

# Penerapan Brute Force dan Decrease and Conquer pada Parameter Tuning XGBoostClassifier

Fajar Muslim / 13517149  
Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132,  
13517149@std.stei.itb.ac.id

**Abstrak** - Perkembangan teknologi juga diiringi dengan perkembangan informasi. Kebutuhan informasi dapat dipenuhi dengan mencari berbagai sumber informasi baik lisan maupun tulisan. Terkadang informasi yang diperoleh tidak dapat digunakan secara langsung. Sehingga, diperlukan suatu algoritma untuk mengolah informasi. Salah satunya adalah XGBoostClassifier yang dapat memberikan suatu prediksi berdasarkan data / informasi yang diberikan. Agar prediksi yang diberikan oleh XGBoostClassifier akurat diperlukan pemberian parameter yang tepat / parameter tuning.

**Keywords**—Informasi, XGBoostClassifier, parameter tuning

## I. INTRODUCTION (HEADING 1)

Industri 4.0 ditandai dengan penggunaan artificial intelligence dan big data dalam meningkatkan produksi suatu industri. Arus laju big data yang sangat cepat dan dinamis menuntut pengolahan data yang cepat pula. Data harus diolah menjadi sebuah informasi penting yang berguna bagi industri. Proses pengolahan data tersebut dapat berupa visualisasi data, ekstraksi fitur, pengisian nilai yang hilang, dan penerapan algoritma machine learning untuk memberikan prediksi yang diperlukan.

XGBoostClassifier yang merupakan salah satu algoritma machine learning memerlukan parameter agar algoritma dapat berjalan dengan baik. Pemberian parameter dilakukan agar algoritma XGBoostClassifier berjalan dengan optimal. Parameter yang dimasukkan kedalam algoritma XGBoostClassifier berbeda beda sesuai data yang diolah. Oleh karena itu, diperlukan suatu metode yang tepat guna dalam menentukan parameter yang optimal untuk dataset yang diolah. Penulis akan menerapkan algoritma Brute Force dan Decrease and Conquer dalam menentukan parameter yang tepat pada algoritma machine learning XGBoostClassifier.

## II. DASAR TEORI

### A. XGBoostClassifier

Xgboost adalah implementasi lanjutan dari algoritma peningkatan gradien (gradient boosting). Xboost menggunakan prinsip ansamble yaitu menggabungkan beberapa set pembelajar (tree) yang lemah menjadi sebuah model yang kuat sehingga menghasilkan prediksi yang kuat. Xgboost Memiliki banyak kelebihan diantaranya : Dapat melakukan pemrosesan paralel yang dapat mempercepat komputasi, memiliki fleksibilitas pengaturan objektif yang tinggi, built in cross validation, mengatasi split saat negative loss. Dengan kelebihan kelebihan tersebut, xgboost sangat cocok untuk mengolah data klasifikasi. xgboost akan membuat tree sebagai cara untuk mengklasifikasikan data train sehingga dapat diperoleh target tertentu. xgboost memiliki beberapa parameter yang dapat kita setting sedemikian sehingga disesuaikan dengan dataset yang diperoleh. Parameter tuning pada xgboost dapat dilakukan dengan menggunakan gridSearchCV dan tuning secara manual.

### B. Parameter tuning algoritma XGBoostClassifier

Parameter tuning merupakan kegiatan untuk menentukan parameter yang tepat untuk mendapatkan algoritma XGBoostClassifier yang mempunyai akurasi tinggi. Parameter tuning dilakukan dengan mengubah ubah parameter yang diberikan terhadap algoritma XGBoostClassifier. Dari parameter yang diberikan, dievaluasi akurasi prediksi yang diberikan. Akurasi tertinggi pada saat parameter tertentu yang diberikan diambil menjadi parameter ideal untuk algoritma XGBoostClassifier.

Parameter pada algoritma XGBoostClassifier diantaranya adalah :

#### 1. learning\_rate

learning rate adalah tingkat pembelajaran yang dilakukan oleh algoritma XGBoostClassifier. Tingkat pembelajaran

tersebut mempengaruhi algoritma XGBoostClassifier dalam membuat classifier yang berbentuk tree.

#### 2. n\_estimators

n\_estimators adalah banyaknya tree yang dibuat oleh classifier

#### 3. max\_depth

max\_depth adalah maksimum kedalaman tree.

#### 4. min\_child\_weight

min\_child\_weight adalah Jumlah minimum berat (hessian) yang diperlukan pada child node.

#### 5. gamma

Gamma adalah minimum loss yang dibutuhkan untuk membuat partisi node pada tree

#### 6. subsample

subsample adalah banyaknya bagian sampel yang dipilih untuk membangun tree. subsampling akan dilakukan setiap iterasi boosting.

### C. Brute Force

Brute Force adalah suatu algoritma penyelesaian masalah dengan membuat solusi berdasarkan definisi yang diberikan. Brute Force dapat dikatakan sebagai algoritma naif. Pencarian solusi brute force umumnya dilakukan sebagai perbandingan dengan terhadap algoritma yang lain yang lebih efisien. Brute force menyelesaikan persoalan dengan sangat sederhana, langsung, dan jelas. Brute force dapat digunakan untuk menyelesaikan permasalahan pencarian elemen terbesar atau terkecil Perkalian matriks, selection sort, bubble sort, dll. Brute force dapat diterapkan untuk persoalan yang relatif kecil yang tidak membutuhkan memori dan kalkulasi yang kompleks.

Langkah langkah dalam algoritma brute force :

1. Mengenumerasi semua kandidat solusi. Jika kontinu dapat dilakukan dengan menggunakan sampel yang mempunyai beda sama. Misalkan solusi terletak pada rentang 11 sampai 1000 inklusif. Maka kita dapat mengambil kandidat solusi mulai dari 1,2,3,4,...,1000.

2. Mengevaluasi kandidat solusi pada langkah ke-1. Solusi ditentukan sesuai definisi permasalahan yang diberikan.

Keunggulan algoritma brute force :

1. Umumnya lebih sederhana
2. Mudah diimplementasikan
3. Dijamin ditemukan solusi optimal

Contoh algoritma brute force :

//prosedur untuk mencari sebuah elemen tertentu pada list dan mengembalikan indeks ditemukanya elemen tersebut. Jika elemen yang dicari tidak terdapat di array. Maka program akan mengembalikan -9999.

Procedure cariIndeksElemen(input: list, input bil, output indeks){

Deklarasi :

i : integer

Algoritma :

i = -9999;

for i = 0 to length(list) do

if (list[i] = bil){

//bilangan ditemukan

break;

}

indeks = i;

}

kompleksitas algoritma pencarian diatas adalah  $O(n)$ .

### D. Decrease and conquer

Decrease and conquer adalah suatu algoritma yang mengurangi lingkup persoalan dengan mengurangi sub persoalan yang dipastikan tidak mengarah ke solusi.

Tahapan decrease and conquer

1. Decrease : Proses mengurangi persoalan menjadi persoalan yang lebih kecil
2. Conquer : Memproses suatu sub persoalan secara rekursif

Jenis jenis decrease and conquer

#### 1. Decrease by constant

Mengurangi suatu persoalan sebesar suatu konstanta tertentu. Biasanya sebesar 1.

Contoh : persoalan perpangkatan, Selection sort,

#### 2. Decrease by constant factor

Mengurangi suatu persoalan sebesar suatu faktor konstanta tertentu.

Contoh : Binary search, Interpolation search

#### 3. Decrease by variable size

Mengurangi suatu persoalan bervariasi pada setiap iterasi algoritma.

Contoh : Menghitung median dan selection problem

Keunggulan decrease and conquer :

1. Penyelesaian masalah yang relatif sulit. Dengan membagi permasalahan menjadi lebih kecil. Solusi lebih mudah ditemukan.
2. Efisien dibanding algoritma lain (brute force)
3. Persoalan berkurang seiring dengan penambahan iterasi

Contoh program decrease and conquer

```
#include <stdio.h>

int binarySearch(int arr[], int kiri, int kanan, int x)
{
    if (kanan >= kiri) {
        int tengah = kiri + (kanan - kiri) / 2;

        // jika elemen berada di tengah
        if (arr[tengah] == x)
            return tengah;

        // Jika elemen lebih kecil dari elemen tengah list. Maka x
        // terletak di sebelah kiri posisi tengah
        if (arr[tengah] > x)
            return binarySearch(arr, kiri, tengah - 1, x);

        // Jika elemen lebih besar dari elemen tengah list. Maka
        // dapat dipastikan x terletak di sebelah kanan posisi tengah
        return binarySearch(arr, tengah + 1, kanan, x);
    }

    // We reach here when element is not
    // present in array
    return -1;
}

int main()
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 10;
    int hasil = binarySearch(arr, 0, n - 1, x);
    if(hasil == -9999) {
        cout<<"Elemen tidak ditemukan di array"<<endl;
    } else {
        cout<<"Elemen ditemukan di indeks ke - "<<i<<endl;
    }
}
```

return 0;

}

Algoritma decrease and conquer memenuhi relasi rekursif sebagai berikut.

$$T(n) = \begin{cases} g(n), n \leq n_0 \\ 2T\left(\frac{n}{2}\right) + f(n), n > n_0 \end{cases}$$

Berdasarkan relasi tersebut, kompleksitas algoritma diatas adalah  $O(\log N)$ .

### III. IMPLEMENTASI ALGORITMA DAN ANALISIS

Parameter pada algoritma XGBoostClassifier beragam. Beberapa parameter mempunyai nilai diskrit. Namun sebagian besar parameter pada algoritma XGBClassifier mempunyai nilai kontinu. Pada nilai diskrit penulis cenderung menggunakan algoritma brute force dalam melakukan parameter tuning. Sedangkan untuk nilai kontinu, perlu dilakukan kuantifikasi terlebih dahulu untuk menentukan kemungkinan parameter optimal yang diinginkan.

Nilai parameter XGBoostClassifier :

1. learning\_rate : 0.01 - 0.2 (kontinu)
2. n\_estimators : 0 - tak hingga (diskrit)
3. max\_depth : biasanya 3 - 10 (diskrit)
4. min\_child\_weight : 0 - tak hingga (diskrit)
5. gamma : 0 - tak hingga, umumnya 0 - 1 (kontinu)
6. subsample 0 - 1 (diskrit)

Pada parameter yang bersifat kontinu. Penulis melakukan kuantifikasi sebagai berikut :

1. learning rate : 0.01, 0.02, 0.03, ....., 0.2
2. gamma : 0.1, 0.2, 0.3, ....., 1

Kemudian untuk setiap nilai pada setiap parameter dilakukan pengujian untuk mendapatkan nilai yang optimal untuk setiap parameter. Nilai yang didapatkan disini belum tentu optimal, karena nilai parameter bersifat kontinu. Oleh karena itu, perlu dilakukan penyesuaian parameter lagi dengan menggunakan decrease and conquer. Misalnya : ditemukan learning rate terbaik adalah 0.05. Penulis mengambil rentang 0.045 sampai 0.045 untuk diimplementasikan decrease and conquer. dalam mencari parameter optimal.

Dalam melakukan eksperimen ini, penulis menggunakan dataset yang disediakan oleh panitia lomba data mining joints UGM 2019. Dataset dapat diakses pada tautan berikut :

<https://www.kaggle.com/c/joints-dm-2019/data>.

Pada dataset tersebut, penulis membuat prediksi gender berdasarkan atribut yang lain. Penulis menggunakan cross validation dalam mengukur tingkat keakuratan prediksi pada dataset.

Berikut algoritma yang diimplementasikan oleh penulis :

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
train = pd.read_csv('train.csv')
test = pd.read_csv('test_data.csv')

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing, model_selection, metrics

gender_mapping = {"putri": 0, "putra": 1, "campur": 2}
train['gender']=train['gender'].map(gender_mapping)

train_data = train.drop('gender', axis=1)
target = train['gender']

from xgboost import XGBClassifier

import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)

clf = XGBClassifier(
    learning_rate=0.1,
    n_estimators=100,
    max_depth=1,
    min_child_weight=1,
    gamma=0,
    subsample=0.3,
```

```
colsample_bytree=0.3,
objective='binary:logistic',
scale_pos_weight=1,
)
```

```
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold,
n_jobs=1, scoring=scoring)
print(score)
```

```
round(np.mean(score)*100, 2)
```

Berdasarkan eksperimen yang penulis lakukan. tanpa menggunakan parameter tuning. algoritma XGBoostClassifier memberikan keakuratan prediksi sebesar 60.70 persen

Langkah Melakukan tuning

#### 1. Learning rate

Dalam menemukan parameter learning rate yang optimal penulis mencoba melakukan dengan cara brute force dan decrease and conquer. Dengan brute force didapatkan hasil sebagai berikut :

Learning rate	Akurasi
0.02	56.8
0.06	60.43
0.10	62.41
0.14	63.23
0.18	62.37

Learning rate naik pada saat 0.14 dan turun pada saat 0.18. Kemudian penulis menggunakan algoritma decrease and conquer untuk mendapatkan parameter learning rate yang optimal. Penulis membuat range antara 0.14 sampai 0.18

Pertama penulis memeriksa di learning rate 0.15, didapatkan akurasi sebesar 62.44. Sehingga penulis memperkecil range lagi dari 0.14 sampai 0.15. Kemudian diperiksa akurasi pada saat learning rate 0.145 yaitu sebesar 63.32. Range diperkecil kembali menjadi 0.145 sampai 0.150. Diperiksa akurasi pada saat learning rate sebesar 0.1475 yaitu sebesar 63.41.

Sehingga untuk learning rate, parameter yang optimal bernilai 0.1475

## 2. Max\_depth

Dalam menemukan parameter max\_depth yang optimal penulis mencoba melakukan dengan cara brute force dan didapatkan hasil sebagai berikut :

max_depth	akurasi
3	62.28
4	62.56
5	63.4
6	63.92
7	64.11
8	64.99
9	66.57
10	66.38

Dalam hal ini max\_depth bernilai diskrit. Sehingga tidak perlu menggunakan algoritma decrease and conquer untuk mendapatkan nilai optimal. Sehingga diperoleh bahwa saat max\_depth = 9. Akurasi prediksi = 66.57

## 3. min\_child\_weight

parameter min\_child\_weight dilakukan tuning menggunakan algoritma brute force dan didapatkan hasil sebagai berikut

min_child_weight	Akurasi
1	66.57
2	65.29
3	65.17
4	64.41
5	63.92

Berdasarkan hasil tersebut didapat bahwa akurasi tertinggi diperoleh saat min\_child\_weight 1.

## 4. gamma

Parameter gamma dilakukan tuning menggunakan algoritma brute force. Didapatkan hasil sebagai berikut

gamma	akurasi
0	66.57
0.2	65.84
0.4	65.71

Berdasarkan hasil tersebut parameter gamma yang optimal adalah saat gamma = 0

## 5. subsample

Dalam menemukan parameter learning rate yang optimal penulis mencoba melakukan dengan cara brute force. Dengan brute force didapatkan hasil sebagai berikut :

subsample	akurasi
0.3	66.57
0.5	67.02
0.7	67.78
0.9	68.42

Berdasarkan hasil tersebut parameter subsample yang optimal didapatkan saat subsample 68.42

## 6. colsample\_bytree

Dalam menemukan parameter learning rate yang optimal penulis mencoba melakukan dengan cara brute force dan decrease and conquer. Dengan brute force didapatkan hasil sebagai berikut :

colsample_bytree	Akurasi
0.3	68.42
0.5	69.02
0.7	69.33
0.9	68.63

Berdasarkan hasil tersebut nilai yang optimal terletak antara 0.7 sampai 0.9. Kemudian penulis membatasi range menjadi 0.7 sampai 0.9. Penulis memeriksa nilai akurasi pada saat 0.75 yaitu sebesar 68.84. Dari hasil tersebut penulis memperkecil range lagi menjadi 0.7 sampai 0.725. Kemudian penulis memeriksa nilai akurasi saat 0.725 yaitu sebesar 69.33. Dari hasil tersebut dapat disimpulkan bahwa akurasi terbesar saat nilai parameter `col_sample_bytree` sebesar 0.725.

### KESIMPULAN

Pada dataset yang diuji coba algoritma XGBoostClassifier memberikan solusi optimal saat parameter yang diberikan sebagai berikut :

1. `learning_rate=0.1475`
2. `n_estimators=100`
3. `max_depth=9,`
4. `min_child_weight=1,`
5. `gamma=0,`
6. `subsample=0.9,`
7. `colsample_bytree=0.75,`
8. `objective='binary:logistic',`
9. `scale_pos_weight=1`

### UCAPAN TERIMAKASIH

Atas terselesainya makalah ini saya selaku penulis mengucapkan terimakasih setulus tulusnya kepada ibu Masayu Leylia Khodra selaku dosen strategi algoritma kelas 02. Saya juga berterimakasih kepada teman teman saya di isaAcademy yang terus men-support proses belajar saya di ITB.

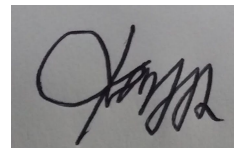
### REFENSI

- [1] <https://www.kaggle.com/c/joints-dm-2019/data>.
- [2] <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>
- [3] <http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/stmik.htm>

### PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2019



(Fajar Muslim / 13517149)