FINAL CHALLENGE **Name**: Manoj Gedela
Kaggle Username: mg56648
Leaderboard/Display name: Achilles last stand

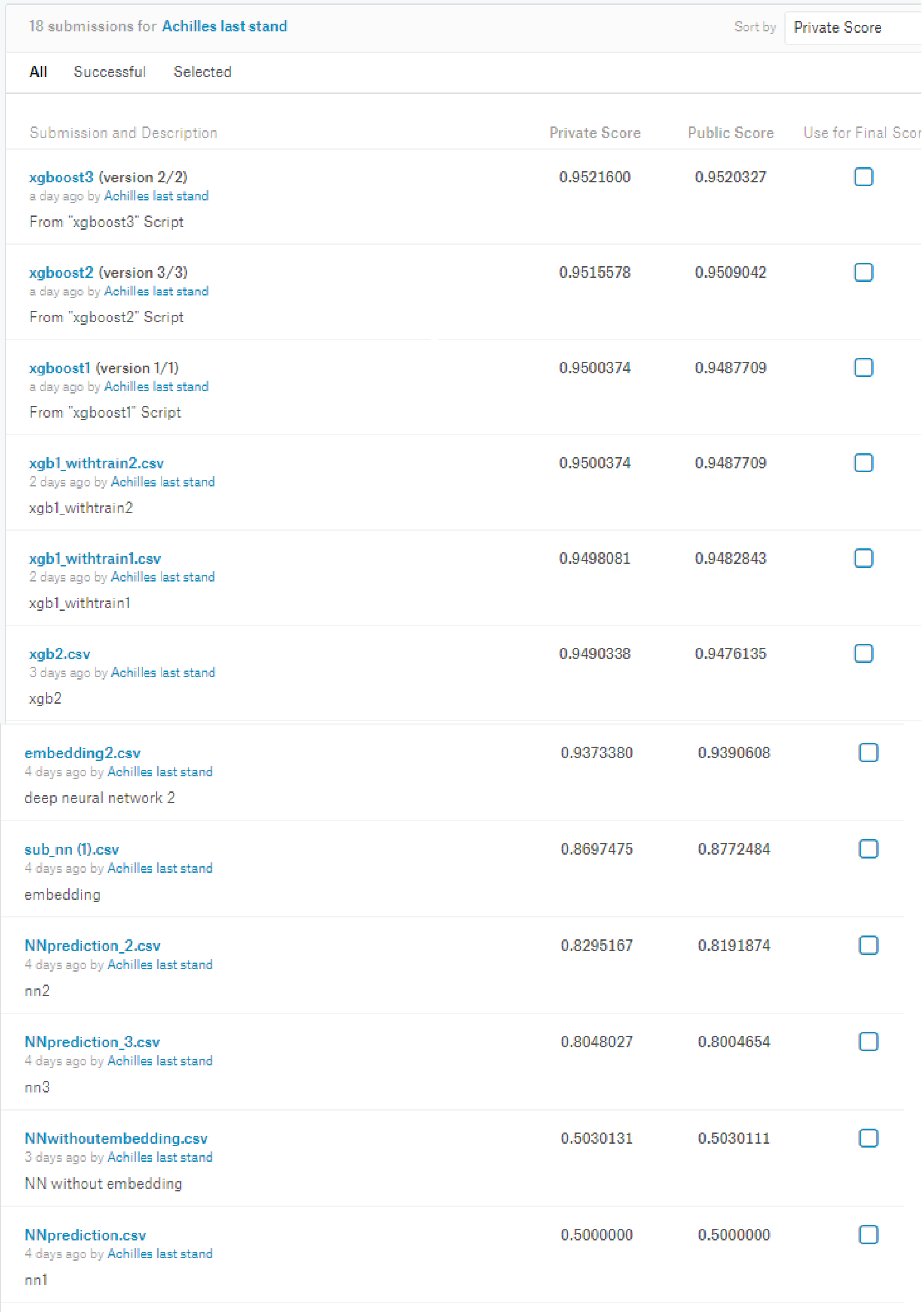
Values:

Model	Kaggle Test AUC(private score)	Comments
Neural network	0.8295167	Kernel_initializer = ‘normal’; activation = ‘relu’; Activation(“sigmoid”), Dense(48), epochs = 3, batch_size = 64
Neural network with embedding	0.937338	Same parameters as above but with embedding
xgboost	**0.95216**	• Train extra 2 was created with train_extra, train_sample and some selected rows in the dataset • parameter tuning was done to find out the optimal parameters • These optimal parameters was used in the final code

Best Model: xgboost

Best AUC: 0.95216

Submissions: Username : Achilles Last stand

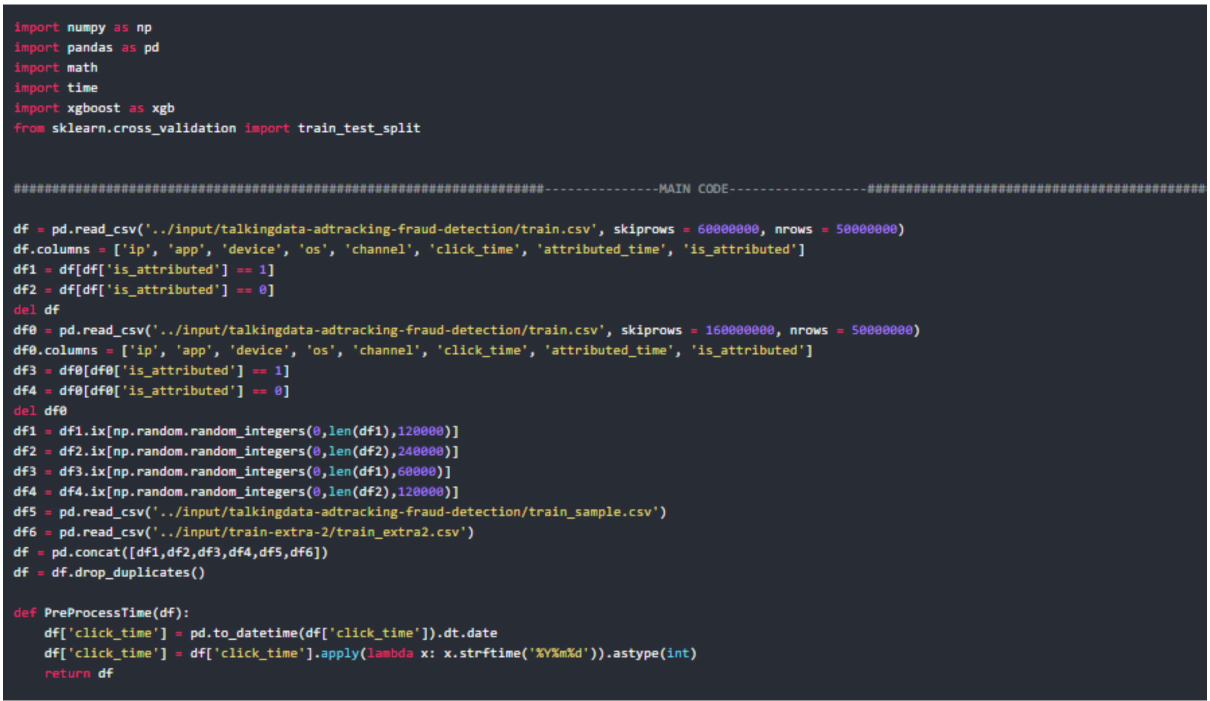


18 submissions for Achilles last stand

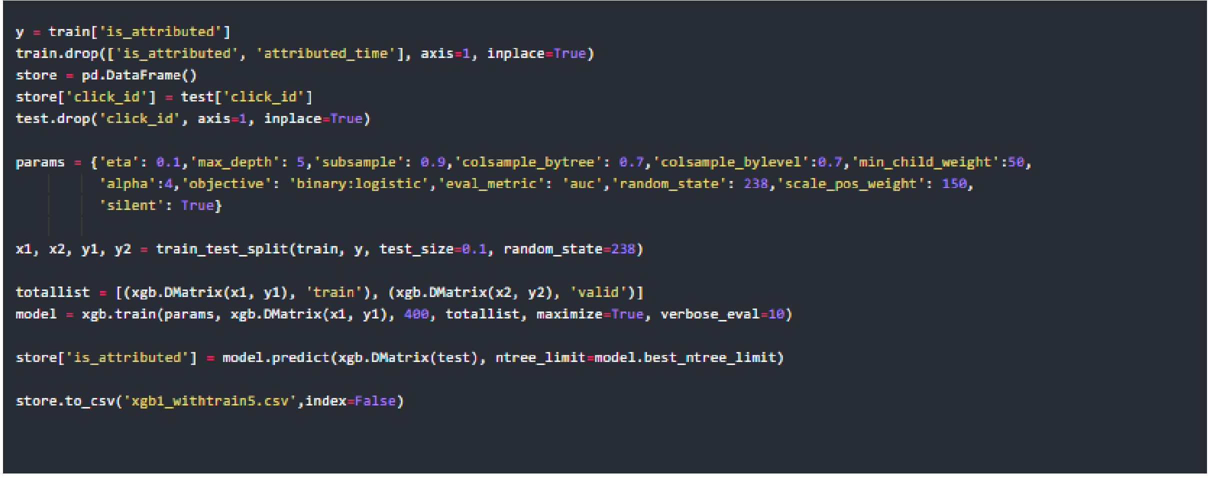
Submission and Description	Private Score	Public Score	Use for Final Scor
xgboost3 (version 2/2) a day ago by Achilles last stand From "xgboost3" Script	0.9521600	0.9520327	☐
xgboost2 (version 3/3) a day ago by Achilles last stand From "xgboost2" Script	0.9515578	0.9509042	☐
xgboost1 (version 1/1) a day ago by Achilles last stand From "xgboost1" Script	0.9500374	0.9487709	☐
xgb1_withtrain2.csv 2 days ago by Achilles last stand xgb1_withtrain2	0.9500374	0.9487709	☐
xgb1_withtrain1.csv 2 days ago by Achilles last stand xgb1_withtrain1	0.9498081	0.9482843	☐
xgb2.csv 3 days ago by Achilles last stand xgb2	0.9490338	0.9476135	☐
embedding2.csv 4 days ago by Achilles last stand deep neural network 2	0.9373380	0.9390608	☐
sub_nn (1).csv 4 days ago by Achilles last stand embedding	0.8697475	0.8772484	☐
NNprediction_2.csv 4 days ago by Achilles last stand nn2	0.8295167	0.8191874	☐
NNprediction_3.csv 4 days ago by Achilles last stand nn3	0.8048027	0.8004654	☐
NNwithoutembedding.csv 3 days ago by Achilles last stand NN without embedding	0.5030131	0.5030111	☐
NNprediction.csv 4 days ago by Achilles last stand nn1	0.5000000	0.5000000	☐

CODE SCREENSHOTS:

Main Code:

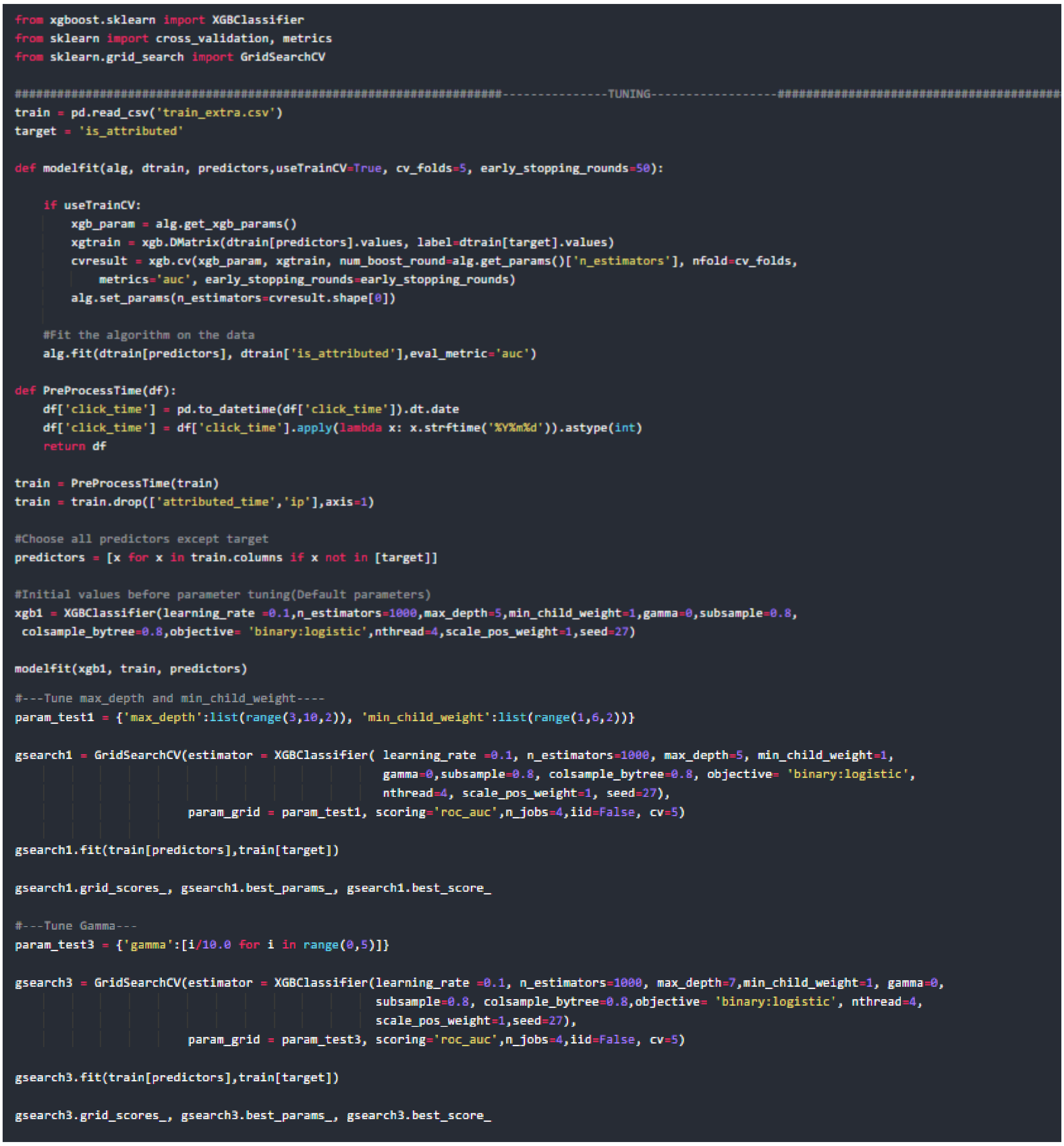


```
import numpy as np
import pandas as pd
import math
import time
import xgboost as xgb
from sklearn.cross_validation import train_test_split


#####################################################################---------------MAIN CODE------------------############################################

df = pd.read_csv('../input/talkingdata-adtracking-fraud-detection/train.csv', skiprows = 60000000, nrows = 50000000)
df.columns = ['ip', 'app', 'device', 'os', 'channel', 'click_time', 'attributed_time', 'is_attributed']
df1 = df[df['is_attributed'] == 1]
df2 = df[df['is_attributed'] == 0]
del df
df0 = pd.read_csv('../input/talkingdata-adtracking-fraud-detection/train.csv', skiprows = 160000000, nrows = 50000000)
df0.columns = ['ip', 'app', 'device', 'os', 'channel', 'click_time', 'attributed_time', 'is_attributed']
df3 = df0[df0['is_attributed'] == 1]
df4 = df0[df0['is_attributed'] == 0]
del df0
df1 = df1.ix[np.random.random_integers(0,len(df1),120000)]
df2 = df2.ix[np.random.random_integers(0,len(df2),240000)]
df3 = df3.ix[np.random.random_integers(0,len(df1),60000)]
df4 = df4.ix[np.random.random_integers(0,len(df2),120000)]
df5 = pd.read_csv('../input/talkingdata-adtracking-fraud-detection/train_sample.csv')
df6 = pd.read_csv('../input/train-extra-2/train_extra2.csv')
df = pd.concat([df1,df2,df3,df4,df5,df6])
df = df.drop_duplicates()

def PreProcessTime(df):
    df['click_time'] = pd.to_datetime(df['click_time']).dt.date
    df['click_time'] = df['click_time'].apply(lambda x: x.strftime('%Y%m%d')).astype(int)
    return df
```

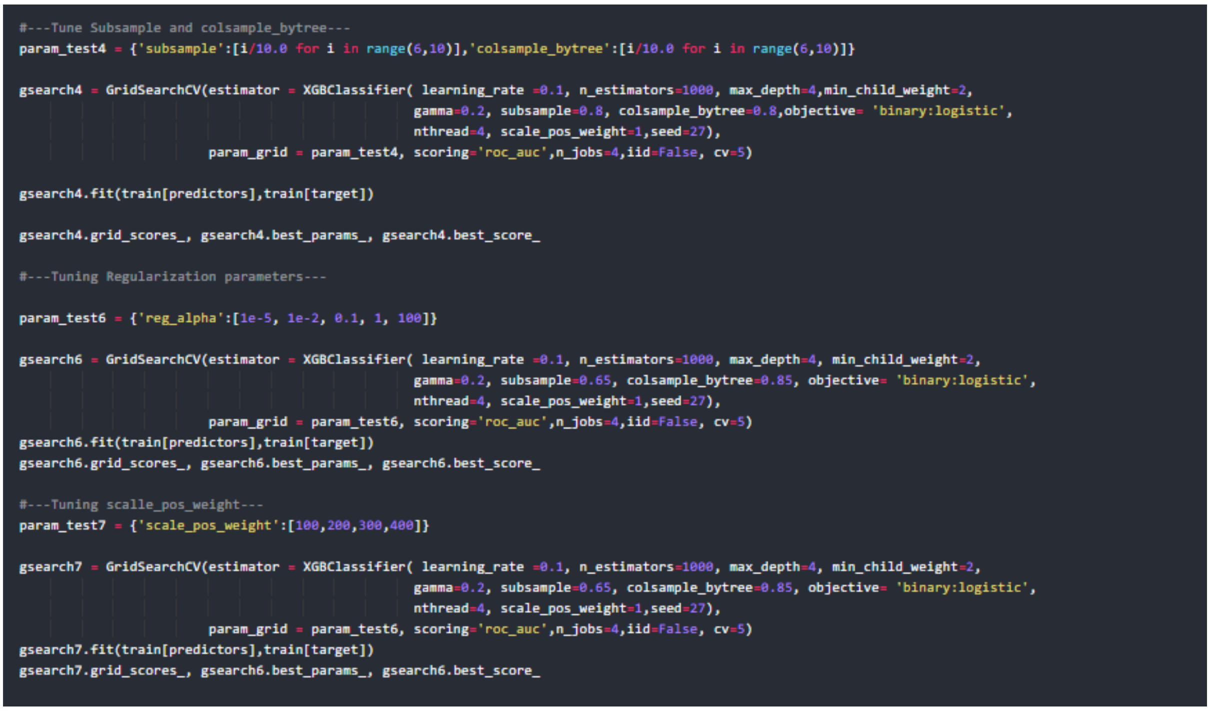


```
y = train['is_attributed']
train.drop(['is_attributed', 'attributed_time'], axis=1, inplace=True)
store = pd.DataFrame()
store['click_id'] = test['click_id']
test.drop('click_id', axis=1, inplace=True)

params = {'eta': 0.1,'max_depth': 5,'subsample': 0.9,'colsample_bytree': 0.7,'colsample_bylevel':0.7,'min_child_weight':50,
          'alpha':4,'objective': 'binary:logistic','eval_metric': 'auc','random_state': 238,'scale_pos_weight': 150,
          'silent': True}

x1, x2, y1, y2 = train_test_split(train, y, test_size=0.1, random_state=238)

totallist = [(xgb.DMatrix(x1, y1), 'train'), (xgb.DMatrix(x2, y2), 'valid')]
model = xgb.train(params, xgb.DMatrix(x1, y1), 400, totallist, maximize=True, verbose_eval=10)

store['is_attributed'] = model.predict(xgb.DMatrix(test), ntree_limit=model.best_ntree_limit)

store.to_csv('xgb1_withtrain5.csv',index=False)
```

Code for tuning to find out the optimal parameters to be used in the final code:



```python
from xgboost.sklearn import XGBClassifier
from sklearn import cross_validation, metrics
from sklearn.grid_search import GridSearchCV

#####################################################################---------------TUNING-----------------########################################
train = pd.read_csv('train_extra.csv')
target = 'is_attributed'

def modelfit(alg, dtrain, predictors,useTrainCV=True, cv_folds=5, early_stopping_rounds=50):

    if useTrainCV:
        xgb_param = alg.get_xgb_params()
        xgtrain = xgb.DMatrix(dtrain[predictors].values, label=dtrain[target].values)
        cvresult = xgb.cv(xgb_param, xgtrain, num_boost_round=alg.get_params()['n_estimators'], nfold=cv_folds,
            metrics='auc', early_stopping_rounds=early_stopping_rounds)
        alg.set_params(n_estimators=cvresult.shape[0])

    #Fit the algorithm on the data
    alg.fit(dtrain[predictors], dtrain['is_attributed'],eval_metric='auc')

def PreProcessTime(df):
    df['click_time'] = pd.to_datetime(df['click_time']).dt.date
    df['click_time'] = df['click_time'].apply(lambda x: x.strftime('%Y%m%d')).astype(int)
    return df

train = PreProcessTime(train)
train = train.drop(['attributed_time','ip'],axis=1)

#Choose all predictors except target
predictors = [x for x in train.columns if x not in [target]]

#Initial values before parameter tuning(Default parameters)
xgb1 = XGBClassifier(learning_rate =0.1,n_estimators=1000,max_depth=5,min_child_weight=1,gamma=0,subsample=0.8,
 colsample_bytree=0.8,objective= 'binary:logistic',nthread=4,scale_pos_weight=1,seed=27)

modelfit(xgb1, train, predictors)

#---Tune max_depth and min_child_weight----
param_test1 = {'max_depth':list(range(3,10,2)), 'min_child_weight':list(range(1,6,2))}

gsearch1 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators=1000, max_depth=5, min_child_weight=1,
                                                    gamma=0,subsample=0.8, colsample_bytree=0.8, objective= 'binary:logistic',
                                                    nthread=4, scale_pos_weight=1, seed=27),
                        param_grid = param_test1, scoring='roc_auc',n_jobs=4,iid=False, cv=5)

gsearch1.fit(train[predictors],train[target])

gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_

#---Tune Gamma---
param_test3 = {'gamma':[i/10.0 for i in range(0,5)]}

gsearch3 = GridSearchCV(estimator = XGBClassifier(learning_rate =0.1, n_estimators=1000, max_depth=7,min_child_weight=1, gamma=0,
                                                   subsample=0.8, colsample_bytree=0.8,objective= 'binary:logistic', nthread=4,
                                                   scale_pos_weight=1,seed=27),
                        param_grid = param_test3, scoring='roc_auc',n_jobs=4,iid=False, cv=5)

gsearch3.fit(train[predictors],train[target])

gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_
```



```
#---Tune Subsample and colsample_bytree---
param_test4 = {'subsample':[i/10.0 for i in range(6,10)],'colsample_bytree':[i/10.0 for i in range(6,10)]}

gsearch4 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators=1000, max_depth=4,min_child_weight=2,
                                                  gamma=0.2, subsample=0.8, colsample_bytree=0.8,objective= 'binary:logistic',
                                                  nthread=4, scale_pos_weight=1,seed=27),
                        param_grid = param_test4, scoring='roc_auc',n_jobs=4,iid=False, cv=5)

gsearch4.fit(train[predictors],train[target])

gsearch4.grid_scores_, gsearch4.best_params_, gsearch4.best_score_

#---Tuning Regularization parameters---

param_test6 = {'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]}

gsearch6 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators=1000, max_depth=4, min_child_weight=2,
                                                  gamma=0.2, subsample=0.65, colsample_bytree=0.85, objective= 'binary:logistic',
                                                  nthread=4, scale_pos_weight=1,seed=27),
                        param_grid = param_test6, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch6.fit(train[predictors],train[target])
gsearch6.grid_scores_, gsearch6.best_params_, gsearch6.best_score_

#---Tuning scalle_pos_weight---
param_test7 = {'scale_pos_weight':[100,200,300,400]}

gsearch7 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators=1000, max_depth=4, min_child_weight=2,
                                                  gamma=0.2, subsample=0.65, colsample_bytree=0.85, objective= 'binary:logistic',
                                                  nthread=4, scale_pos_weight=1,seed=27),
                        param_grid = param_test6, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch7.fit(train[predictors],train[target])
gsearch7.grid_scores_, gsearch6.best_params_, gsearch6.best_score_
```