Reply

CLASSIFICATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/CLASSIFICATION/)

INTERMEDIATE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/INTERMEDIATE/)

MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/)

PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)

STRUCTURED DATA (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/STRUCTURED-DATA/)

SUPERVISED (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/SUPERVISED/)

TECHNIQUE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/TECHNIQUE/)

Complete Guide to Parameter Tuning in XGBoost with codes in Python

Unlock 2020

AARSHAY JAIN (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/AARSHAY/), MARCH 1, 2016

Overview

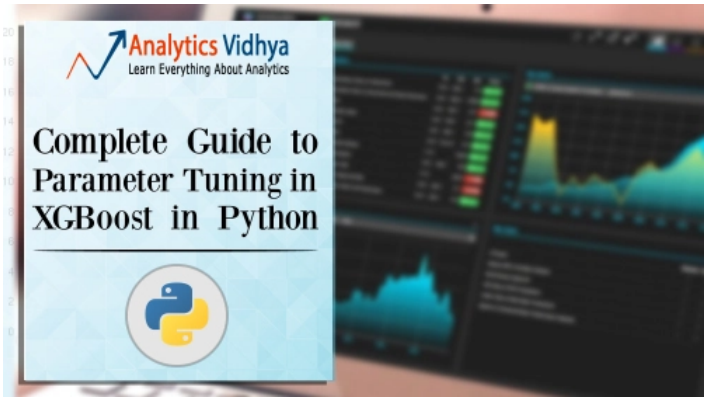
- XGBoost is a powerful machine learning (https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=parameter-tuning-xgboost) algorithm especially where speed and accuracy are concerned
- We need to consider different parameters and their values to be specified while implementing an XGBoost model
- The XGBoost model requires parameter tuning to improve and fully leverage its advantages over other algorithms

Introduction

If things don't go your way in predictive modeling, use XGboost. XGBoost algorithm has become the ultimate weapon of many data scientist. It's a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data.

Building a model using XGBoost is easy. But, improving the model using XGBoost is difficult (at least I struggled a lot). This algorithm uses multiple parameters. To improve the model, parameter tuning is must. It is very difficult to get answers to practical questions like – Which set of parameters you should tune ? What is the ideal value of these parameters to obtain optimal output ?

This article is best suited to people who are new to XGBoost. In this article, we'll learn the art of parameter tuning along with some useful information about XGBoost. Also, we'll practice this algorithm using a data set in Python.



POPULAR POSTS

6 Open Source Data Science Projects to Impress your Interviewer (https://www.analyticsvidhya.com/blog/2020/06/6-open-source-data-science-projects-interviewer/)

5 Powerful Python IDEs for Writing Analytics and Data Science Code (https://www.analyticsvidhya.com/blog/2020/06/5-python-ide-analytics-data-science-programming/)

22 Widely Used Data Science and Machine Learning Tools in 2020 (https://www.analyticsvidhya.com/blog/2020/06/22-tools-data-science-machine-learning/)

5 Powerful Excel Dashboards for Analytics Professionals (https://www.analyticsvidhya.com/blog/2020/06/5-excel-dashboards-analytics/)

3 Advanced Excel Charts Every Analytics Professional Should Try (https://www.analyticsvidhya.com/blog/2020/06/3-advanced-excel-charts-every-analytics-professional-should-try/)

40 Questions to test a data scientist on Machine Learning [Solution: SkillPower – Machine

I've always admired the boosting capabilities that this algorithm infuses in a predictive model. When I explored more about its performance and science behind its high accuracy, I discovered many advantages:

1. Regularization:
   - Standard GBM implementation has no regularization (https://www.analyticsvidhya.com/blog/2015/02/avoid-over-fitting-regularization/) like XGBoost, therefore it also helps to reduce overfitting.
   - In fact, XGBoost is also known as a '**regularized boosting**' technique.

2. Parallel Processing:
   - XGBoost implements parallel processing and is **blazingly faster** as compared to GBM.
   - But hang on, we know that boosting (https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/) is a sequential process so how can it be parallelized? We know that each tree can be built only after the previous one, so what stops us from making a tree using all cores? I hope you get where I'm coming from. Check this link (http://zhanpengfang.github.io/418home.html) out to explore further.
   - XGBoost also supports implementation on Hadoop.

3. High Flexibility
   - XGBoost allows users to define **custom optimization objectives and evaluation criteria**.
   - This adds a whole new dimension to the model and there is no limit to what we can do.

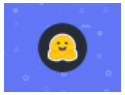
4. Handling Missing Values
   - XGBoost has an in-built routine to handle missing values.
   - The user is required to supply a different value than other observations and pass that as a parameter. XGBoost tries different things as it encounters a missing value on each node and learns which path to take for missing values in future.

5. Tree Pruning:
   - A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a **greedy algorithm**.
   - XGBoost on the other hand make **splits upto the max_depth** specified and then start **pruning** the tree backwards and remove splits beyond which there is no positive gain.
   - Another advantage is that sometimes a split of negative loss say -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGBoost will go deeper and it will see a combined effect of +8 of the split and keep both.

6. Built-in Cross-Validation
   - XGBoost allows user to run a **cross-validation at each iteration** of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.
   - This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

7. Continue on Existing Model

JUNE 29, 2020



(https://www.analyticsvidhya.com/blog/2020/06/hugging-face-tokenizers-nlp-library/)

Hugging Face Releases New NLP 'Tokenizers' Library Version (v0.8.0) (https://www.analyticsvidhya.com/blog/2020/06/hugging-face-tokenizers-nlp-library/)

JUNE 27, 2020

(https://www.analyticsvidhya.com/blog/2020/06/22-tools-data-science-machine-learning/)

22 Widely Used Data Science and Machine Learning Tools in 2020 (https://www.analyticsvidhya.com/blog/2020/06/22-tools-data-science-machine-learning/)

JUNE 27, 2020

(https://www.analyticsvidhya.com/blog/2020/06/3-building-blocks-machine-learning-data-scientist/)

3 Building Blocks of Machine Learning you Should Know as a Data Scientist (https://www.analyticsvidhya.com/blog/2020/06/3-building-blocks-machine-learning-data-scientist/)

JUNE 26, 2020

- User can start training an XGBoost model from its last iteration of previous run. This can be of significant advantage in certain specific applications.
- GBM implementation of sklearn also has this feature so they are even on this point.

I hope now you understand the sheer power XGBoost algorithm. Note that these are the points which I could muster. You know a few more? Feel free to drop a comment below and I will update the list.

Did I whet your appetite ? Good. You can refer to following web-pages for a deeper understanding:

- XGBoost Guide – Introduction to Boosted Trees (http://xgboost.readthedocs.org/en/latest/model.html)
- Words from the Author of XGBoost (https://www.youtube.com/watch?v=X47SGnTMZIU) [Video]

2. XGBoost Parameters

The overall parameters have been divided into 3 categories by XGBoost authors:

1. **General Parameters:** Guide the overall functioning
2. **Booster Parameters:** Guide the individual booster (tree/regression) at each step
3. **Learning Task Parameters:** Guide the optimization performed

I will give analogies to GBM here and highly recommend to read this article (https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/) to learn from the very basics.

General Parameters

These define the overall functionality of XGBoost.

1. **booster [default=gbtree]**
    - Select the type of model to run at each iteration. It has 2 options:
        - gbtree: tree-based models
        - gblinear: linear models

2. **silent [default=0]:**
    - Silent mode is activated is set to 1, i.e. no running messages will be printed.
    - It's generally good to keep it 0 as the messages might help in understanding the model.

3. **nthread [default to maximum number of threads available if not set]**
    - This is used for parallel processing and number of cores in the system should be entered
    - If you wish to run on all cores, value should not be entered and algorithm will detect automatically

There are 2 more parameters which are set automatically by XGBoost and you need not worry about them. Lets move on to Booster parameters.

What should you know?

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm. Since I covered Gradient Boosting Machine in detail in my previous article – Complete Guide to Parameter Tuning in Gradient Boosting (GBM) in Python (https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/), I highly recommend going through that before reading further. It will help you bolster your understanding of boosting in general and parameter tuning for GBM.

Special Thanks: Personally, I would like to acknowledge the timeless support provided by Mr. Sudalai Rajkumar (https://www.linkedin.com/in/sudalairajkumar) (aka SRK), currently AV Rank 2 (http://datahack.analyticsvidhya.com/user/profile/SRK). This article wouldn't be possible without his help. He is helping us guide thousands of data scientists. A big thanks to SRK!

Project to apply XGBoost

Problem Statement

HR analytics is revolutionizing the way human resources departments operate, leading to higher efficiency and better results overall. Human resources have been using analytics for years.

However, the collection, processing, and analysis of data have been largely manual, and given the nature of human resources dynamics and HR KPIs, the approach has been constraining HR. Therefore, it is surprising that HR departments woke up to the utility of machine learning (https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=parameter-tuning-xgboost) so late in the game. Here is an opportunity to try predictive analytics in identifying the employees most likely to get promoted.

Practice Now (http://datahack.analyticsvidhya.com/contest/wns-analytics-hackathon-2018-1/?utm_source=av_blog&utm_medium=practice_blog_xgboost)

Table of Contents

1. The XGBoost Advantage

Learning, DataFest 2017] (https://www.analyticsvidhya.com/blog/2017/04/40-questions-test-data-scientist-machine-learning-solution-skillpower-machine-learning-datafest-2017/)

Commonly used Machine Learning Algorithms (with Python and R Codes) (https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/)

40 Questions to test a Data Scientist on Clustering Techniques (Skill test Solution) (https://www.analyticsvidhya.com/blog/2017/02/test-data-scientist-clustering/)

RECENT POSTS

(https://www.analyticsvidhya.com/blog/2020/06/nlp-project-information-extraction/)

Hands-on NLP Project: A Comprehensive Guide to Information Extraction using Python (https://www.analyticsvidhya.com/blog/2020/06/nlp-project-information-extraction/)

Booster Parameters

Though there are 2 types of boosters, I'll consider only **tree booster** here because it always outperforms the linear booster and thus the later is rarely used.

1. **eta [default=0.3]**
    - Analogous to learning rate in GBM
    - Makes the model more robust by shrinking the weights on each step
    - Typical final values to be used: 0.01-0.2

2. **min_child_weight [default=1]**
    - Defines the minimum sum of weights of all observations required in a child.
    - This is similar to **min_child_leaf** in GBM but not exactly. This refers to min "sum of weights" of observations while GBM has min "number of observations".
    - Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
    - Too high values can lead to under-fitting hence, it should be tuned using CV.

3. **max_depth [default=6]**
    - The maximum depth of a tree, same as GBM.
    - Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
    - Should be tuned using CV.
    - Typical values: 3-10

4. **max_leaf_nodes**
    - The maximum number of terminal nodes or leaves in a tree.
    - Can be defined in place of max_depth. Since binary trees are created, a depth of 'n' would produce a maximum of 2^n leaves.
    - If this is defined, GBM will ignore max_depth.

5. **gamma [default=0]**
    - A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
    - Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

6. **max_delta_step [default=0]**
    - In maximum delta step we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative.
    - Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.
    - This is generally not used but you can explore further if you wish.

7. **subsample [default=1]**
    - Same as the subsample of GBM. Denotes the fraction of observations to be randomly samples for each tree.
    - Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting.

- Typical values: 0.5-1

8. **colsample_bytree [default=1]**
   - Similar to max_features in GBM. Denotes the fraction of columns to be randomly samples for each tree.
   - Typical values: 0.5-1

9. **colsample_bylevel [default=1]**
   - Denotes the subsample ratio of columns for each split, in each level.
   - I don't use this often because subsample and colsample_bytree will do the job for you. but you can explore further if you feel so.

10. **lambda [default=1]**
    - L2 regularization term on weights (analogous to Ridge regression)
    - This used to handle the regularization part of XGBoost. Though many data scientists don't use it often, it should be explored to reduce overfitting.

11. **alpha [default=0]**
    - L1 regularization term on weight (analogous to Lasso regression)
    - Can be used in case of very high dimensionality so that the algorithm runs faster when implemented

12. **scale_pos_weight [default=1]**
    - A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence.

Learning Task Parameters

These parameters are used to define the optimization objective the metric to be calculated at each step.

1. **objective [default=reg:linear]**
   - This defines the loss function to be minimized. Mostly used values are:
     - **binary:logistic** –logistic regression for binary classification, returns predicted probability (not class)
     - **multi:softmax** –multiclass classification using the softmax objective, returns predicted class (not probabilities)
       - you also need to set an additional **num_class** (number of classes) parameter defining the number of unique classes
     - **multi:softprob** –same as softmax, but returns predicted probability of each data point belonging to each class.

2. **eval_metric [default according to objective]**
   - The metric to be used for validation data.
   - The default values are rmse for regression and error for classification.
   - Typical values are:
     - **rmse** – root mean square error
     - **mae** – mean absolute error
     - **logloss** – negative log-likelihood
     - **error** – Binary classification error rate (0.5 threshold)

- **merror** – Multiclass classification error rate
- **mlogloss** – Multiclass logloss
- **auc:** Area under the curve

3. **seed [default=0]**
    - The random number seed.
    - Can be used for generating reproducible results and also for parameter tuning.

If you've been using Scikit-Learn till now, these parameter names might not look familiar. A good news is that xgboost module in python has an sklearn wrapper called XGBClassifier. It uses sklearn style naming convention. The parameters names which will change are:

1. eta –> learning_rate
2. lambda –> reg_lambda
3. alpha –> reg_alpha

You must be wondering that we have defined everything except something similar to the "n_estimators" parameter in GBM. Well this exists as a parameter in XGBClassifier. However, it has to be passed as "num_boosting_rounds" while calling the fit function in the standard xgboost implementation.

I recommend you to go through the following parts of xgboost guide to better understand the parameters and codes:

1. XGBoost Parameters (official guide) (http://xgboost.readthedocs.org/en/latest/parameter.html#general-parameters)
2. XGBoost Demo Codes (xgboost GitHub repository) (https://github.com/dmlc/xgboost/tree/master/demo/guide-python)
3. Python API Reference (official guide) (http://xgboost.readthedocs.org/en/latest/python/python_api.html)

3. Parameter Tuning with Example

We will take the data set from Data Hackathon 3.x AV hackathon, same as that taken in the GBM article (https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/). The details of the problem can be found on the competition page (http://datahack.analyticsvidhya.com/contest/data-hackathon-3x). You can download the data set from here (https://www.analyticsvidhya.com/wp-content/uploads/2016/02/Dataset.rar). I have performed the following steps:

1. City variable dropped because of too many categories
2. DOB converted to Age | DOB dropped
3. EMI_Loan_Submitted_Missing created which is 1 if EMI_Loan_Submitted was missing else 0 | Original variable EMI_Loan_Submitted dropped
4. EmployerName dropped because of too many categories
5. Existing_EMI imputed with 0 (median) since only 111 values were missing
6. Interest_Rate_Missing created which is 1 if Interest_Rate was missing else 0 | Original variable Interest_Rate dropped
7. Lead_Creation_Date dropped because made little intuitive impact on outcome
8. Loan_Amount_Applied, Loan_Tenure_Applied imputed with median values

9. Loan_Amount_Submitted_Missing created which is 1 if Loan_Amount_Submitted was missing else 0 | Original variable Loan_Amount_Submitted dropped
10. Loan_Tenure_Submitted_Missing created which is 1 if Loan_Tenure_Submitted was missing else 0 | Original variable Loan_Tenure_Submitted dropped
11. LoggedIn, Salary_Account dropped
12. Processing_Fee_Missing created which is 1 if Processing_Fee was missing else 0 | Original variable Processing_Fee dropped
13. Source – top 2 kept as is and all others combined into different category
14. Numerical and One-Hot-Coding performed

For those who have the original data from competition, you can check out these steps from the data_preparation iPython notebook in the repository.

Lets start by importing the required libraries and loading the data:

```
#Import libraries:
import pandas as pd
import numpy as np
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
from sklearn import cross_validation, metrics   #Additional scklearn functions
from sklearn.grid_search import GridSearchCV   #Perforing grid search

import matplotlib.pylab as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 12, 4

train = pd.read_csv('train_modified.csv')
target = 'Disbursed'
IDcol = 'ID'
```

Note that I have imported 2 forms of XGBoost:

1. **xgb** – this is the direct xgboost library. I will use a specific function “cv” from this library
2. **XGBClassifier** – this is an sklearn wrapper for XGBoost. This allows us to use sklearn’s Grid Search with parallel processing in the same way we did for GBM

Before proceeding further, lets define a function which will help us create XGBoost models and perform cross-validation. The best part is that you can take this function as it is and use it later for your own models.

```
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, show_progress=False)
        alg.set_params(n_estimators=cvresult.shape[0])

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

    #Predict training set:
    dtrain_predictions = alg.predict(dtrain[predictors])
    dtrain_predprob = alg.predict_proba(dtrain[predictors])[:,1]

    #Print model report:
    print "\nModel Report"
    print "Accuracy : %.4g" % metrics.accuracy_score(dtrain['Disbursed'].values, dtrain_predictions)
    print "AUC Score (Train): %f" % metrics.roc_auc_score(dtrain['Disbursed'], dtrain_predprob)

    feat_imp = pd.Series(alg.booster().get_fscore()).sort_values(ascending=False)
    feat_imp.plot(kind='bar', title='Feature Importances')
    plt.ylabel('Feature Importance Score')
```

This code is slightly different from what I used for GBM. The focus of this article is to cover the concepts and not coding. Please feel free to drop a note in the comments if you find any challenges in understanding any part of it. Note that xgboost's sklearn wrapper doesn't have a "feature_importances" metric but a get_fscore() function which does the same job.

General Approach for Parameter Tuning

We will use an approach similar to that of GBM here. The various steps to be performed are:

1. Choose a relatively **high learning rate**. Generally a learning rate of 0.1 works but somewhere between 0.05 to 0.3 should work for different problems. Determine the **optimum number of trees for this learning rate**. XGBoost has a very useful function

called as “cv” which performs cross-validation at each boosting iteration and thus returns the optimum number of trees required.
2. **Tune tree-specific parameters** (max_depth, min_child_weight, gamma, subsample, colsample_bytree) for decided learning rate and number of trees. Note that we can choose different parameters to define a tree and I’ll take up an example here.
3. Tune **regularization parameters** (lambda, alpha) for xgboost which can help reduce model complexity and enhance performance.
4. **Lower the learning rate** and decide the optimal parameters .

Let us look at a more detailed step by step approach.

Step 1: Fix learning rate and number of estimators for tuning tree-based parameters

In order to decide on boosting parameters, we need to set some initial values of other parameters. Lets take the following values:

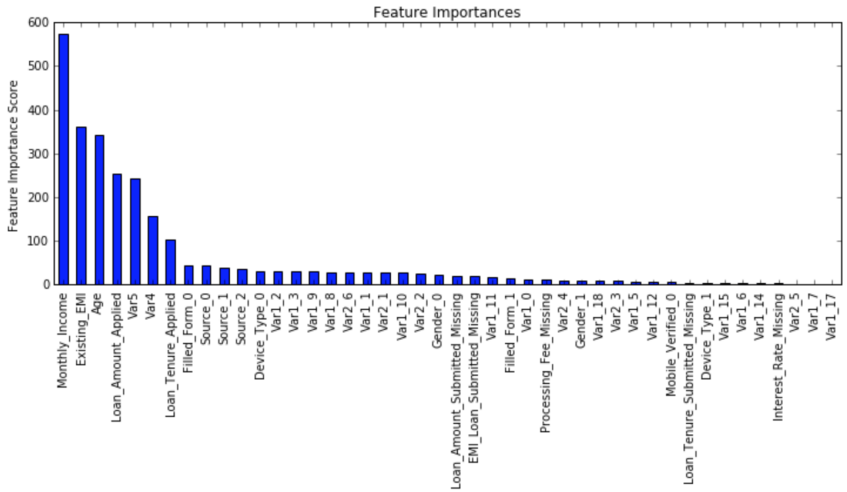
1. **max_depth = 5** : This should be between 3-10. I’ve started with 5 but you can choose a different number as well. 4-6 can be good starting points.
2. **min_child_weight = 1** : A smaller value is chosen because it is a highly imbalanced class problem and leaf nodes can have smaller size groups.
3. **gamma = 0** : A smaller value like 0.1-0.2 can also be chosen for starting. This will anyways be tuned later.
4. **subsample, colsample_bytree = 0.8** : This is a commonly used used start value. Typical values range between 0.5-0.9.
5. **scale_pos_weight = 1**: Because of high class imbalance.

Please note that all the above are just initial estimates and will be tuned later. Lets take the default learning rate of 0.1 here and check the optimum number of trees using cv function of xgboost. The function defined above will do it for us.

```
#Choose all predictors except target & IDcols
predictors = [x for x in train.columns if x not in [target, IDcol]]
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)
```

```
Will train until cv error hasn't decreased in 50 rounds.
Stopping. Best iteration:
[140] cv-mean:0.843638  cv-std:0.0141274405467
```

```
Model Report
Accuracy : 0.9854
AUC Score (Train): 0.899857
AUC Score (Test): 0.847934
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/1.-inital.png)

As you can see that here we got 140 as the optimal estimators for 0.1 learning rate. Note that this value might be too high for you depending on the power of your system. In that case you can increase the learning rate and re-run the command to get the reduced number of estimators.

Note: You will see the test AUC as "AUC Score (Test)" in the outputs here. But this would not appear if you try to run the command on your system as the data is not made public. It's provided here just for reference. The part of the code which generates this output has been removed here.

Step 2: Tune max_depth and min_child_weight

We tune these first as they will have the highest impact on model outcome. To start with, let's set wider ranges and then we will perform another iteration for smaller ranges.

Important Note: I'll be doing some heavy-duty grid searched in this section which can take 15-30 mins or even more time to run depending on your system. You can vary the number of values you are testing based on what your system can handle.

```
param_test1 = {
 'max_depth':range(3,10,2),
 'min_child_weight':range(1,6,2)
}
gsearch1 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators
=140, 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_
```

```
([mean: 0.83690, std: 0.00821, params: {'max_depth': 3, 'min_child_weight': 1},
  mean: 0.83730, std: 0.00858, params: {'max_depth': 3, 'min_child_weight': 3},
  mean: 0.83713, std: 0.00847, params: {'max_depth': 3, 'min_child_weight': 5},
  mean: 0.84051, std: 0.00748, params: {'max_depth': 5, 'min_child_weight': 1},
  mean: 0.84112, std: 0.00595, params: {'max_depth': 5, 'min_child_weight': 3},
  mean: 0.84123, std: 0.00619, params: {'max_depth': 5, 'min_child_weight': 5},
  mean: 0.83772, std: 0.00518, params: {'max_depth': 7, 'min_child_weight': 1},
  mean: 0.83672, std: 0.00579, params: {'max_depth': 7, 'min_child_weight': 3},
  mean: 0.83658, std: 0.00355, params: {'max_depth': 7, 'min_child_weight': 5},
  mean: 0.82690, std: 0.00622, params: {'max_depth': 9, 'min_child_weight': 1},
  mean: 0.82909, std: 0.00560, params: {'max_depth': 9, 'min_child_weight': 3},
  mean: 0.83211, std: 0.00707, params: {'max_depth': 9, 'min_child_weight': 5}],
 {'max_depth': 5, 'min_child_weight': 5},
 0.84123292820257589)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/2.-tree-base-1.png)

Here, we have run 12 combinations with wider intervals between values. The ideal values are **5 for max_depth** and **5 for min_child_weight**. Lets go one step deeper and look for optimum values. We'll search for values 1 above and below the optimum values because we took an interval of two.

```
param_test2 = {
 'max_depth':[4,5,6],
 'min_child_weight':[4,5,6]
}
gsearch2 = GridSearchCV(estimator = XGBClassifier( learning_rate=0.1, n_estimators=
140, max_depth=5,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test2, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch2.fit(train[predictors],train[target])
gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_
```

```
([mean: 0.84031, std: 0.00658, params: {'max_depth': 4, 'min_child_weight': 4},
  mean: 0.84061, std: 0.00700, params: {'max_depth': 4, 'min_child_weight': 5},
  mean: 0.84125, std: 0.00723, params: {'max_depth': 4, 'min_child_weight': 6},
  mean: 0.83988, std: 0.00612, params: {'max_depth': 5, 'min_child_weight': 4},
  mean: 0.84123, std: 0.00619, params: {'max_depth': 5, 'min_child_weight': 5},
  mean: 0.83995, std: 0.00591, params: {'max_depth': 5, 'min_child_weight': 6},
  mean: 0.83905, std: 0.00635, params: {'max_depth': 6, 'min_child_weight': 4},
  mean: 0.83904, std: 0.00656, params: {'max_depth': 6, 'min_child_weight': 5},
  mean: 0.83844, std: 0.00682, params: {'max_depth': 6, 'min_child_weight': 6}],
 {'max_depth': 4, 'min_child_weight': 6},
 0.84124915179964577)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/3.-tree-base-2.png)

Here, we get the optimum values as **4 for max_depth** and **6 for min_child_weight**. Also, we can see the CV score increasing slightly. Note that as the model performance increases, it becomes exponentially difficult to achieve even marginal gains in performance. You would have noticed that here we got 6 as optimum value for min_child_weight but we haven't tried values more than 6. We can do that as follow:.

```
param_test2b = {
 'min_child_weight':[6,8,10,12]
}
gsearch2b = GridSearchCV(estimator = XGBClassifier( learning_rate=0.1, n_estimators
=140, max_depth=4,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test2b, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch2b.fit(train[predictors],train[target])
```

```
modelfit(gsearch3.best_estimator_, train, predictors)
gsearch2b.grid_scores_, gsearch2b.best_params_, gsearch2b.best_score_
```

```
([mean: 0.84125, std: 0.00723, params: {'min_child_weight': 6},
  mean: 0.84028, std: 0.00710, params: {'min_child_weight': 8},
  mean: 0.83920, std: 0.00674, params: {'min_child_weight': 10},
  mean: 0.83996, std: 0.00729, params: {'min_child_weight': 12}],
 {'min_child_weight': 6},
 0.84124915179964577)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/4.-tree-base-3.png)

We see 6 as the optimal value.

Step 3: Tune gamma

Now lets tune gamma value using the parameters already tuned above. Gamma can take various values but I'll check for 5 values here. You can go into more precise values as.

```
param_test3 = {
 'gamma':[i/10.0 for i in range(0,5)]
}
gsearch3 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators
=140, max_depth=4,
 min_child_weight=6, 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_
```

```
([mean: 0.84125, std: 0.00723, params: {'gamma': 0.0},
  mean: 0.83996, std: 0.00695, params: {'gamma': 0.1},
  mean: 0.84045, std: 0.00639, params: {'gamma': 0.2},
  mean: 0.84032, std: 0.00673, params: {'gamma': 0.3},
  mean: 0.84061, std: 0.00692, params: {'gamma': 0.4}],
 {'gamma': 0.0},
 0.84124915179964577)
```

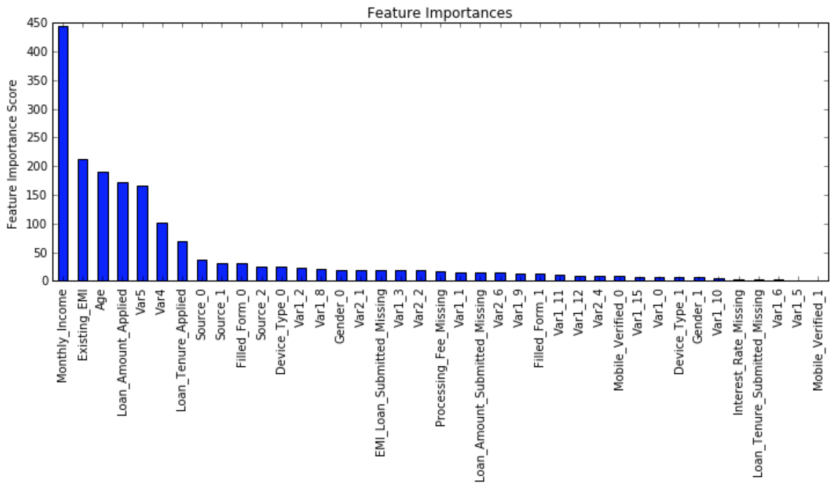
(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/5.-gamma.png)

This shows that our original value of gamma, i.e. **0 is the optimum one**. Before proceeding, a good idea would be to re-calibrate the number of boosting rounds for the updated parameters.

```
xgb2 = XGBClassifier(
 learning_rate =0.1,
 n_estimators=1000,
 max_depth=4,
 min_child_weight=6,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb2, train, predictors)
```

```
Will train until cv error hasn't decreased in 50 rounds.
Stopping. Best iteration:
[177] cv-mean:0.8451166 cv-std:0.0123406045006
```

```
Model Report
Accuracy : 0.9854
AUC Score (Train): 0.883836
AUC Score (Test): 0.848967
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/6.-xgb2.png)Here, we can see the improvement in score. So the final parameters are:

- max_depth: 4
- min_child_weight: 6
- gamma: 0

Step 4: Tune subsample and colsample_bytree

The next step would be try different subsample and colsample_bytree values. Lets do this in 2 stages as well and take values 0.6,0.7,0.8,0.9 for both to start with.

```
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
=177, max_depth=4,
 min_child_weight=6, gamma=0, 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_
```

```
([mean: 0.83688, std: 0.00849, params: {'subsample': 0.6, 'colsample_bytree': 0.6},
  mean: 0.83834, std: 0.00772, params: {'subsample': 0.7, 'colsample_bytree': 0.6},
  mean: 0.83946, std: 0.00813, params: {'subsample': 0.8, 'colsample_bytree': 0.6},
  mean: 0.83845, std: 0.00831, params: {'subsample': 0.9, 'colsample_bytree': 0.6},
  mean: 0.83816, std: 0.00651, params: {'subsample': 0.6, 'colsample_bytree': 0.7},
  mean: 0.83797, std: 0.00668, params: {'subsample': 0.7, 'colsample_bytree': 0.7},
  mean: 0.83956, std: 0.00824, params: {'subsample': 0.8, 'colsample_bytree': 0.7},
  mean: 0.83892, std: 0.00626, params: {'subsample': 0.9, 'colsample_bytree': 0.7},
  mean: 0.83914, std: 0.00794, params: {'subsample': 0.6, 'colsample_bytree': 0.8},
  mean: 0.83974, std: 0.00687, params: {'subsample': 0.7, 'colsample_bytree': 0.8},
  mean: 0.84102, std: 0.00715, params: {'subsample': 0.8, 'colsample_bytree': 0.8},
  mean: 0.84029, std: 0.00645, params: {'subsample': 0.9, 'colsample_bytree': 0.8},
  mean: 0.83881, std: 0.00723, params: {'subsample': 0.6, 'colsample_bytree': 0.9},
  mean: 0.83975, std: 0.00706, params: {'subsample': 0.7, 'colsample_bytree': 0.9},
  mean: 0.83975, std: 0.00648, params: {'subsample': 0.8, 'colsample_bytree': 0.9},
  mean: 0.83954, std: 0.00698, params: {'subsample': 0.9, 'colsample_bytree': 0.9}],
 {'colsample_bytree': 0.8, 'subsample': 0.8},
 0.8410246925643593)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/7.-gsearch-4.png)

Here, we found 0.8 as the optimum value for both subsample and colsample_bytree. Now we should try values in 0.05 interval around these.

```
param_test5 = {
 'subsample':[i/100.0 for i in range(75,90,5)],
 'colsample_bytree':[i/100.0 for i in range(75,90,5)]
}
gsearch5 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators
=177, max_depth=4,
 min_child_weight=6, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test5, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch5.fit(train[predictors],train[target])
```

```
([mean: 0.83881, std: 0.00795, params: {'subsample': 0.75, 'colsample_bytree': 0.75},
  mean: 0.84037, std: 0.00638, params: {'subsample': 0.8, 'colsample_bytree': 0.75},
  mean: 0.84013, std: 0.00685, params: {'subsample': 0.85, 'colsample_bytree': 0.75},
  mean: 0.83967, std: 0.00694, params: {'subsample': 0.75, 'colsample_bytree': 0.8},
  mean: 0.84102, std: 0.00715, params: {'subsample': 0.8, 'colsample_bytree': 0.8},
  mean: 0.84087, std: 0.00693, params: {'subsample': 0.85, 'colsample_bytree': 0.8},
  mean: 0.83836, std: 0.00738, params: {'subsample': 0.75, 'colsample_bytree': 0.85},
  mean: 0.84067, std: 0.00698, params: {'subsample': 0.8, 'colsample_bytree': 0.85},
  mean: 0.83978, std: 0.00689, params: {'subsample': 0.85, 'colsample_bytree': 0.85}],
 {'colsample_bytree': 0.8, 'subsample': 0.8},
 0.8410246925643593)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/8.-gsearcg-5.png)

Again we got the same values as before. Thus the optimum values are:

- subsample: 0.8
- colsample_bytree: 0.8

Step 5: Tuning Regularization Parameters

Next step is to apply regularization to reduce overfitting. Though many people don't use this parameters much as gamma provides a substantial way of controlling complexity. But we should always try it. I'll tune 'reg_alpha' value here and leave it upto you to try different values

of ‘reg_lambda’.

```
param_test6 = {
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]
}
gsearch6 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators
=177, max_depth=4,
 min_child_weight=6, gamma=0.1, subsample=0.8, colsample_bytree=0.8,
 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_
```

```
([mean: 0.83999, std: 0.00643, params: {'reg_alpha': 1e-05},
  mean: 0.84084, std: 0.00639, params: {'reg_alpha': 0.01},
  mean: 0.83985, std: 0.00831, params: {'reg_alpha': 0.1},
  mean: 0.83989, std: 0.00707, params: {'reg_alpha': 1},
  mean: 0.81343, std: 0.01541, params: {'reg_alpha': 100}],
 {'reg_alpha': 0.01},
 0.84084269674772316)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/9.-gsearcg-6.png)

We can see that the CV score is less than the previous case. But the values tried are very widespread, we should try values closer to the optimum here (0.01) to see if we get something better.

```
param_test7 = {
 'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05]
}
gsearch7 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators
=177, max_depth=4,
 min_child_weight=6, gamma=0.1, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test7, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch7.fit(train[predictors],train[target])
gsearch7.grid_scores_, gsearch7.best_params_, gsearch7.best_score_
```

```
([mean: 0.83999, std: 0.00643, params: {'reg_alpha': 0},
  mean: 0.83978, std: 0.00663, params: {'reg_alpha': 0.001},
  mean: 0.84118, std: 0.00651, params: {'reg_alpha': 0.005},
  mean: 0.84084, std: 0.00639, params: {'reg_alpha': 0.01},
  mean: 0.84008, std: 0.00690, params: {'reg_alpha': 0.05}],
 {'reg_alpha': 0.005},
 0.84118352535245489)
```

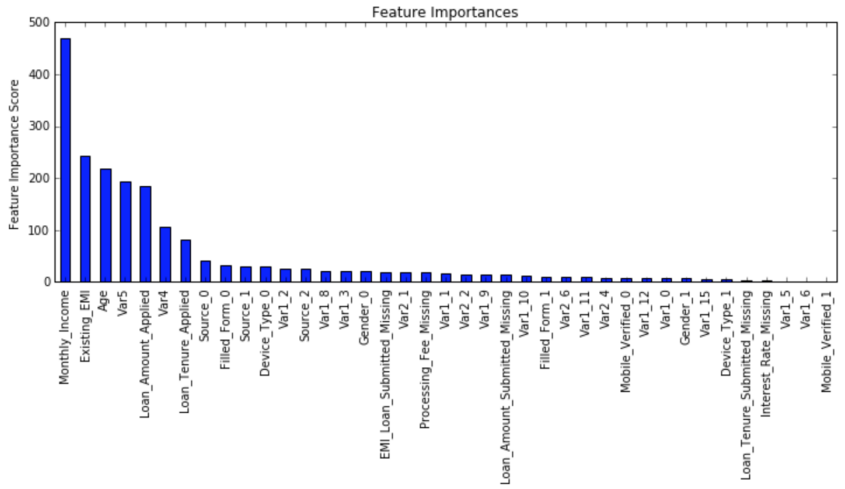
(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/10.-gsearch-7.png)

You can see that we got a better CV. Now we can apply this regularization in the model and look at the impact:

```
xgb3 = XGBClassifier(
 learning_rate =0.1,
 n_estimators=1000,
 max_depth=4,
 min_child_weight=6,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 reg_alpha=0.005,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb3, train, predictors)
```

```
Will train until cv error hasn't decreased in 50 rounds.
Stopping. Best iteration:
[188] cv-mean:0.844475  cv-std:0.0129019770268
```

```
Model Report
Accuracy : 0.9854
AUC Score (Train): 0.887149
AUC Score (Test): 0.848972
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/11.-final.png)

Again we can see slight improvement in the score.

Step 6: Reducing Learning Rate

Lastly, we should lower the learning rate and add more trees. Lets use the cv function of XGBoost to do the job again.

You can also read this article on our Mobile APP 

(//play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm_source=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1) 

(https://apps.apple.com/us/app/analytics-vidhya/id1470025572)

TAGS : GRADIENT BOOSTING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GRADIENT-BOOSTING/), GRID SEARCH (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GRID-SEARCH/), LIVE CODING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LIVE-CODING/), PARAMETER TUNING IN XGBOOST (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PARAMETER-TUNING-IN-XGBOOST/), PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON/), SKLEARN (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/SKLEARN/), XGBOOST (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/XGBOOST/), XGBOOST MODEL (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/XGBOOST-MODEL/)

PREVIOUS ARTICLE
A Complete Tutorial to learn Data Science in R from Scratch
(https://www.analyticsvidhya.com/blog/2016/02/complete-tutorial-learn-data-science-scratch/)

NEXT ARTICLE
Data Scientist (3+ years experience) – New Delhi, India
(https://www.analyticsvidhya.com/blog/2016/03/data-scientist-3-years-experience-delhi-india/)


(https://www.analyticsvidhya.com/blog/author/aarshay/)

Aarshay Jain (Https://Www.Analyticsvidhya.Com/Blog/Author/Aarshay/)

Aarshay graduated from MS in Data Science at Columbia University in 2017 and is currently an ML Engineer at Spotify New York. He works at an intersection or applied research and engineering while designing ML solutions to move product metrics in the required direction. He specializes in designing ML system architecture, developing offline models and deploying them in production for both batch and real time prediction use cases.

(mailto:aarshay.jain@gmail.com) (https://in.linkedin.com/in/aarshay) (https://github.com/aarshayj)

(aarshay)

This article is quite old and you might not get a prompt response from the author. We request you to post this comment on Analytics Vidhya's **Discussion portal** (https://discuss.analyticsvidhya.com/) to get your queries resolved

99 COMMENTS

PRATEEK **Reply**

March 2, 2016 at 5:18 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106464)

Please provide the R code as well.

Thnkx

ANKUR BHARGAVA **Reply**

March 2, 2016 at 6:14 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106466)

It is a great article , but if you could provide codes in R , it would be more beneficial to us. Thanks

---

BEN **Reply**

February 18, 2018 at 7:10 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-151443)

Nowadays less people are using R already. Python is the way to go

---

AARSHAY JAIN **Reply**

March 2, 2016 at 7:07 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106467)

Hi guys,

Thanks for reaching out!

I've given a link to an article (http://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/ (http://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/)) in my above article. This has some R codes for implementing XGBoost in R.

This won't replicate the results I found here but will definitely help you. Also, I don't use R much but think it should not be very difficult for someone to code it in R. I encourage you to give it a try and share the code as well if you wish :D.

In the meanwhile, I'll also try to get someone to write R codes. I'll get back to you if I find something.

Cheers,
Aarshay

LUCA **Reply**

March 2, 2016 at 7:40 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106470)

I am wondering whether in practice it is useful such an extreme tuning of the parameters … it seems that often the standard deviation on the cross validation folds does not allow to really distinguish between different parameters sets… any thoughts on that?

AARSHAY JAIN **Reply**

March 2, 2016 at 8:09 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106472)

Agree but partially. Some thoughts:

1. Though the standard deviations are high, as the mean comes down, their individual values should also come down (though theoretically not necessary). Actually the point is that some basic tuning helps but as we go deeper, the gains are just marginal. If you think practically, the gains might not be significant. But when you in a competition, these can have an impact because people are close and many times the difference between winning and loosing is 0.001 or even smaller.

2. As we tune our models, it becomes more robust. Even is the CV increases just marginally, the impact on test set may be higher. I've seen Kaggle master's taking AWS instances for hyper-parameter tuning to test out very small differences in values.

3. I actually look at both mean and std of CV. There are instances where the mean is almost the same but std is lower. You can prefer those models at times.

4. As I mentioned in the end, techniques like feature engineering and blending have a much greater impact than parameter tuning. For instance, I generally do some parameter tuning and then run 10 different models on same parameters but different seeds. Averaging their results generally gives a good boost to the performance of the model.

Hope this helps. Please share your thoughts.

LUCA **Reply**

March 3, 2016 at 4:04 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106536)

Hi, first of all thank you for writing the article (I forgot to thank you for that in my previous post :-)).

Regarding your points a few more thoughts:

1.-2. My gut feeling is that if the uncertainty on the mean is high (and usually it is proportional to the std) an apparent small average improvement maybe be actually due to stochastic effects (choice of a particular training set): hence would probably in general, not transfer to an independent test set. I wouldn't know how to make this argument more precise though.

3. That is probably useful indeed: another common choice is to choose the parameter set which provides the model of lowest complexity within one or half std from the minimum.

4. Yes, if the learning of these models is done by solving a non-convex optimization problem, that blending will in general help (indeed you have a chance of effectively averaging different models). It should work even better if you blend intrinsically different models (like linear + other types of nonlinear classifiers) since then you are even more sure that the decision boundaries are not correlated.

AARSHAY JAIN **Reply**

March 3, 2016 at 4:29 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106538)

Thanks a lot for sharing your feedback.

1-2: I'm getting your point. I think you are right. Very small improvements might actually be due to randomness. Probably we should consider model tuning in the end and use some moderate models to test out feature engineering.

3. Valid point. But how do we judge complexity in case of models like GBM or XGBoost? Is it related to training accuracy?

4. Agree totally.

Thanks for your comments. There is still so much for me to learn and what's better than interacting with experienced folks 🙂

BORUN DEV CHOWDHURY **Reply**

April 18, 2016 at 8:57 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109640)

Luca if you want to make more precise what you are saying the following is the way. Suppose you want to check the null hypothesis that two groups have different spending habits given their sample means and sample variances. How would you go about it. One method is ANOVA and another is to realise that under the assumption that each is normally distributed, the difference is also normally distributed with variance std_A/\sqrt(n_A) +std_B/\sqrt(n_B) and asking for the p-value of the observed difference in sample means.

This is the same problem. You have two difference means and you want to ask if the difference is statistically significant. Given that you are doing 5-fold CV the square-root factors are about 2 so the roughly the standard deviation of the difference in sample means is about the standard deviation you observe and you can see that if the difference in sample means is within one-sigma it is 65% likely to be 'statistical fluctuation' as you put it (correctly).

If you want to be more rigorous using t-distributions as n=5 either you can do that but as a ball park estimate I would say that in this problem is standard deviation is comparable to mean, an improvement much smaller than the mean means nothing (technically said, it does not rule out the null hypothesis that the parameter tuning did not buy you anything.)

JAY **Reply**

March 2, 2016 at 9:52 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106477)

Wow this seems to be very interesting I am new to Python and R programming I am really willing to learn this programming. Will be grateful if anyone here can guide me through that what should I learn first or from where should I start.

Thanks
Jay

AARSHAY JAIN **Reply**

March 2, 2016 at 9:55 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106479)

Well Jay you have come to the right place!

Check out this learning path for Python – http://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/learning-path-data-science-python/ (http://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/learning-path-data-science-python/)

You can start with this complete tutorial on python as well – http://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/ (http://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/)

You'll find similar resources for R as well here. Along with programming, there are detailed tutorials on data science concepts like this one. You're in for a treat!!

Cheers,
Aarshay

SHAN **Reply**

March 3, 2016 at 11:40 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106524)

Hi..
Nice article with lots of informations.
I was wondering if I can clear my understandings on following :

a) On Handling Missing Values, XGBoost tries different things as it encounters a missing value on each node and learns which path to take for missing values in future.
Please elaborate on this.

b) In function modelfit; the following has been used
xgb_param = alg.get_xgb_params()
Is get_xgb_params() available in xgb , what does it passes to xgb_param

Please explain:
alg.set_params(n_estimators=cvresult.shape[0])

Thanks.

AARSHAY JAIN **Reply**

March 3, 2016 at 3:33 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106534)

Glad you liked it.. My responses below:

a) When xgboost encounters a missing value at a node, it tries both left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on testing data.

b) Yes it is available in sklearn wrapper of xgboost package. It will pass the parameters in actual xgboost format (not sklearn wrapper). The cv function requires parameters in that format itself.

c) cvresults is a dataframe with the number of rows being equal to the optimum number of parameters selected. You can try printing cvresults and it'll be clear.

Hope this helps.

STALLAB Reply

March 4, 2016 at 9:45 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106566)

Fantastic work ! thanks a lot.

Now let's hope that we will be able to install XGBoost with a simple pip command 🙂

AARSHAY JAIN Reply

March 4, 2016 at 5:24 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106588)

Thanks 🙂

i think installation is not that simple. depending on the OS, you can refer to different sections of this page – https://github.com/dmlc/xgboost/blob/master/doc/build.md (https://github.com/dmlc/xgboost/blob/master/doc/build.md)

JULIEN NEL Reply

March 4, 2016 at 4:03 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106583)

Hi Guys,

I cant seem to predict probabilities, the gbm.predict is only giving me 0's and 1's..

I put objective="binary:logistic" in but I still only get 0 or 1..

Any tips?

AARSHAY JAIN Reply

March 4, 2016 at 5:16 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106586)

sklearn model classes have a function "predict_proba" for predicting the probabilities. Please use that.

JULIEN NEL Reply

March 6, 2016 at 5:26 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106722)

Great thank you!!

VIKAS REDDY Reply

Unlock 2020

March 4, 2016 at 6:31 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106591)

During feature engineering, if I want to check if a simple change is producing any effect on performance, should I go through the entire process of fine tuning the parameters, which is obviously better than keeping the same parameter values but takes lot of time. So, how often do you tune your parameters?

AARSHAY JAIN Reply

March 5, 2016 at 3:12 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106657)

Hi Vikas,

I don't think that should be required. Once you tune your model on a baseline input, it should be good enough to check if the features are working.

If you're experimenting a lot, it might be a good idea to use random forest to check if feature improved the accuracy. RF models run faster and are not much affected by tuning.

Hope this helps.

ANURAG Reply

March 5, 2016 at 8:05 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106633)

excellent article….. We want Neural Networks as well.

AARSHAY JAIN Reply

March 5, 2016 at 3:03 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106656)

Thanks.. NN is in the pipeline.. 🙂

ANDRE LOPES Reply

March 7, 2016 at 2:53 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106747)

At section 3 : – 3.Parameter With tuning,

xgtest = xgb.DMatrix(dtest[predictors].values)

dtest doesnt exist.
Where did you get it?

Im trying to learn with your code!
Thanks in advance

AARSHAY JAIN **Reply**

March 7, 2016 at 5:57 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106774)

Hi Andre,
Thanks for reaching out. Valid point. My bad I should have removed it. I've updated the code above.

The reason it was present is that I used the test file on my end for checking the result of each model, which can be seen as "AUC Score (Test)". You would not get this output when you run it locally on your system. Hope this clears the confusion.

GIANNI **Reply**

March 7, 2016 at 4:49 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106816)

Hi Jain thanks for you effort, this guide is simply awesome !

But just because I wasn't able to find the modified Train Data from the repository (in effect I wasn't able to find the repository, my fault for sure, but I'm working on it), I had to rebuild the modified train data (good exercise !) and I want to share with everyone my code:

```
train.ix[ train['DOB'].isnull(), 'DOB' ] = train['DOB'].max()
train['Age'] = (pd.to_datetime( train['DOB'].max(), dayfirst=True ) – pd.to_datetime(
train['DOB'], dayfirst=True )).astype('int64')
train.ix[ train['EMI_Loan_Submitted'].isnull(), 'EMI_Loan_Submitted_Missing' ] = 1
train.ix[ train['EMI_Loan_Submitted'].notnull(), 'EMI_Loan_Submitted_Missing' ] = 0
train.ix[ train['Existing_EMI'].isnull(), 'Existing_EMI' ] = train['Existing_EMI'].median()
train.ix[ train['Interest_Rate'].isnull(), 'Interest_Rate_Missing' ] = 1
train.ix[ train['Interest_Rate'].notnull(), 'Interest_Rate_Missing' ] = 0
train.ix[ train['Loan_Amount_Applied'].isnull(), 'Loan_Amount_Applied' ] =
train['Loan_Amount_Applied'].median()
train.ix[ train['Loan_Tenure_Applied'].isnull(), 'Loan_Tenure_Applied' ] =
train['Loan_Tenure_Applied'].median()
train.ix[ train['Loan_Amount_Submitted'].isnull(), 'Loan_Amount_Submitted_Missing' ] = 1
train.ix[ train['Loan_Amount_Submitted'].notnull(), 'Loan_Amount_Submitted_Missing' ] = 0
train.ix[ train['Loan_Tenure_Submitted'].isnull(), 'Loan_Tenure_Submitted_Missing' ] = 1
train.ix[ train['Loan_Tenure_Submitted'].notnull(), 'Loan_Tenure_Submitted_Missing' ] = 0
train.ix[ train['Processing_Fee'].isnull(), 'Processing_Fee_Missing' ] = 1
train.ix[ train['Processing_Fee'].notnull(), 'Processing_Fee_Missing' ] = 0
train.ix[ ( train['Source'] != train['Source'].value_counts().index[0] ) &
( train['Source'] != train['Source'].value_counts().index[1] ), 'Source' ] = 'S000'
# Numerical Categorization
from sklearn.preprocessing import LabelEncoder
var_mod = [] # Nessun valore numerico da categorizzare, in caso contrario avremmo avuto
una lista di colonne
le = LabelEncoder()
for i in var_mod:
train[i] = le.fit_transform(train[i])
#One Hot Coding:
train = pd.get_dummies(train, columns=['Source', 'Gender', 'Mobile_Verified', 'Filled_Form',
```

'Device_Type','Var1','Var2'])
train.drop(['City','DOB','EMI_Loan_Submitted','Employer_Name','Interest_Rate','Lead_Creation_D
'Loan_Tenure_Submitted','LoggedIn','Salary_Account','Processing_Fee'], axis=1, inplace=True)

Just because the way I constructed my "age" column, results are a little different, but plus or minus all ought to be right.

Thanks everyone, this site is pure gold for me. I learned here in a month more than I learned everywhere in years … I'm just guessing where I will be in a year from now.

AARSHAY JAIN **Reply**

March 7, 2016 at 4:55 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-106817)

Hi Gianni,

Thanks for your effort and for sharing the code. The data set has been uploaded and a link provided inside the article at section 3. Parameter Tuning with Example line 3.

You can also download the same from my GitHub repository:
https://github.com/aarshayj/Analytics_Vidhya/tree/master/Articles/Parameter_Tu
(https://github.com/aarshayj/Analytics_Vidhya/tree/master/Articles/Parameter_Tu
The filename is 'train_modified.zip'

Cheers,
Aarshay

MAHESH **Reply**

March 12, 2016 at 12:24 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107179)

Guys,

Please help me with xgboost installation on windows

AARSHAY JAIN **Reply**

March 13, 2016 at 5:41 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107239)

I use a MAC OS so I haven't tried on windows. I think installing on R is pretty straight forward but Python is a challenge. I guess the discussion forum is the right place to reach out to a wider audience who can help. 🙂

PRAVEEN GUPTA SANKA **Reply**

March 16, 2016 at 4:21 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107465)

I followed instructions from the below link and it worked for me
http://stackoverflow.com/a/35119904 (http://stackoverflow.com/a/35119904)

Long story short, I have installed “mingw64” and “Cygwin shell” on my laptop and ran the commands provided in the above answer.

VITALIY RADCHENKO **Reply**

March 13, 2016 at 5:56 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107268)

I have the error
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, show_progress=False)

raise ValueError('Check your params.'\
ValueError: Check your params.Early stopping works with single eval metric only.

How can I fix it? Thank you in advance.

AARSHAY JAIN **Reply**

March 13, 2016 at 6:10 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107269)

What I can understand from the error is that multiple metrics have been defined. But here it's just 'auc'. Please check your xgb_param value. Is it setting a different value for metric?

If problem persists for long, I suggest you start a discussion thread with code and error snapshot. It'll be easier to debug.

VITALIY RADCHENKO **Reply**

March 13, 2016 at 6:28 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107272)

Params are the same as in tutorial
xgb1 = XGBClassifier(
learning_rate =0.1,
n_estimators=294,
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)

AARSHAY JAIN **Reply**

March 14, 2016 at 5:58 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107314)

Do you have the latest version of xgboost? I just checked and this was an issue in one of the older versions!

STELLA **Reply**

March 18, 2016 at 7:43 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107653)

I am using version 0.4 on ubuntu 15.10. I checked the xgboost.cv document, and found the parameter metrics must be “list of strings”. So I changed to metric = [“auc”], and it worked.

DANIEL **Reply**

March 14, 2016 at 6:16 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107316)

Hi Aarshay,

quick question: if I try to do multi-class classification, python send error as follows:
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,

n_class=4,
objective=”multi:softmax”,
nthread=4,
scale_pos_weight=1,
seed=27)
Traceback (most recent call last):

File “”, line 15, in
seed=27)

TypeError: __init__() got an unexpected keyword argument ‘n_class’

When i try “num_class” instead it does not work either nor with “n_classes” the sklearn wrapper I assume,

Any Thoughts?

thanks,

Daniel

AARSHAY JAIN **Reply**

March 14, 2016 at 7:23 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107322)

Hi Daniel,
I don't think the 'n_classes' or any other variant of argument is needed in the sklearn wrapper. It works for me without this argument. Please try removing it.

DANIEL **Reply**

March 14, 2016 at 1:28 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107352)

Hi Aarshay!
Thanks for your prompt response. Yes, you are right I can train without the argument 'n_classes´.
However, when I want to use xgb.cv(…) it gives an error:
"XGBoostError: must set num_class to use softmax" (the log is below).

So I guess my question is if one can use xgb.cv() for parameter tuning with multi-class classification.

Thanks again in advance!

cvresult = xgb.cv(xgb_param, dtrain,
num_boost_round=xgb1.get_params()['n_estimators'], nfold=5,
early_stopping_rounds=50, show_progress=False)
Will train until cv error hasn't decreased in 50 rounds.
Traceback (most recent call last):

File "", line 2, in
early_stopping_rounds=50, show_progress=False)

File "//anaconda/lib/python2.7/site-packages/xgboost/training.py", line 418, in cv
fold.update(i, obj)

File "//anaconda/lib/python2.7/site-packages/xgboost/training.py", line 257, in update
self.bst.update(self.dtrain, iteration, fobj)

File "//anaconda/lib/python2.7/site-packages/xgboost/core.py", line 694, in update
_check_call(_LIB.XGBoosterUpdateOneIter(self.handle, iteration,
dtrain.handle))

File "//anaconda/lib/python2.7/site-packages/xgboost/core.py", line 97, in _check_call
raise XGBoostError(_LIB.XGBGetLastError())

XGBoostError: must set num_class to use softmax

AARSHAY JAIN **Reply**

March 14, 2016 at 1:30 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107353)

Hi Daniel,

Yes it can be used. You have to add the parameter 'num_class' to the xgb_param dictionary. Use something like this before calling xgb,cv:

xgb_param['num_class'] = k #k = number of classes.

It should work. I use xgb.cv for multi-class problems a lot!

SHAN **Reply**

March 14, 2016 at 1:56 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107355)

Hi.. Daniel.

Can you please share how you installed xgboost in anaconda and which OS you are using.
Thanks.

AARSHAY JAIN **Reply**

March 16, 2016 at 3:08 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107497)

@shan – look at Preveen Gupta's answer above!

PRAVEEN GUPTA SANKA **Reply**

March 23, 2016 at 6:57 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108060)

Hi Shan,

As per instructions given in the link that I mentioned above,
I first installed MINGW-64 from the below website
http://sourceforge.net/projects/mingw-w64/ (http://sourceforge.net/projects/mingw-w64/)
then I installed cygwin from the below link
https://cygwin.com/setup-x86_64.exe (https://cygwin.com/setup-x86_64.exe)

Hope this helps.

MICHELLE **Reply**

October 4, 2016 at 5:54 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-116779)

Hi Daniel,

I met the same problem as you. Can not figure out how to add “num_class” parameter to XGBClassifer(). If you figure it out, could you please show us how to solve this problem?
Thanks a lot!

Michelle

PRAVEEN GUPTA SANKA **Reply**

March 16, 2016 at 1:39 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107440)

Hi Aarshay,

The youtube video link you posted is not working. (Error is “This video is private”)
https://www.youtube.com/watch?v=X47SGnTMZIU (https://www.youtube.com/watch?v=X47SGnTMZIU)

Is there any other source where we can watch the video?

Thanks,
Praveen

AARSHAY JAIN **Reply**

March 16, 2016 at 3:07 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-107496)

try this – https://www.youtube.com/watch?v=ufHo8vbk6g4 (https://www.youtube.com/watch?v=ufHo8vbk6g4)

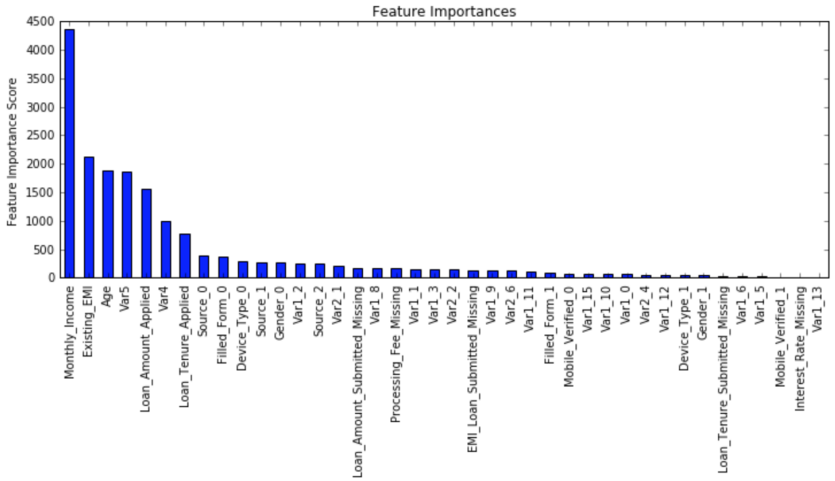
PRAVEEN GUPTA SANKA **Reply**

March 16, 2016 at 4:04 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-

```
xgb4 = XGBClassifier(
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=4,
 min_child_weight=6,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 reg_alpha=0.005,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb4, train, predictors)
```

```
Will train until cv error hasn't decreased in 50 rounds.
Stopping. Best iteration:
[1732] cv-mean:0.8452782        cv-std:0.0126670016879
```

```
Model Report
Accuracy : 0.9854
AUC Score (Train): 0.885261
AUC Score (Test): 0.849430
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/12.-final-0.01.png)

Here is a live coding window where you can try different parameters and test the results.

(https://id.analyticsvidhya.com/auth/login/?next=https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/?&utm_source=coding-window-blog&source=coding-window-blog)


Unlock 2020

Now we can see a significant boost in performance and the effect of parameter tuning is clearer.

As we come to the end, I would like to share 2 key thoughts:

1. It is **difficult to get a very big leap** in performance by just using **parameter tuning** or **slightly better models**. The max score for GBM was 0.8487 while XGBoost gave 0.8494. This is a decent improvement but not something very substantial.
2. A significant jump can be obtained by other methods like **feature engineering**, creating **ensemble** of models, **stacking**, etc

You can also download the iPython notebook with all these model codes from my GitHub account (https://github.com/aarshayj/Analytics_Vidhya/tree/master/Articles/Parameter_Tuning_XGBoost_with_Example). For codes in R, you can refer to this article (https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/).

End Notes

This article was based on developing a XGBoost model end-to-end. We started with discussing **why XGBoost has superior performance over GBM** which was followed by detailed discussion on the **various parameters** involved. We also defined a generic function which you can re-use for making models.

Finally, we discussed the **general approach** towards tackling a problem with XGBoost and also worked out the **AV Data Hackathon 3.x problem** through that approach.

I hope you found this useful and now you feel more confident to apply XGBoost in solving a data science problem. You can try this out in out upcoming hackathons.

Did you like this article? Would you like to share some other hacks which you implement while making XGBoost models? Please feel free to drop a note in the comments below and I'll be glad to discuss.

You want to apply your analytical skills and test your potential?
Then participate in our Hackathons (http://datahack.analyticsvidhya.com/contest/all) and compete with Top Data Scientists from all over the world.

parameter-tuning-xgboost-with-codes-python/#comment-107503)

Thanks a lot.. This link is working

PMITRA **Reply**

April 5, 2016 at 5:36 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109006)

Hi Praveen,

I followed the steps to install XGB on Windows 7 as mentioned in your comment above i.e using mingw64 and cygwin/ Everything went fine until the last steps as below:

cp make/mingw64.mk config.mk
make -j4 —>>> where (make = mingw32-make)

By running the above lines I get the error as follows::

g++ -m64 -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iincl ude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -MM -MT build/logging.o src/logging.cc >build/logging.d
g++ -m64 -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iincl ude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -MM -MT build/learner.o src/learner.cc >build/learner.d
g++ -m64 -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iincl ude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -MM -MT build/c_api/c_api.o src/c_api/c_api.cc >build/c_api/c_api.d
g++ -m64 -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iincl ude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -MM -MT build/data/simple_dmatrix.o src/data/simple_dmatrix.cc >build/data/simple_d matrix.d
g++ -m64 -c -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iinclude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -c src/logging.cc -o build/logging.o
g++ -m64 -c -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iinclude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -c src/c_api/c_api.cc -o build/c_api/c_api.o
g++ -m64 -c -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iinclude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -c src/data/simple_dmatrix.cc -o build/data/simple_dmatrix.o
In file included from include/xgboost/./base.h:10:0,
from include/xgboost/logging.h:13,
from src/logging.cc:7:
dmlc-core/include/dmlc/omp.h:9:17: fatal error: omp.h: No such file or directory
compilation terminated.
g++ -m64 -c -std=c++0x -Wall -O3 -msse2 -Wno-unknown-pragmas -funroll-loops -Iinclude -DDMLC_ENABLE_STD_THREAD=0 -Idmlc-core/include -Irabit/include -fopenmp -c src/learner.cc -o build/learner.o
Makefile:97: recipe for target ‘build/logging.o’ failed
make: *** [build/logging.o] Error 1
make: *** Waiting for unfinished jobs....

In file included from include/xgboost/./base.h:10:0,
from include/xgboost/logging.h:13,
from src/learner.cc:7:
dmlc-core/include/dmlc/omp.h:9:17: fatal error: omp.h: No such file or directory
compilation terminated.
Makefile:97: recipe for target 'build/learner.o' failed
make: *** [build/learner.o] Error 1
In file included from include/xgboost/./base.h:10:0,
from include/xgboost/data.h:15,
from src/data/simple_dmatrix.cc:7:
dmlc-core/include/dmlc/omp.h:9:17: fatal error: omp.h: No such file or directory
compilation terminated.
Makefile:97: recipe for target 'build/data/simple_dmatrix.o' failed
make: *** [build/data/simple_dmatrix.o] Error 1
In file included from include/xgboost/./base.h:10:0,
from include/xgboost/data.h:15,
from src/c_api/c_api.cc:3:
dmlc-core/include/dmlc/omp.h:9:17: fatal error: omp.h: No such file or directory
compilation terminated.
Makefile:97: recipe for target 'build/c_api/c_api.o' failed
make: *** [build/c_api/c_api.o] Error 1

I don't understand the reason behind this error. I have stored the mingw64 files under C:\mingw64\mingw64 And I have stored the xgboost files under C:\xgboost. I also added the paths to Environment.as well.
I even tried to install the same way in my oracle virtual box but it threw the same building error there too.

Please could you throw some light on this and let me know if I am missing anything ??

PRAVEEN GUPTA SANKA **Reply**

March 23, 2016 at 6:23 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108056)

Hi Aarshay,

As always, a great article.

I have two doubts
1. n_estimators=cvresult.shape[0] we have set this while fitting the algorithm for XGBoost. Any specific reason why we did in that way.
2. In the model fit function, we are not generating CV score as the output.. How are we automatically able to get it in box with red background. I am not getting CV value. Am I missing something?

Can you please clarify

Regards,
Praveen

SHAN Reply

March 23, 2016 at 6:52 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108059)

Hi..Praveen Gupta Sanka,

Can you please share how to install xgboost in python/ anaconda env. ? r
I followed instructions from the below link and it worked for me
http://stackoverflow.com/a/35119904 (http://stackoverflow.com/a/35119904)

Can you please share how you installed “mingw64” and “Cygwin shell” on laptop ?
Need hand holding on the same.

Thanks in advance,

AARSHAY JAIN Reply

March 23, 2016 at 10:09 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108077)

Thanks Praveen! My responses:

1. I’ve used xgb.cv here for determining the optimum number of estimators for a given learning rate. After running xgb.cv, this statement overwrites the default number of estimators to that obtained from xgb.cv. The variable cvresults is a dataframe with as many rows as the number of final estimators.

2. The red box is also a result of the xgb.cv function call.

VZ Reply

April 5, 2016 at 9:57 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108970)

When I try the GridSearchCV my system does not do anything. It sits there for a long time, but I can check the activity monitor and nothing happens, no crash, no message, no activity. Any clue?

AARSHAY JAIN Reply

April 5, 2016 at 10:08 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108972)

This is strange indeed. Right off the bat, I think of following diagnosis:
1. Run the GridSearchCV for a very small sample of data, the one which you are sure your system can handle easily. This will check the installation of sklearn
2. If it works fine, it might be a system computing power issue. If it doesn’t work try re-installing sklearn.

VZ Reply

April 5, 2016 at 11:45 am
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-

parameter-tuning-xgboost-with-codes-python/#comment-108984)

This is the line where it hangs.
gsearch1.fit(train_data[predictors],train_data[target])
Is there any verbose parameter I can add?

AARSHAY JAIN **Reply**
April 5, 2016 at 11:46 am
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108985)

I don't think so. Have you tried the diagnostic I suggested above?

VZ **Reply**
April 5, 2016 at 1:00 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108992)

Yes, it is not the data size, and sklearn installation went fine. modelfit function runs fine.

AARSHAY JAIN **Reply**
April 5, 2016 at 1:05 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108994)

I'm sorry I didn't get your point. If the sklearn installation is fine and modelfit runs on small data, then it looks more likely to be the data size issue. Any other reason you can think of?

VZ **Reply**
April 5, 2016 at 1:27 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-108995)

No, it does not run on small data either. The modelfit function above works fine on either large or small data, but gsearch1.fit does not work on either.

Unlock 2020

AARSHAY JAIN **Reply**

April 5, 2016 at 1:30 pm (https://www.analyticsvidhya.com/blguide-parameter-tuning-xgboost-with-codes-python/#comment-108996)

I guess it is an installation issue then. You can try re-installing python or contacting the sklearn developers by raising a ticket and sharing your details.

VZ **Reply**

April 5, 2016 at 1:42 pm (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-108997)

Honestly I don't think it is a python or sklearn issue since they both work fine with everything else, but thank you for your time.

AARSHAY JAIN **Reply**

April 5, 2016 at 4:04 pm (https://www.analyticsvidhya.com/blguide-parameter-tuning-xgboost-with-codes-python/#comment-109003)

Might be the case. Difficult to diagnose remotely with the available information. You might want to use the discussion forum (discuss.analyticsvidhya.com) to reach to a wider audience and seek help.

VZ **Reply**

April 5, 2016 at 9:24 pm (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-109024)

Thank you for all your time, and by the way, excellent tutorial. I am going to try to debug it and let you know what I find. By the way, what exactly gives us the modelfit function, what exactly represents the best iteration in the parameters we are trying to tune?

AARSHAY JAIN **Reply**

April 6, 2016 at 6:22 am (https://www.analyticsvidhya.com/blguide-parameter-tuning-xgboost-with-codes-python/#comment-109044)

I am sorry I didn't get your question. Please elaborate.

VZ **Reply**

April 6, 2016 at 7:25 am (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-109051)

I am sorry I as not clear. In step 1 you use a function, modelfit. This function will output something like "stopping. Best iteration [n]". In your case that number is 140. I am not sure I understand how you use this information, is this used with the n_estimators parameters?

By the way, I debugged the issue and it appears a problem with n_jobs. If I do not pass that variable, the issues goes away. It looks then like a bug in the library, not an installation issue.

AARSHAY JAIN **Reply**

April 6, 2016 at 7:34 am (https://www.analyticsvidhya.com/blguide-parameter-tuning-xgboost-with-codes-python/#comment-109053)

Its great that you debugged the issue. Yes you got it right. We use it with the n_estimators parameter. The modelfit function automatically does that using the following command: alg.set_params(n_estimators=cvresult.s This replaces the n_estimators to that obtained from cvresult. Here cvresult is a dataframe with as many rows as the number of optimum trees, say 140 in the case you were referring.

DAVID COMFORT **Reply**

Unlock 2020

April 6, 2016 at 2:30 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109032)

I get an error:

XGBClassifier' object has no attribute 'feature_importances_'

It looks like it a known issue with XGBClassifier.

See https://www.kaggle.com/c/homesite-quote-conversion/forums/t/18669/xgb-importance-question-lost-features-advice/106421 (https://www.kaggle.com/c/homesite-quote-conversion/forums/t/18669/xgb-importance-question-lost-features-advice/106421)

and https://github.com/dmlc/xgboost/issues/757#issuecomment-174550974 (https://github.com/dmlc/xgboost/issues/757#issuecomment-174550974)

I can get the feature importances with the following:

```
def importance_XGB(clf):
impdf = []
for ft, score in clf.booster().get_fscore().iteritems():
impdf.append({'feature': ft, 'importance': score})
impdf = pd.DataFrame(impdf)
impdf = impdf.sort_values(by='importance', ascending=False).reset_index(drop=True)
impdf['importance'] /= impdf['importance'].sum()
return impdf

importance_XGB(xgb1)
```

DAVID COMFORT **Reply**

April 6, 2016 at 3:44 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109037)

I actually got it working by updating to the latest version of XGBoost. However, I had to change

metrics='auc' to metrics={'auc'}

Also, early_stopping_rounds does not appear to work anymore

AARSHAY JAIN **Reply**

April 6, 2016 at 6:23 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109045)

Which function are you using early_stopping_rounds as a parameter?

DAVID COMFORT **Reply**

April 6, 2016 at 11:16 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-

guide-parameter-tuning-xgboost-with-codes-python/#comment-109084)

Never Mind, I did get it working.

However, I have another question. Once you have optimized your model parameters, how would you save your model and then use it to predict on a test set?

AARSHAY JAIN **Reply**

April 7, 2016 at 12:34 pm (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-109116)

If you observe the modelfit function carefully, the following lines are used to make predictions on test data:
#Predict training set:
dtrain_predictions = alg.predict(dtrain[predictors])
dtrain_predprob = alg.predict_proba(dtrain[predictors])[:,1]

VZ **Reply**

April 9, 2016 at 7:18 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109230)

Sorry to bother you again, but would you mind elaborating a little more on the code in modelfit, in particular:

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, show_progress=False)
alg.set_params(n_estimators=cvresult.shape[0])

Thank you very for your time.

AARSHAY JAIN **Reply**

April 10, 2016 at 3:25 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109260)

sure. this part of the code would check the optimal number of estimators using the "cv" function of xgboost. This works only if the useTrainCV argument of the function is set as True. If True, this will run "xgb.cv", determine the optimal value for n_estimators and replace the value set by the user with this value. While using this case, you should remember to set a very high value for n_estimators, i.e. higher than the expected optimal value range. Hope this makes sense.

VZ **Reply**

April 10, 2016 at 5:40 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109263)

Thank you for your answer. I do understand that, but I was wondering about what DMatrix and get_xgb_params exactly do.

AARSHAY JAIN **Reply**

April 11, 2016 at 7:02 am
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109281)

As mentioned above, there are 2 ways to use xgboost:
1. sklearn wrapper – allows pandas dataframe as input
2. raw xgboost functions – requires a DMatrix format provided by xgboost. So this is just a necessary pre-processing step if you are not using sklearn wrapper.

Similarly,, get_xgb_params() return the parameters in the format required by the raw xgboost functions.

All this is needed because xgboost.cv has not been implemented in the sklearn wrapper and we have to use the original functions for that.

DEEPISH (HTTP://XPLOREANALYTICS.BLOGSPOT.IN/) **Reply**

April 14, 2016 at 3:52 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109471)

Nice article @Aarshah
One question on setting the parameters for xgb here.
Can the value of n_estimators be only set or we can derive different parameters like max_depth,seed, etc??

If we can derive all the parameters then how is this different from GridSearchCV?

AARSHAY JAIN **Reply**

April 14, 2016 at 5:16 pm
(https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109475)

I'm sorry i didn't get what you mean by deriving variables?

DEEPISH **Reply**

April 14, 2016 at 5:29 pm (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-109476)

I am sorry i should have been more clear with the question.

My question was more conceptual in nature. In the modelfit() method you have show that setting the value of estimators using the n_estimators=cvresult.shape[0] is possible, but there are more parameters to the xgb classifier eg. max_depth,seed, colsample_bytree, nthread etc. Is it possible to find out optimal values of these parameters also via cv method.

I surely know that this can be done by GridSearchCV, just wondering if at all its possible by the sklearn wrapper cv() method?

Thanks for the help.

AARSHAY JAIN **Reply**

April 14, 2016 at 5:33 pm (https://www.analyticsvidhya.com/blog/2016/03/guide-parameter-tuning-xgboost-with-codes-python/#comment-109477)

Thanks for clarifying. cv is only for determining n_estimators and other parameters cannot be determined using this. It basically gives the optimum n_estimators value corresponding to the other set of parameters.

CURTIS (HTTP://CURTISNORTHCUTT.COM) **Reply**

April 11, 2016 at 12:17 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109272)

Thanks for your work here – great job! Is it be possible to be notified when a similar article to this one is released for Neural Networks?

AARSHAY JAIN **Reply**

April 11, 2016 at 7:04 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109282)

They are already out there:
1. http://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/ (http://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/)
2. http://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/ (http://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/)

JOSE MAGANA Reply

April 14, 2016 at 4:27 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109435)

Hello,
really great article, I have learnt a lot from it.

One question, you mention the default value for scale_pos_weight is 0. Where have you got this information from? Checking the source code (regresion_obj.cc) I have found the value to be 1 by default, with a lower bound of 0. In the R version, that I use, the parameter does not appear explicitly.

Can you please clarify?

Thanks in advance

AARSHAY JAIN Reply

April 14, 2016 at 10:44 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109451)

I just checked again. Yes you're right the default value is 1 and not 0. Thanks for pointing this out. I'll make the correction.

DIEGO Reply

April 21, 2016 at 5:50 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-109810)

I'm getting this strange error:"WindowsError: exception: access violation reading 0x000000000D92066C"
Any Idea what may be causing it?

FYI, if I don't include the [] on the metric parameter, I get: "ValueError: Check your params.Early stopping works with single eval metric only." (same as the user above)

cvresult = xgb.cv(xgb_param, xgtrain, num_boost_round=alg.get_params()['n_estimators'], nfold=5,
metrics=['logloss'], early_stopping_rounds=25, show_progress=False)

Will train until cv error hasn't decreased in 25 rounds.
Traceback (most recent call last):

File “”, line 2, in
metrics=[‘logloss’], early_stopping_rounds=25, show_progress=False)

File “C:\Anaconda2\lib\site-packages\xgboost-0.4-py2.7.egg\xgboost\training.py”, line 415, in cv
cvfolds = mknfold(dtrain, nfold, params, seed, metrics, fpreproc)

File “C:\Anaconda2\lib\site-packages\xgboost-0.4-py2.7.egg\xgboost\training.py”, line 275, in mknfold
dtrain = dall.slice(np.concatenate([idset[i] for i in range(nfold) if k != i]))

File “C:\Anaconda2\lib\site-packages\xgboost-0.4-py2.7.egg\xgboost\core.py”, line 494, in slice
ctypes.byref(res.handle)))

WindowsError: exception: access violation reading 0x000000000D92066C

AARSHAY JAIN **Reply**

April 25, 2016 at 5:49 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-110006)

not sure man. have you tried searching? posting on discussion forum might be a good idea to crowd-source the issue.

JOSE MAGANA **Reply**

May 10, 2016 at 9:00 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-110750)

According to this: https://www.kaggle.com/c/santander-customer-satisfaction/forums/t/20662/overtuning-hyper-parameters-especially-re-xgboost (https://www.kaggle.com/c/santander-customer-satisfaction/forums/t/20662/overtuning-hyper-parameters-especially-re-xgboost)
If you are using logistic trees, as I understand your article describes, alpha and lambda don’t play any role.
I would appreciate your feedback
Thanks in advance

AARSHAY JAIN **Reply**

May 10, 2016 at 2:30 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-110765)

Hi Jose,

I’m not sure which part of the post you are referring to. If it is the part which says “reg_alpha, reg_lambda are not used in tree booster”, then this is right.

But the parameters which I’ve mentioned are alpha and lambda and not reg_alpha and reg_lambda. Regularization is used in tree-booster as well where the constraint is put on the score of each leaf in the tree.

Please let me know if its still unclear.

Cheers!

JOSE MAGANA **Reply**

May 11, 2016 at 4:16 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-110796)

If you check the source code, you would observe that alpha is nothing but an alias for reg_alpha. Files> param.h and gblinear.cc.
In section 2 of your article you mention a similar mapping of names for the case of Python.
Can you tell me where in the code is alpha used in the case of trees? What is the effect?
Furthermore, the improvements in your CV are smaller than your std still you claim the improvement is due to the tuning of these parameters and not to the data separation for example.

AARSHAY JAIN **Reply**

May 11, 2016 at 6:24 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-110800)

I guess the nomenclature varies in different implementations. If you read the Tree Boosting part here – http://xgboost.readthedocs.io/en/latest/model.html (http://xgboost.readthedocs.io/en/latest/model.html), you'll understand how regularization is used for tree boosters. I haven't gone into the coding yet. I was trusting that these guys implement what they say. I don't have time to look into it now but will do sometime later.

Regarding the other point, I agree with you partially. Typically we should use the same folds and see if there is improvement in most of the folds (atleast 3 out of 5). I just used mean here for simplicity and because mostly it works out. The standard deviation being similar, a higher mean generally means an improvement in most folds. It'll be a rare case where 1 fold increases drastically and other decreases. But I agree we should check those things. I didn't want this to become too overwhelming for beginners so decided to stick with the mean.

LIMING HU **Reply**

May 17, 2016 at 11:19 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-111112)

It is a great blog. It will be better, if you can give a parameter tuning for a regression problem, although a lot
of stuff will be similiar to the classification problem.

AARSHAY JAIN **Reply**

May 18, 2016 at 4:20 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-111120)

Yes its mostly similar. If you understand this, the regression part should be easy to manage.

SUNIL SANGWAN **Reply**

June 6, 2016 at 7:00 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-111935)

Thanks great article.

EMRAH YIGIT **Reply**

June 9, 2016 at 6:15 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-112042)

Great article. Thank you.

TANGUY **Reply**

September 4, 2016 at 3:17 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-115584)

Thanks for the article, very useful 🙂

I was wondering if an article on “stacking” was in the pipe?

MICHELLE **Reply**

October 5, 2016 at 7:17 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-116830)

Hi Jian,
Just a quick question.
you use test_results.csv in modelfit function. Where is the cvs file? I couldn’t find it.
test_results = pd.read_csv(‘test_results.csv’)

Thank you.
Michelle

- **INSTALLING XGBOOST ON MAC OSX (IT BEST KEPT SECRET IS OPTIMIZATION) – CLOUD DATA ARCHITECT (HTTP://WWW.DATAARCHITECT.CLOUD/INSTALLING-XGBOOST-ON-MAC-OSX-IT-BEST-KEPT-SECRET-IS-OPTIMIZATION/)**

January 5, 2017 at 6:24 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-120588)

[…] I explain how to enable multi threading for XGBoost, let me point you to this excellent Complete Guide to Parameter Tuning in XGBoost (with codes in Python). I found it useful as I started using XGBoost. And I assume that you could be interested if you […]

- **INSTALLING XGBOOST ON MAC OSX (IT BEST KEPT SECRET IS OPTIMIZATION) – IOT PORTAL (HTTP://IOTPORTAL.TK/INSTALLING-XGBOOST-ON-MAC-OSX-IT-BEST-KEPT-SECRET-IS-OPTIMIZATION/)**
  January 25, 2017 at 2:36 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-121565)

  […] I explain how to enable multi threading for XGBoost, let me point you to this excellent Complete Guide to Parameter Tuning in XGBoost (with codes in Python). I found it useful as I started using XGBoost. And I assume that you could be interested if you […]

KAI Reply

May 20, 2017 at 6:02 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-128787)

Hi,
Thanks for sharing. One question: how do you decide what random seed to use. Is 27 just a random pick?

JHAKIR MIAH Reply

June 30, 2017 at 3:00 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-131336)

This is a great article, Aarshay. Thank you so much for writing it.
I am a newbie in data science. Once I follow this article and tune my parameters, how do I get the model to make a prediction on test data and see the prediction?

Please help me with sample code.
Thank you in advance.

JIMMY CHEN Reply

March 6, 2018 at 8:22 am (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-151713)

Hello Aarshay
This is really great article, I have learnt a lot from it.
One question, when i tuning the model using dataset (size = 1gb), the model ran very slowly , do you know why it ran too slowly ?
Thanks

JOHAN RENSINK Reply

April 11, 2018 at 3:32 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-152509)

Very impressive, I learned a lot. thanks for writing this!

JR

AISHWARYA SINGH **Reply**

April 13, 2018 at 3:31 pm (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#comment-152553)

Hi Johan,
Thank you for the feedback!

Unlock 2020