



# Knowledge Discovery Community

Open Recruitment  
OmahTi 2019/2020



**IRVIN ANDERSEN**

Computer Science UGM 2018

18/425519/PA/18411

[Irvinn1294@gmail.com](mailto:Irvinn1294@gmail.com)

087776718595

[linkedin.com/in/irvinn](https://www.linkedin.com/in/irvinn)

# Daftar isi

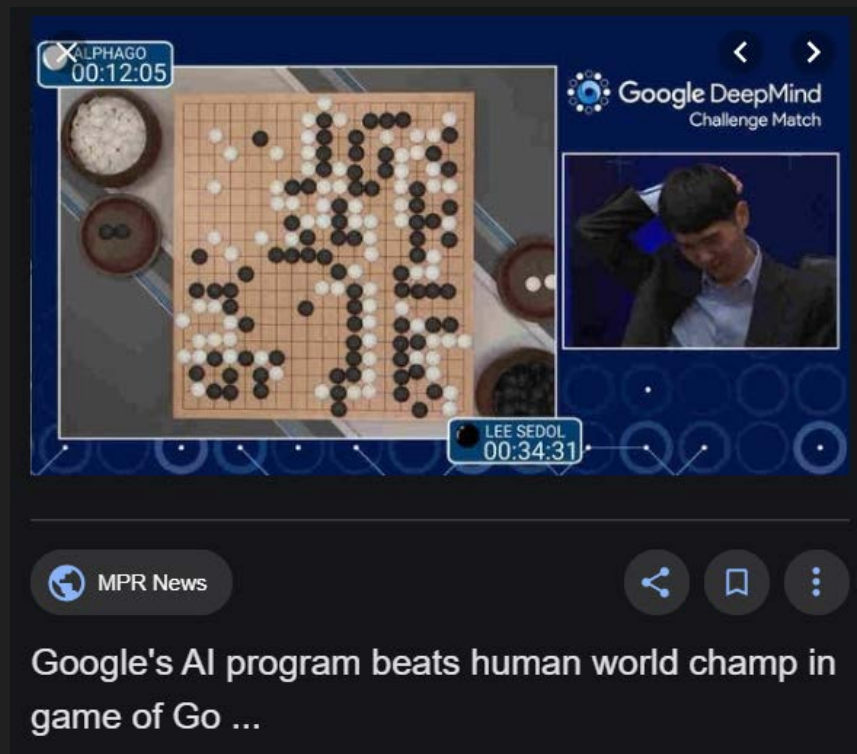
HALAMAN JUDUL .....	1
BIODATA .....	2
DAFTAR ISI .....	3
TEORI	
PEMBAHASAN NOMOR SATU .....	4
PEMBAHASAN NOMOR DUA .....	14
PEMBAHASAN NOMOR TIGA .....	22
PRAKTIK	
PEMBAHASAN NOMOR EMPAT (PERCOBAAN SATU) .....	30
PEMBAHASAN NOMOR EMPAT (PERCOBAAN DUA) .....	44
JAWABAN AKHIR NOMOR EMPAT .....	58
PEMBAHASAN NOMOR LIMA .....	61
JAWABAN AKHIR NOMOR LIMA .....	75
PEMBAHASAN NOMOR ENAM .....	77

# Teori

1. Jelaskan secara detail menggunakan bahasa anda tentang apa itu Data Science !

## Apa itu data science?

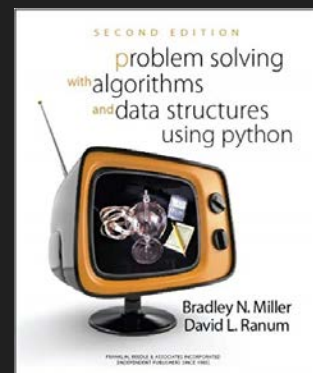
Yang mungkin kalian bayangkan, Data Science hanya seputar AI dan Deep Learning. Karena biasanya itu yang disorot oleh media.



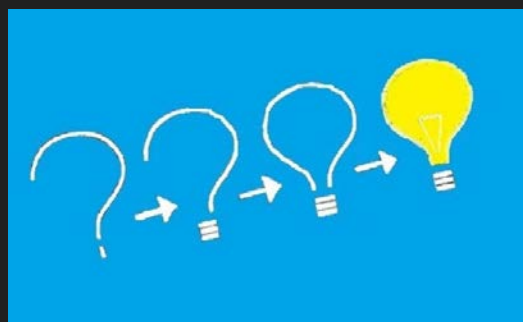
## TAPI?

Data science tidak terbatas pada hal tersebut, Data science juga memiliki beberapa tahapan, Data Scientist yang sering dibutuhkan oleh perusahaan adalah seorang Data Scientist Analyst.

Tugas dari Data Scientist adalah menyelesaikan masalah yang di alami perusahaan menggunakan data.



Yang nyatanya Data Science adalah seorang yang dibutuhkan untuk membuat dampak untuk perusahaan, yaitu menjadi Problem Solver bagi perusahaan, kita akan diberikan pertanyaan yang membingungkan dan perusahaan mengharapkan menentukan jalan bagi perusahaan.



# Definisi dari Data Science itu sendiri

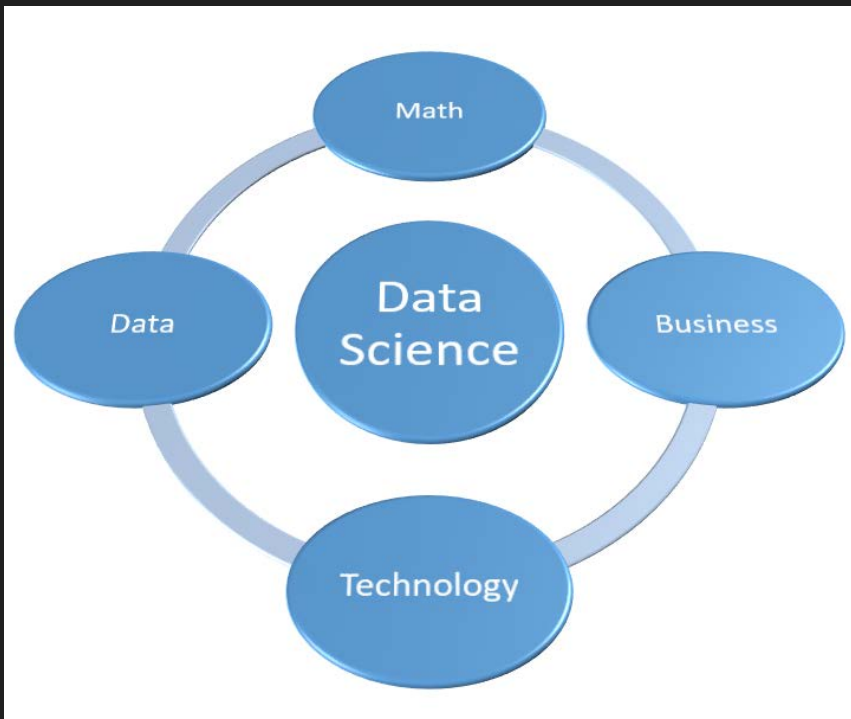
Data Science adalah bidang multi-disiplin yang menggunakan metode, proses, algoritma dan sistem ilmiah untuk mengekstraksi pengetahuan dan wawasan dari data terstruktur dan tidak terstruktur.

Konsep dari Data Science sendiri adalah gunakan perangkat keras yang paling kuat, sistem pemrograman yang paling kuat, dan algoritma yang paling efisien untuk menyelesaikan masalah

Data Science adalah konsep untuk menyatukan statistik, analisis data, machine learning dan metode terkait" untuk "memahami dan menganalisis fenomena aktual" dengan data. Ini menggunakan teknik dan teori yang diambil dari banyak bidang dalam konteks matematika, statistik, ilmu komputer, dan ilmu informasi.

# Awal terbentuknya Data Science

Untuk konsep dari Data Science sendiri adalah Data Mining yang dikembangkan dengan menggabungkan dengan Computer Science. Data mining ini digunakan metode statistik untuk menghasilkan informasi yang lebih berguna dan metode ini di namai Data Science.



# Testimoni sebagai seorang Data Scientist

Data Science tidak bekerja sendiri, data science butuh masalah yang di berikan oleh job desc lain, disaat itu kita bertugas untuk menemukan informasi yang berguna dan akan mengembangkan perusahaan.

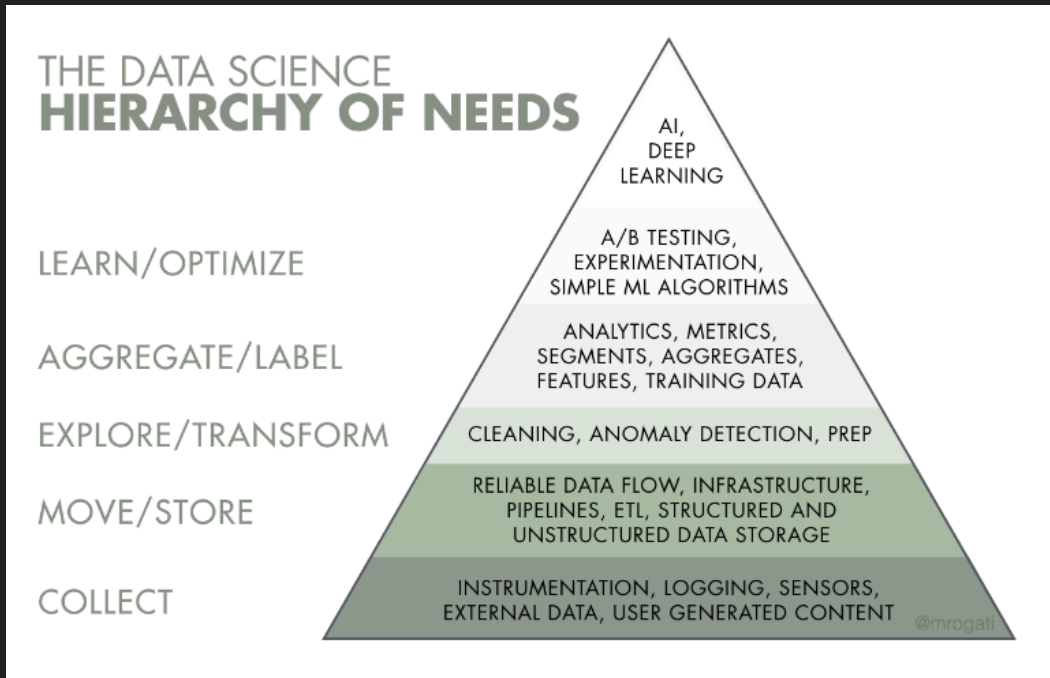


Semua masalah yang bisa berhubungan dengan arimatik ataupun masalah yang bisa jadi tidak masuk akal akan diberikan kepada anda sebagai Data Scientist. Kita akan menyelesaikan dengan metode statistik, analisis data, machine learning dan metode terkait yang di butuhkan.

Jawaban untuk masalah ini bisa dalam berbagai bentuk yaitu informasi, data product, dan rekomendasi



# Pekerjaan dari Data Scientist



Pekerjaan Data Scientist memiliki beberapa tahapan. Tetapi keperluan ini akan berbeda-beda di setiap level Perusahaan.

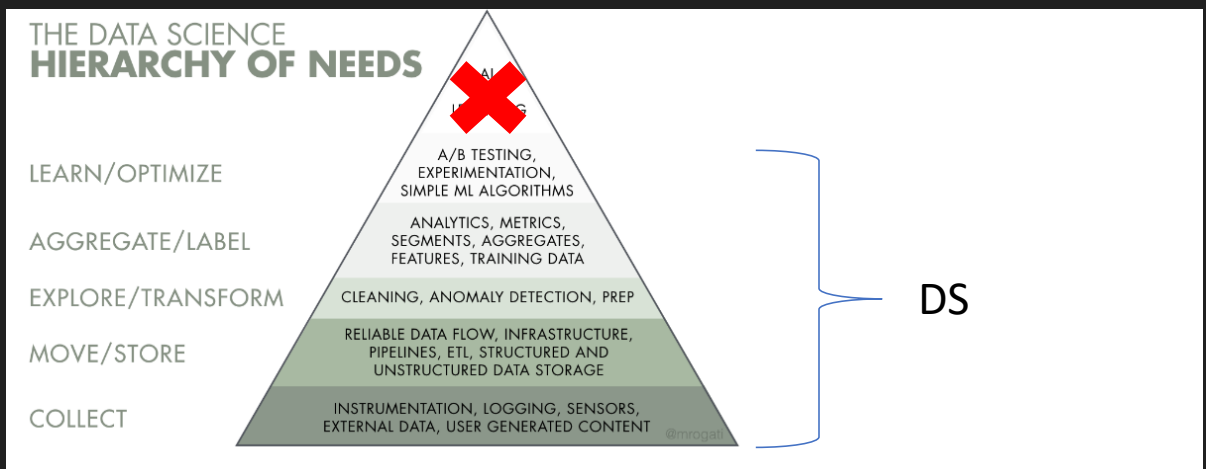
# Pekerjaan dari Data Scientist

- Startup / Perusahaan Kecil

Di sebuah Startup, di karenakan kurang nya sumber daya manusia ,mungkin seorang Data Scientist akan melakukan semua pekerjaan, kecuali AI dan deep learning.

Kenapa tidak AI dan deep learning?

Karena masih begitu banyak hal lain yang perlu di kerjakan untuk mengembangkan dan menyelesaikan permasalahan perusahaan.

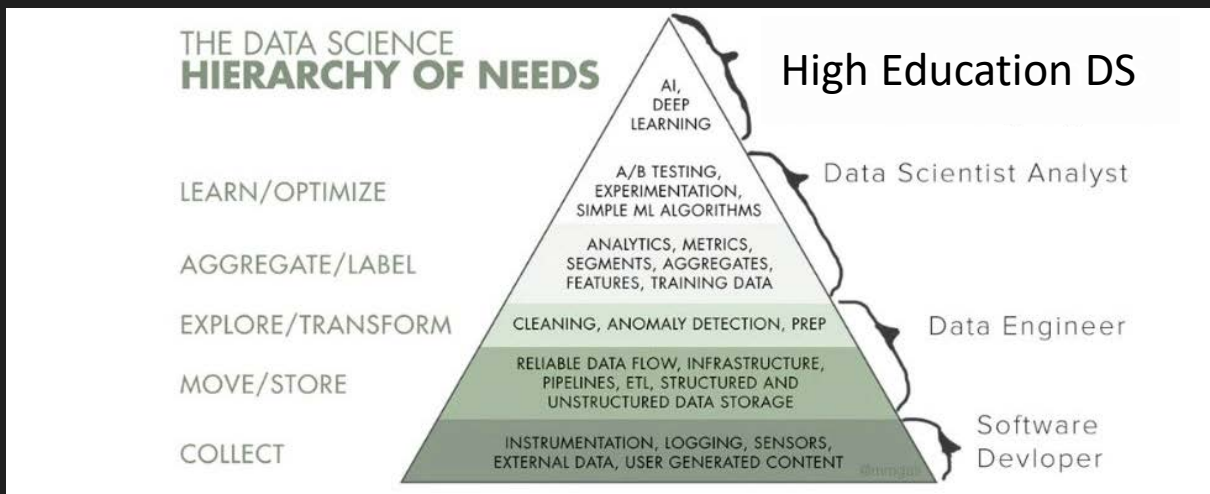


Data scientist akan membuat software untuk mengumpulkan data, menganalisa sendiri, membuat metrics sendiri dan menguji sendiri.

## Pekerjaan dari Data Scientist

- Perusahaan taraf Sedang / Medium

Di sebuah perusahaan Medium, akan lebih banyak memiliki tenaga kerja dan akan mulai di bagi pekerjaan.



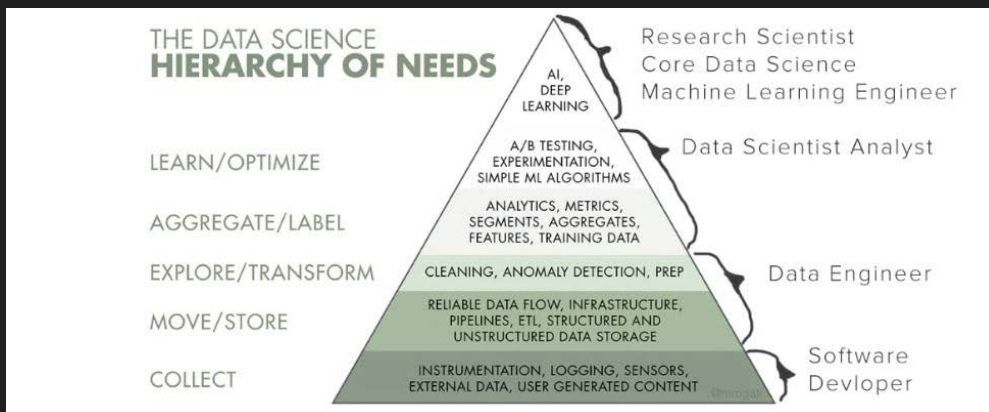
Collecting data akan di lakukan oleh Software Engineer, Move and Explore akan di lakukan oleh seorang Data engineer , dan Aggregate and Learn akan dilakukan oleh seorang Data Science.

Beberapa mungkin akan mulai mengerjakan AI dan deep learning untuk yang membutuhkan dan akan dibutuhkan orang dengan berpendidikan tinggi dengan gelar PhD ataupun Master untuk mengerjakan hal ini.

## Pekerjaan dari Data Scientist

- Perusahaan Tinggi / High Level

Disebuah perusahaan taraf besar, perusahaan telah memiliki banyak uang dan dapat memilih tenaga kerja yang berkualitas, disini pekerja tidak lagi perlu memikirkan pekerjaan yang tidak harus di kerjakan olehnya dan bisa fokus ke pekerjaan mereka sendiri.



Data Scientist tidak lagi mengerjakan AI, tetapi akan di handling oleh Research Scientist ataupun Machine Learning Enginner.

Di tahap ini yang menjadi penganalisa adalah Data Science Analyst

## Sebuah Curhatan Data Scientist Newbie

Pada awalnya saya tidak percaya apa yang dikatakan oleh atasan kerja saya, bahwa AI dan Machine Learning belum akan di pakai di sebuah perusahaan kecil ataupun medium. Tetapi semakin hari saya baru menyadari bahwa hal itu benar.

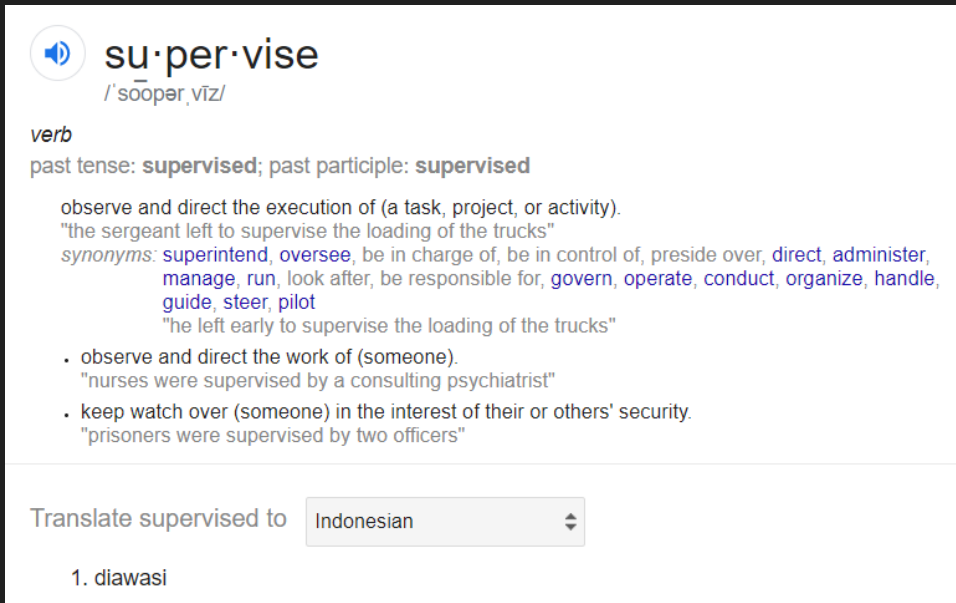
Karena banyak pekerjaan lain yang bisa membuat perusahaan lebih berkembang di banding membuat AI dan Deeplearning. Karena penting untuk membangun fondasi yang kuat terlenbih dahulu.

Masih banyak dasar yang kita perlu pelajari, dan butuh langkah dalam belajar hal itu. AI, Deep Learning, dan Machine Learning bisa di terapkan nanti ketika fondasi dari perusahaan sudah kuat dan tidak lagi banyak masalah yang perlu di kerjakan.



## 2. Jelaskan perbedaan Supervised Learning, Semi Supervised Learning, dan Unsupervised Learning dalam bahasa anda!

Untuk mendefinisikan supervised dan unsupervised learning, kita akan melihat dari definisi dari kata itu sendiri.



**su·per·vise**  
/'sööpəˌvīz/

*verb*  
past tense: **supervised**; past participle: **supervised**

observe and direct the execution of (a task, project, or activity).  
"the sergeant left to supervise the loading of the trucks"

*synonyms:* **superintend, oversee**, be in charge of, be in control of, preside over, **direct, administer, manage, run**, look after, be responsible for, **govern, operate, conduct, organize, handle, guide, steer, pilot**  
"he left early to supervise the loading of the trucks"

- observe and direct the work of (someone).  
"nurses were supervised by a consulting psychiatrist"
- keep watch over (someone) in the interest of their or others' security.  
"prisoners were supervised by two officers"

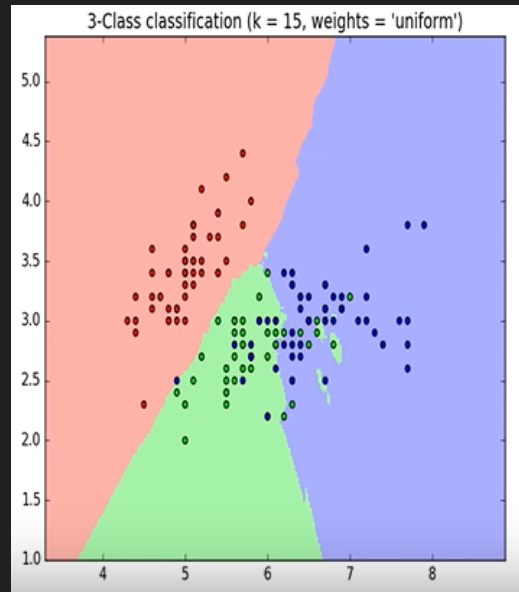
Translate supervised to Indonesian

1. diawasi

Arti dari supervised sendiri adalah mengawasi atau mengarahkan sesuatu. Yang kita supervise disini adalah model dari machine learning yang dapat melakukan klasifikasi.



Contoh Grafik dari klasifikasi yang di hasilkan adalah seperti ini



Lalu bagaimana kita akan mengarahkan model ini?  
Jadi yang kita lakukan adalah mengajarkan model ini dengan memberikan data ini dengan informasi sehingga model ini bisa digunakan untuk memprediksi nantinya

Lalu bagaimana kita mengajarkan model ini?  
Kita akan mengajarkannya dengan cara melatih dengan beberapa dataset yang memiliki label  
Penting untuk di garis bawahi bahwa data yang digunakan adalah data yang berlabel.

## Seperti apa data yang memiliki Label?

Rerata UN	Rata-rata	Hasil
87	83	Lolos
83	91	Lolos
91	84	Lolos
90	100	Lolos
82	75	Coba lagi
80	78	Coba lagi
78	77	Coba lagi
81	72	Coba lagi
82	72	Coba lagi
80	84	Lolos
79	91	Lolos
91	79	Lolos
83	75	Coba lagi
80	75	Coba lagi
81	77	Lolos
90	88	Lolos
91	91	Lolos
92	87	Lolos
79	96	Lolos
93	84	Lolos
93	71	Coba lagi
95	84	Lolos
82	97	Lolos
84	84	Lolos
80	78	Coba lagi
79	96	Lolos
90	81	Lolos
92	95	Lolos

Ini contoh dari data yang berlabel, data ini dimuat dengan jawaban yang benar.

Data-data yang diatas ini yang bernama Rata-rata UN, Rata-rata Sekolah dan Hasil merupakan Attribute dari dataset ini.

Dan kolom Hasil di sini adalah fitur

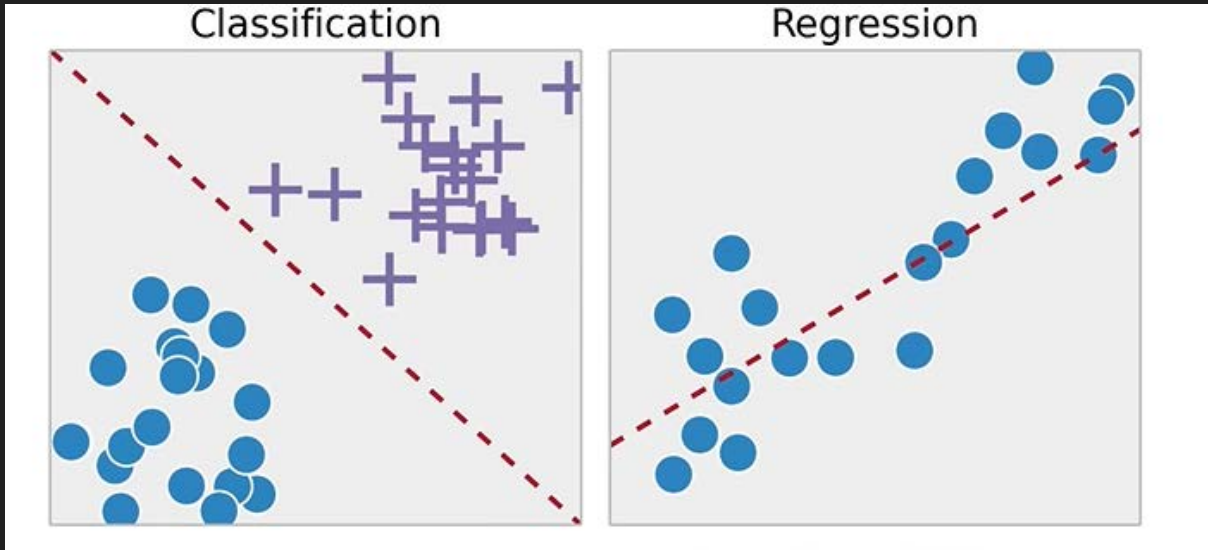
1 row dari data ini memiliki semua atribut dari data ini, dan di namakan observation

Jenis data ini memiliki 2 jenis data,

- Numerikal value, valu yang paling umum digunakan dalam Machine Learning
- Non numerical, berisikan kategorial untuk klasifikasi



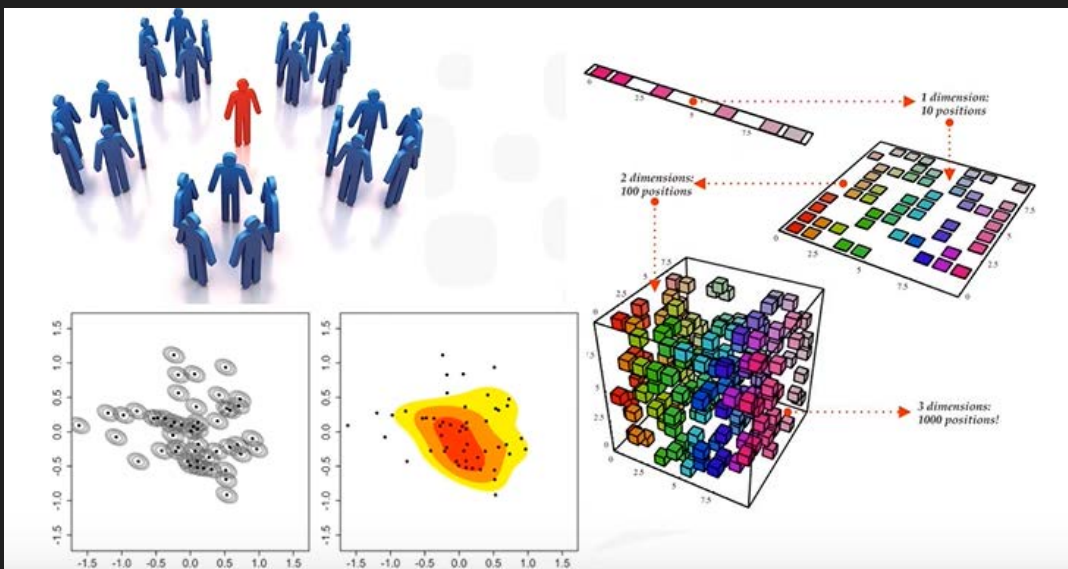
Ada 2 tipe dalam supervised learning, yang pertama adalah Classification dan Regression.



Ini tidak akan saya jelaskan disini karena akan menjadi sangat banyak dan sedikit off topic.

Algoritma dari Unsupervised Learning memiliki kekompleksan yang lebih sulit dari model supervised karena kita tidak tahu informasi tentang data yang diharapkan.

Dengan unsupervised learning kita bisa mendapatkan informasi-informasi seperti group atau clustering, estimasi kerapatan, dan pengurangan dimensi



Dibandingkan dengan supervised learning, test yang dimiliki oleh unsupervised learning lebih sedikit dan model untuk memastikan data tersebut akurat juga lebih sedikit. Maka Hasil nya tidak akan terkontrol.

# Perbedaan Supervised dan Unsupervised Learning

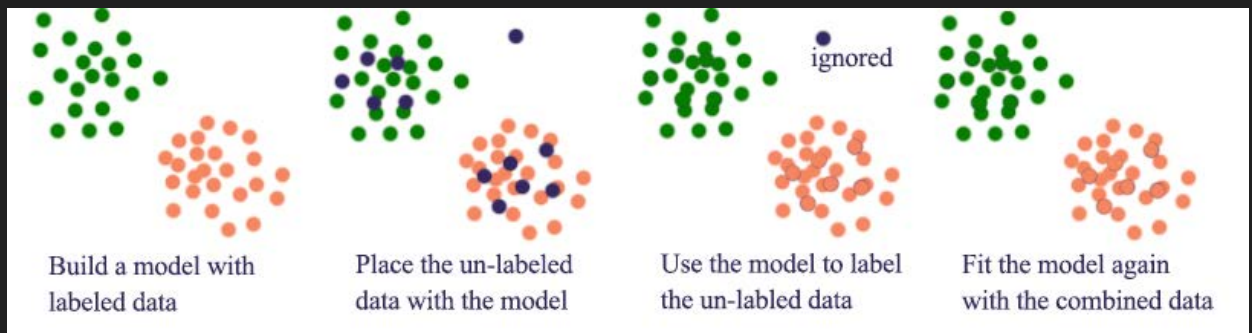
Supervised	Unsupervised
<ul style="list-style-type: none"><li>• Data yang digunakan data yang berlabel</li><li>• Supervised memiliki algoritma untuk hal seperti klasifikasi dan regresi</li></ul>	<ul style="list-style-type: none"><li>• Data yang digunakan data yang tidak memiliki label</li><li>• Unsupervised memiliki algoritma seperti clustering, dimana kita menganalisis pola dan mengelompokkan data yang tidak berlabel</li></ul>

## Dan, apa itu semi-supervised Learning?

Semi-supervised Learning adalah kombinasi dari Supervised dan Unsupervised Learning. Data yang kita gunakan adalah data yang berlabel dan data yang tidak berlabel.

Contoh mudah dari semi-supervised

Kita memiliki data yang tidak memiliki label, dan untuk melakukannya kita akan memberikan label, tetapi memberikan label pada data secara manual tidaklah mudah dan tidak praktis.

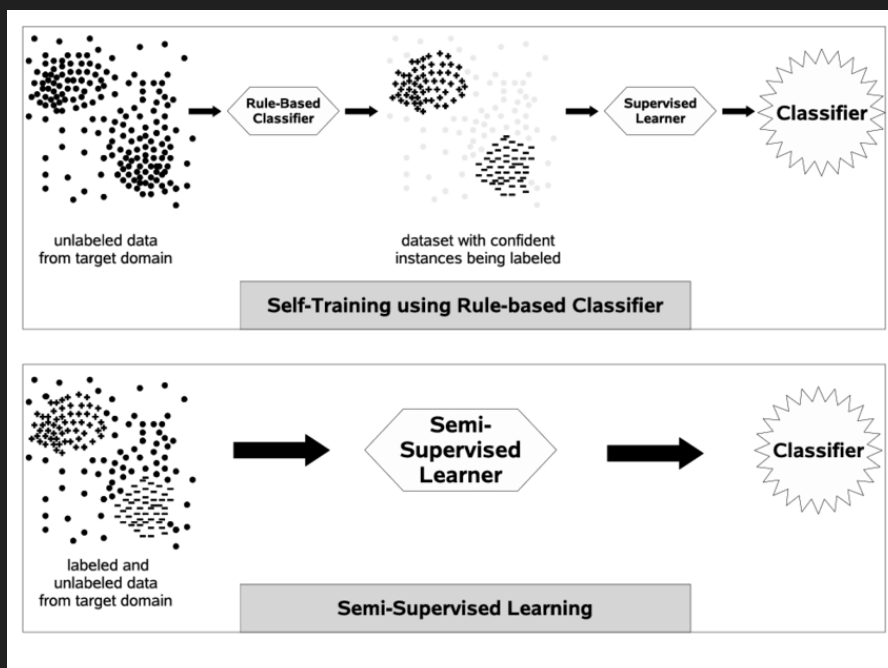


Yang akan kita lakukan adalah melakukan labelling manual pada beberapa data dan menggunakan data in untuk menjadi model training. Biasanya model ini digunakan dalam neural network untuk labelling.

Metode ini bernama Pseudo Labeling, cara kerja labelling ini adalah memberikan label pada beberapa model dan akan digunakan untuk training data dan mendapatkan label. Lalu kita akan membuat model dari data yang memiliki label ini (Supervised)

Dan setelah itu dengan model yang telah kita buat, kita akan menggunakan untuk data yang tidak memiliki label dan idetifikasi labelnya.

Lalu kita akan menggabungkan data-data itu dan mendapatkan model yang kita inginkan



### 3. Jelaskan secara detail proses pengolahan data dari Raw Data hingga menjadi sebuah Informasi !

Sebelum masuk kedalam tahapan Data Processing, kita perlu mengetahui komponen dari Data Processing itu sendiri terlebih dahulu.

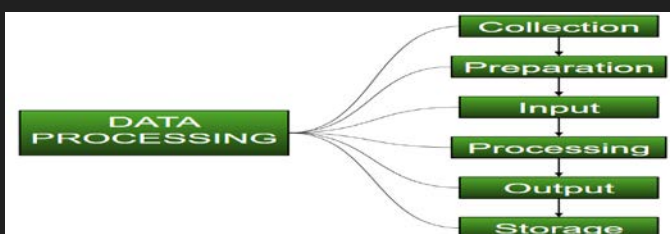
Definisi dari Raw Data adalah

“Raw Data adalah istilah untuk data yang dikumpulkan pada sumber yang belum mengalami pemrosesan atau manipulasi lainnya.”  
(wikipedia.org)

Definisi dari Data Processing adalah

Data Processing adalah konversi Raw Data ke dalam bentuk informasi yang bermakna melalui proses. Data akan di proses untuk menghasilkan hasil yang digunakan untuk menyelesaikan masalah.

Raw Data ini akan di proses menggunakan Software untuk menghasilkan sebuah output. Untuk menghasilkan informasi akan melalui proses ringkasan, perhitungan, penyimpanan dan di sajikan dalam bentuk diagram, laporan, grafik.



# Proses dalam Data Processing

## I. Collecting

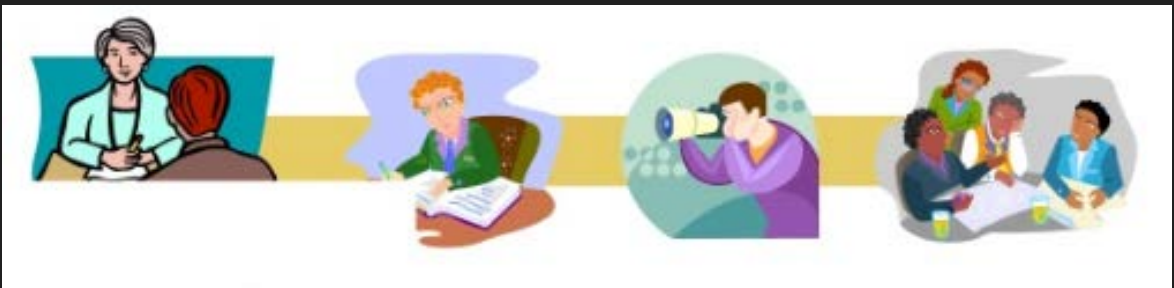
Tahap pertama dalam processing adalah pengumpulan data. Tahap ini sangatlah penting, karena kualitas data akan menentukan output dari Data processing tersebut.

Pengumpulan juga harus memastikan bahwa data akurat sehingga hasilnya akan valid.

Pengumpulan data ini juga memiliki berbagai jenis

- Sensus : Pengumpulan data tentang segala sesuatu dalam kelompok atau populasi statistic
- Survei Sampel : metode pengumpulan yang hanya mencakup sebagian dari total populasi
- Administrative by-product : Pengumpulan data produk dari operasi hari ke hari

Dalam dunia kerja tugas ini di akan di jalankan oleh Software Developer. Dimana dengan sebuah program akan menggali data untuk di proses nantinya.



# Proses dalam Data Processing

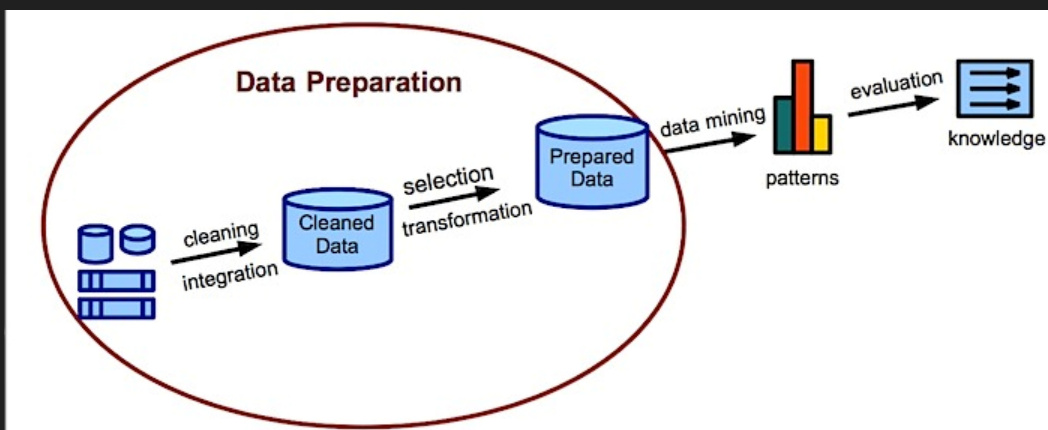
## 2. Preparation

Preparasi ini dilakukan dengan mengkonversi data ke dalam bentuk yang bisa digunakan untuk analisis dan pemrosesan lebih lanjut. Raw Data tentu tidak bisa langsung diproses dan harus di jaga keakuratannya.

Persiapan ini bisa dilakukan dalam mengabungkan informasi yang cocok dari beberapa dataset untuk di eksplor dan di proses.

Data yang belum di screen dengan benar dapat menghasilkan data yang salah karena tergantung pada kualitas data

Dalam dunia kerja tugas ini di akan di jalankan oleh Data Engineer. Dimana data di konversi ke dalam bentuk yang bisa digunakan .





# Proses dalam Data Processing

## 3. Input

Input adalah data yang telah preparasi dan akan di coding dan di ubah ke dalam bentuk yang dapat di baca oleh mesin.

Data entry dapat dilakukan dengan berbagai macam metode seperti keyboard, digitizer, scanner, or data entry sumber lain.

Proses ini memakan waktu banyak dan butuh ketepatan dalam menginput. Data harus mengikuti format yang formal karena akan membutuhkan kekuatan pemrosesan dalam memcah data.

Dalam dunia kerja tugas ini di akan di jalankan oleh Data Engineer. Dimana data itu disiapkan ke dalam bentuk yang bisa dipakai untuk coding.



# Proses dalam Data Processing

## 4. Processing

Pemrosesan adalah ketika data di manipulasi menggunakan metode. Dimana program akan dijalankan yang menjalankan instruksi code-code.

Proses dapat terdiri dari beberapa eksekusi yang secara bersamaan menjalankan instruksi, tergantung pada sistem operasi.

Suatu proses adalah eksekusi aktual dari instruksi tersebut. Banyak program perangkat lunak tersedia untuk memproses data dalam volume besar dalam waktu yang sangat singkat.

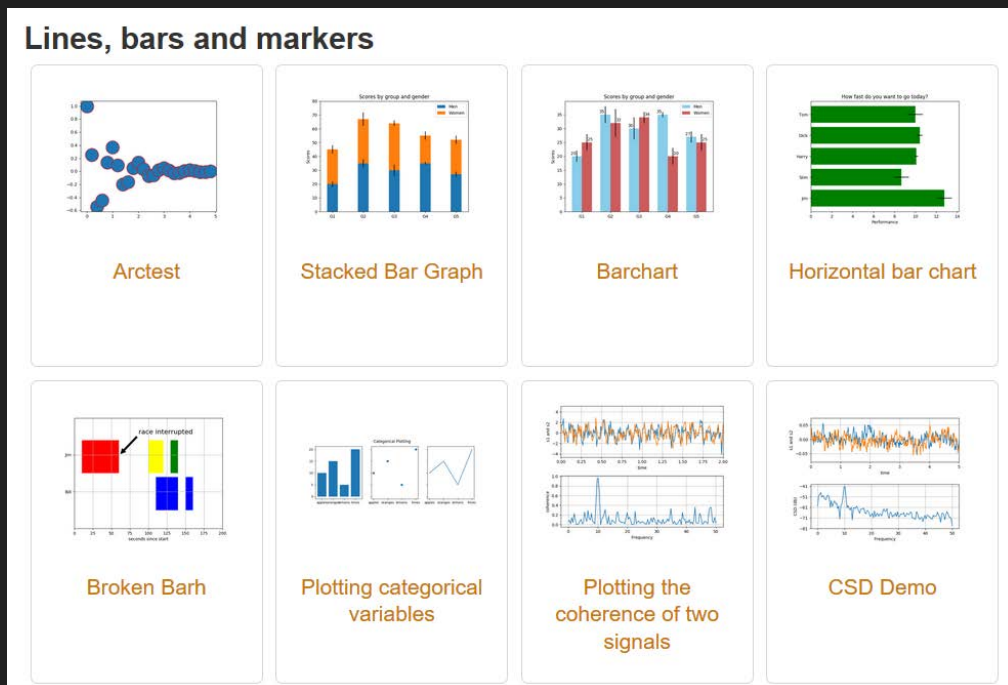
Dalam dunia kerja tugas ini di akan di jalankan oleh Data Scientist. Dimana data diproses menggunakan program yang berisikan logika codingan.



# Proses dalam Data Processing

## 5. Output and interpretation

Di tahap ini data yang telah diolah menjadi informasi akan di sampaikan kepada pengguna. Output ini di sajikan kepada pengguna dalam berbagai format seperti audio maupun visual.



Output ini harus menjelaskan secara detail sehingga dapat memberikan informasi yang berguna bagi perusahaan nantinya.

Dalam dunia kerja tugas ini di akan di jalankan oleh Data Scientist. Dimana data yang telah di proses di buat visualisasinya menggunakan software komputer.

# Proses dalam Data Processing

## 6. Storage

Penyimpanan adalah tahap terakhir dalam siklus pemrosesan data, di mana data, instruksi, dan informasi disimpan untuk penggunaan di masa mendatang.

Pentingnya siklus ini adalah ia memungkinkan akses cepat dan pengambilan informasi yang diproses, memungkinkan untuk diteruskan ke tahap berikutnya secara langsung, bila diperlukan.

Setiap komputer menggunakan penyimpanan untuk menyimpan perangkat lunak sistem dan aplikasi.



Dalam dunia kerja tugas ini di akan di jalankan oleh Artificial Intelligence. Dimana data-data yang disimpan akan membutuhkan storage dan data akan terus diolah untuk membentuk model baru dalam pembelajaran AI.

# Proses dalam Data Processing

## Kesimpulan

Data Processing adalah serangkaian langkah yang dilakukan untuk mendapatkan informasi dari Raw Data.

Meskipun setiap langkah harus dilakukan secara berurutan, urutannya adalah cyclic.

Tahap output dan penyimpanan dapat menyebabkan pengulangan tahap pengumpulan data, menghasilkan siklus pemrosesan data yang lain.

Siklus ini memberikan pandangan tentang bagaimana data bergerak dan berubah dari pengumpulan ke interpretasi, dan akhirnya, digunakan dalam keputusan bisnis yang efektif.

# Praktik

## Nomor 4 :

### • PERCOBAAN SATU

Di permasalahan ini, metode yang saya gunakan adalah CART atau Classification and Regression Trees.

CART adalah metode partisi rekursif, dengan membuat classification tree dan regression untuk memprediksi variable dependent (regression) dan prediksi kategori (classification).

Langkah pertama adalah membuat Root Node dari tree yang menerima input seluruh row dari training set

Training Set

Lalu di setiap node akan menanyakan true atau false dari sebuah fitur, dan respon dari input tersebut akan memisahkan menjadi 2 data.

Training Set



\*pertanyaan\*

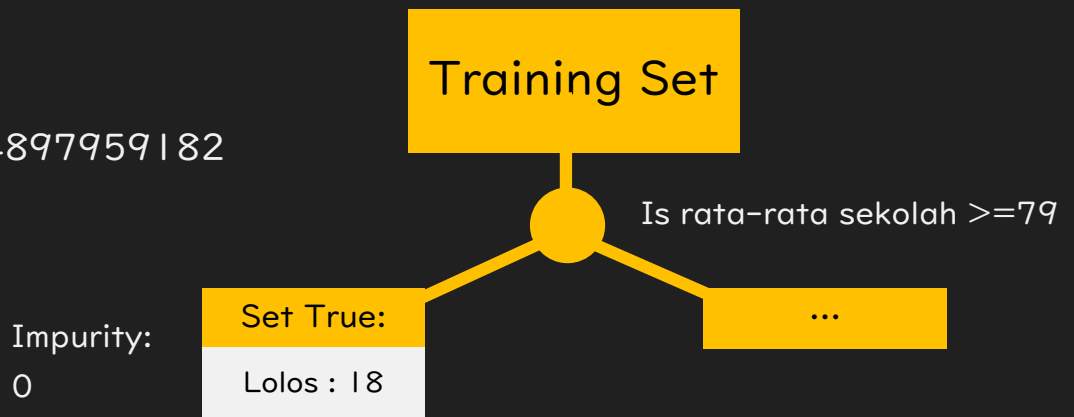


Tujuan dari pertanyaan ini adalah untuk memisahkan label dan mendapatkan kemungkinan yang paling pasti. Sebuah pertanyaan yang bagus akan mengurangi ketidakpastian dari set tersebut.

Cara kita untuk menentukan pertanyaan yang sesuai, waktu digunakan dan banyak nya pertanyaan adalah menggunakan Gini Impurity, semakin kecil impurity maka data semakin terpisah.

Impurity:

0.4362244897959182

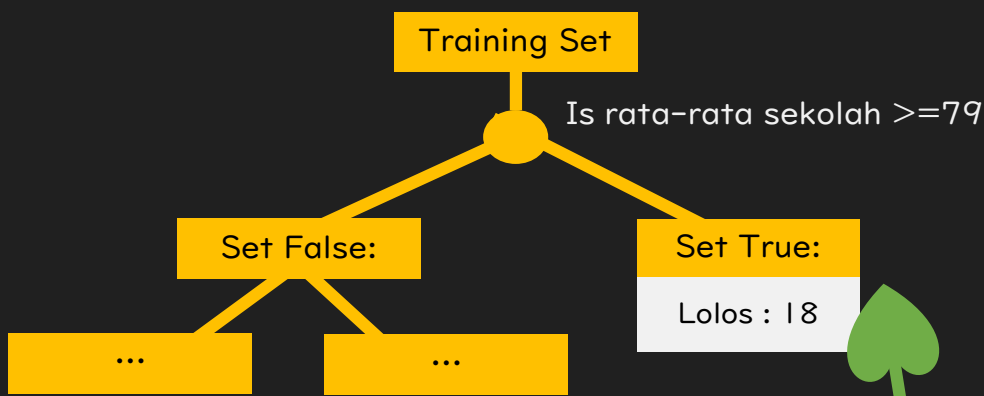


Impurity:

0

Setelah mengetahui impurity kita bisa mendapatkan Information Gain untuk memilih pertanyaan yang paling tepat. Information gain akan menentukan kualitas dari sebuah pertanyaan.

Cara mendapatkan Information gain ini adalah dengan mengurangi impurity awal dengan rata-rata impurity partisinya.



Proses akan terus melakukan pengulangan untuk melihat satu persatu sampai tidak ada ada yang bisa pisah lagi dan mendapatkan leaf.

Dan tree ini akan digunakan untuk memprediksi di data selanjutnya





Untuk mengolah data ini saya menggunakan Python

- Langkah pertama adalah mengimport library yang di perlukan

```
In [1]: # mengimport Library yang di butuhkan
from __future__ import print_function
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- Setelah itu kita mengimport data training yang kita miliki

```
In [2]: # mengimport dataset train
dataset = pd.read_csv('Train1.csv', sep = ';')
dataset.head(5)
```

Out[2]:

	NISN	Nama	Umur	Kelas	Rerata UN	Ratarata Nilai Sekolah	Hasil
0	27786	Akbar	17	IPA 1	87	83	Lolos
1	38539	Opang	17	IPA 2	83	91	Lolos
2	47031	Erwan	18	IPA 2	91	84	Lolos
3	58624	Kaka	17	IPS 3	90	100	Lolos
4	68091	Cimo	17	IPS 4	82	75	Coba lagi

- Kita akan menghapus variable yang tidak diperlukan untuk menjadi bahan pertimbangan

```
In [3]: # menghapus dan memilih data yang menjadi bahan pertimbangan data train
dataset = dataset.drop(dataset.columns[0:4],axis = 1)
dataset.head(5)
```

Out[3]:

	Rerata UN	Ratarata Nilai Sekolah	Hasil
0	87	83	Lolos
1	83	91	Lolos
2	91	84	Lolos
3	90	100	Lolos
4	82	75	Coba lagi



- Dari bentuk dataframe kita rubah ke dalam bentuk list untuk mempermudah pengolahan

```
In [4]: # merubah dataset ke dalam bentuk list
training_data = dataset.values.tolist()
training_data
```

```
Out[4]: [[87, 83, 'Lolos'],
[83, 91, 'Lolos'],
[91, 84, 'Lolos'],
[90, 100, 'Lolos'],
[82, 75, 'Coba lagi'],
[80, 78, 'Coba lagi'],
[78, 77, 'Coba lagi'],
[81, 72, 'Coba lagi'],
[82, 72, 'Coba lagi'],
[80, 84, 'Lolos'],
[79, 91, 'Lolos'],
[91, 79, 'Lolos'],
[83, 75, 'Coba lagi'],
[80, 75, 'Coba lagi'],
```

- Kita memberi nama label dari setiap kolom

```
In [5]: # memberi nama label dari kolom
header = ["Rerata UN", "Rata-rata Nilai Sekolah", "Hasil"]
```

- Kita akan mendeklare function yang mencari data unik (distinct)

```
In [6]: # mencari value unik
def unique_vals(rows, col):
    return set([row[col] for row in rows])
```

```
In [7]: unique_vals(training_data, 1)
```

```
Out[7]: {71, 72, 75, 77, 78, 79, 81, 83, 84, 87, 88, 91, 95, 96, 97, 100}
```

- Kita juga akan membuat fitur menghitung jumlah items

```
In [8]: # menghitung jumlah fitur dari sebuah dataset
def class_counts(rows):
    counts = {} # deklarasi label ke count
    for row in rows:
        # label adalah kolom terakhir
        label = row[-1]
        if label not in counts:
            counts[label] = 0
        counts[label] += 1
    return counts
```

```
In [9]: class_counts(training_data)
```

```
Out[9]: {'Lolos': 19, 'Coba lagi': 9}
```



- Lalu kita membuat function untuk menganalisa suatu item numerik atau tidak

```
In [10]: # tes apakah numerik atau bukan
def is_numeric(value):
    return isinstance(value, int) or isinstance(value, float)

In [11]: is_numeric(4)

Out[11]: True
```

- Setelah itu membuat function yang memunculkan pertanyaan dengan input row data train dan value

```
In [12]: # function ini membuat sebuah pertanyaan digunakan untuk mempartisi dataset
class Question:

    def __init__(self, column, value): #digunakan untuk merecord column number dan value yang di tentu
        self.column = column
        self.value = value

    def match(self, example):
        # menentukan ketika value adalah numerik dan ketika fitur tidak berupa numerik
        val = example[self.column]
        if is_numeric(val):
            return val >= self.value
        else:
            return val == self.value

    def __repr__(self):
        # This is just a helper method to print
        # the question in a readable format.
        condition = "=="
        if is_numeric(self.value):
            condition = ">="
        return "Is %s %s %s?" % (
            header[self.column], condition, str(self.value))

In [13]: #tes pertanyaan numerik
Question(1, 79)

Out[13]: Is Rata-rata Nilai Sekolah >= 79?

In [14]: q = Question(1, 75)
q

Out[14]: Is Rata-rata Nilai Sekolah >= 75?

In [15]: # contoh
example = training_data[0]
q.match(example)
# benar selama rata2 sekolah di atas 75

Out[15]: True
```

- Kita juga butuh function yang mempartisi data ke dalam set true dan false

```
In [16]: def partition(rows, question):
#function partition menerima 2 input yaitu row list dan pertanyaan
#function akan menganalisa benar atau tidak dan akan di klasifikasikan ke masing-masing row
true_rows, false_rows = [], []
for row in rows:
    if question.match(row):
        true_rows.append(row)
    else:
        false_rows.append(row)
return true_rows, false_rows

In [17]: true_rows, false_rows = partition(training_data, Question(2, 'Lolos'))
true_rows
```

```
Out[17]: [[87, 83, 'Lolos'],
[83, 91, 'Lolos'],
[91, 84, 'Lolos'],
[90, 100, 'Lolos'],
[80, 84, 'Lolos'],
[79, 91, 'Lolos'],
[91, 79, 'Lolos'],
[81, 77, 'Lolos'],
[90, 88, 'Lolos'],
[91, 91, 'Lolos'],
[92, 87, 'Lolos'],
[79, 96, 'Lolos'],
[93, 84, 'Lolos'],
[95, 84, 'Lolos'],
[82, 97, 'Lolos'],
[84, 84, 'Lolos'],
[79, 96, 'Lolos'],
[90, 81, 'Lolos'],
[92, 95, 'Lolos']]
```

```
In [18]: false_rows

Out[18]: [[82, 75, 'Coba lagi'],
[80, 78, 'Coba lagi'],
[78, 77, 'Coba lagi'],
[81, 72, 'Coba lagi'],
[82, 72, 'Coba lagi'],
[83, 75, 'Coba lagi'],
[80, 75, 'Coba lagi'],
[93, 71, 'Coba lagi'],
[80, 78, 'Coba lagi']]
```

- Setelah itu kita akan membuat function yang menghitung impurity atau gini Impurity

```
In [19]: def gini(rows): #function ini digunakan untuk menghitung impurity

counts = class_counts(rows)
impurity = 1
for lbl in counts:
    prob_of_lbl = counts[lbl] / float(len(rows))
    impurity -= prob_of_lbl**2
return impurity
```

```
In [20]: # Contoh menentukan rasio dari mixing
some_mixing = [[1],
[2]]
gini(some_mixing)
#hasil menjadi 0.5 dikarenakan adanya 50% salah penentuan
```

```
Out[20]: 0.5
```



- Setelah mendapatkan Impurity kita juga bisa mendapatkan Infogain dengan mengurangi impurity atasan dan branch nya

```
In [21]: # function ini menghasilkan information gain dari mengurangi impurity induk dan nodes nya
def info_gain(left, right, current_uncertainty):

    p = float(len(left)) / (len(left) + len(right))
    return current_uncertainty - p * gini(left) - (1 - p) * gini(right)

In [22]: # mengecek uncertainty dari data training kita
current_uncertainty = gini(training_data)
current_uncertainty

Out[22]: 0.4362244897959182
```

- Di function ini kita akan menggabungkan function yang kita sudah buat sebelumnya untuk mencari pertanyaan yang paling tepat dengan mengulangi pertanyaan dan membandingkan jumlah Infogainnya

```
In [25]: # function ini mencari pertanyaan yang paling sesuai untuk di gunakan dengan satu persatu menghitung i
def find_best_split(rows):
    best_gain = 0 # keep track of the best information gain
    best_question = None # keep train of the feature / value that produced it
    current_uncertainty = gini(rows)
    n_features = len(rows[0]) - 1 # banyak kolom

    # disini kita akan melihat setiap value dari fitur tersebut kita akan menghasilkan pertanyaan serta
    # kita akan membuang pertanyaan yang gagal menghasilkan pecahan
    for col in range(n_features): # for each feature

        values = set([row[col] for row in rows]) # unique values in the column

        for val in values:

            question = Question(col, val)

            # splitting dataset
            true_rows, false_rows = partition(rows, question)

            # skip ketika dataset tidak splittig
            if len(true_rows) == 0 or len(false_rows) == 0:
                continue

            # Calculate the information gain from this split
            gain = info_gain(true_rows, false_rows, current_uncertainty)

            if gain >= best_gain:
                best_gain, best_question = gain, question

    return best_gain, best_question

In [26]: # contoh pertanyaan yang paling sesuai
best_gain, best_question = find_best_split(training_data)
best_question

Out[26]: Is Rata-rata Nilai Sekolah >= 79?
```



- Kita membutuhkan leaf untuk menaruh data yang berhasil di pisahkan sebelumnya

```
In [27]: class Leaf: #Leaf memuat dictionary dari data yang berhasil di olah

    def __init__(self, rows):
        self.predictions = class_counts(rows)
```

- Decision Node akan menempatkan data yang di pisahkan ke tempat yang seharusnya

```
In [28]: # Decision Node menanyakan pertanyaan
class Decision_Node:

    def __init__(self,
                 question,
                 true_branch,
                 false_branch):
        self.question = question
        self.true_branch = true_branch
        self.false_branch = false_branch
```

- Function build tree akan membuat tree dari semua yang di olah sebelumnya membentuk tree dengan nodenya

```
In [29]: def build_tree(rows):

    # disini kita akan mempartisi data dan memakai pertanyaan yang menghasilkan information dengan gain
    gain, question = find_best_split(rows)

    #ketika tidak ada lagi information gain, kita akan return Leaf
    if gain == 0:
        return Leaf(rows)

    # If we reach here, we have found a useful feature / value
    # to partition on.
    true_rows, false_rows = partition(rows, question)

    # memanggil tree kembali untuk menambahkan node
    true_branch = build_tree(true_rows)

    # false juga akan menjadi Leaf ketika tidak ada lagi gain yang bisa di dapatkan
    false_branch = build_tree(false_rows)

    # ketika selesai maka akan terbentuk decision node
    return Decision_Node(question, true_branch, false_branch)
```

- Setelah membuat treenya, sekarang kita akan melihat tree nya dengan function print tree

```
In [30]: def print_tree(node, spacing=""):

    # Base case: we've reached a leaf
    if isinstance(node, Leaf):
        print (spacing + "Predict", node.predictions)
        return

    # Print the question at this node
    print (spacing + str(node.question))

    # Call this function recursively on the true branch
    print (spacing + '--> True:')
    print_tree(node.true_branch, spacing + " ")

    # Call this function recursively on the false branch
    print (spacing + '--> False:')
    print_tree(node.false_branch, spacing + " ")
```

```
In [31]: #ketika di panggil pertama kali tree ini akan menerima semua data training sebagai input
my_tree = build_tree(training_data)
```

```
In [32]: print_tree(my_tree)
```

```
Is Rata-rata Nilai Sekolah >= 79?
--> True:
    Predict {'Lolos': 18}
--> False:
    Is Rata-rata Nilai Sekolah >= 77?
    --> True:
        Is Rerata UN >= 81?
        --> True:
            Predict {'Lolos': 1}
        --> False:
            Predict {'Coba lagi': 3}
    --> False:
        Predict {'Coba lagi': 6}
```

- Classify digunakan untuk dataset yang akan di prediksi untuk menentukan ke arah false atau true

```
In [33]: def classify(row, node):

    # mencapai Leaf
    if isinstance(node, Leaf):
        return node.predictions

    #menentukan untuk ke arah true atau false branch
    if node.question.match(row):
        return classify(row, node.true_branch)
    else:
        return classify(row, node.false_branch)
```

```
In [34]: #contoh prediksi dari confidence tertinggi
classify(training_data[0], my_tree)
```

```
Out[34]: {'Lolos': 18}
```



- Setelah di classify kita akan mengeluarkan prediksi kita atau leaf yang sudah di dapatkan dari classify tersebut

```
In [36]: print_leaf(classify(training_data[0], my_tree))
Out[36]: {'Lolos': '100%'}
```

- Semuanya sudah siap, kita akan mengimport data yang akan kita prediksi dan akan mengklasifikasi lolos atau tidak

```
In [37]: newset = pd.read_csv('Test1.csv', sep = ';')
newset = newset.drop(newset.columns[0:4], axis = 1)
newset.head(5)
```

```
Out[37]:
```

	Rerata UN	Rerata Nilai Sekolah	Hasil
0	80	94	?
1	70	97	?
2	84	77	?
3	90	89	?
4	92	92	?

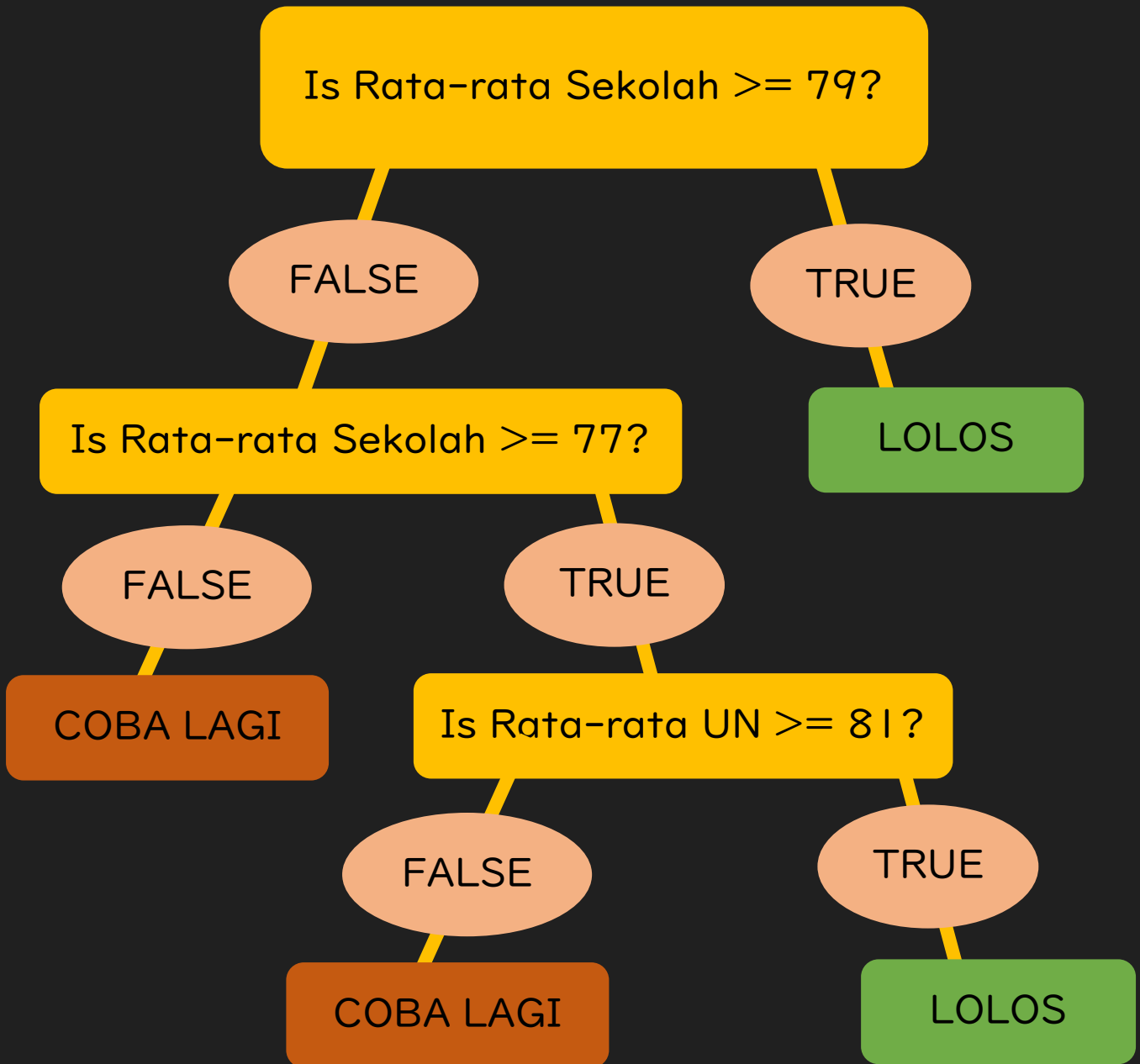
```
In [38]: testing_data = newset.values.tolist()
for row in testing_data:
    print ("Actual: %s. Predicted: %s" %
          (row[1], print_leaf(classify(row, my_tree))))
```

```
Actual: 94. Predicted: {'Lolos': '100%'}
Actual: 97. Predicted: {'Lolos': '100%'}
Actual: 77. Predicted: {'Lolos': '100%'}
Actual: 89. Predicted: {'Lolos': '100%'}
Actual: 92. Predicted: {'Lolos': '100%'}
Actual: 81. Predicted: {'Lolos': '100%'}
Actual: 92. Predicted: {'Lolos': '100%'}
Actual: 78. Predicted: {'Coba lagi': '100%'}
Actual: 78. Predicted: {'Lolos': '100%'}
Actual: 95. Predicted: {'Lolos': '100%'}
Actual: 83. Predicted: {'Lolos': '100%'}
```

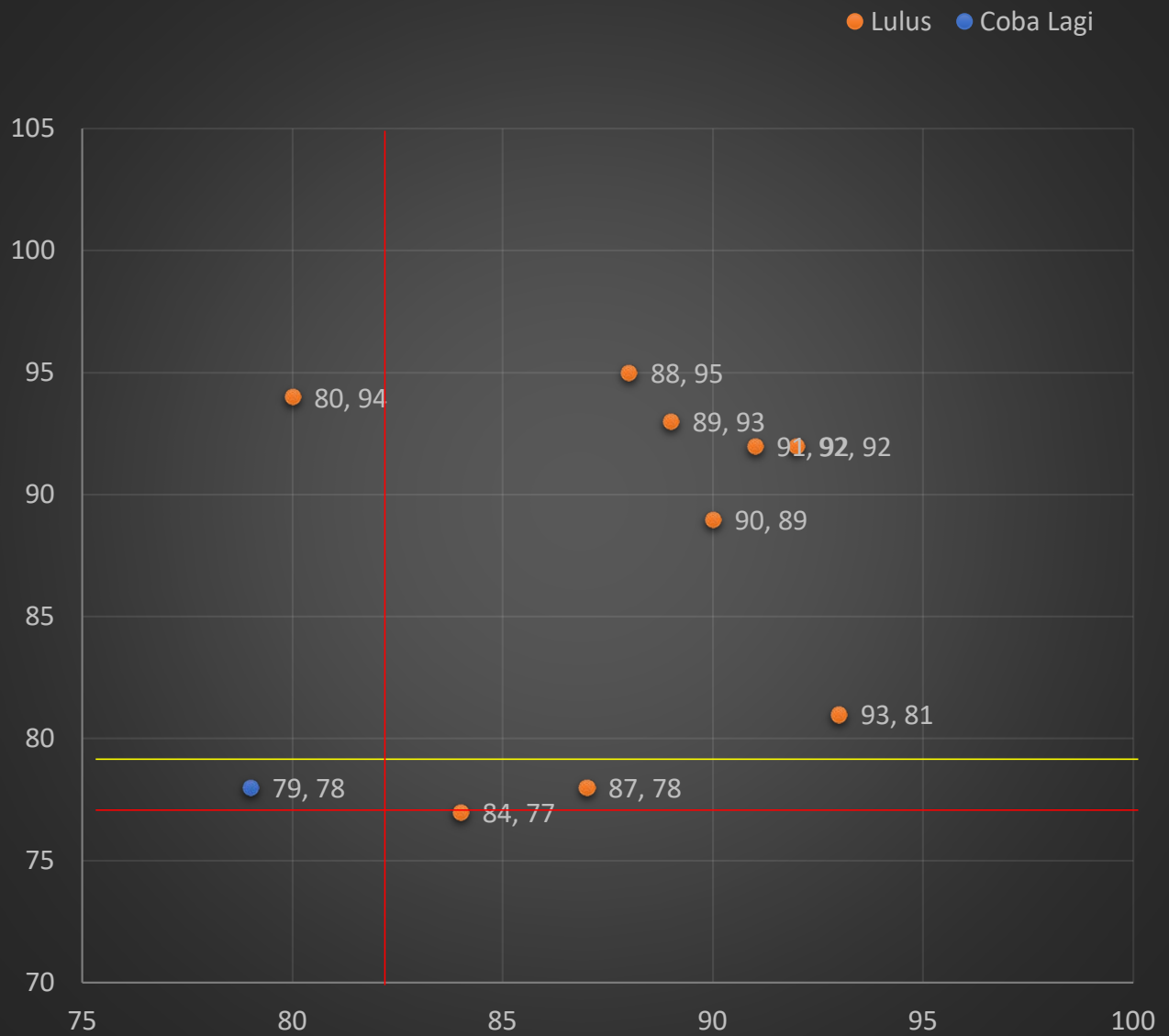




# Decision Tree



# DATA TEST



Dengan menggunakan Decision Tree of classification di atas, set test akhirnya dapat diprediksi.

NISN	NAMA	UMUR	KELAS	RERATA UN	RERATA NILAI SEKOLAH	HASIL
12345	Risa	16	IPA 1	80	94	LOLOS
56789	Putri	18	IPS 2	70	97	LOLOS
12452	Aris	17	IPA 3	84	77	LOLOS
24566	Ida	17	IPA 2	90	89	LOLOS
78657	Tuti	17	IPS 4	92	92	LOLOS
28428	Nugraha	16	IPA 1	93	81	LOLOS
93020	Adi	18	IPA 2	91	92	LOLOS
24748	Sora	16	IPA 6	79	78	COBA LAGI
85326	Eka	18	IPA 2	87	78	LOLOS
37944	Ahmad	16	IPA 1	88	95	LOLOS
27424	Andre	18	IPS 2	89	93	LOLOS

## Nomor 4 :

### • PERCOBAAN DUA

Dalam percobaan kedua kali ini saya akan menggunakan metode yang lebih advanced. Di permasalahan ini saya akan menggunakan Binary Classifiers. Yang akan saya gunakan antara lain adalah Logistic Regression, Random Forest, dan LinearSVC. Dan saya akan membandingkan dan menggunakan prediksi yang paling akurat.

Yang pertama akan saya coba adalah Logistic Regression, Logistic Regression adalah satu analisi multivariate, yang berguna untuk memprediksi dependent variabel berdasarkan variabel independent

Rumus dari Linear regression Equation ini adalah

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

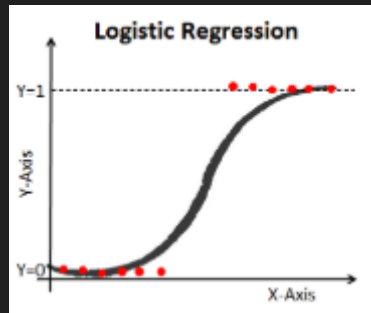
Dimana y adalah dependent variable dan x sebagai explanatory variable.

Sigmoid function dari Linear Regression adalah

$$p = 1 / (1 + e^{-y})$$



Contoh dari dari grafik Logistic regression antara lain seperti ini



Yang memberikan bentuk S dari grafik ini adalah sigmoid function yang rumus yang adalah

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

- Langkah pertama yang perlu kita lakukan adalah mengimport library yang di butuhkan

```
In [1]: #import Library yang di butuhkan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
```

- Lalu kita membuat Classifier untuk digunakan nanti

```
In [2]: #create classifier
lr = LogisticRegression(solver='lbfgs')
svc = LinearSVC(C = 1.0)
rfc = RandomForestClassifier(n_estimators = 100)
```

- Lalu kita import dataset train

```
In [3]: #mengimport dataset train
df = pd.read_csv('Train1.csv', sep = ';')
df.head(5)
```

```
Out[3]:
```

	NISN	Nama	Umur	Kelas	Rerata UN	Ratarata Nilai Sekolah	Hasil
0	27786	Akbar	17	IPA 1	87	83	Lolos
1	38539	Opang	17	IPA 2	83	91	Lolos
2	47031	Erwan	18	IPA 2	91	84	Lolos
3	58624	Kaka	17	IPS 3	90	100	Lolos
4	68091	Cimo	17	IPS 4	82	75	Coba lagi

- Lalu disini saya akan mengambil variable2 yang layak untuk menjadi bahan pertimbangan

```
In [4]: # menghapus kolom yang tidak dibutuhkan
df = df.drop(["NISN", "Kelas", "Nama "], axis=1)
df.head(5)
```

```
Out[4]:
```

	Umur	Rerata UN	Ratarata Nilai Sekolah	Hasil
0	17	87	83	Lolos
1	17	83	91	Lolos
2	18	91	84	Lolos
3	17	90	100	Lolos
4	17	82	75	Coba lagi

- Untuk mengetes keakuratan dari prediksi saya akan menggunakan data train yang di pisah

```
In [6]: # memisahkan data menjadi train dan test untuk menguji prediksi menjadi 20%
# disini data di acak menjadi random
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size = 0.2)
```

```
In [7]: #memisahkan dataset menjadi features dan target
train_feat = train.iloc[:, :3]
train_targ = train["Hasil"]
```

```
In [8]: test_feat = test.iloc[:, :3]
test_targ = test["Hasil"]
```

- Lalu kita regresikan data train yang sudah di pisah menjadi train untuk mendapatkan rumus

```
In [9]: #fit LR Models
lr.fit(train_feat, train_targ)

Out[9]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, max_iter=100, multi_class='warn',
    n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
    tol=0.0001, verbose=0, warm_start=False)
```

- Setelah di fit kita akan mengecek seberapa akurat prediksi yang di hasilkan data train dan juga data test

```
In [10]: # seberapa akurat dari data train
lr.score(train_feat, train_targ)
```

```
Out[10]: 0.9545454545454546
```

```
In [11]: # seberapa akurat dari data test
lr.score(test_feat, test_targ)
```

```
Out[11]: 1.0
```

- Kita gunakan coef untuk melihat variable mana yang paling berpengaruh terhadap hasil

```
In [12]: # melihat koefisien variable yang paling berpengaruh
lr.coef_
```

```
Out[12]: array([[ -0.30600825,  0.3044778 ,  0.77178214]])
```



- Setelah itu kita akan melihat dari hasil prediksi

```
In [13]: # hasil dari prediksi data train
lr.predict(train_feat)

Out[13]: array(['Coba lagi', 'Coba lagi', 'Coba lagi', 'Coba lagi', 'Lolos',
                'Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Coba lagi', 'Lolos',
                'Lolos', 'Coba lagi', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos',
                'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos'], dtype=object)

In [14]: # dari seluruh prediksi terdapat 1 error di data train
from sklearn.metrics import confusion_matrix
confusion_matrix(lr.predict(train_feat), train_targ)

Out[14]: array([[ 8,  1],
                [ 0, 13]], dtype=int64)

In [15]: # dari seluruh prediksi tidak terdapat error di data train
confusion_matrix(lr.predict(test_feat), test_targ)

Out[15]: array([[1,  0],
                [0,  5]], dtype=int64)
```

- Ternyata prediksi cukup tepat dengan keberhasilan 95%, terdapat 1 error dari 21 percobaan
- Setelah ini kita akan melihat prediksi menggunakan seluruh data training terhadap data test yang di sediakan





- Pertama kita akan mengimport data test yang akan diprediksi

```
In [17]: #mengimport data yang akan di prediksi
test1 = pd.read_csv('Test1.csv', sep = ';')
test1 = test1.drop(["NISN", "Kelas", "Nama"], axis=1)
test1
```

```
Out[17]:
```

	Umur	Rerata UN	Rerata Nilai Sekolah	Hasil
0	16	80	94	?
1	18	70	97	?
2	17	84	77	?
3	17	90	89	?
4	17	92	92	?
5	16	93	81	?
6	18	91	92	?
7	16	79	78	?
8	18	87	78	?
9	16	88	95	?
10	18	89	83	?

```
In [18]: #memisahkan dataset menjadi features dan target
test1_feat = test1.iloc[:, :3]
```

```
In [19]: df_feat = df.iloc[:, :3]
df_targ = df["Hasil"]
```

- Lalu akan kita fit seluruh data training dengan logistic regression

```
In [20]: # #fit LR Models
lr.fit(df_feat, df_targ)
```

```
Out[20]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, max_iter=100, multi_class='warn',
    n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
    tol=0.0001, verbose=0, warm_start=False)
```

```
In [21]: # seberapa akurat prediksi dari data
lr.score(df_feat, df_targ)
```

```
Out[21]: 0.9642857142857143
```

```
In [22]: # melihat koefisien variable yang paling berpengaruh
lr.coef_
```

```
Out[22]: array([[ -0.30387279,  0.30747268,  0.7800669 ]])
```



- Hasil nya cukup akurat yaitu 96% dengan rata-rata nilai sekolah yang paling berpengaruh terhadap hasil kelulusan
- Dan berikut hasil test yang di hasilkan dari prediksi Logistic regression

```
In [23]: # hasil dari prediksi data train
         lr.predict(test1_feat)

Out[23]: array(['Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
                'Coba lagi', 'Lolos', 'Lolos', 'Lolos'], dtype=object)
```



## Hasil Prediksi Logistic Regression

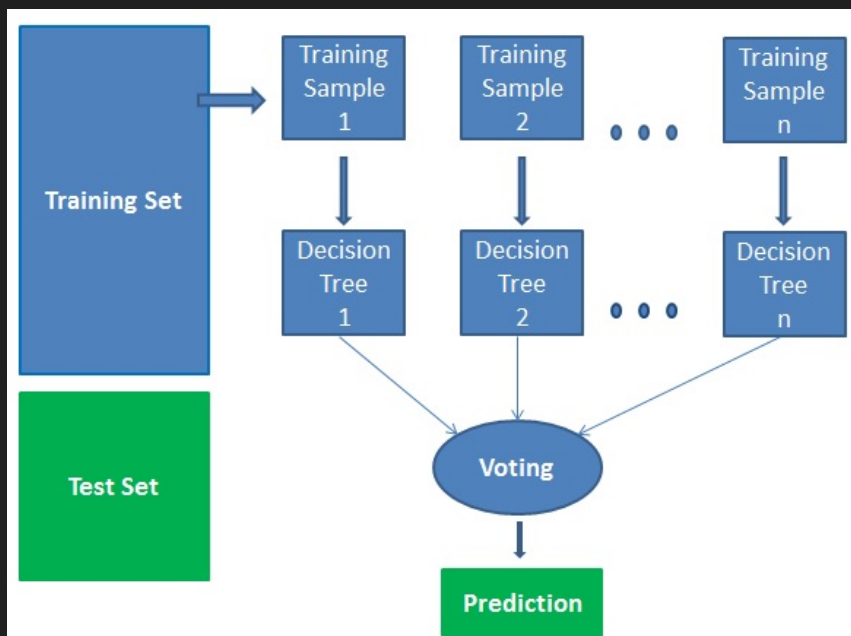
NISN	NAMA	UMUR	KELAS	RERATA UN	RERATA NILAI SEKOLAH	HASIL
12345	Risa	16	IPA 1	80	94	LOLOS
56789	Putri	18	IPS 2	70	97	LOLOS
12452	Aris	17	IPA 3	84	77	COBA LAGI
24566	Ida	17	IPA 2	90	89	LOLOS
78657	Tuti	17	IPS 4	92	92	LOLOS
28428	Nugraha	16	IPA 1	93	81	LOLOS
93020	Adi	18	IPA 2	91	92	LOLOS
24748	Sora	16	IPA 6	79	78	COBA LAGI
85326	Eka	18	IPA 2	87	78	LOLOS
37944	Ahmad	16	IPA 1	88	95	LOLOS
27424	Andre	18	IPS 2	89	93	LOLOS

## Metode yang kedua adalah Random Forest

Random Forest menciptakan decision tree pada sampel data yang dipilih secara acak, mendapatkan prediksi dari setiap tree dan memilih solusi terbaik dengan cara memilih

### Cara kerja Random Forest

- Pilih sampel acak dari dataset yang diberikan.
- Bangun decision tree untuk setiap sampel dan dapatkan hasil prediksi dari setiap pohon keputusan.
- Lakukan vote untuk setiap hasil yang diprediksi.
- Pilih hasil prediksi dengan suara terbanyak sebagai prediksi akhir.



- Cara mengimplementasi kannya sama seperti sebelumnya, yaitu kita fit terlebih dahulu data yang sudah di pisahkan

```
In [25]: rfc.fit(train_feat,train_targ)
Out[25]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                                oob_score=False, random_state=None, verbose=0,
                                warm_start=False)

In [26]: rfc.score(train_feat,train_targ)
Out[26]: 1.0

In [27]: rfc.score(test_feat,test_targ)
Out[27]: 1.0

In [28]: rfc.predict(train_feat)
Out[28]: array(['Coba lagi', 'Coba lagi', 'Coba lagi', 'Coba lagi', 'Lolos',
                'Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Coba lagi', 'Lolos',
                'Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos',
                'Coba lagi', 'Lolos', 'Lolos'], dtype=object)

In [29]: confusion_matrix(rfc.predict(train_feat),train_targ)
Out[29]: array([[ 8,  0],
                [ 0, 14]], dtype=int64)

In [30]: confusion_matrix(rfc.predict(test_feat),test_targ)
Out[30]: array([[1, 0],
                [0, 5]], dtype=int64)

## Mengimplementasikan prediksi menggunakan data uji
```

- Hasil yang sungguh mengejutkan, keakuratan prediksi 100% di data yang sudah di pisahkan, sehingga ini akan menjadi metode utama saya nantinya



- Lalu kita gunakan seluruh data untuk memprediksi data test

```
In [32]: rfc.fit(df_feat,df_targ)
Out[32]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                                oob_score=False, random_state=None, verbose=0,
                                warm_start=False)

In [33]: rfc.score(df_feat,df_targ)
Out[33]: 1.0

In [34]: rfc.predict(test1_feat)
Out[34]: array(['Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
                'Coba lagi', 'Lolos', 'Lolos', 'Lolos'], dtype=object)
```

- Dengan metode Random forest di data ini akan menghasilkan 100% prediksi akurat dan jawabannya sama seperti Logistic Regression yang mempunyai keakuratan 96%

```
In [34]: rfc.predict(test1_feat)
Out[34]: array(['Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
                'Coba lagi', 'Lolos', 'Lolos', 'Lolos'], dtype=object)
```



## Hasil Prediksi Random Forest

NISN	NAMA	UMUR	KELAS	RERATA UN	RERATA NILAI SEKOLAH	HASIL
12345	Risa	16	IPA 1	80	94	LOLOS
56789	Putri	18	IPS 2	70	97	LOLOS
12452	Aris	17	IPA 3	84	77	COBA LAGI
24566	Ida	17	IPA 2	90	89	LOLOS
78657	Tuti	17	IPS 4	92	92	LOLOS
28428	Nugraha	16	IPA 1	93	81	LOLOS
93020	Adi	18	IPA 2	91	92	LOLOS
24748	Sora	16	IPA 6	79	78	COBA LAGI
85326	Eka	18	IPA 2	87	78	LOLOS
37944	Ahmad	16	IPA 1	88	95	LOLOS
27424	Andre	18	IPS 2	89	93	LOLOS

Yang terakhir adalah metode SFM

SFM adalah teknik pencitraan rentang fotogrametrik untuk memperkirakan struktur tiga dimensi dari urutan gambar dua dimensi yang dapat digabungkan dengan sinyal gerak lokal.

- Lagi2 sama seperti sebelumnya kita akan memprediksi dari data yang dipisah terlebih dahulu

```
In [36]: import warnings
with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    svc.fit(train_feat, train_targ)

In [37]: svc.score(train_feat, train_targ)
Out[37]: 0.7727272727272727

In [38]: svc.score(test_feat, test_targ)
Out[38]: 0.8333333333333334

In [39]: svc.predict(train_feat)
Out[39]: array(['Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
'Lolos', 'Lolos', 'Coba lagi', 'Lolos', 'Lolos', 'Lolos',
'Coba lagi', 'Lolos', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
'Lolos', 'Lolos'], dtype=object)
```

- Keakuratan yang didapatkan cukup buruk yaitu 77% di train dan 83% di test, tapi kita akan melihat prediksi dari metode ini





- Kali ini hasil prediksi berbeda dari sebelumnya

```
In [44]: with warnings.catch_warnings():
          warnings.simplefilter('ignore')
          svc.fit(df_feat, df_targ)

In [45]: svc.score(df_feat, df_targ)
Out[45]: 0.7857142857142857

In [46]: svc.predict(test1_feat)
Out[46]: array(['Lolos', 'Lolos', 'Lolos', 'Lolos', 'Lolos', 'Lolos', 'Lolos',
                'Lolos', 'Coba lagi', 'Lolos', 'Lolos'], dtype=object)
```

- Hasilnya sama seperti menggunakan metode CART

NISN	NAMA	UMUR	KELAS	RERATA UN	RERATA NILAI SEKOLAH	HASIL
12345	Risa	16	IPA 1	80	94	LOLOS
56789	Putri	18	IPS 2	70	97	LOLOS
12452	Aris	17	IPA 3	84	77	LOLOS
24566	Ida	17	IPA 2	90	89	LOLOS
78657	Tuti	17	IPS 4	92	92	LOLOS
28428	Nugraha	16	IPA 1	93	81	LOLOS
93020	Adi	18	IPA 2	91	92	LOLOS
24748	Sora	16	IPA 6	79	78	COBA LAGI
85326	Eka	18	IPA 2	87	78	LOLOS
37944	Ahmad	16	IPA 1	88	95	LOLOS
27424	Andre	18	IPS 2	89	93	LOLOS

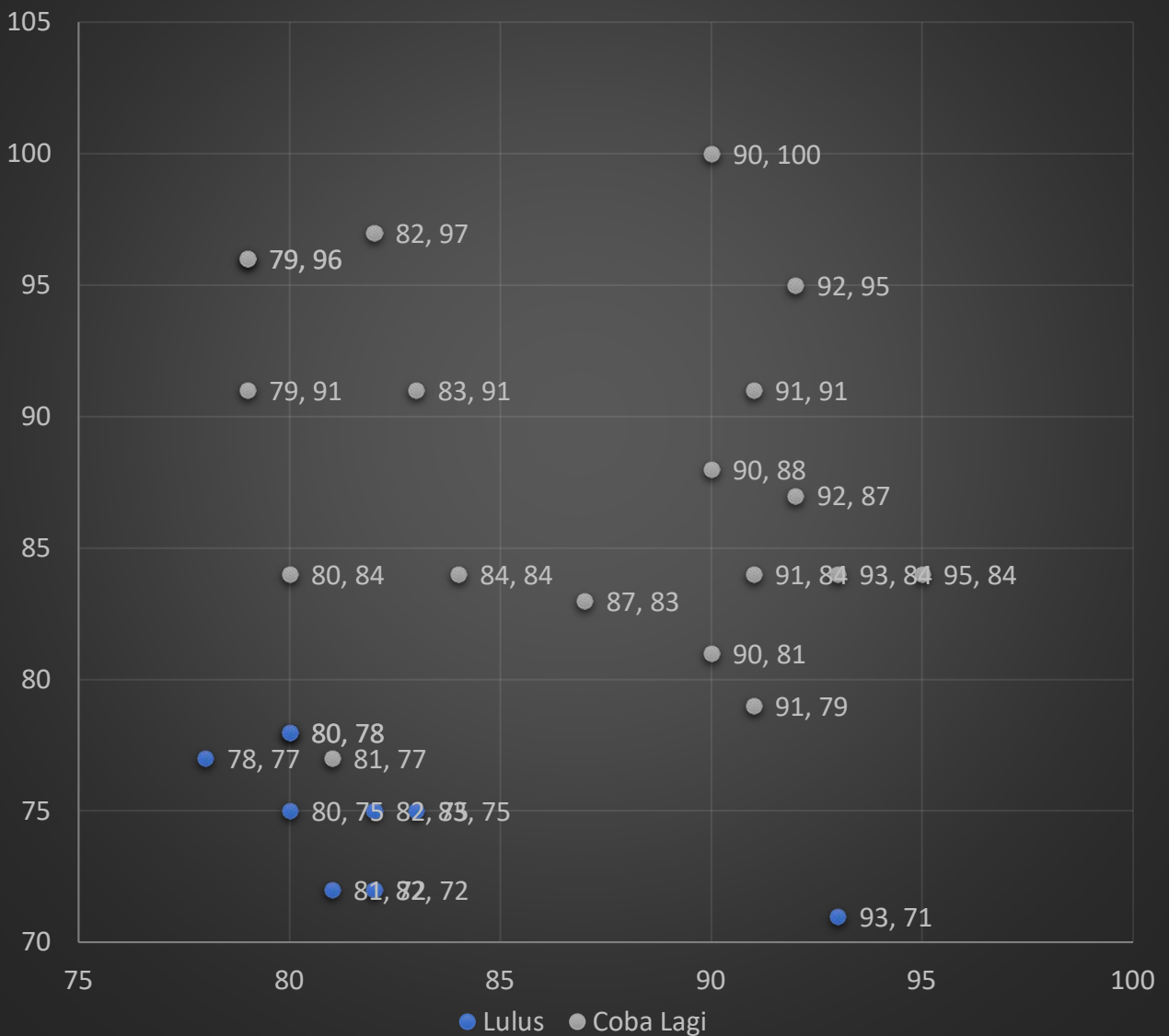
## JAWABAN AKHIR NOMOR 4

Ya, setelah memprediksi hasil dengan banyak metode di atas maka saya akan memutuskan menggunakan hasil dengan keakuratan prediksi tertinggi yaitu Random Forest dengan keakuratan 100% yang hasilnya sama dengan Logistic Regression yang keakuratannya 96%

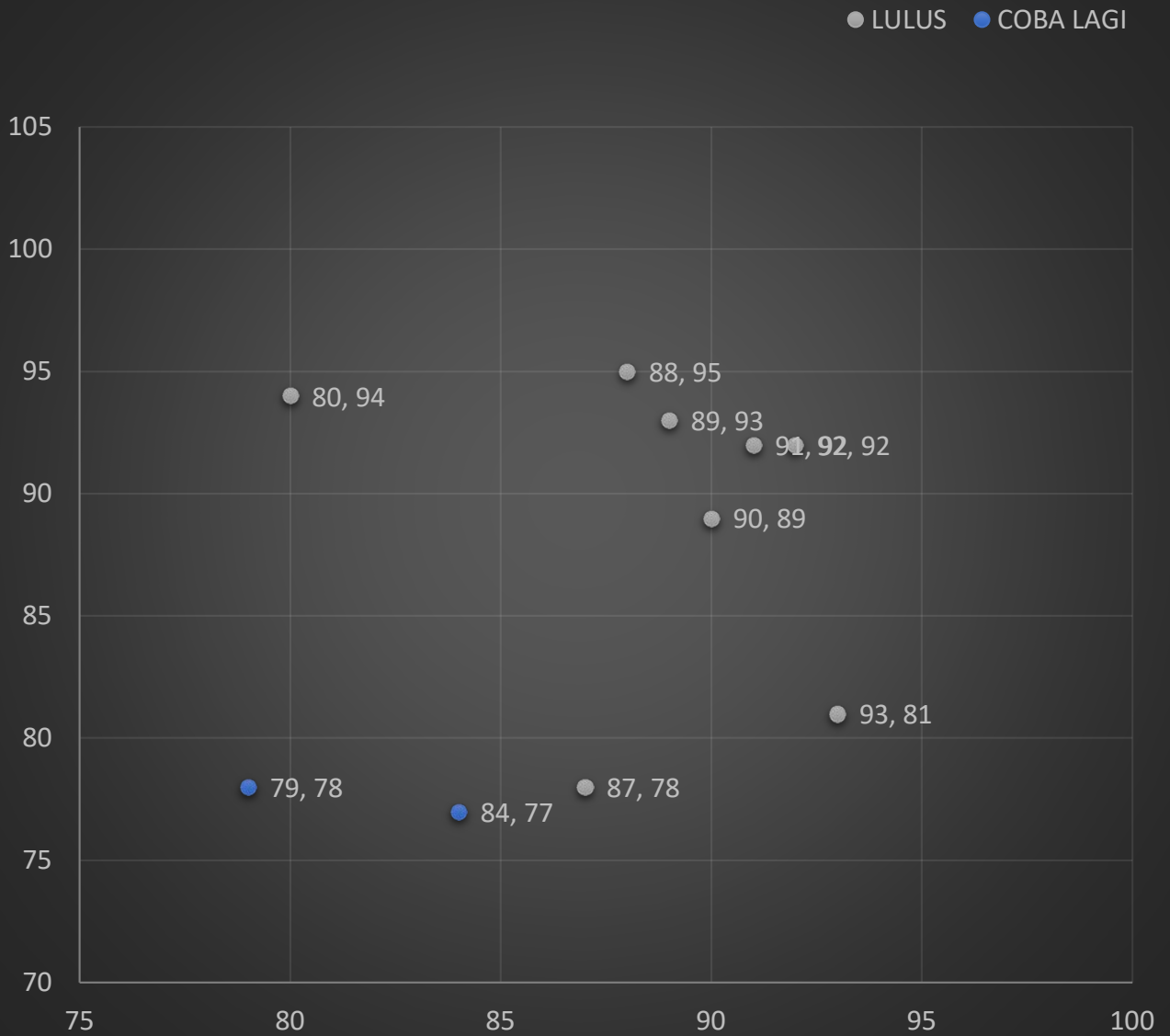
Saya tidak menggunakan Hasil dari SFM dan CART karena tingkat keakuratan prediski nya jauh rendah di banding dengan Random Forest dan Logistic Regression. Ini adalah table dari Hasil Prediksi saya.

NISN	NAMA	UMUR	KELAS	RERATA UN	RERATA NILAI SEKOLAH	HASIL
12345	Risa	16	IPA 1	80	94	LOLOS
56789	Putri	18	IPS 2	70	97	LOLOS
12452	Aris	17	IPA 3	84	77	COBA LAGI
24566	Ida	17	IPA 2	90	89	LOLOS
78657	Tuti	17	IPS 4	92	92	LOLOS
28428	Nugraha	16	IPA 1	93	81	LOLOS
93020	Adi	18	IPA 2	91	92	LOLOS
24748	Sora	16	IPA 6	79	78	COBA LAGI
85326	Eka	18	IPA 2	87	78	LOLOS
37944	Ahmad	16	IPA 1	88	95	LOLOS
27424	Andre	18	IPS 2	89	93	LOLOS

# DATA TRAIN



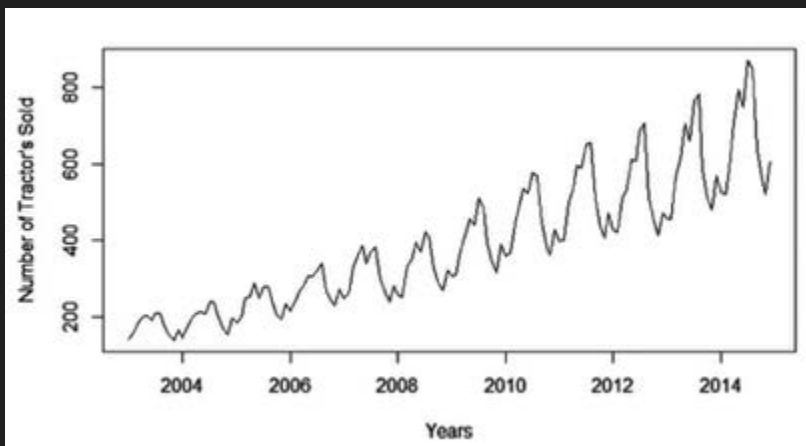
## DATA TEST (HASIL PREDIKSI)



## Nomor 5:

Di permasalahan ini, metode yang saya gunakan adalah time series.

Time series adalah observasi pada value-value yang di ambil dari suatu variable pada waktu yang berbeda-beda. Ada beberapa pattern dari time-series, dan pattern yang ada di data ini adalah Trend. Trend adalah data yang bergerak ke atas atau ke bawah bergeser dalam kumpulan data dari waktu ke waktu. Ini adalah contoh grafik Trend.



Data trend ini juga menghasilkan sebuah pattern atau seasonality. Berikut kita akan menggunakan metode AR dan ARIMA.

## AutoRegressive (AR)

AR didapatkan dengan meregresikan data itu ke data itu sendiri dengan time lag.

$$Y_t = f(Y_{t-1}, Y_{t-2}, Y_{t-3} \dots)$$

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \beta_3 Y_{t-3} \dots$$

AR ini mirip dengan linear regression dimana  $y = ax + B$ , tetapi tidak menggunakan  $x$  melainkan menggunakan data dari sebelumnya sebagai parameter.

## Moving Average (MA)

Moving average di dapatkan dari error dari perhitungan sebelumnya.

$$Y_t = f(\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \epsilon_{t-3}, \dots)$$

or

$$Y_t = \beta + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3} + \dots$$

Error ini dilambangkan dengan epsilon, dengan error dari  $y_t$  sebelumnya akan menjadi moving average.

## ARMA

ARMA didapatkan dengan menggabungkan AR dan MA dan digunakan di model data yang stationary dengan ciri-ciri mean dan variance dalam konstan dalam periode waktu. Dikatakan mean akan selalu sama dalam bagian graph tersebut.

## ARIMA

ARIMA didapatkan dengan menggabungkan ketiga metode tersebut AR, differencing dan ARIMA, digunakan di data yang tidak stationary. Differencing di sini akan mengubah data yang tidak stationary menjadi stationary. ARIMA menerima 3 parameter AR, I, MA sebagai p, d, q, contoh ketika kita menggunakan ARIMA(1,1,1) berarti kita menggunakan 1 data sebelumnya sebagai parameter, dan mengubah data menjadi stationary dengan 1 differencing dan moving average menggunakan error.

## Akaike Information Criteria (AIC)

Disetiap parameter ARIMA akan menghasilkan AIC, semakin kecil AIC maka semakin bagus parameter tersebut. AIC didefinisikan oleh persamaan sederhana dari jumlah kuadrat dan jumlah derajat kebebasan kedua model

$$\Delta AIC = N \times \ln \left( \frac{SS2}{SS1} \right) + 2\Delta DF$$



Untuk mengolah data ini saya menggunakan Python

- Pertama import library yang dibutuhkan

```
In [1]: # mengimport library yang di butuhkan  
  
import pandas as pd  
from pandas import datetime  
import matplotlib.pyplot as plt  
import numpy as np
```

- Lalu disini saya membuat function untuk merubah kolom tanggal menjadi bentuk datetime

```
In [2]: # mengubah bentuk string dari Tanggal ke dalam bentuk datetime  
  
def parser(x):  
    return datetime.strptime(x,'%d') # menuliskan bahwa data yang saat ini adalah date
```

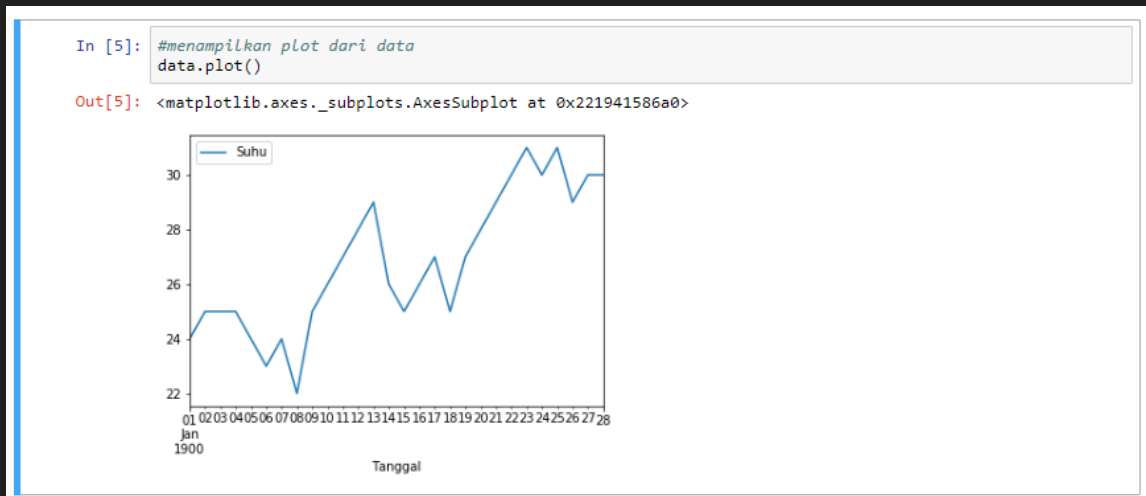
- Setelah itu kita import file csv dengan parameter sepetaror (;) karena csv kita di batasi semicolon, yang membuat kolom tanggal kita menjadi index dari dataframe kita

```
In [3]: # mengimport data berdasarkan format dari csv yang di pisahkan semicolon  
# dan menjadikan kolom Tanggal sebagai index  
# parse_date menjelaskan kolom mana yang menjadi tanggal di data tersebut  
  
data = pd.read_csv('Cuaca_Juli_2020.csv', sep = ';', index_col=0, parse_dates=[0], date_parser = parser)
```



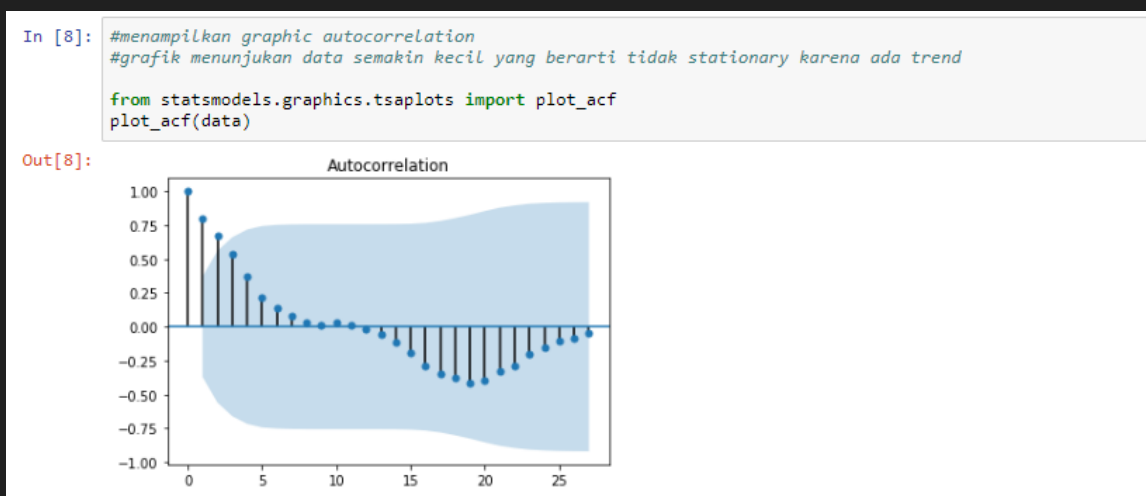


- Kita gunakan plot untuk melihat bentuk dari data kita



Dari grafik ini terlihat bahwa ini memiliki trend dan menghasilkan sebuah pattern. Yang berarti data ini tidak stationary

- Untuk lebih jelas lagi kita akan melihat dari korelasinya

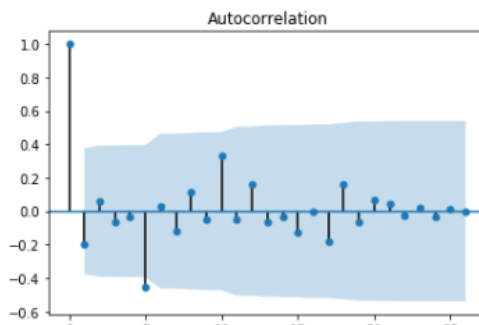


Dari grafik auto korelasi tersebut semakin terlihat bahwa data ini adalah trend karena terlihat bar semakin mengecil seiring waktu

- Agar data menjadi stationary kita gunakan data diff diaman mengurangi sata sebelumnya sebanyak 1 periode, nanatinya diff ini akan digunakan untuk menentukan d di ARIMA

```
In [9]: # disini data sudah terlihat sebagai data stationary
plot_acf(data_diff)
```

Out[9]:



- Sebelum masuk ke dalam Autoregressive Model saya akan membuat data train dan set untuk mengecek yang lebih optimal

```
In [11]: #mengambil suhu dari data
X = data.values
X.size
```

Out[11]: 28

```
In [12]: #memecah data untuk digunakan sebagai tes perbandingan nantinya
train = X[0:21]
test = X[21:]
test.size
```

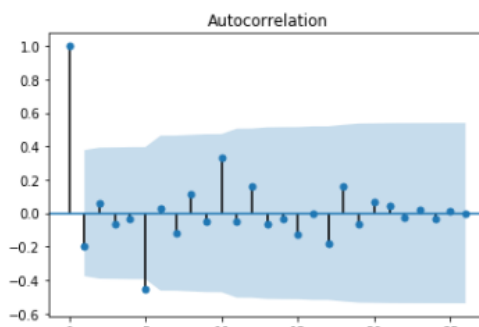
Out[12]: 7

Dari grafik auto korelasi tersebut semakin terlihat bahwa data ini adalah trend karena terlihat bar semakin mengecil seiring waktu

- Agar data menjadi stationary kita gunakan data diff diaman mengurangi sata sebelumnya sebanyak 1 periode, nanatinya diff ini akan digunakan untuk menentukan d di ARIMA

```
In [9]: # disini data sudah terlihat sebagai data stationary
plot_acf(data_diff)
```

Out[9]:



- Sebelum masuk ke dalam Autoregressive Model saya akan membuat data train dan set untuk mengecek yang lebih optimal

```
In [11]: #mengambil suhu dari data
X = data.values
X.size
```

Out[11]: 28

```
In [12]: #memecah data untuk digunakan sebagai tes perbandingan nantinya
train = X[0:21]
test = X[21:]
test.size
```

Out[12]: 7



- Berikut kita akan mencoba model AR atau Autoregressive , kita akan mengimport library dari AR dan menjalankannya. Di sini kita menggunakan data train untuk mengecek optimal atau tidaknya menggunakan metode ini. Setelah di AR akan kita fit untuk bisa di olah atau visualize.

### Autoregressive Model

```
In [14]: #mengimport library dari AR
from statsmodels.tsa.ar_model import AR
from sklearn.metrics import mean_squared_error

#menjalankan AR pada train
model_ar_test = AR(train)
model_ar_fittest = model_ar_test.fit()
```

- Setelah mendapatkan AR tersebut, maka kita akan set untuk mulai dari tanggal 21 - 29 terlebih dahulu. Dan setelah itu akan kita gabungkan dengan data train.

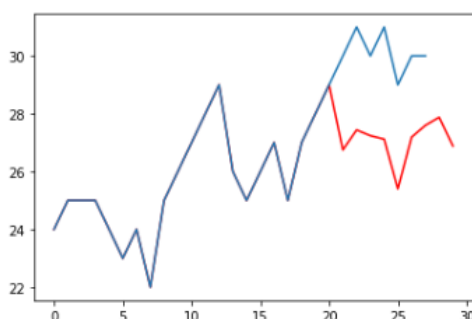
```
In [30]: #memprediksi berdasarkan data train
predictions_test = model_ar_fittest.predict(start=21,end=29)
```

```
In [31]: #menggabungkan data train dan prediksi
ar_testgoal = np.concatenate((train, predictions_test), axis = None)
```

- Setelah itu akan kita tampilkan hasl dari prediksi data train tersebut

```
In [32]: #menampilkan data asli beserta prediksi
plt.plot(ar_testgoal, color = 'red')
plt.plot(X)
```

Out[32]: [<matplotlib.lines.Line2D at 0x22197895c88>]



Jika dilihat menggunakan metode AR tidak optimal dan menbuahkan prediksi yang cukup jauh dari yang di harapkan, maka saya tidak gunakan hasil dari AR ini

- Setelah itu saya akan mencoba memprediksi di data set aslinya hanya sebagai bahan pertimbangan, dan inilah hasil dari prediksi menggunakan data asli tanggal 29 sampai 39

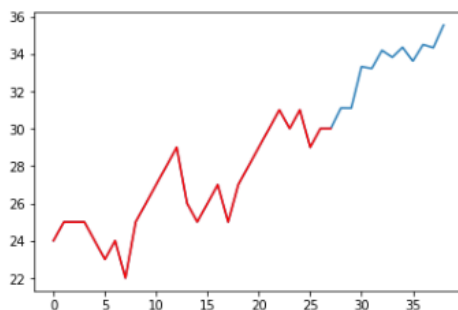
```
In [33]: # kali ini kita menggunakan data aslinya sebagai bahan pertimbangan
model_ar = AR(X)
model_ar_fit = model_ar.fit()
```

```
In [34]: # kita akan memprediksi dari tanggal 29 sampai 39 (10 hari)
predictions_ar = model_ar_fit.predict(start=28,end=38)
```

```
In [35]: # menggabungkan hasil prediksi kita dengan data awal
ar_goal = np.concatenate((X, predictions_ar), axis = None)
```

```
In [36]: #menampilkan data hasil gabungan prediksi dan wal
plt.plot(ar_goal)
plt.plot(X, color= 'red')
```

```
Out[36]: [matplotlib.lines.Line2D at 0x221978e5470]
```



- Hasil dari prediksi ini tidak cukup memuaskan maka dari itu saya akan beralih ke metode ARIMA

Jika dilihat menggunakan metode AR tidak optimal dan menbuahkan prediksi yang cukup jauh dari yang di harapkan, maka saya tidak gunakan hasil dari AR ini

- Setelah itu saya akan mencoba memprediksi di data set aslinya hanya sebagai bahan pertimbangan, dan inilah hasil dari prediksi menggunakan data asli tanggal 29 sampai 39

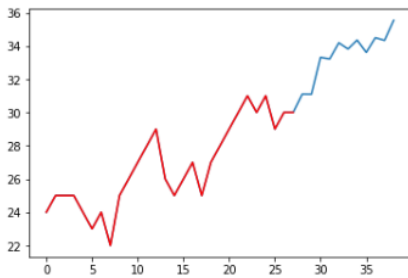
```
In [33]: # kali ini kita menggunakan data aslinya sebagai bahan pertimbangan
model_ar = AR(X)
model_ar_fit = model_ar.fit()
```

```
In [34]: # kita akan memprediksi dari tanggal 29 sampai 39 (10 hari)
predictions_ar = model_ar_fit.predict(start=28,end=38)
```

```
In [35]: # menggabungkan hasil prediksi kita dengan data awal
ar_goal = np.concatenate((X, predictions_ar), axis = None)
```

```
In [36]: #menampilkan data hasil gabungan prediksi dan wal
plt.plot(ar_goal)
plt.plot(X, color= 'red')
```

```
Out[36]: [<matplotlib.lines.Line2D at 0x221978e5470>]
```



```
In [22]: # ini adalah data dari prediksi menggunakan AR
# yang kita butuhkan hanyalah 3 data pertama
# tanggal 29 : 31.10268156
# tanggal 30 : 31.09482315
# tanggal 31 : 33.31349727

predictions_ar
```

```
Out[22]: array([31.10268156, 31.09482315, 33.31349727, 33.21792786, 34.18767262,
33.81403407, 34.35056097, 33.60859264, 34.49673619, 34.33214847,
35.54021757])
```

- Hasil dari prediksi ini tidak cukup memuaskan maka dari itu saya akan beralih ke metode ARIMA

- Berikut saya akan gunakan metode ARIMA, di metode ARIMA ini kita menggunakan AR sebagai p, I sebagai d, dan q sebagai MA. Pertama saya akan membuat loop untuk memuat semua kemungkinan p,d,dan q.

```
In [37]: #digunakan untuk memprediksi semua kemungkinan pdq
#p = periode yang digunakan untuk menganalisa
#d = selisih ( bisa dilihat dari diff sebelumnya)
#q = moving average

import itertools
p=d=q=range(0,10)
pdq = list(itertools.product(p,d,q))
pdq
```

```
Out[37]: [(0, 0, 0),
(0, 0, 1),
(0, 0, 2),
(0, 0, 3),
(0, 0, 4),
(0, 0, 5),
(0, 0, 6),
(0, 0, 7),
(0, 0, 8),
(0, 0, 9),
(0, 1, 0),
(0, 1, 1),
(0, 1, 2),
(0, 1, 3),
(0, 1, 4),
```

- Setelah itu function pdq itu akan kita panggil untuk mencari AIC terkecil dengan melakukan iterasi

```
In [*]: #ini digunakan untuk mencari RSS yang paling kecil dengan melakukan pengulangan
import warnings
warnings.filterwarnings('ignore')
for param in pdq:
    try:
        model = ARIMA(X,order = param)
        model_fit = model.fit()
        print(model_fit.aic)
        print(param,model_fit.aic)
    except:
        continue
```

```
134.9494237039258
(0, 0, 0) 134.9494237039258
120.55902307408532
(0, 0, 1) 120.55902307408532
113.88830050479086
(0, 0, 2) 113.88830050479086
112.68940443948176
(0, 0, 3) 112.68940443948176
108.34407459250563
(0, 0, 4) 108.34407459250563
112.34461497689323
(0, 0, 5) 112.34461497689323
113.2156999834196
```

- Setelah itu kita akan menggunakan aic terkecil yaitu order 5,1,1 dan kita gunakan order itu ke dalam data training untuk melihat prediksi yang di lakukan ARIMA

```
In [23]: #menimport library dan menjalankan prediksi arima terhadap data train
# dan menggunakan order yang menghasilkan RRS paling kecil

from statsmodels.tsa.arima_model import ARIMA
model_testarima = ARIMA(train,order = (5,1,1))
model_fitarima = model_testarima.fit()
print(model_fitarima.aic)

73.4508558968715
```

- Jarak prediksi yang saya gunakan dalah 10 step dimana berarti 10 hari kedepan. Setelah itu saya gabungkan dengan data train.

```
In [24]: #menampilkan prediksi 10 hari sesudah data awal
predictions_testarima = model_fitarima.forecast(steps=10)[0]
predictions_testarima

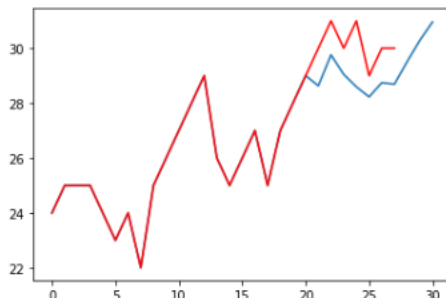
Out[24]: array([28.63200818, 29.7600298 , 29.04590978, 28.59367494, 28.22641025,
28.74143993, 28.69160084, 29.51029815, 30.28667839, 30.95753702])

In [25]: #menggabungkan hasil prediksi dari training data dan data asli sebagai perbandingan
goaltest = np.concatenate((train, predictions_testarima), axis = None)
```

- Setelah itu kita akan melihat plot dari hasil prediksi yang di hasilkan dari data train

```
In [26]: # hasil untuk membandingkan hasil prediksi dengan data aslinya
plt.plot(goaltest)
plt.plot(X,color='red')

Out[26]: [<matplotlib.lines.Line2D at 0x19af37947f0>]
```





- Dilihat dari data yang dihasilkan prediksi yang dilakukan data train cukup mendekati, maka dari itu saya akan menggunakan metode ARIMA sebagai metode prediksi saya dan AIC 5,1,1

```
In [28]: #setelah mendapatkan RSS paling kecil lalu kita masukan ke dalam order
#Lagi kita menjalankan arima untuk data aslinya

model = ARIMA(X,order = (5,1,1))
model_fit = model.fit()
print(model_fit.aic)

92.42304528129736
```

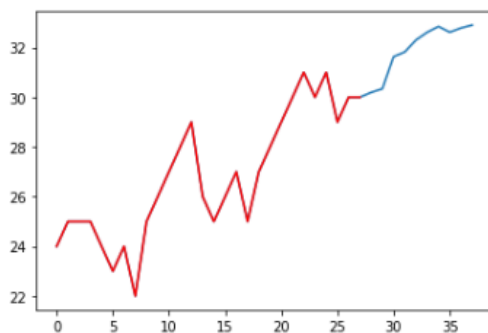
```
In [29]: #menampilkan prediksi untuk 10 hari kedepan
predictions = model_fit.forecast(steps=10)[0]
```

```
In [30]: #menggabungkan hasil prediksi dengan data awal
goal = np.concatenate((X, predictions), axis = None)
```

- Sama seperti sebelumnya saya memakai 10 step atau 10 hari prediksi kedepan dan menggabungkan data awal dengan hasil prediksi yang kita dapatkan. Dan inilah hasil prediksi yang saya dapatkan.

```
In [31]: #menampilkan hasil dari prediksi
plt.plot(goal)
plt.plot(X,color='red')
```

```
Out[31]: [<matplotlib.lines.Line2D at 0x19af48a0b70>]
```



- Berikut adalah hasil prediksi untuk tanggal 29,30, dan 31.

```
In [32]: # ini adalah data dari prediksi menggunakan AR
# yang kita butuhkan hanyalah 3 data pertama
# tanggal 29 : 30.19262905
# tanggal 30 : 30.33919401
# tanggal 31 : 31.62847663

predictions

Out[32]: array([30.19262905, 30.33919401, 31.62847663, 31.82050289, 32.30376381,
                32.61402052, 32.84605322, 32.61422582, 32.7843486 , 32.90269673])
```

Tanggal 29 : 30.19262905

Tanggal 30 : 30.33919401

Tanggal 31 : 31.62847663

## JAWABAN AKHIR NOMOR 5

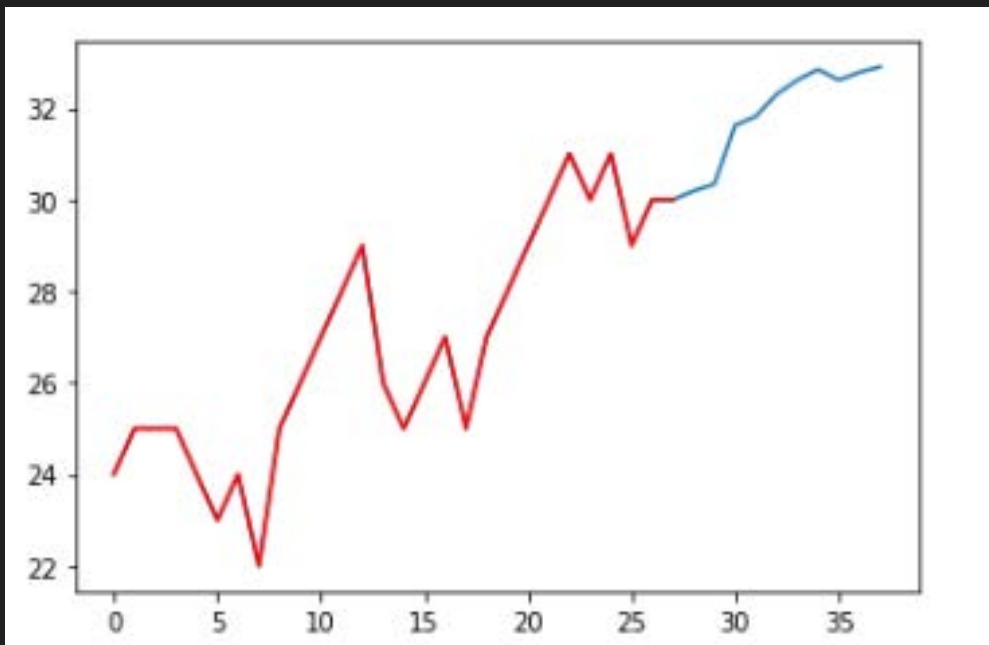
Setelah Menggunakan 2 metode di atas yaitu AR dan ARIMA, jelas saya memutuskan untuk menggunakan hasil prediksi ARIMA karena lebih akurat di bandung AR.

Pola yang dihasilkan juga sangat cocok untuk di muat ke dalam logika. Sehingga ini adalah jawaban akhir dari nomor 5 ini.

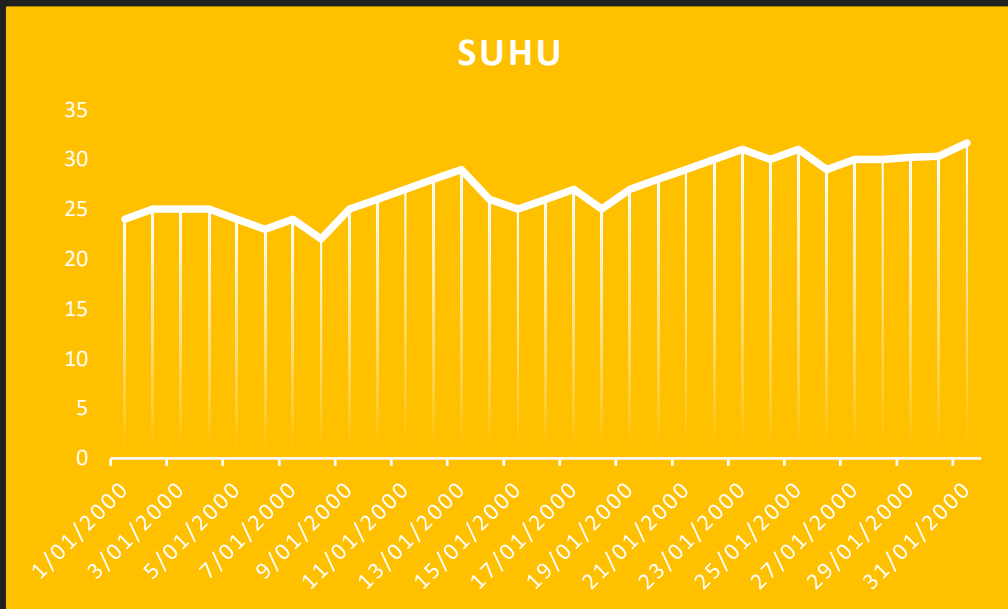
Tanggal 29 : 30.19262905

Tanggal 30 : 30.33919401

Tanggal 31 : 31.62847663




## GRAFIK PREDIKSI





## TABEL PREDIKSI

Tanggal	Suhu		
1/01/2000	24	16/01/2000	26
2/01/2000	25	17/01/2000	27
3/01/2000	25	18/01/2000	25
4/01/2000	25	19/01/2000	27
5/01/2000	24	20/01/2000	28
6/01/2000	23	21/01/2000	29
7/01/2000	24	22/01/2000	30
8/01/2000	22	23/01/2000	31
9/01/2000	25	24/01/2000	30
10/01/2000	26	25/01/2000	31
11/01/2000	27	26/01/2000	29
12/01/2000	28	27/01/2000	30
13/01/2000	29	28/01/2000	30
14/01/2000	26	29/01/2000	30.19263
15/01/2000	25	30/01/2000	30.33919
		31/01/2000	31.62848


# JAWABAN NOMOR 6





Introduction to Python

Practice Now




Intermediate Python for Data Science

Practice Now


SOLIDIFY YOUR KNOWLEDGE FROM PREVIOUS COURSES





**PROJECT**

**TV, Halftime Shows, and the Big Game**


Load, clean, and explore Super Bowl data in the age of soaring ad costs and flashy halftime shows.





Python Data Science Toolbox (Part 1)

Practice Now




Python Data Science Toolbox (Part 2)


Practice Now

Selain Intro to Python for Data Science dan Intermediate Python for Data Science saya berhasil menyelesaikan course lain yang lebih Advanced.


Licenses & Certifications




**SQL for Data Science**  
 Coursera  
 Issued Sep 2019 - No Expiration Date  
 Credential ID ZUAAWMB2CGWK  
[See credential](#)




**Business Intelligence: Strategies, Tools & Techniques**  
 Udemy  
 Issued Aug 2019 - No Expiration Date  
 Credential ID UC-JRB9UCJ2  
[See credential](#)




**DATA SCIENCE - OmahTi Learning Center**  
 OmahTI UGM (Organisasi Mahasiswa Ahli Teknologi Informasi)  
 Issued May 2019 - No Expiration Date  
[See credential](#)




**Python Data Science Toolbox (Part 1)**  
 DataCamp  
 Issued May 2019 - No Expiration Date  
 Credential ID #9084695  
[See credential](#)



**Python Data Science Toolbox (Part 2)**  
 DataCamp  
 Issued May 2019 - No Expiration Date  
 Credential ID #9572710  
[See credential](#)

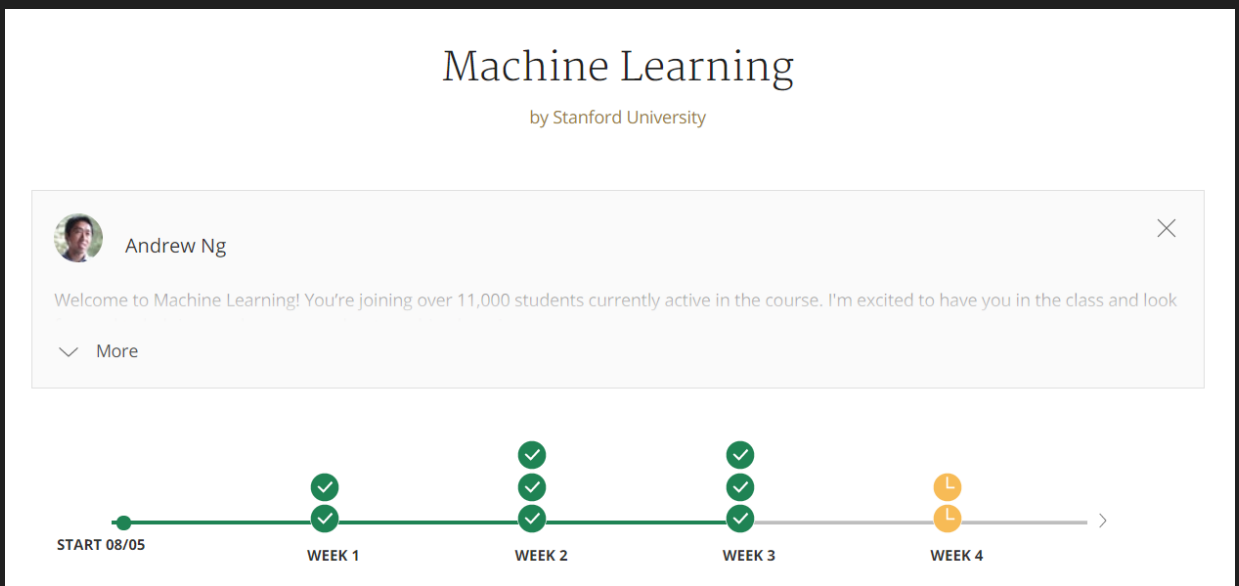


**Intermediate Python for Data Science**  
 DataCamp  
 Issued Jan 2019 - No Expiration Date  
 Credential ID #8274604  
[See credential](#)



**Introduction to Python**  
 DataCamp  
 Issued Jan 2019 - No Expiration Date  
 Credential ID #8189321  
[See credential](#)

Saat ini saya sedang menjalani Course di Coursera mengenai Machine Learning yang di adakan oleh Stanford University dan berniat mengambil course Deeplearning



Untuk informasi lebih lengkap mengenai saya bisa dikunjungi LinkedIn saya.  
[linkedin.com/in/irvinn/](https://www.linkedin.com/in/irvinn/)

