06XgbCV

March 18, 2019

1 XGB + CV

In [1]:
```
import matplotlib.pyplot as plt
from planar_utils import plot_decision_boundary, sigmoid, load_planar_dataset, load_ext
import numpy as np
import pandas as pd
import os
from collections import Counter

from sklearn.neighbors import KNeighborsClassifier  ## KNN
from sklearn.linear_model import LogisticRegressionCV  ## logistic regression
from sklearn.tree import DecisionTreeClassifier  ## decision tree
from sklearn.svm import SVC  ## SVM

from sklearn.tree import DecisionTreeClassifier  ## decision tree
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from xgboost import XGBClassifier

import math
import string
import re

import xgboost

from preprocess import preprocess
plt.rcParams['figure.figsize'] = [10, 8]
```

2 鐵達尼號資料集

In [2]:
```
df = pd.read_csv('train.csv')
df = preprocess(df)
df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Cabin	\
0	1	0	3	1	22.0	1	0	2	0	
1	2	1	1	0	38.0	1	0	5	3	

```
2            3         1       3    0  26.0      0      0       7      0
3            4         1       1    0  35.0      1      0       1      3
4            5         0       3    1  35.0      0      0       1      0

   Embarked  Has_Cabin  Age_Cat  Fare_log2  Fare_Cat  Name_Length  \
0         0          0        1   2.857981         0           23
1         2          1        2   6.155492         5           51
2         0          0        1   2.986411         0           22
3         0          1        2   5.730640         4           44
4         0          0        2   3.008989         0           24

   Name_With_Special_Char  Family_Size  Title
0                       0            1      1
1                       1            1      3
2                       0            0      2
3                       1            1      3
4                       0            0      1
```

In [3]:
```
X = df[['PassengerId', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch',
        'Ticket', 'Cabin', 'Embarked', 'Has_Cabin', 'Age_Cat', 'Fare_log2',
        'Fare_Cat', 'Name_Length', 'Name_With_Special_Char', 'Family_Size',
        'Title']].values
Y = df['Survived'].values
```

In [4]:
```
from sklearn.model_selection  import train_test_split

X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, test_size =0.3, random_sta
print(X_train.shape)  ## (445, 17)
print(X_valid.shape)  ## (446, 17)
print(Y_train.shape)  ## (445,)
print(Y_valid.shape)  ## (446,)
```

```
(623, 17)
(268, 17)
(623,)
(268,)
```

In [5]:
```
def get_accuracy(clf):
    #=============your works starts==============#
    clf =
    clf =
    y_pred =
    accuracy =
    #==============your works ends================#
    return accuracy

print('SVM: ', get_accuracy(SVC))
print('DecisionTree: ', get_accuracy(DecisionTreeClassifier))
```

```
        print('RandomForest: ', get_accuracy(RandomForestClassifier))
        print('AdaBoost: ', get_accuracy(AdaBoostClassifier))  ## Boosting 的演算法
        print('XGB: ', get_accuracy(XGBClassifier))

        # SVM:  0.609865470852
        # DecisionTree:  0.764573991031
        # RandomForest:  0.795964125561
        # AdaBoost:  0.784753363229
        # XGB:  0.80269058296
```

```
SVM:  0.6455223880597015
DecisionTree:  0.7686567164179104
RandomForest:  0.832089552238806
AdaBoost:  0.7910447761194029


  "avoid this warning.", FutureWarning)
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


XGB:  0.8432835820895522
```

```
In [6]: # Set our parameters for xgboost
        params = {}
        # 請填入以下參數:
        # 目標函數: 二元分類
        # 評價函數: logloss
        # 學習速度: 0.04
        # 最大深度: 5
        #=============your works starts===============#
        params['objective'] =
        params['eval_metric'] =
        params['eta'] =
        params['max_depth'] =
        #==============your works ends================#

        d_train = xgboost.DMatrix(X_train, label=Y_train)
        d_valid = xgboost.DMatrix(X_valid, label=Y_valid)

        watchlist = [(d_train, 'train'), (d_valid, 'valid')]

        bst = xgboost.train(params, d_train, 100, watchlist, early_stopping_rounds=100, verbose
        y_pred = bst.predict(xgboost.DMatrix(X_valid))
        print("Accuracy: ", str(sum(Y_valid == (y_pred > 0.5))/Y_valid.shape[0]))
```

```
Accuracy:  0.835820895522388
```

3 空氣品質

In [7]:
```
# dateparse = lambda x: pd.datetime.strptime(x, '%d/%m/%Y %H:%M:%S')
# dateparse_1 = lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')
# EPA_6 = pd.read_csv('air_pollution_data/EPA_OD_201806.csv', parse_dates=['PublishTim
# EPA_7 = pd.read_csv('air_pollution_data/EPA_OD_201807.csv', parse_dates=['PublishTim
# EPA_8 = pd.read_csv('air_pollution_data/EPA_OD_201808.csv', parse_dates=['PublishTim
# EPA_9 = pd.read_csv('air_pollution_data/EPA_OD_201809.csv', parse_dates=['PublishTim
# EPA_10 = pd.read_csv('air_pollution_data/EPA_OD_201810.csv', parse_dates=['PublishTi
# EPA_11 = pd.read_csv('air_pollution_data/EPA_OD_201811.csv', parse_dates=['PublishTi
# EPA_12 = pd.read_csv('air_pollution_data/EPA_OD_201812.csv', parse_dates=['PublishTi
# frames = [EPA_6, EPA_7, EPA_8, EPA_9, EPA_10, EPA_11, EPA_12]
# df_AQI = pd.concat(frames)
# df_AQI.to_pickle('air_pollution_data.pkl')
# df_AQI.head()
```

In [8]:
```
# df_AQI = pd.read_pickle('air_pollution_data.pkl')
# df_AQI.sort_values(by='PublishTime', inplace=True)
# df_AQI = df_AQI.loc[df_AQI['SiteName'] == '麥寮', ['SiteName', 'AQI', 'PM2.5', 'SO2
# df_AQI.to_pickle('df_AQI_gl.pkl')
```

In [9]:
```
df_AQI = pd.read_pickle('df_AQI_gl.pkl')
```

In [10]:
```
window = 7
shift = 1
segments = int((df_AQI.shape[0] - window) // shift) + 1

train = np.zeros((segments-1, 2 * window))
target = np.zeros((segments-1,))
for segment in range(segments -1):
    seg = df_AQI.iloc[segment:segment+window][['SO2', 'PM2.5']]
    target[segment] = df_AQI.iloc[segment+window][['AQI']]
    train[segment] = np.append(seg['SO2'].values, seg['PM2.5'].values)
```

In [11]:
```
train = np.where(np.isnan(train), -1, train)
target = np.where(np.isnan(target), int(np.nanmean(target)), target)
```

In [12]:
```
train.shape, target.shape
```

Out[12]:
```
((4721, 14), (4721,))
```

In [13]:
```
X_train, X_valid, Y_train, Y_valid = train_test_split(train, target, test_size=0.2, ra
X_train.shape, Y_train.shape, X_valid.shape, Y_valid.shape
```

Out[13]:
```
((3776, 14), (3776,), (945, 14), (945,))
```

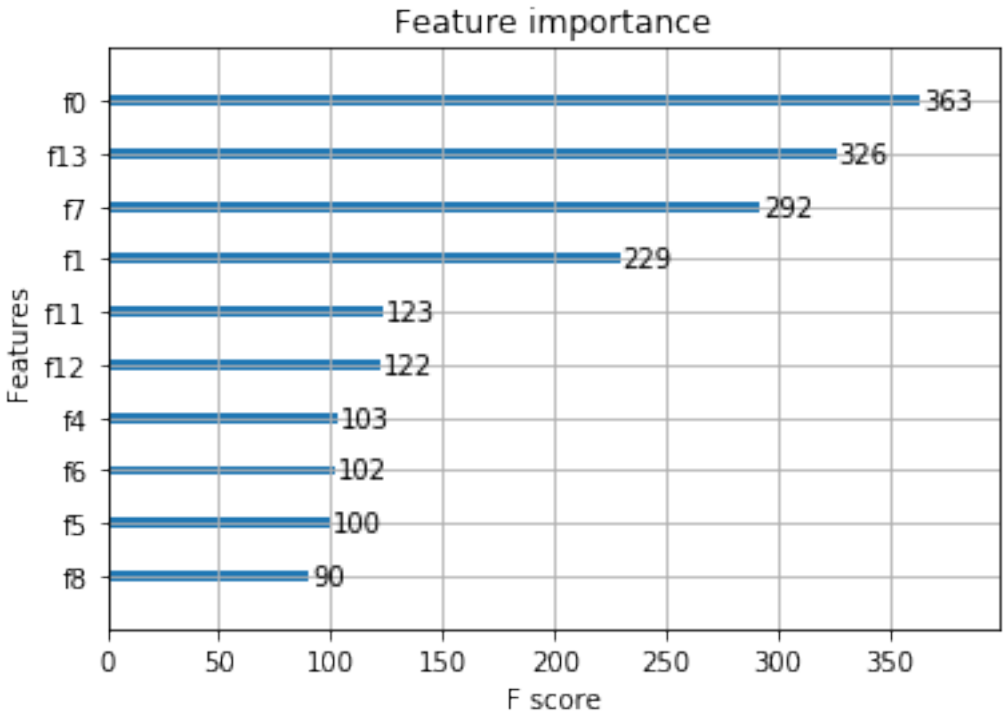
In [14]:
```
# Set our parameters for xgboost
params = {}
```

```
        # 請填入以下參數:
        # 目標函數: 線性回歸
        # 評價函數: rmse
        # 學習速度: 0.01
        # 最大深度: 5
        # bst = xgboost.train(params, d_train, 3000, watchlist, early_stopping_rounds=50, ver
        #=============your works starts===============#
        params['objective'] =
        params['eval_metric'] =
        params['eta'] =
        params['max_depth'] =
        d_train =
        d_valid =
        watchlist =
        bst =
        Y_pred =
        #==============your works ends================#
```

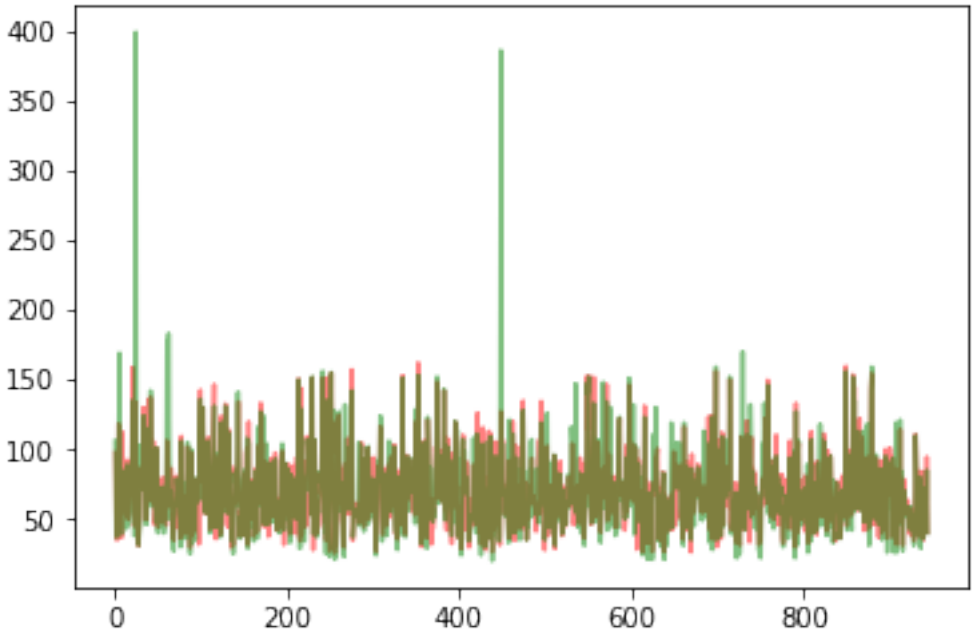
```
[0]        train-rmse:74.7182        valid-rmse:75.092
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 10 rounds.
[10]        train-rmse:56.2433        valid-rmse:57.2731
[20]        train-rmse:42.8466        valid-rmse:44.581
[30]        train-rmse:33.2595        valid-rmse:35.6529
[40]        train-rmse:26.5246        valid-rmse:29.5798
[50]        train-rmse:21.8925        valid-rmse:25.5427
[60]        train-rmse:18.8046        valid-rmse:22.983
[70]        train-rmse:16.7946        valid-rmse:21.4035
[80]        train-rmse:15.4931        valid-rmse:20.4358
[90]        train-rmse:14.6551        valid-rmse:19.8031
[100]        train-rmse:14.1008        valid-rmse:19.4283
[110]        train-rmse:13.7185        valid-rmse:19.2046
[120]        train-rmse:13.4201        valid-rmse:19.0727
[130]        train-rmse:13.1949        valid-rmse:19.0049
[140]        train-rmse:13.0321        valid-rmse:18.9568
[150]        train-rmse:12.8907        valid-rmse:18.9242
[160]        train-rmse:12.7914        valid-rmse:18.8938
[170]        train-rmse:12.711        valid-rmse:18.8637
[180]        train-rmse:12.6417        valid-rmse:18.8474
[190]        train-rmse:12.576        valid-rmse:18.8277
[200]        train-rmse:12.5151        valid-rmse:18.8179
[210]        train-rmse:12.4596        valid-rmse:18.8122
[220]        train-rmse:12.4095        valid-rmse:18.803
[230]        train-rmse:12.3449        valid-rmse:18.7866
[240]        train-rmse:12.2967        valid-rmse:18.7779
[250]        train-rmse:12.2428        valid-rmse:18.7633
[260]        train-rmse:12.1924        valid-rmse:18.7449
```

```
[270]        train-rmse:12.1467        valid-rmse:18.7386
[280]        train-rmse:12.1038        valid-rmse:18.7303
[290]        train-rmse:12.0681        valid-rmse:18.7282
[300]        train-rmse:12.02        valid-rmse:18.7229
Stopping. Best iteration:
[299]        train-rmse:12.0274        valid-rmse:18.7228
```

In [15]:
```
# 請使用 xgboost.plot_importance，並設定 max_num_features=10
#!=============your works starts===============!#

#!==============your works ends================!#

plt.show()
```



In [16]:
```
plt.plot(Y_pred, label='Y_pred', alpha=0.5, c='r')
plt.plot(Y_valid, label='Y_valid', alpha=0.5, c='g')
plt.show()
```



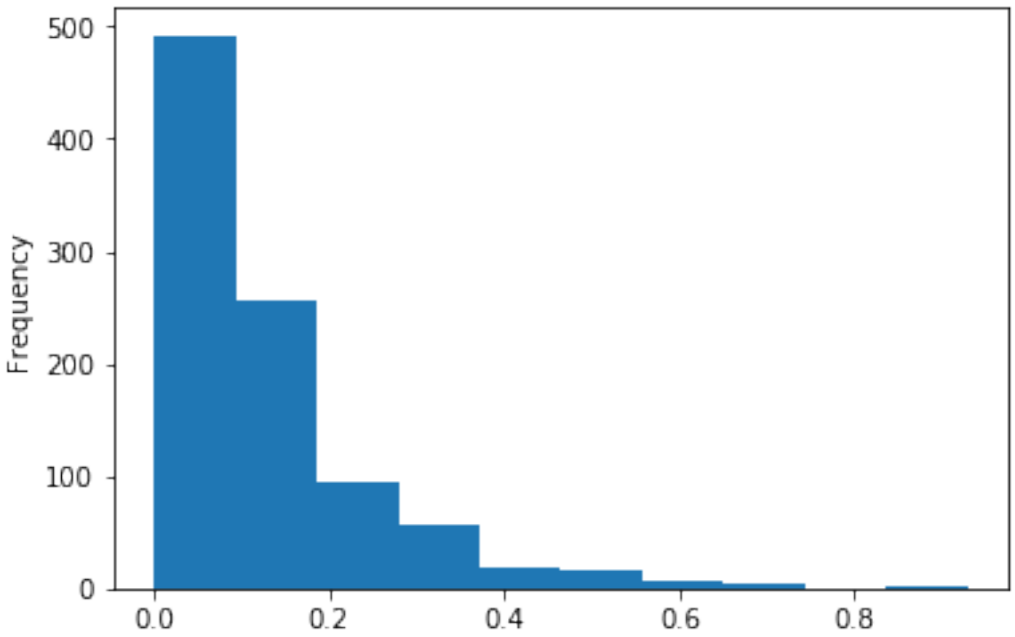
In [17]:
```
df_result = pd.DataFrame()

# 1. 使用 X_valid 去評價此模型
# 2. 使用 ['predict', 'truth', 'error'] 三個欄位的 DataFrame 去使決畫呈現預測結果
#    (1). 請注意與測結果 (Y_pred) 與真實值 (Y_valid) 都必須取 exp 方能反映實際情況
#    (2). error 請使用計算 np.abs(predict-truth)/truth 計算誤差百分比
#=============your works starts===============#
Y_pred =
df_result['predict'] =
df_result['truth'] =
df_result['error'] =
df_result_sort =
#==============your works ends================#

df_result.head()
```

Out[17]:

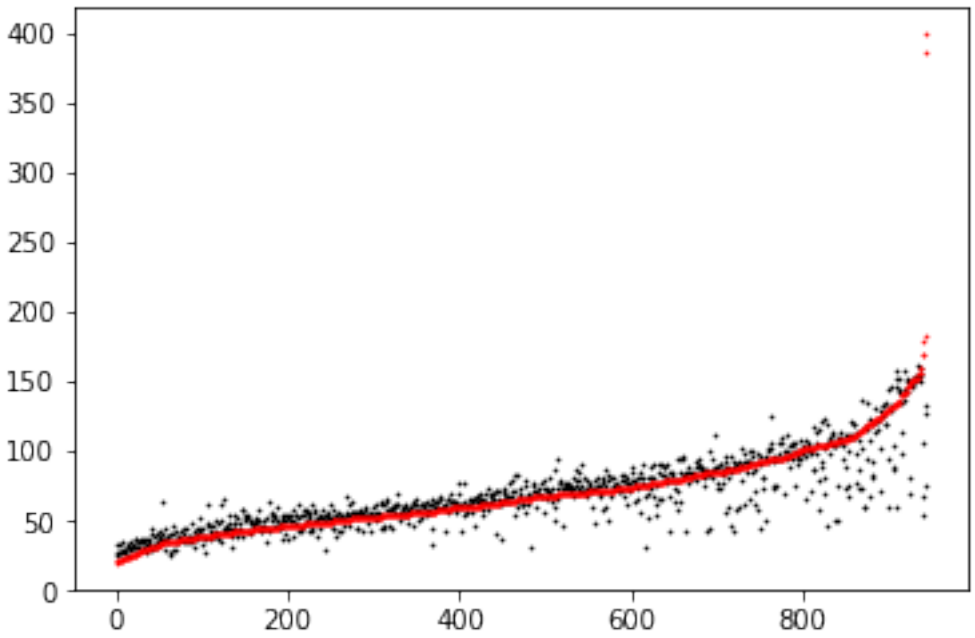
	predict	truth	error
0	97.562889	106.0	0.079595
1	48.304008	37.0	0.305514
2	34.956127	37.0	0.055240
3	59.692013	67.0	0.109074
4	118.139969	116.0	0.018448

In [18]:
```
# 請使用 df_result_sort 濾掉 error 大於 1 的部分畫出 error 的分布圖
#!=============your works starts===============!#
```

```
#!==============your works ends================!#

plt.show()
```



In [19]:
```
# 請使用 plt.scatter，以 0~len(df_result) 作為 x，預測值 (黑色) 與實際值 (紅色) 作為 y。
#!==============your works starts===============!#


#!==============your works ends================!#

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

4 房價資料集

In [20]:
```
# import urllib.request
# if 'df_realestate_processed.csv' not in os.listdir():
#     url = 'https://s3.amazonaws.com/datasets-jeremy/df_realestate_processed.csv'
#     urllib.request.urlretrieve(url, 'df_realestate_processed.csv')

# # processed
# path = "df_realestate_processed.csv"
# df_realestate_processed = pd.read_csv(path)
# X = df_realestate_processed.drop(["price_per_meter", "total_price"], axis=1)
# Y = df_realestate_processed['total_price']
```

In [21]:
```
# X_train = X.iloc[:-1000]
# Y_train = Y.iloc[:-1000]
# Y_train = np.log(Y_train)

# X_valid = X.iloc[-1000:]
# Y_valid = Y.iloc[-1000:]
# Y_valid = np.log(Y_valid)
```

In [22]:
```
# # Set our parameters for xgboost
# params = {}

# # 請填入以下參數:
# # 目標函數: 線性回歸
# # 評價函數: rmse
# # 學習速度: 0.01
# # 最大深度: 5
# # bst = xgboost.train(params, d_train, 3000, watchlist, early_stopping_rounds=50, v
# #=============your works starts===============#
# params['objective'] =
# params['eval_metric'] =
# params['eta'] =
# params['max_depth'] =
# d_train =
# d_valid =
# watchlist =
# bst =
# Y_pred =
# #==============your works ends================#
```

In [23]:
```
#   模型 save 與 load 的方式自己看
# bst.save_model("bst_subtotal_log_with_cross.pickle.dat")
# bst = xgboost.Booster({'nthread':1}) #init model
# bst.load_model("bst_subtotal_log_with_cross.pickle.dat") # load data
```

In [24]:
```
# # 請使用 xgboost.plot_importance，並設定 max_num_features=10
# #!=============your works starts===============!#

# #!==============your works ends================!#

# plt.show()
```

In [25]:
```
# df_result = pd.DataFrame()

# # 1. 使用 X_valid 去評價此模型
# # 2. 使用 ['predict', 'truth', 'error'] 三個欄位的 DataFrame 去使決畫呈現預測結果
# #    (1). 請注意與測結果 (Y_pred) 與真實值 (Y_valid) 都必須取 exp 方能反映實際情況
# #    (2). error 請使用計算 np.abs(predict-truth)/truth 計算誤差百分比
# #=============your works starts===============#
# Y_pred =
```

```
        # df_result['predict'] =
        # df_result['truth'] =
        # df_result['error'] =
        # df_result_sort =
        # #==============your works ends================#

        # df_result.head()
```

In [26]:
```
        # # 請使用 df_result_sort 濾掉 error 大於 1 的部分畫出 error 的分布圖
        # #!==============your works starts===============!#

        # #!==============your works ends================!#

        # plt.show()
```

In [27]:
```
        # # 請使用 plt.scatter，以 0~len(df_result) 作為 x，預測值 (黑色) 與實際值 (紅色) 作為 y
        # #!==============your works starts===============!#


        # #!==============your works ends================!#

        # plt.show()
```

In [ ]: