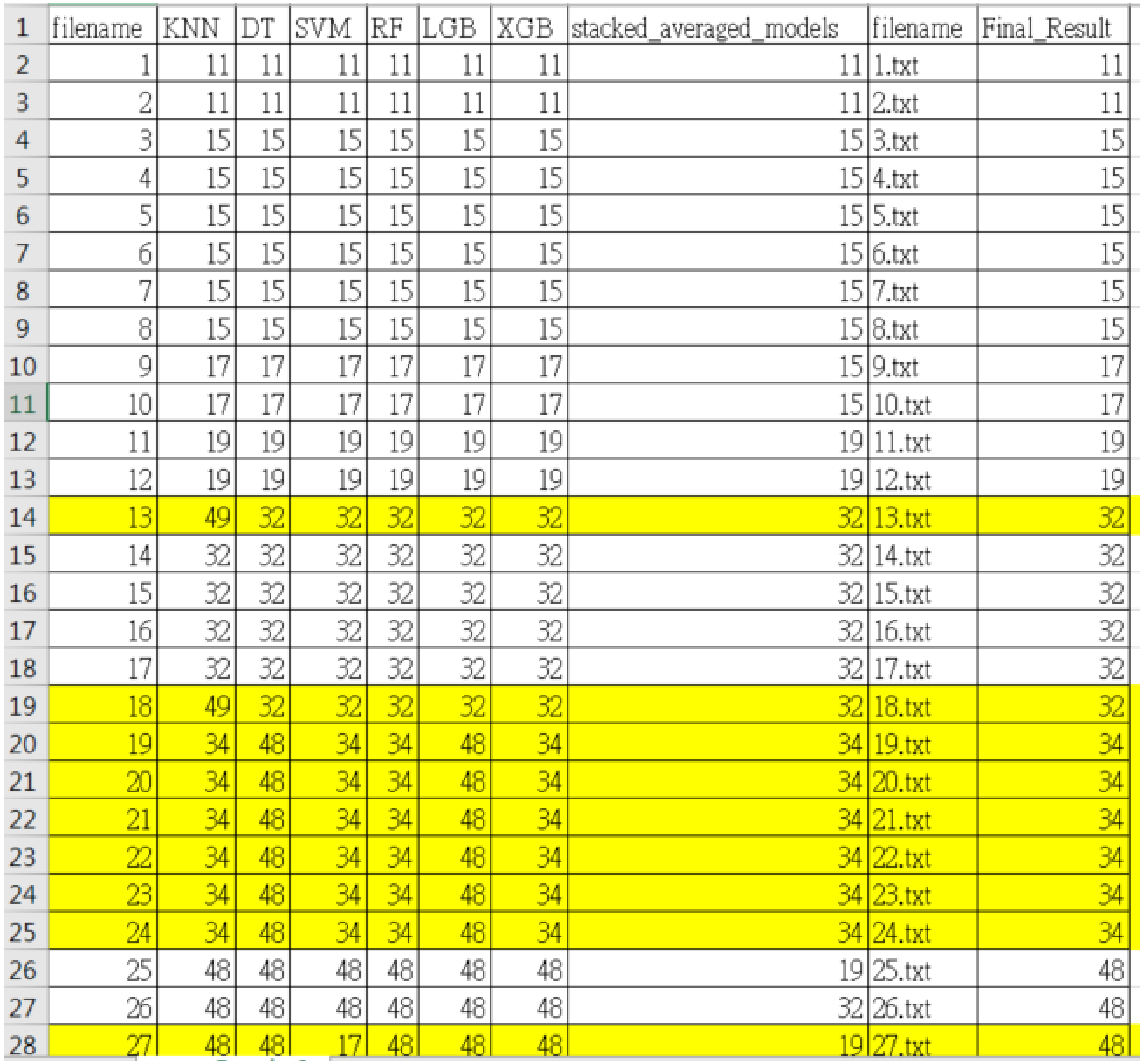
	算法名	得分
1	K Nearest Neighbors(KNN)	0.9971
2	Support Vector Machine (SVM)	0.996
3	RandomForest	0.9943
4	LightGBM (LGBM)	0.9942
5	Xgboost	0.9857
6	Decision Tree	0.9821
7	Meta-model Stacking	0.9312

三、結果選擇



```
examEnsemble.apply(lambda x:x.mode(),axis = 1)
```

三個演算法做Voting，選擇出現次數最多的當作Exam最後的結果。



1	filename	KNN	DT	SVM	RF	LGB	XGB	stacked_averaged_models	filename	Final_Result
2	1	11	11	11	11	11	11	11	1.txt	11
3	2	11	11	11	11	11	11	11	2.txt	11
4	3	15	15	15	15	15	15	15	3.txt	15
5	4	15	15	15	15	15	15	15	4.txt	15
6	5	15	15	15	15	15	15	15	5.txt	15
7	6	15	15	15	15	15	15	15	6.txt	15
8	7	15	15	15	15	15	15	15	7.txt	15
9	8	15	15	15	15	15	15	15	8.txt	15
10	9	17	17	17	17	17	17	15	9.txt	17
11	10	17	17	17	17	17	17	15	10.txt	17
12	11	19	19	19	19	19	19	19	11.txt	19
13	12	19	19	19	19	19	19	19	12.txt	19
14	13	49	32	32	32	32	32	32	13.txt	32
15	14	32	32	32	32	32	32	32	14.txt	32
16	15	32	32	32	32	32	32	32	15.txt	32
17	16	32	32	32	32	32	32	32	16.txt	32
18	17	32	32	32	32	32	32	32	17.txt	32
19	18	49	32	32	32	32	32	32	18.txt	32
20	19	34	48	34	34	48	34	34	19.txt	34
21	20	34	48	34	34	48	34	34	20.txt	34
22	21	34	48	34	34	48	34	34	21.txt	34
23	22	34	48	34	34	48	34	34	22.txt	34
24	23	34	48	34	34	48	34	34	23.txt	34
25	24	34	48	34	34	48	34	34	24.txt	34
26	25	48	48	48	48	48	48	19	25.txt	48
27	26	48	48	48	48	48	48	32	26.txt	48
28	27	48	48	17	48	48	48	19	27.txt	48