06XgbCV

November 11, 2018

1 XGB + CV

```
In [2]: 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
```

2 鐵達尼號資料集

```
In [3]: df = pd.read_csv('train.csv')
        df = preprocess(df)
        df.head()

Out[3]:    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
```

```python
In [4]: 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
```

```python
In [5]: 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_stat
        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,)
```

```python
In [6]: 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))
```

```python
        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:  0.7611940298507462
RandomForest:  0.8544776119402985


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


AdaBoost:  0.7910447761194029
XGB:  0.8432835820895522
```

```python
In [7]: # 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 房價資料集

```
In [8]: 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 [9]: 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 [10]: # 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:15.8112      valid-rmse:15.6215
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:11.6632      valid-rmse:11.5317
[20]    train-rmse:8.60487      valid-rmse:8.51713
```

```
[30]          train-rmse:6.35023        valid-rmse:6.30023
[40]          train-rmse:4.68857        valid-rmse:4.66973
[50]          train-rmse:3.46457        valid-rmse:3.46551
[60]          train-rmse:2.56373        valid-rmse:2.58316
[70]          train-rmse:1.90185        valid-rmse:1.93611
[80]          train-rmse:1.41686        valid-rmse:1.46651
[90]          train-rmse:1.06302        valid-rmse:1.12422
[100]         train-rmse:0.806951       valid-rmse:0.876425
[110]         train-rmse:0.624079       valid-rmse:0.700146
[120]         train-rmse:0.496058       valid-rmse:0.577805
[130]         train-rmse:0.408926       valid-rmse:0.49603
[140]         train-rmse:0.351423       valid-rmse:0.442858
[150]         train-rmse:0.314738       valid-rmse:0.409589
[160]         train-rmse:0.291941       valid-rmse:0.387703
[170]         train-rmse:0.277808       valid-rmse:0.373504
[180]         train-rmse:0.26903        valid-rmse:0.364483
[190]         train-rmse:0.263451       valid-rmse:0.358139
[200]         train-rmse:0.259615       valid-rmse:0.352701
[210]         train-rmse:0.256954       valid-rmse:0.349393
[220]         train-rmse:0.254944       valid-rmse:0.347053
[230]         train-rmse:0.253445       valid-rmse:0.345315
[240]         train-rmse:0.252139       valid-rmse:0.343419
[250]         train-rmse:0.251006       valid-rmse:0.342147
[260]         train-rmse:0.250072       valid-rmse:0.341061
[270]         train-rmse:0.249214       valid-rmse:0.340131
[280]         train-rmse:0.248427       valid-rmse:0.339297
[290]         train-rmse:0.247735       valid-rmse:0.338445
[300]         train-rmse:0.247081       valid-rmse:0.33738
[310]         train-rmse:0.24648        valid-rmse:0.337037
[320]         train-rmse:0.245882       valid-rmse:0.336335
[330]         train-rmse:0.245386       valid-rmse:0.335948
[340]         train-rmse:0.244862       valid-rmse:0.335533
[350]         train-rmse:0.244334       valid-rmse:0.33528
[360]         train-rmse:0.243893       valid-rmse:0.334766
[370]         train-rmse:0.243417       valid-rmse:0.334686
[380]         train-rmse:0.242968       valid-rmse:0.334314
[390]         train-rmse:0.242505       valid-rmse:0.333969
[400]         train-rmse:0.242057       valid-rmse:0.333667
[410]         train-rmse:0.241676       valid-rmse:0.333362
[420]         train-rmse:0.241232       valid-rmse:0.332969
[430]         train-rmse:0.240818       valid-rmse:0.332734
[440]         train-rmse:0.24045        valid-rmse:0.332447
[450]         train-rmse:0.240066       valid-rmse:0.332321
[460]         train-rmse:0.239724       valid-rmse:0.331935
[470]         train-rmse:0.239354       valid-rmse:0.331688
[480]         train-rmse:0.238993       valid-rmse:0.331617
[490]         train-rmse:0.23864        valid-rmse:0.331323
[500]         train-rmse:0.238285       valid-rmse:0.331034
```

```
[510]          train-rmse:0.23793          valid-rmse:0.331124
Stopping. Best iteration:
[502]          train-rmse:0.238198          valid-rmse:0.33097
```

In [ ]: # 　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 [11]: # 請使用 *xgboost.plot_importance*，並設定 *max_num_features=10*
        #!==============*your works starts*==============!#

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

        plt.show()

Feature importance
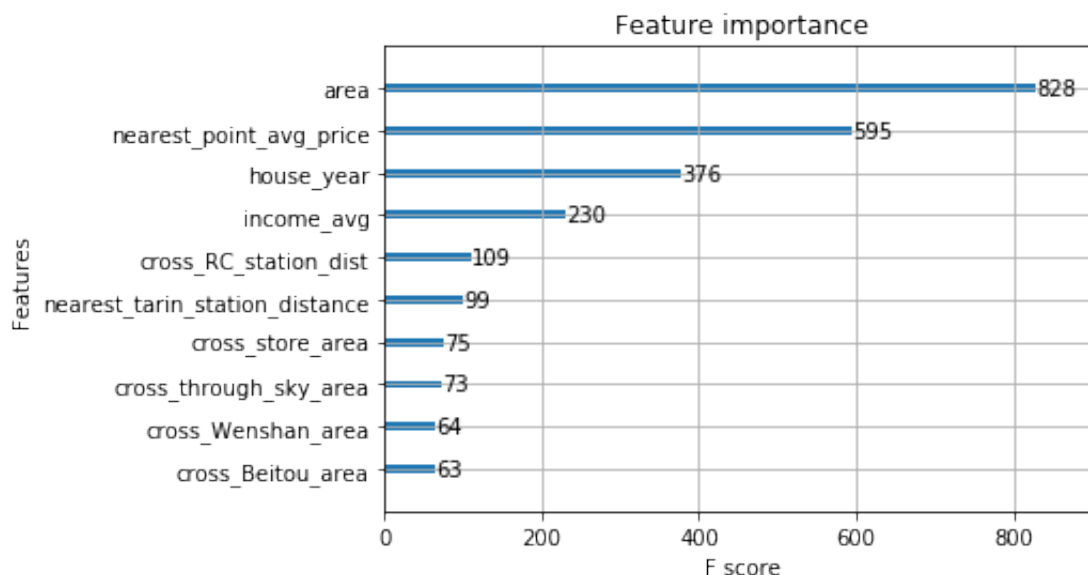
In [12]: df_result = pd.DataFrame()

        # 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 *(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[12]:
```
      predict       truth     error
0  15413886.0  15880006.78  0.029353
1  12065383.0  10999982.00  0.096855
2  29951496.0  28199982.04  0.062110
3  23521218.0  21920043.69  0.073046
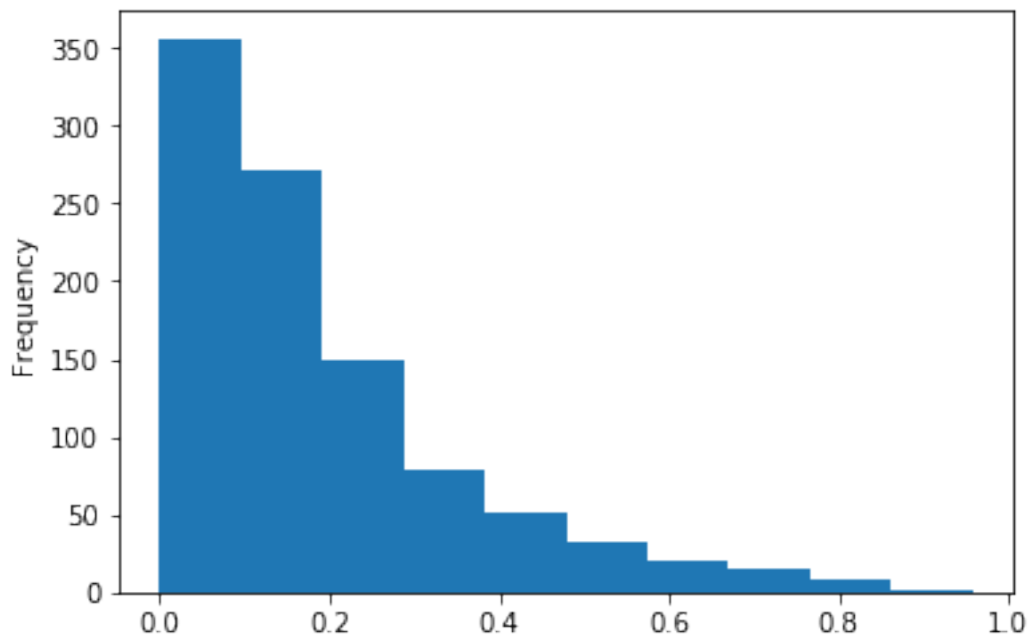4   5400714.5   3220663.36  0.676895
```

In [13]:
```
# 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 [14]:
```
# 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()
```