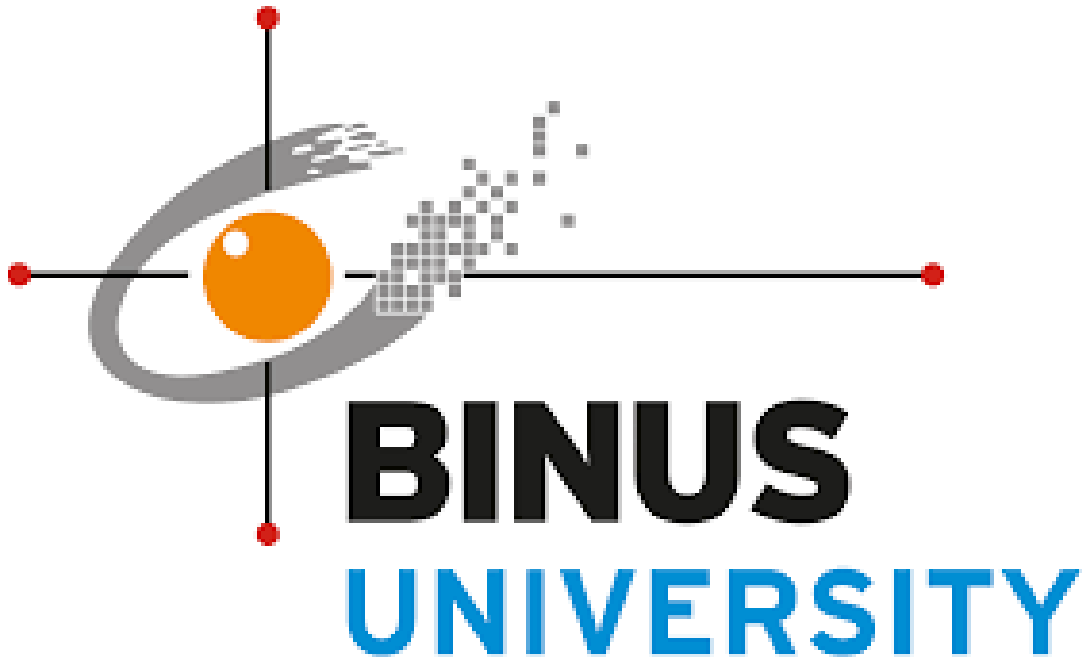


LAPORAN ASSURANCE OF LEARNING (AOL)

# **Implementasi Algoritma Tree Untuk Klasifikasi Tingkat Kebugaran Seseorang**



MATA KULIAH:  
MACHINE LEARNING

Oleh:

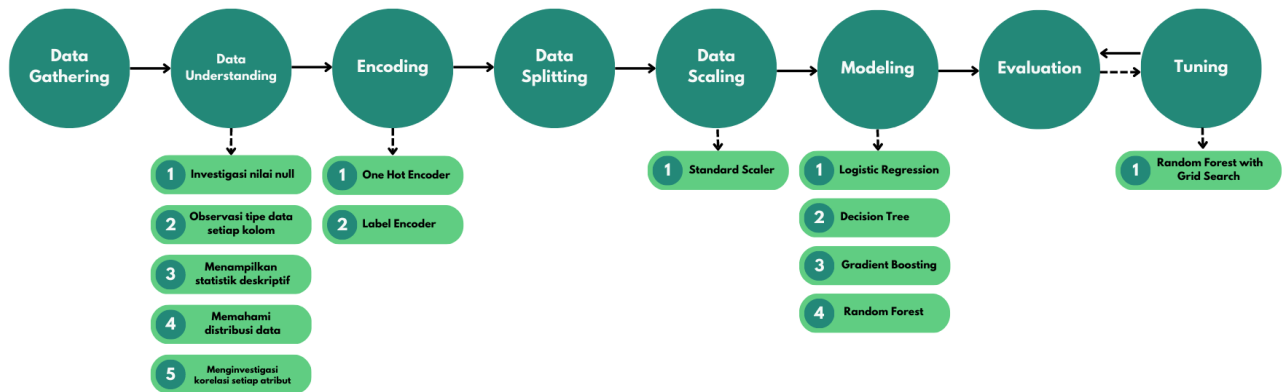
2502018064 - Narendra Nusantara Handradika

2540120452 - Jayagatha Saputra

2501967096 - Ryan Bertrand

**DATA SCIENCE  
SCHOOL OF COMPUTER SCIENCE  
UNIVERSITAS BINA NUSANTARA  
JAKARTA  
2023**

## BAB I PENDAHULUAN



### I.I Latar Belakang

Penilaian tingkat kebugaran merupakan evaluasi yang dilakukan untuk mengetahui kondisi fisik seseorang dan mengukur tingkat kesehatan yang dimiliki. Tujuan utama dari penilaian tingkat kebugaran adalah untuk mengetahui kekuatan, daya tahan, fleksibilitas, dan keseimbangan tubuh seseorang. Hal ini penting dilakukan untuk mengetahui kondisi fisik seseorang sebelum melakukan aktivitas fisik atau olahraga yang lebih intens, sehingga dapat diidentifikasi potensi risiko cedera atau masalah kesehatan lainnya.

Data yang kami gunakan adalah data Body Performance yang berisikan data seperti umur, gender, dan performance atau batasan maksimal dalam berolahraga maupun action yang dilakukan yang bisa dilakukan dari setiap individu berdasarkan jumlah rata-rata yang bisa dilakukan juga nilai maksimal yang bisa dilakukan.

Penilaian tingkat kebugaran dikelompokkan ke dalam kelas A, B, C, dan D untuk memudahkan interpretasi hasil dan memberikan ukuran yang sesuai dengan tingkat kondisi fisik seseorang. Kelas A merupakan tingkat kebugaran yang paling tinggi, yang menunjukkan bahwa individu tersebut memiliki kondisi fisik yang sangat baik dan dapat menjalani aktivitas fisik atau olahraga dengan intensitas tinggi tanpa risiko cedera atau masalah kesehatan. Kelas B menunjukkan tingkat kebugaran yang baik, dengan kondisi fisik yang cukup untuk menjalani aktivitas fisik atau olahraga dengan intensitas sedang. Kelas C menunjukkan tingkat kebugaran yang rendah, dengan kondisi fisik yang kurang baik dan memerlukan program latihan yang lebih intens untuk meningkatkan kondisi fisik. Kelas D menunjukkan tingkat kebugaran yang sangat rendah, dengan kondisi fisik yang buruk dan memerlukan perawatan medis atau program latihan yang sangat intens.

Penilaian tingkat kebugaran dikelompokkan ke dalam kelas-kelas ini adalah untuk mempermudah interpretasi hasil dan memberikan ukuran yang sesuai dengan tingkat kondisi fisik seseorang. Berikut merupakan beberapa variabel yang terdapat dalam dataset

- |   |   |
|---|---|
| 1. age (20-64) = numerikal                                  | 8. gripForce = numerikal                  |
| 2. Gender (F/M) = kategorikal                               | 9. sit and bend forward_cm =<br>numerikal |
| 3. height_cm = numerikal                                    | 10. sit-ups counts = numerikal            |
| 4. Weight_kg = numerikal                                    | 11. broad jump_cm = numerikal             |
| 5. body fat_% = numerikal                                   | 12. Class (A,B,C,D) = kategorikal         |
| 6. diastolic (diastolic blood pressure<br>(min)): numerikal |   |
| 7. systolic (systolic blood pressure<br>(min)) = numerikal  |   |

Terdapat dua kategorikal variabel yaitu Class dan Gender, sedangkan yang lainnya merupakan variabel numerik. Dari data yang kami dapatkan, kami memutuskan untuk menggunakan metode *Supervised Learning* lebih tepatnya dalam klasifikasi yang akan dibahas lebih lanjut pada BAB berikutnya,

## BAB II

### METODE PENELITIAN

#### II.I Pemilihan data

Data yang dipilih adalah data tentang tingkat kebugaran fisik dari sampel yang diambil dari semua kalangan umur. Data ini dipilih karena sesuai dengan tujuan penelitian yang ingin mengetahui faktor-faktor yang mempengaruhi tingkat kebugaran fisik lewat beberapa aktivitas yang dapat dilakukan setiap individu.


#### II.II Pemanggilan Data & Library

```
import pandas as pd
import numpy as np
import seaborn as sns
data = pd.read_csv("bodyPerformance.csv")
data.head()
```

#### II.III Exploratory Data Analysis

##### II.III.I Missing Value

```
data.isna().sum()
```



age	0
gender	0
height_cm	0
weight_kg	0
body fat_%	0
diastolic	0
systolic	0
gripForce	0
sit and bend forward_cm	0
sit-ups counts	0
broad jump_cm	0
class	0
dtype: int64	

Langkah awal dalam Exploratory Data Analysis disini dilakukan pengecekan missing values. Didapati hasil tidak ditemukan bahwa adanya missing value dengan munculnya nilai 0 pada setiap variabel.

## II.III.II Data Description

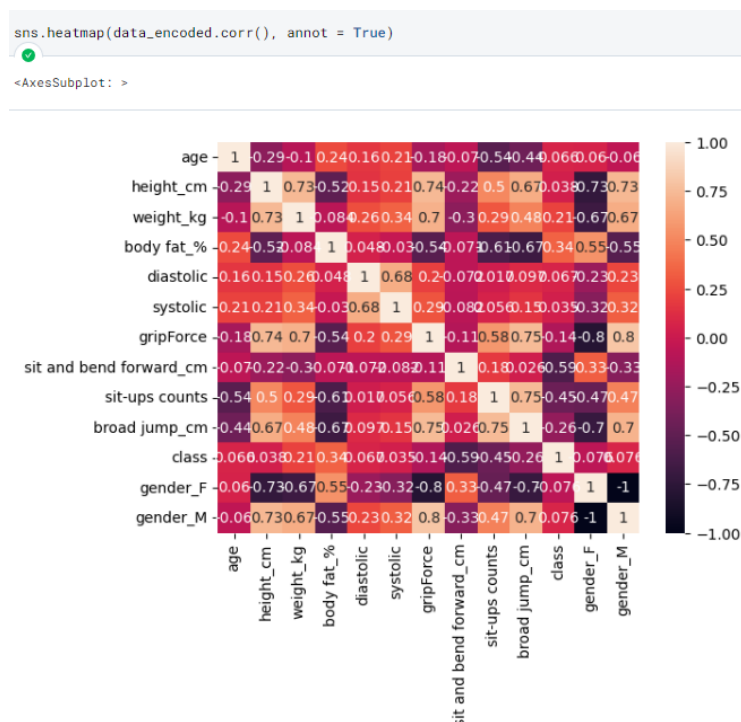
data.describe()									
	age float64	height_cm float64	weight_kg float64	body fat_% float64	diastolic float64	systolic float64	gripForce float64	sit and bend forward_cm	
count	13393.0	13393.0	13393.0	13393.0	13393.0	13393.0	13393.0	13393.0	13393.0
mean	36.77510639886508	168.5598073620548	67.4473157619652	23.240164950869858	78.79684163368923	130.23481669528857	36.96387739864108	15.20926827447174	
std	13.625639475291313	8.426582550560243	11.94966634270741	7.256844079929906	10.742033099909756	14.713953521704253	10.624864027335336	8.456677009240204	
min	21.0	125.0	26.3	3.0	0.0	0.0	0.0	-25.0	
25%	25.0	162.4	58.2	18.0	71.0	120.0	27.5	10.9	
50%	32.0	169.2	67.4	22.8	79.0	130.0	37.9	16.2	
75%	48.0	174.8	75.3	28.0	86.0	141.0	45.2	20.7	
max	64.0	193.8	138.1	78.4	156.2	201.0	70.5	213.0	

8 rows, showing 10 per page

<< < Page 1 of 1 > >>

Berikutnya dilakukan analisis data statistik menggunakan fungsi describe() untuk mengetahui nilai-nilai dari setiap variabel. Seperti pada variabel umur, tinggi, berat dan body\_fat yang rata-ratanya disekitar angka 36 tahun, tinggi 168cm, berat 67kg, dan body\_fat 23%. Dari sana muncul apakah itu merupakan hal yang sudah cukup baik, atau belum?

## II.III.II Correlation



Korelasi disini digunakan untuk mencari data yang berpotensi dengan menargetkan variabel (y) “Class” dan mendrop data yang korelasinya rendah . Namun setelah dipertimbangkan, dalam klasifikasi drop data tidak terlalu berpengaruh dibandingkan untuk regresi. Oleh sebab itu, tidak ada data yang akan di drop dalam persiapan pembuatan model.

## II.IV Data Pre-Processing

### II.IV.I Encoding

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
onehot_encoder = OneHotEncoder()
label_encoder = LabelEncoder()

data_encoded = data.copy()
gender_encoded = pd.DataFrame(onehot_encoder.fit_transform(data_encoded[['gender']]).toarray(),
                              columns=onehot_encoder.get_feature_names_out())
data_encoded = pd.concat([data_encoded, gender_encoded], axis=1)
data_encoded.drop(['gender'], axis=1, inplace=True)

data_encoded['class'] = label_encoder.fit_transform(data_encoded['class'])
data_encoded
```

```
data['class'].value_counts()
```

```
C    3349
D    3349
A    3348
B    3347
Name: class, dtype: int64
```

```
data_encoded['class'].value_counts()
```

```
2    3349
3    3349
0    3348
1    3347
Name: class, dtype: int64
```

Encoding dilakukan dalam bentuk mengubah variabel kategorikal yaitu “Class” dan “Gender” menjadi numerikal. Tujuannya agar mesin dapat memproses setiap variabel yang ada dalam pembuatan model.

### II.IV.II Data Splitting

```
drop_class = data_encoded.drop(["class"], axis = 1)
```

```
x = (drop_class).values
y = data_encoded['class'].values
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 100, stratify = y)
```

Data Splitting dilakukan untuk membagi data menjadi beberapa bagian yang berbeda sebagai keperluan evaluasi model machine learning. Tujuannya adalah untuk memastikan bahwa model tersebut dapat bekerja dengan baik pada data baru dan untuk mencegah overfitting, yaitu ketika model terlalu mempelajari data latih dan tidak dapat membuat generalisasi yang baik pada data baru. Bagian data digunakan sebagai data training (latihan dan validasi), dan data testing.

### II.IV.III Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```


Scaling dilakukan untuk menyesuaikan skala data agar memiliki skala yang sama. Scaling disini menggunakan StandardScaler (standarisasi). Tujuan diadakannya scaling agar dapat meningkatkan akurasi, stabilitas model dan juga mempercepat konvergensi algoritma optimasi.

## II.V Membuat Model

### II.V.I Classification Report

```
from sklearn.metrics import classification_report

# Independent Prediction
y_pred = model.predict(x_test_scaled)
y_pred
```



```
array([3, 2, 3, ..., 1, 0, 3])
```

Classification report digunakan untuk menilai kinerja model pembelajaran mesin pada klasifikasi. Laporan ini menyediakan informasi tentang akurasi model, seperti precision, recall, f1-score, dan support yang mana berguna untuk menentukan apakah model bekerja dengan baik dan dapat membantu dalam perbandingan seleksi model yang berbeda.

## BAB III

### HASIL DAN PEMBAHASAN

#### III.I Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train_scaled, y_train)
```

```
LogisticRegression()
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.70	0.73	0.72	670
1	0.47	0.45	0.46	669
2	0.52	0.51	0.52	670
3	0.79	0.79	0.79	670
accuracy			0.62	2679
macro avg	0.62	0.62	0.62	2679
weighted avg	0.62	0.62	0.62	2679

Logistic Regression adalah metode klasifikasi biner yang memodelkan probabilitas label positif menggunakan fungsi logistik dan memprediksi label akhir menggunakan threshold. Dari pemodelan diatas didapat hasil akurasi 62%. Akurasi tersebut dinilai masih cukup kecil sehingga dibuatlah pemodelan yang lain.

#### III.II Decision Tree

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
dtree = dtree.fit(x_train_scaled, y_train)
features = data_encoded.drop(['class'],axis=1).columns

y_pred_tree = dtree.predict(x_test_scaled)
print(classification_report(y_test, y_pred_tree))
```

	precision	recall	f1-score	support
0	0.69	0.70	0.70	670
1	0.54	0.54	0.54	669
2	0.60	0.57	0.58	670
3	0.78	0.81	0.79	670
accuracy			0.65	2679
macro avg	0.65	0.65	0.65	2679
weighted avg	0.65	0.65	0.65	2679


Decision Tree adalah algoritma yang digunakan untuk membuat model pembuatan keputusan berdasarkan aturan if-then. Membagi data menjadi cabang-cabang berdasarkan fitur-fitur dan membuat prediksi label akhir pada cabang akhir. Dari pemodelan diatas didapat hasil akurasi 65%. Akurasi tersebut lebih baik dari model Logistic Regression namun masih cenderung kurang bagus sehingga dibuatlah pemodelan yang lain.



### III.III Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

boosting = GradientBoostingClassifier()
boosting.fit(x_train_scaled, y_train)
y_pred_boosting = boosting.predict(x_test_scaled)
print(classification_report(y_test, y_pred_boosting))
```




	precision	recall	f1-score	support
0	0.71	0.87	0.78	670
1	0.59	0.60	0.59	669
2	0.71	0.62	0.66	670
3	0.93	0.82	0.87	670
accuracy			0.73	2679
macro avg	0.73	0.73	0.73	2679
weighted avg	0.73	0.73	0.73	2679

Gradient Boosting adalah teknik pembelajaran mesin yang menggabungkan beberapa model menjadi satu yang lebih kuat dengan memperbaiki kesalahan dari model sebelumnya (decision tree, logistic etc.). Akurasi mulai meningkat drastis dari awalnya 65% menjadi 73%. Karena masih berharap ada model yang memiliki akurasi lebih baik, dilanjutkanlah pencarian model tersebut.

### III.IV Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(x_train_scaled, y_train)
y_pred_rfc = rfc.predict(x_test_scaled)
print(classification_report(y_test, y_pred_rfc))
```



	precision	recall	f1-score	support
0	0.75	0.86	0.80	670
1	0.62	0.61	0.62	669
2	0.71	0.67	0.69	670
3	0.91	0.84	0.87	670
accuracy			0.74	2679
macro avg	0.75	0.74	0.74	2679
weighted avg	0.75	0.74	0.74	2679

Random Forest adalah algoritma yang menggabungkan beberapa decision tree menjadi satu model. Setiap tree dalam forest dibangun dari subset acak data, fitur, dan prediksi label akhir dengan cara mengambil rata-rata atau voting dari semua tree. Diantara semua model yang sudah dibuat, Random Forest memiliki akurasi terbaik yaitu sebesar 74%. Karena ingin meningkatkan akurasi, setelah ini akan dilakukan tahapan tuning pada model Random Forest.

### III.V Evaluasi Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train_scaled, y_train)
y_pred_rfc = rfc.predict(x_test_scaled)
print(classification_report(y_test, y_pred_rfc))
```

	precision	recall	f1-score	support
0	0.75	0.86	0.80	670
1	0.62	0.61	0.62	669
2	0.71	0.67	0.69	670
3	0.91	0.84	0.87	670
accuracy			0.74	2679
macro avg	0.75	0.74	0.74	2679
weighted avg	0.75	0.74	0.74	2679

```
from sklearn.model_selection import GridSearchCV

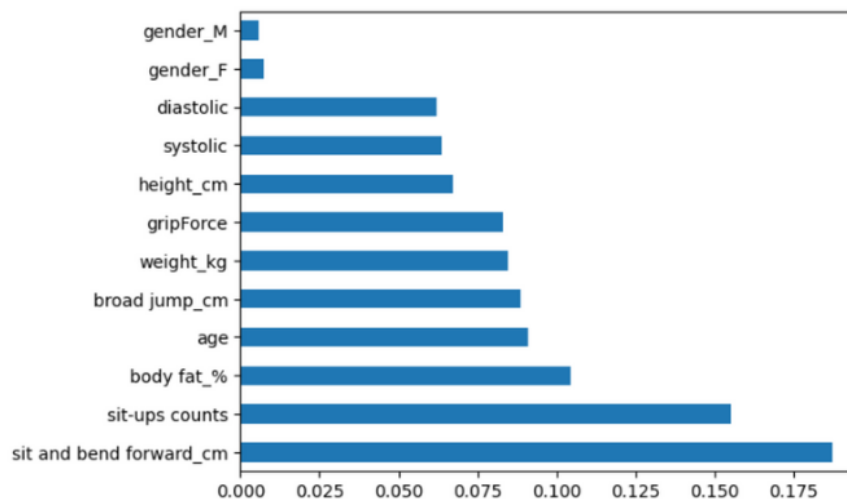
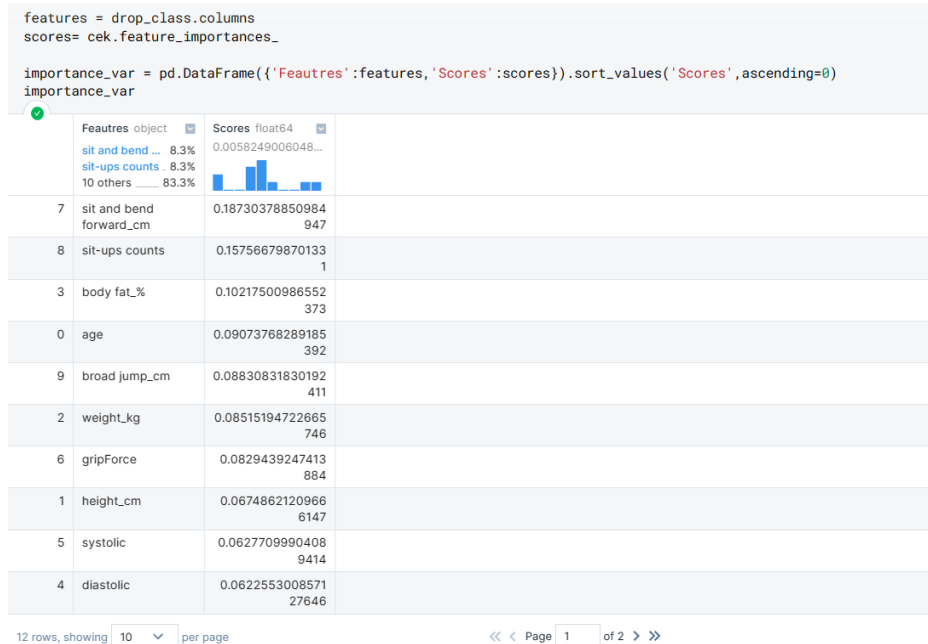
params = {'max_depth':[100,200,300], 'n_estimators':[50,100,200]}
rfcGrid = GridSearchCV(RandomForestClassifier(class_weight='balanced'),
                        params,cv=5,scoring='accuracy').fit(x_train_scaled,y_train)

rfcGridPred = rfcGrid.predict(x_test_scaled)
print(classification_report(rfcGridPred,y_test))
```

	precision	recall	f1-score	support
0	0.87	0.75	0.80	772
1	0.62	0.63	0.62	653
2	0.67	0.71	0.69	635
3	0.84	0.91	0.88	619
accuracy			0.75	2679
macro avg	0.75	0.75	0.75	2679
weighted avg	0.75	0.75	0.75	2679

Percobaan untuk menaikkan akurasi Random Forest dibantu dengan menggunakan metode GridSearch sebagai Tuning untuk meningkatkan akurasi dari model awal. Sehingga didapat untuk model yang akan di deploy akurasi sebesar 75% diikuti oleh F1-Score yang juga ikut turut meningkat menjadi 88%.

## BAB IV KESIMPULAN



Random Forest Classifiers memprediksi Penilaian tingkat kebugaran seseorang dimana model ini memiliki akurasi yang lumayan tinggi dan dapat disimpulkan bahwa model ini ditampilkan dengan baik dalam memprediksi Body Performance. Sit and bend forward merupakan data terpenting karena data ini digunakan untuk menentukan fleksibilitas seseorang dan berpengaruh dalam meningkatkan strength dan endurance yang mana dapat mempengaruhi hasil dari body performance.