

SCHEDULING DATA NON-STREAM PADA KASUS PENYAKIT CARDIOVASCULAR

Dyah Ayu Wulandari¹, Lenny Raufi Syafitri², Anggasta Aji Azhari³

¹Fakultas Ilmu Komputer

²Fakultas Ilmu Komputer

³Fakultas Ilmu Komputer

Email: dyahaw_@student.ub.ac.id, lennyraufi@student.ub.ac.id, anggazaazhari12@student.ub.ac.id

Abstrak

Penelitian ini bertujuan untuk mengklasifikasikan pasien yang memiliki gangguan penyakit kardiovaskuler dengan melihat hubungan antar fitur dataset tersebut. Data ini memiliki label biner untuk mengelompokkan pasien yang memiliki gangguan kardiovaskuler dan juga terdapat demografi dari pasien-pasien tersebut. Pengolahan data ini menggunakan algoritma klasifikasi yaitu *Random Forest Classifier* dengan pembagian data latih dan data uji sebesar 80 persen dan 20 persen. Hasil akurasi yang didapatkan dalam pengolahan data ini terhadap algoritma tersebut sebesar persen.

Kata kunci: *Kardiovaskuler, Klasifikasi, Random Forest Classifier, dan Decision Tree*

SCHEDULING DATA NON-STREAM IN CASE OF CARDIOVASCULAR DISEASE

Abstract

This Study is to classify patients who have cardiovascular disease disorders to see the relationship between the features of the dataset. This data has a binary label to classify patients with cardiovascular disorders and also includes the demographics of these patients. This Study uses a classification algorithm, namely the Random Forest Classifier with the distribution of training data and test data of 80% and 20%. The accuracy results obtained in processing this data against the algorithm are 69%.

Keywords: *Cardiovascular, Classification, Random Forest Classifier and Decision Tree*

1. PENDAHULUAN

Penyakit jantung dan pembuluh darah atau biasa disebut penyakit kardiovaskuler merupakan kondisi dimana terjadi penyempitan atau penyumbatan pembuluh darah yang bisa menyebabkan beberapa penyakit lain seperti serangan jantung dan stroke. Penyakit kardiovaskular ini memerlukan penanganan segera ditangani karena berhubungan dengan jantung sebagai organ vital. Penyakit jantung tergolong penyakit yang sering terjadi dan penyakit ini disebut sebagai penyebab kematian terbanyak di seluruh dunia.

World Health Organization (WHO) (2002) melaporkan Noncommunicable Disease (NCDs) atau penyakit non infeksi menyumbang 60 persen mortalitas dan 47 persen beban penyakit di dunia akan terus meningkat dengan prediksi pada tahun 2020 kematian akibat NCDs adalah 73 persen dan merupakan 60 persen beban penyakit di dunia. Penyakit non infeksi utama yang menduduki proporsi tertinggi adalah penyakit jantung koroner, stroke, diabetes (DM), kanker dan penyakit paru.

Persepsi sampai saat ini penyakit jantung adalah tipikal penyakit laki-laki, namun data-data menunjukkan telah terjadi pergeseran angka kejadian penyakit jantung antara laki-laki dan perempuan. Angka kejadian

penyakit jantung dan stroke akhir-akhir ini menunjukkan tidak terdapat perbedaan antara laki-laki dan perempuan dimana terdapat kecenderungan perempuan meningkat angka kejadiannya. Untuk itu, kami akan meneliti klasifikasi dari penderita penyakit kardiovaskular ini terhadap data demografi serta kebiasaan sehari-hari pasien. Penelitian ini bertujuan sebagai jawaban untuk persepsi-persepsi yang masih salah tentang penyakit kardiovaskular.

2. TINJAUAN PUSTAKA

2.1 KARDIOVASKULAR

Penyakit Kardiovaskular menurut Rosjidi (2014: 2) adalah penyakit yang berhubungan dengan pola perilaku modern sehingga penyakit ini tidak hanya menyerang Negara-negara maju saja tetapi sudah menjadi ancaman bagi Negara yang sedang menuju kearah modernisasi. Jenis-jenis penyakit kardiovaskular antara lain Aterosklerosis, Jantung koroner, Aritmia, dan lain-lain. Penyakit kardiovaskular ini masih belum dapat disembuhkan secara total artinya, seorang yang didiagnosis terserang penyakit ini akan terus memiliki penyakit ini sepanjang hidup.

2.2 RANDOM FOREST

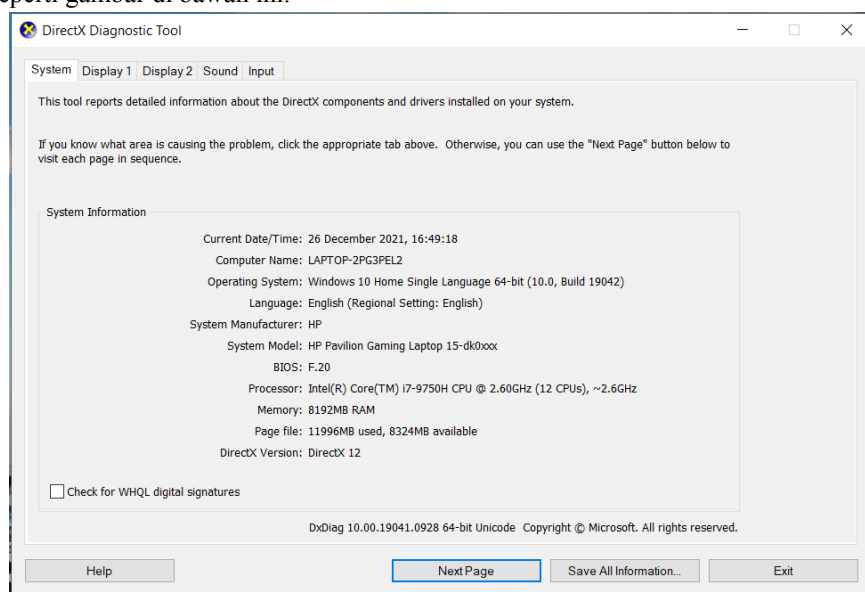
Penelitian ini menggunakan algoritma klasifikasi yang dinilai mampu mendeteksi data dengan baik yakni *Random Forest Classifier*. *Random forest* adalah algoritma pembelajaran yang diusulkan oleh Breiman [Mach. Learn. 45 (2001) 5-32] yang menggabungkan beberapa algoritma decision tree dan menggabungkan prediksi menggunakan rata-rata.

Penggunaan *Random Forest Classifier* ini luas dan penggunaannya yang sangat praktis. Namun, dibalik itu semua hanya sedikit yang dapat diketahui dari prosedur matematika algoritma ini. Perbedaan antara praktik dan teori ini dapat berasal dari ketika kita secara bersamaan menganalisis praktik dan teori sehingga akan sangat menyulitkan. Proses pengacakan dan struktur *decision tree* dari algoritma ini sangat bergantung pada data. Selain untuk mengklasifikasi, *Random Forest* ini sebelumnya dapat digunakan untuk model regresi. *Random forest* yang digunakan untuk model regresi bernama *Random Forest Regressor*.

3. METODOLOGI PENELITIAN

3.1. Instalasi dan Konfigurasi Docker

Dalam melakukan scheduling data non stream, kami menggunakan docker dan apache airflow. Dalam menginstall docker dan WSL menggunakan operating system windows dengan spesifikasi laptop yang digunakan yaitu seperti gambar di bawah ini.



Gambar 1 Spesifikasi Laptop yang Digunakan

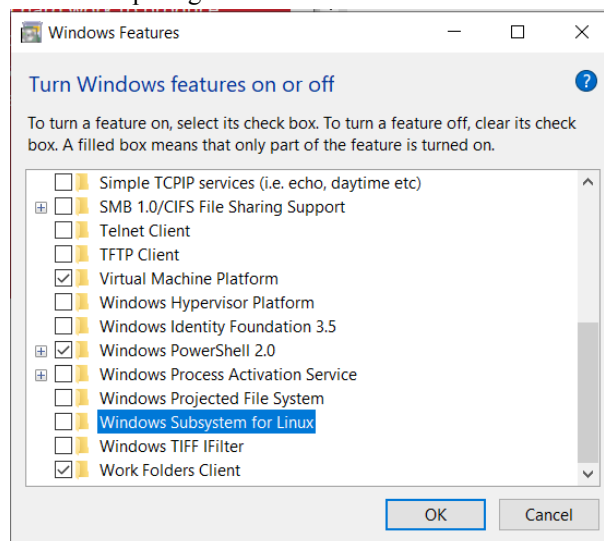
Untuk menginstall docker pada windows 10, requirement system minimumnya antara lain yaitu :

- Install Windows 10 Pro atau Enterprise 64-bit.
- Enable fitur WSL 2 pada Windows
- 64 bit processor dengan Second Level Address Translation (SLAT)
- 4GB system RAM.
- Support BIOS-level hardware virtualization.
- Download dan install the Linux kernel update package.

Docker adalah software atau tools yang digunakan untuk membuat(create), menjalankan(run) serta melakukan deployment suatu aplikasi dengan menggunakan container. Sumber kode (source code) serta libraries terkait (jika ada) akan dipaketkan dalam sebuah container. Untuk melakukan scheduling langkah pertama yang dilakukan yaitu melakukan instalasi dan konfigurasi docker, berikut adalah tahapan-tahapannya :

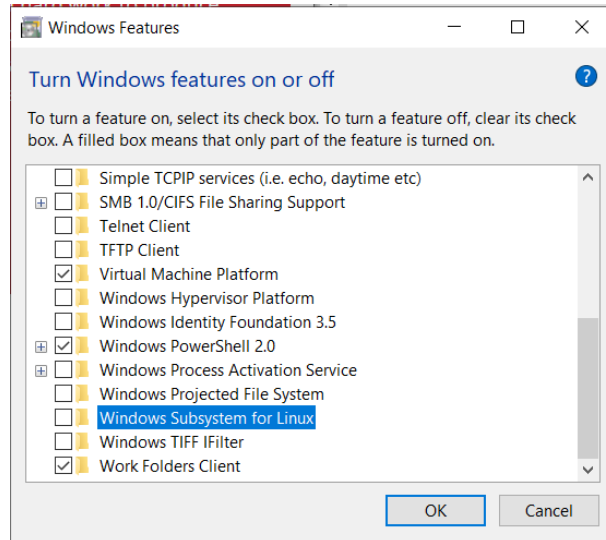
- Sebelum melakukan download docker dan instalasi docker, dapat mengaktifkan fitur WSL 2 di Windows terlebih dahulu. Docker untuk Windows 10 berjalan dengan memanfaatkan WSL 2 (WSL = “Windows Subsystem for Linux”) sehingga sebelum menginstal backend docker desktop WSL 2 harus mengaktifkan fitur WSL 2 di windows terlebih dahulu. Dengan WSL 2, user dapat menggunakan workspace Linux pada Windows tanpa harus memaintain build scripts keduanya. WSL 2 menyediakan perbaikan pada sistem berbagi file (file sharing), waktu booting (boot time), dan mengizinkan akses pada fitur-fitur terbaru untuk pengguna Docker Desktop. Dengan WSL 2 ini, waktu yang dibutuhkan untuk menghidupkan Docker daemon dari cold start pun jauh lebih cepat, yaitu kurang dari 10 detik (bandingkan dengan versi Docker Desktop sebelumnya, yaitu hampir 1 menit). Langkah-langkah dalam mengaktifkan fitur WSL 2 di windows yaitu :

- Membuka start on Windows 10.
- Setelah itu, melakukan search turn windows feature on or off dan menekan hasil paling atas untuk membuka. Maka akan muncul seperti gambar di bawah ini.



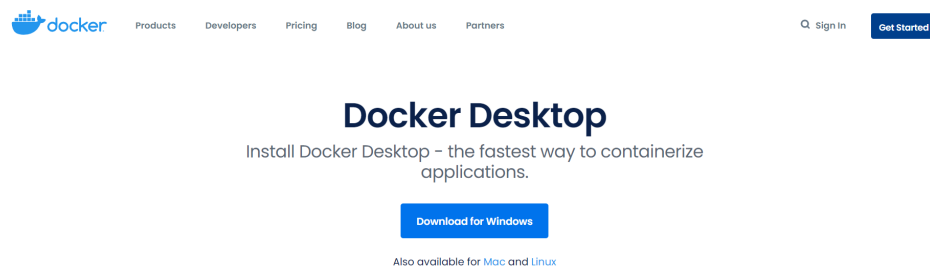
Gambar 2 Turn Windows Features on or off

- Lalu, mencentang “Windows Subsystem for Linux” seperti pada gambar di bawah ini.



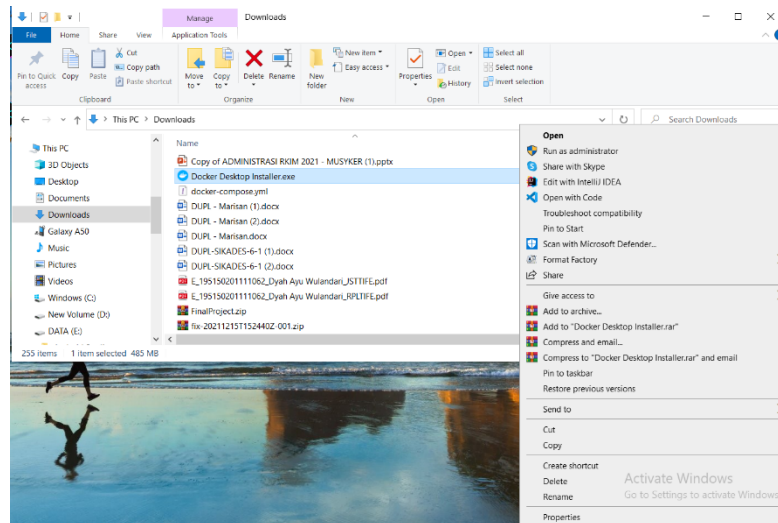
Gambar 3 Enable WSL on Windows 10

- Setelah itu, menekan tombol “OK” dan menekan tombol “Restart”.
- b. Melakukan download docker desktop untuk windows.
Download docker dapat dilakukan pada halaman resmi yaitu <https://www.docker.com/products/docker-desktop>. Setelah membuka link tersebut maka akan menampilkan halaman seperti gambar di bawah ini. Kemudian, memilih versi windows karena sistem operasi yang digunakan yaitu windows sehingga menekan “Download for Windows”.



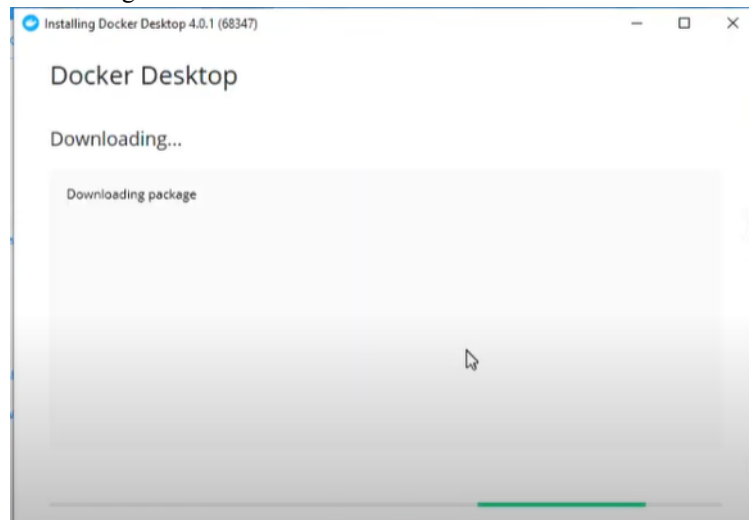
Gambar 4 Halaman Resmi Docker

- c. Setelah menekan “Download for Windows” maka akan mengunduh docker, dan setelah proses pengunduhan selesai maka dapat menjalankan file installer yang sudah diunduh tersebut. Untuk menjalankan proses instalasi dapat menggunakan mode administrator dengan cara me-klik kanan file yang akan diinstal lalu me-klik “Run as administrator”.



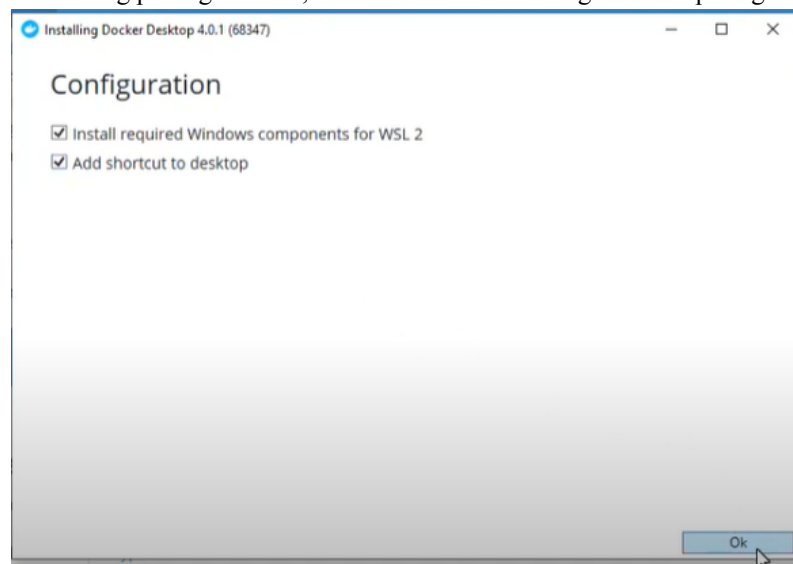
Gambar 5 Meng-klik Run as administrator

- d. Setelah me-klik run as administrator maka tampilan akan seperti di bawah ini untuk proses downloading package dan melakukan konfigurasi docker.



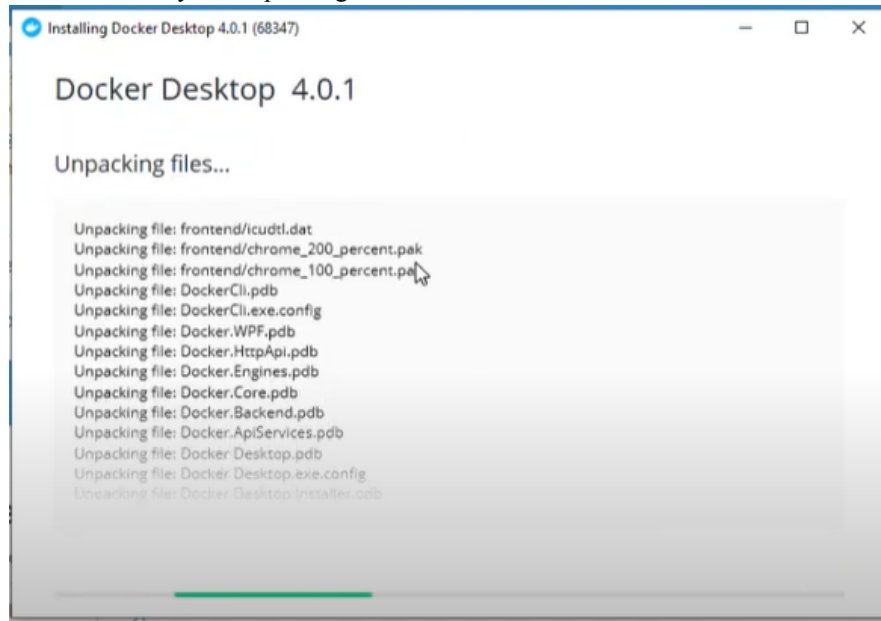
Gambar 6 Tahap Downloading Package

Setelah proses downloading package selesai, maka akan muncul configuration seperti gambar di bawah ini.



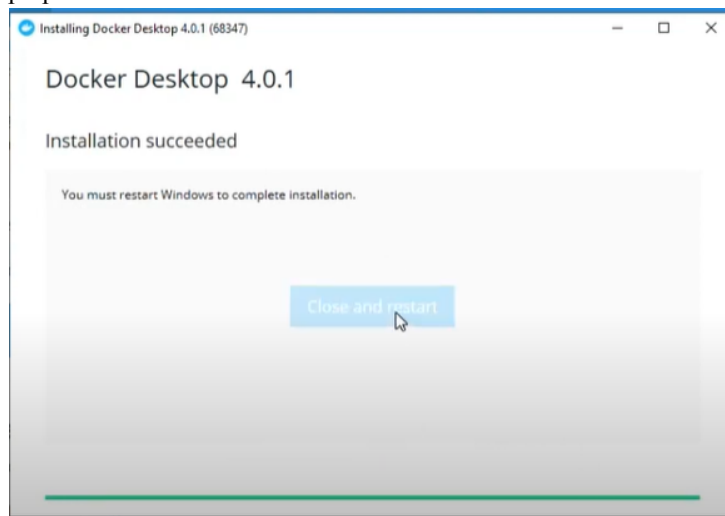
Gambar 7 Tahap Configuration Docker

Untuk configuration saya centang semua lalu mengklik tombol “Ok”. Setelah itu akan muncul tampilan seperti pada gambar di bawah ini yaitu unpacking files.



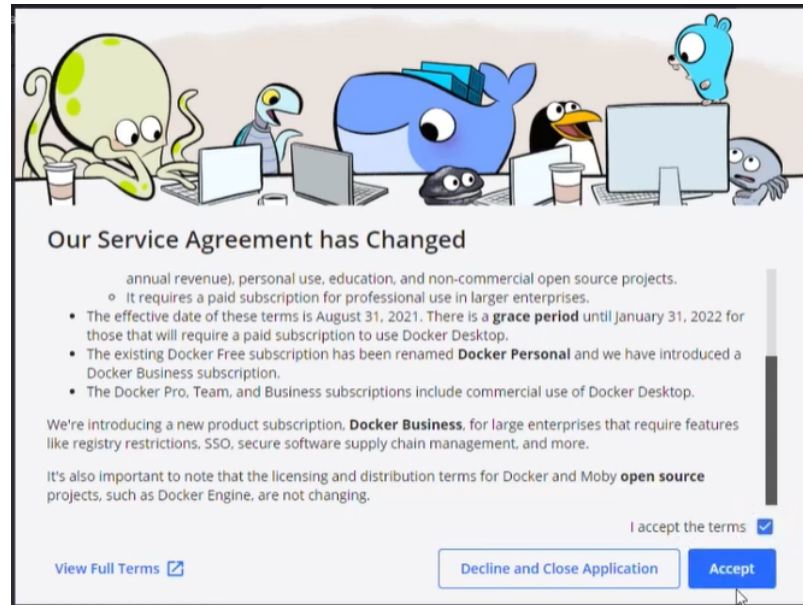
Gambar 8 Tahap Unpacking Files

- e. Setelah proses instalasi selesai maka akan muncul tampilan seperti di bawah ini lalu dapat menekan “Close dan Restart”. Lalu laptop akan terestart.



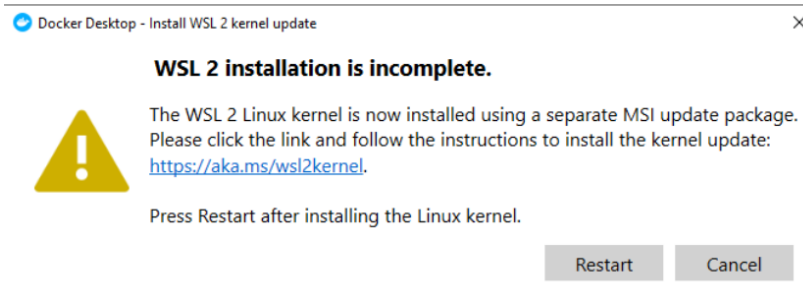
Gambar 9 Tahap Instalasi Success

- f. Setelah instalasi berhasil dan setelah restart laptop selesai maka akan muncul agreement seperti gambar di bawah ini. Lalu mencentang “I accept the terms” dan menekan tombol “Accept”.



Gambar 10 Tahap Our Service Agreement

- g. Setelah itu jika terdapat permasalahan pada instalasi WSL 2 atau harus diupdate maka akan muncul seperti gambar di bawah ini.



Gambar 11 Perintah untuk Update WSL 2

- h. Setelah itu, dapat menekan url di atas. Lalu akan menampilkan halaman seperti gambar di bawah ini.

Step 4 - Download the Linux kernel update package

1. Download the latest package:

- [WSL2 Linux kernel update package for x64 machines](#)

① Catatan

If you're using an ARM64 machine, please download the [ARM64 package](#) instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: `systeminfo | find "System Type"`. **Caveat:** On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German `systeminfo | find '"Systemtyp"'`.

2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)

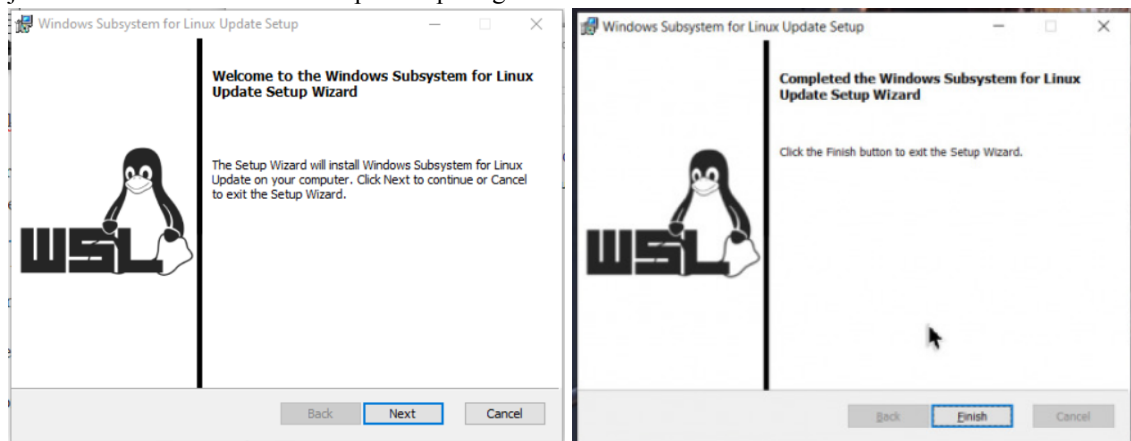
Once the installation is complete, move on to the next step - setting WSL 2 as your default version when installing new Linux distributions. (Skip this step if you want your new Linux installs to be set to WSL 1).

① Catatan

For more information, read the article [changes to updating the WSL2 Linux kernel](#), available on the [Windows Command Line Blog](#).

Gambar 12 Halaman <https://aka.ms/ws2kernel>

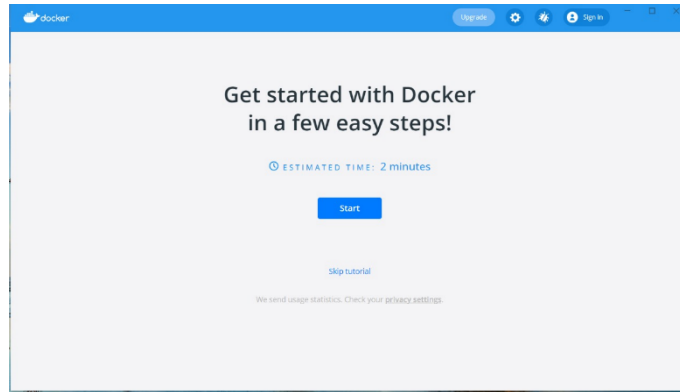
- i. Kemudian, dapat melakukan download WSL2 Linux kernel update package for x64 machines seperti perintah no.1 gambar di atas. Setelah berhasil terdownload maka dapat menjalankan installer. Jika installer dijalankan maka akan muncul tampilan seperti gambar di bawah ini.



Gambar 13 Setup Update WSL2

Setelah itu klik next untuk melanjutkan tahap instalasi lalu jika instalasi selesai dapat menekan tombol "Finish".

- j. Setelah proses selesai maka docker akan berjalan seperti pada gambar di bawah ini. Lalu dapat menekan "Start" setelah selesai maka sudah dapat digunakan.



Gambar 14 Tampilan Docker

3.2. Persiapan Data

Setelah menginstall docker dan konfigurasinya, maka langkah selanjutnya yaitu mempersiapkan data. Data yang kami gunakan yaitu dataset cardiovascular disease. Dataset yang kami gunakan dari Kaggle. Kaggle merupakan sebuah website yang menampilkan dataset serta kompetisi untuk para pencinta data scientist. Data ini diambil dengan menggunakan API Kaggle. Link dataset cardiovascular dapat diakses melalui link berikut <https://www.kaggle.com/sulianova/cardiovascular-disease-dataset>. Data ini merupakan data yang diambil dari pemeriksaan medis, informasi faktual, dan informasi yang didapat dari pasien. Data ini digunakan untuk melihat hubungan antara informasi yang telah diberikan dengan cardiovascular. Ada 3 jenis fitur masukan, antara lain yaitu : Objective (informasi factual), Examination (hasil pemeriksaan kesehatan), dan Subjective (Informasi yang diberikan oleh pasien). Dataset ini memiliki 12 fitur, antara lain :

1. Age | Objective Feature | age | int (days)
2. Height | Objective Feature | height | int (cm) |
3. Weight | Objective Feature | weight | float (kg) |
4. Gender | Objective Feature | gender | categorical code |
5. Systolic blood pressure | Examination Feature | ap_hi | int |
6. Diastolic blood pressure | Examination Feature | ap_lo | int |
7. Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |
8. Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |
9. Smoking | Subjective Feature | smoke | binary |
10. Alcohol intake | Subjective Feature | alco | binary |
11. Physical activity | Subjective Feature | active | binary |
12. Presence or absence of cardiovascular disease | Target Variable | cardio | binary |

3.3. ETL (Extract, Transform, Load) dan Scheduling

Setelah menyiapkan data, maka langkah Selanjutnya yaitu melakukan extract yang merupakan proses mengambil data dari sumber. Untuk data yang kami gunakan dari Kaggle. Setelah itu, dilanjutkan untuk proses transform yang merupakan proses untuk mengolah data menjadi format yang sesuai serta dalam proses transform ini dilakukan cleaning data. Kemudian, setelah ditransform maka dapat dilakukan load ke dalam database. Sebelum dilakukan load ke dalam database, dapat dilakukan data validation terlebih dahulu menggunakan great expectations. Great Expectations adalah pustaka Python yang dapat digunakan untuk memvalidasi, mendokumentasikan, dan membuat profil data serta untuk memastikan data sesuai dengan yang diharapkan. Dalam proses ETL ini, dilakukan scheduling. Untuk langkah-langkah scheduling proses ETL, dapat dilakukan antara lain :

- a. Sebelum melakukan scheduling pada ETL, proses yang dilakukan yaitu membuat file docker-compose.yml dan diletakan pada direktori lokal. Saya meletakan pada direktori C:\Users\WIN 10\airflow. Berikut adalah kodingan dari file docker-compose.yml

Tabel 1 Kodingan docker-compose up

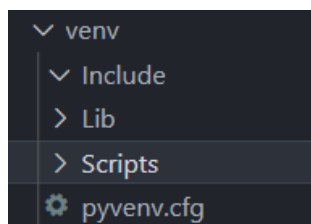
docker-compose.yml	
1	version: '3.7'
2	services:
3	postgres:

```

4      image: postgres:9.6
5      environment:
6        - POSTGRES_USER=airflow
7        - POSTGRES_PASSWORD=airflow
8        - POSTGRES_DB=airflow
9      logging:
10       options:
11         max-size: 10m
12         max-file: "3"
13
14     webserver:
15       image: puckel/docker-airflow:1.10.9
16       restart: always
17       depends on:
18         - postgres
19       environment:
20         - LOAD_EX=y
21         - EXECUTOR=Local
22       logging:
23         options:
24           max-size: 10m
25           max-file: "3"
26       volumes:
27         - ./dags:/usr/local/airflow/dags
28         # - ./plugins:/usr/local/airflow/plugins
29       ports:
30         - "8080:8080"
31       command: webserver
32       healthcheck:
33         test:
34           [
35             "CMD-SHELL",
36             "[ -f /usr/local/airflow/airflow-webserver.pid ]"
37           ]
38         interval: 30s
39         timeout: 30s
40         retries: 3

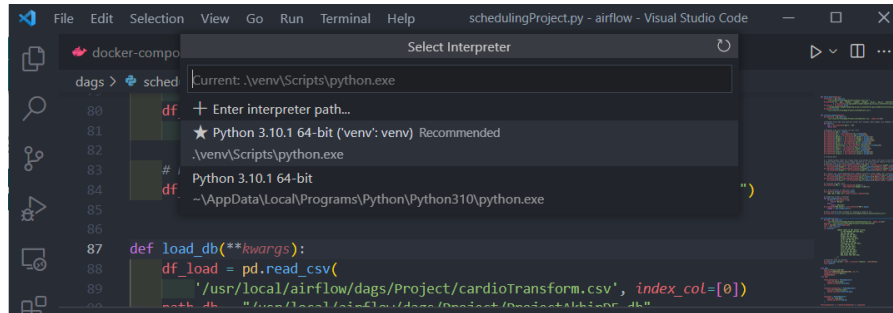
```

- b. Setelah membuat file docker-compose.yml maka langkah selanjutnya yaitu membuat virtual environment dengan perintah `python -m venv venv` pada terminal. Karena saya menggunakan visual studio code, maka saya menuliskan perintah tersebut pada terminal studio code. Setelah itu, akan muncul folder baru yang bernama "venv" seperti pada gambar di bawah ini.



Gambar 15 Folder Baru yang bernama venv

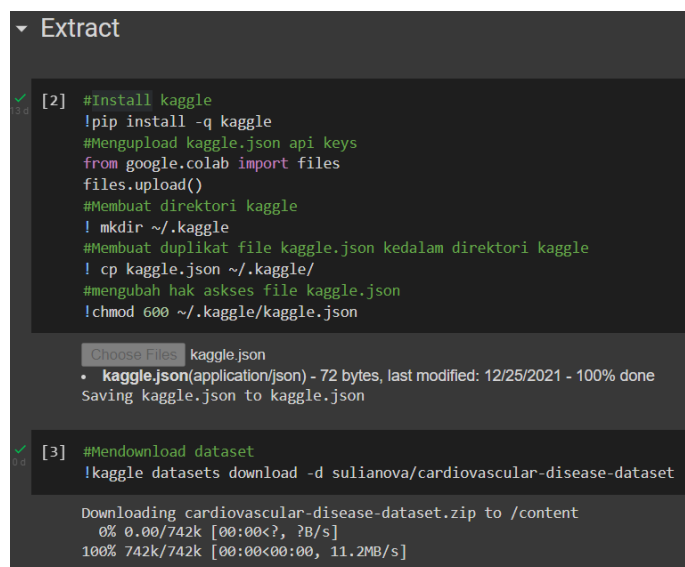
- c. Kemudian, melakukan select interpreter dengan menekan `ctrl+Shift+P` dan memilih python dari folder `venv\Scripts\python.exe` seperti gambar di bawah ini



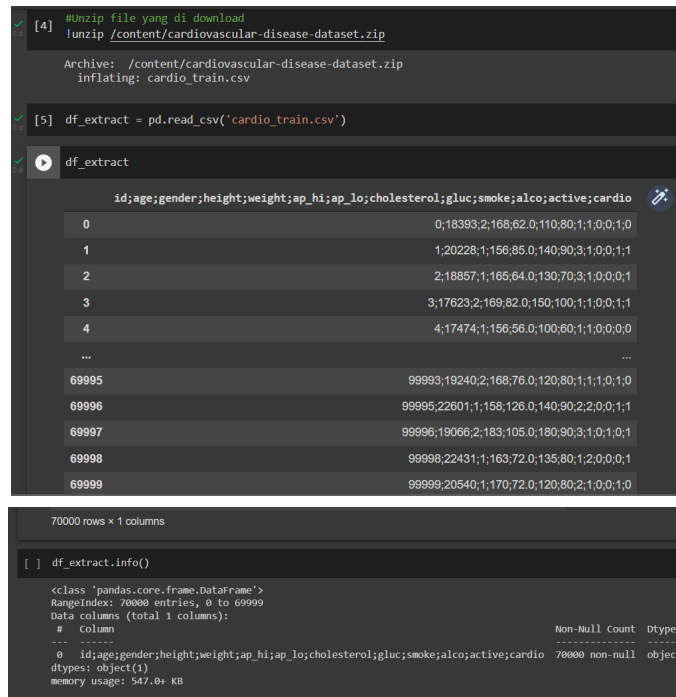
Gambar 16 Select Interpreter

- d. Setelah itu, dapat membuat kodingan untuk extract, transform, load yang akan discheduling. Berikut adalah tahapannya :
1. Extract

Sebelum melakukan extract dapat melakukan import library yang diperlukan terlebih dahulu yaitu numpy, pandas, matplotlib, seaborn, dan sklearn. Setelah itu, dilanjutkan proses extract dataset dari kaggle dengan cara menginstall kaggle terlebih dahulu lalu mengupload kaggle.json dan membuat direktori kaggle. Setelah itu, membuat duplikat file kaggle.json ke dalam direktori kaggle dan mengubah hak akses file kaggle.json. Kemudian, dilakukan download dataset dari kaggle dengan perintah `!kaggle datasets download -d (dataset)` dan diextract dengan perintah `unzip` maka file `cardio_train.csv` berhasil terextract dan terdapat di dalam notebook. Dalam mengextract data ini, kami menggunakan notebook google collaboratory. Setelah itu, membaca file `cardio_train` yang telah berhasil di download sebelumnya serta dimasukkan ke dalam variabel `df_extract`. Untuk melihat info dataset dapat menggunakan perintah `info()`. Dapat diketahui pada kodingan di bawah ini bahwa dataset belum tersplit per kolom dan tipe data adalah masih object untuk semua fitur.



Gambar 17 Tahapan Extract Data



```
[4] #Unzip file yang di download
unzip /content/cardiovascular-disease-dataset.zip

Archive: /content/cardiovascular-disease-dataset.zip
  inflating: cardio_train.csv

[5] df_extract = pd.read_csv('cardio_train.csv')

df_extract
```

	id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio
0	0;18393;2;168;62.0;110;80;1;1;0;0;1;0
1	1;20228;1;156;85.0;140;90;3;1;0;0;1;1
2	2;18857;1;165;64.0;130;70;3;1;0;0;0;1
3	3;17623;2;169;82.0;150;100;1;1;0;0;1;1
4	4;17474;1;156;56.0;100;60;1;1;0;0;0;0
...	...
69995	99993;19240;2;168;76.0;120;80;1;1;1;0;1;0
69996	99995;22601;1;158;126.0;140;90;2;2;0;0;1;1
69997	99996;19066;2;183;105.0;180;90;3;1;0;1;0;1
69998	99998;22431;1;163;72.0;135;80;1;2;0;0;0;1
69999	99999;20540;1;170;72.0;120;80;2;1;0;0;1;0

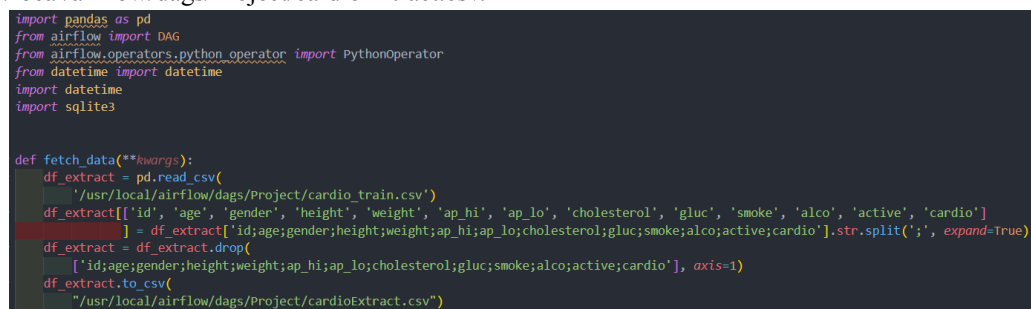
```
70000 rows x 1 columns

[ ] df_extract.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column                                                                                               Non-Null Count  Dtype
---  --
 0   id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio 70000 non-null  object
dtypes: object(1)
memory usage: 547.0+ KB
```

Gambar 18 Tahapan Extract Data dan Melihat Info Data

Setelah data dari kaggle berhasil di extract maka tahap selanjutnya dilakukan pada visual studi code, yaitu melakukan import library yang dibutuhkan yaitu pandas, airflow, PythonOperator, datetime, dan sqlite3. Setelah itu, membuat fungsi fetch data. Fungsi fetch data ini melakukan membaca file cardio_train yang sudah diextract sebelumnya dan didownload serta diletakkan ke path /usr/local/airflow/dags/Project/cardio_train.csv tersebut. Karena tabel dataset sebelumnya belum kesplit jadi dilakukan split semicolon untuk memisahkan per kolom. Setelah per kolom sudah terpisah maka dataset tersebut dibuat ke dalam file csv dengan nama cardioExtract dan diletakkan ke dalam path yang sesuai dengan keinginan, dalam kasus ini saya letakkan pada path /usr/local/airflow/dags/Project/cardioExtract.csv.



```
import pandas as pd
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime
import datetime
import sqlite3

def fetch_data(**kwargs):
    df_extract = pd.read_csv(
        '/usr/local/airflow/dags/Project/cardio_train.csv')
    df_extract[['id', 'age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio']]
    df_extract = df_extract[['id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio']].str.split(';', expand=True)
    df_extract = df_extract.drop(
        ['id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio'], axis=1)
    df_extract.to_csv(
        "/usr/local/airflow/dags/Project/cardioExtract.csv")
```

Gambar 19 Import Library dan Fetch Data

2. Transform

Selanjutnya melakukan transform dengan membuat fungsi transform_data. Fungsi transform_data ini melakukan membaca data dari cardioExtract yang sudah dibuat sebelumnya dan dimasukkan ke dalam variabel df_transform. Lalu membuat fungsi years() yang digunakan untuk mengubah umur yang semula dalam satuan hari menjadi tahun (dengan cara membagi umur dengan /365). Setelah itu dimasukkan ke dalam dataframe kolom 'age' sehingga age sudah dalam satuan tahun. Kemudian, mengubah datatype id, age, gender, height, ap_hi, ap_lo, cholesterol, gluc, smoke, alco, active, cardio menjadi int dan weight menjadi float karena datatype sebelumnya adalah object.

```
def transform_data(**kwargs):
    df_transform = pd.read_csv(
        '/usr/local/airflow/dags/Project/cardioExtract.csv', index_col=[0])

    # Mengubah kolom Age yang memiliki satuan hari menjadi tahun dengan cara membagi /365
    def years():
        umur = df_transform['age'] / 365
        return umur

    # Mengubah tipe data menjadi int dan float
    df_transform['age'] = years()
    df_transform['id'] = df_transform['id'].astype(int)
    df_transform['age'] = df_transform['age'].astype(int)
    df_transform['gender'] = df_transform['gender'].astype(int)
    df_transform['height'] = df_transform['height'].astype(int)
    df_transform['weight'] = df_transform['weight'].astype(float)
    df_transform['ap_hi'] = df_transform['ap_hi'].astype(int)
    df_transform['ap_lo'] = df_transform['ap_lo'].astype(int)
    df_transform['cholesterol'] = df_transform['cholesterol'].astype(int)
    df_transform['gluc'] = df_transform['gluc'].astype(int)
    df_transform['smoke'] = df_transform['smoke'].astype(int)
    df_transform['alco'] = df_transform['alco'].astype(int)
    df_transform['active'] = df_transform['active'].astype(int)
    df_transform['cardio'] = df_transform['cardio'].astype(int)
```

Gambar 20 Mengubah Kolom Age menjadi Satuan Tahun dan Mengubah Tipe Data

Setelah itu, melakukan cleaning data. Pertama melakukan cleaning data dengan menghilangkan bobot dan tinggi badan yang berada di bawah 2,5% atau di atas 97,5% dari kisaran tertentu karena dapat diketahui bahwa tinggi minimum 55 cm, berat minimum adalah 10 kg serta usia minimum 29kg, berat maksimum 250 cm dan berat maksimum 200kg. Data tersebut tidak relevan. Yang kedua, dalam beberapa kasus tekanan diastolik lebih tinggi dari sistolik, yang juga data tersebut adalah salah sehingga data tersebut perlu didrop/dihapus. Setelah itu, membuat fitur baru yaitu BMI (Body Mass Index) atau Indeks Massa Tubuh (IMT) adalah angka yang menjadi penilaian standar untuk menentukan apakah berat badan tergolong normal, atau abnormal. Yang ketiga, menghapus outliers di body mass index. Setelah itu, mengkategorikan apakah dengan BMI tersebut termasuk normal atau abnormal dengan membuat fitur baru yaitu BMI_State. Lalu, dataset yang sudah diproses tersebut dimasukkan ke dalam file csv cardioTransform pada path yang sesuai keinginan, dalam kasus ini path yang saya gunakan yaitu /usr/local/airflow/dags/Project/cardioTransform.csv.

```
# Cleaning Data

# 1. Menghilangkan bobot dan tinggi badan yang berada di bawah 2,5% atau di atas 97,5% dari kisaran tertentu
# Karena dapat diketahui bahwa tinggi minimum 55 cm, berat minimum adalah 10 kg serta usia minimum 29kg, berat
# maksimum 250 cm dan berat maksimum 200kg yang tidak relevan.
df_transform.drop(df_transform[(df_transform['height'] > df_transform['height'].quantile(0.975)) | (
    df_transform['height'] < df_transform['height'].quantile(0.025))].index, inplace=True)
df_transform.drop(df_transform[(df_transform['weight'] > df_transform['weight'].quantile(0.975)) | (
    df_transform['weight'] < df_transform['weight'].quantile(0.025))].index, inplace=True)

# 2. Selain itu, dalam beberapa kasus tekanan diastolik lebih tinggi dari sistolik, yang juga salah.
df_transform.drop(df_transform[(df_transform['ap_hi'] > df_transform['ap_hi'].quantile(0.975)) | (
    df_transform['ap_hi'] < df_transform['ap_hi'].quantile(0.025))].index, inplace=True)
df_transform.drop(df_transform[(df_transform['ap_lo'] > df_transform['ap_lo'].quantile(0.975)) | (
    df_transform['ap_lo'] < df_transform['ap_lo'].quantile(0.025))].index, inplace=True)

# calculate the BMI score
df_transform['BMI'] = ((df_transform['weight']) /
    (df_transform['height']/100)**2)

# 3. Drop outliers in body mass index
df_transform.drop(df_transform.query(
    'BMI >60 or BMI <15').index, axis=0, inplace=True)

# Hasil Transform Data sesudah di cleaning di Dump ke csv
df_transform.to_csv("/usr/local/airflow/dags/Project/cardioTransform.csv")

# categorize normal & abnormal
def bmi_categorize(bmi_score):
    if 18.5 <= bmi_score <= 25:
        return "Normal"
    else:
        return "Abnormal"
df_transform["BMI_State"] = df_transform["BMI"].apply(
    lambda x: bmi_categorize(x))

# Hasil Transform Data sesudah di cleaning di Dump ke csv
df_transform.to_csv("/usr/local/airflow/dags/Project/cardioTransform.csv")
```

Gambar 21 Cleaning Data

Setelah melakukan cleaning data, dapat melakukan data validation dengan cara menginstall great_expectations lalu melakukan import library great expectations. Lalu, melakukan read data csv cardioTransform dan dimasukkan ke dalam variabel df. Setelah itu melakukan Validasi data memastikan apakah kolom id, age, gender, height, weight, ap_hi, ap_lo, cholesterol, smoke, alco, active, cardio, BMI, dan BMI_State ada. Metode expect_column_to_exist () diterapkan ke daftar kolom DataFrame yang ditentukan untuk memastikan bahwa kolom tersebut ada, dan didapatkan bahwa hasilnya yaitu success : true.

```
!pip install great_expectations

import great_expectations as ge

df = ge.read_csv('cardioTransform.csv', index_col=[0])
```

Gambar 22 Install great_expectations dan import library great expectations

```
df.expect_column_to_exist('id')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "id",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}

df.expect_column_to_exist('age')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "age",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}

df.expect_column_to_exist('gender')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "gender",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}
```

Gambar 23 Data Validation Kolom id, age, dan gender

```
df.expect_column_to_exist('height')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "height",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}

df.expect_column_to_exist('weight')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "weight",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}

df.expect_column_to_exist('ap_hi')
{
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  },
  "success": true,
  "expectation_config": {
    "expectation_type": "expect_column_to_exist",
    "meta": {},
    "kwargs": {
      "column": "ap_hi",
      "result_format": "BASIC"
    }
  },
  "meta": {},
  "result": {}
}
```

Gambar 24 Data Validation Kolom height, weight, dan ap_hi

<pre>df.expect_column_to_exist('ap_lo') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "ap_lo", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>	<pre>df.expect_column_to_exist('cholesterol') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "cholesterol", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>	<pre>df.expect_column_to_exist('smoke') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "smoke", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>
--	--	--

Gambar 25 Data Validation Kolom ap_lo, cholesterol, dan smoke

<pre>df.expect_column_to_exist('alco') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "alco", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>	<pre>df.expect_column_to_exist('active') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "active", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>	<pre>df.expect_column_to_exist('cardio') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "cardio", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>
--	--	--

Gambar 26 Data Validation Kolom alco, active, dan cardio

<pre>df.expect_column_to_exist('BMI') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "BMI", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>	<pre>df.expect_column_to_exist('BMI_State') { "exception_info": { "raised_exception": false, "exception_traceback": null, "exception_message": null }, "success": true, "expectation_config": { "expectation_type": "expect_column_to_exist", "meta": {}, "kwargs": { "column": "BMI_State", "result_format": "BASIC" } }, "meta": {}, "result": {} }</pre>
--	--

Gambar 27 Data Validation Kolom BMI dan BMI_State

3. Load ke Database

Setelah melakukan fetch data dan transform data maka langkah selanjutnya yaitu load data dengan membuat fungsi load_db. Fungsi load_db ini digunakan untuk memasukkan data yang sudah ditransform dari file cardioTransform.csv dan telah diproses sebelumnya ke dalam database ProjectAkhirDE.db pada tabel Cardio. Tabel Cardio ini memiliki beberapa variabel yaitu id, age, gender, height, weight, ap_hi, ap_lo, cholesterol, smoke, alco, active, cardio, BMI, dan BMI_State.

```
def load_db(**kwargs):
    df_load = pd.read_csv(
        '/usr/local/airflow/dags/Project/cardioTransform.csv', index_col=[0]
    )
    path_db = "/usr/local/airflow/dags/Project/ProjectAkhirDE.db"
    conn = sqlite3.connect(path_db)
    cur = conn.cursor()
    cur.execute("""
        CREATE TABLE IF NOT EXISTS Cardio(
            id INT PRIMARY KEY NOT NULL,
            age INT NOT NULL,
            gender INT NOT NULL,
            height INT NOT NULL,
            weight FLOAT NOT NULL,
            ap_hi INT NOT NULL,
            ap_lo INT NOT NULL,
            cholesterol INT NOT NULL,
            gluc INT NOT NULL,
            smoke INT NOT NULL,
            alco INT NOT NULL,
            active INT NOT NULL,
            cardio INT NOT NULL,
            BMI FLOAT NOT NULL,
            BMI_State TEXT NOT NULL
        );
    """)
    # memasukan data ke database
    df_load.to_sql("Cardio", conn, if_exists='replace', index=False)
    conn.commit()
```

Gambar 28 Load ke Database

4. Membuat DAG

Pada tahap membuat DAG ini, DAG memiliki id yaitu Project_AkhirDE, terdapat start_date atau tanggal mulai dan shedule_interval yaitu @daily yang berarti akan dijalankan setiap hari. Di dalam DAG terdapat beberapa task yaitu fetch_dataCardio, transform_dataCardio dan load_data. Untuk fetch_dataCardio terdapat task_id yaitu fetch_data dan akan menjalankan fungsi fetch_data. Pada task transform_dataCardio terdapat id tasknya yaitu tranform_data yang akan menjalankan fungsi transform_data. Lalu, task load data terdapat id task yaitu load_data dan akan menjalankan fungsi load_db. Kemudian, untuk alur DAG sendiri yaitu dimulai dari fetch_dataCardio kemudian menuju transform_dataCardio dan yang terakhir yaitu load_data.

```
with DAG(
    dag_id="Project_AkhirDE",
    start_date=datetime.datetime(2021, 12, 7),
    schedule_interval="@daily",
    catchup=False
) as dag:

    fetch_dataCardio = PythonOperator(
        task_id='fetch_data',
        python_callable=fetch_data,
    )

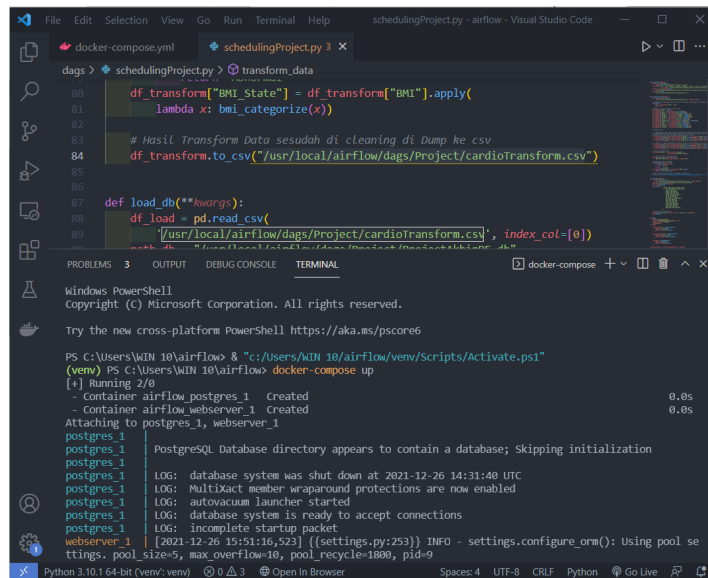
    transform_dataCardio = PythonOperator(
        task_id='transform_data',
        python_callable=transform_data,
    )

    load_data = PythonOperator(
        task_id='load_data',
        python_callable=load_db,
    )

    fetch_dataCardio >> transform_dataCardio >> load_data
```

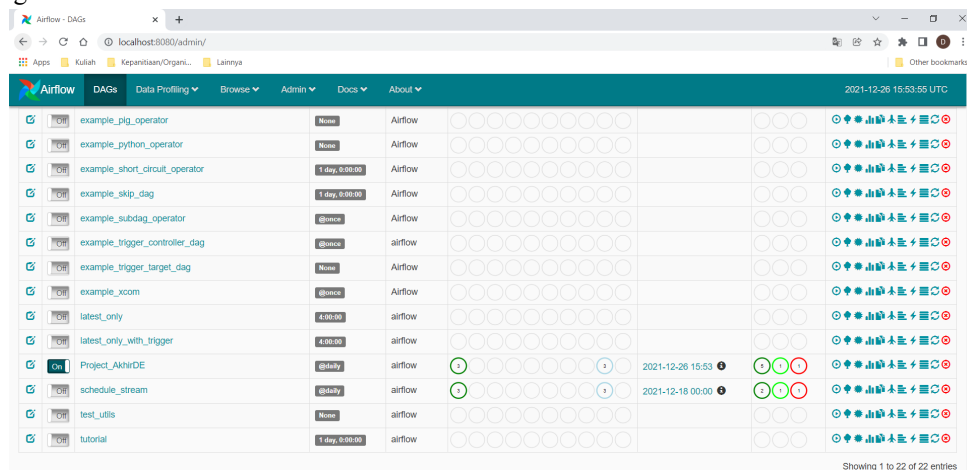
Gambar 29 Membuat DAG

Setelah proses ETL dibuat maka dapat dijalankan dengan perintah docker-compose up pada terminal visual studio code untuk menjalankan scheduling seperti gambar di bawah ini. Sebelumnya, pastikan terlebih dahulu bahwa aplikasi docker sudah dibuka.



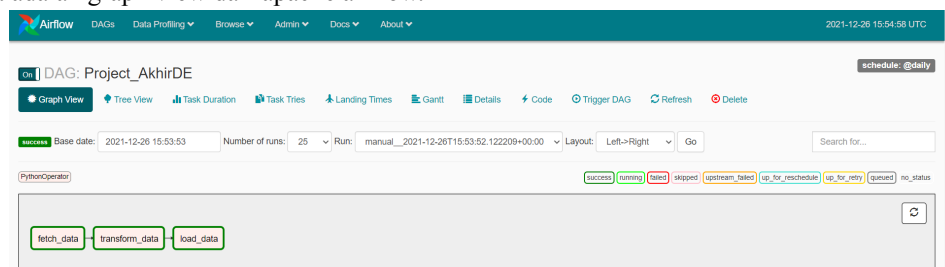
Gambar 30 Menjalankan Perintah docker-compose up

Setelah berjalan, maka dapat membuka <http://localhost:8080/> dan dapat terlihat tampilan apache airflow seperti gambar di bawah ini.



Gambar 31 Tampilan Apache Airflow

Berikut adalah graph view dari apache airflow.



Gambar 32 Graph View

3.4. EDA (Exploratory Data Analysis)

- Describe
Menggunakan fungsi describe untuk menampilkan statistik dari data seperti minimum, maksimum, rata-rata, dan lain-lain.

We can use describe() to display sample statistics such as min, max, mean, std for each attribute:

```

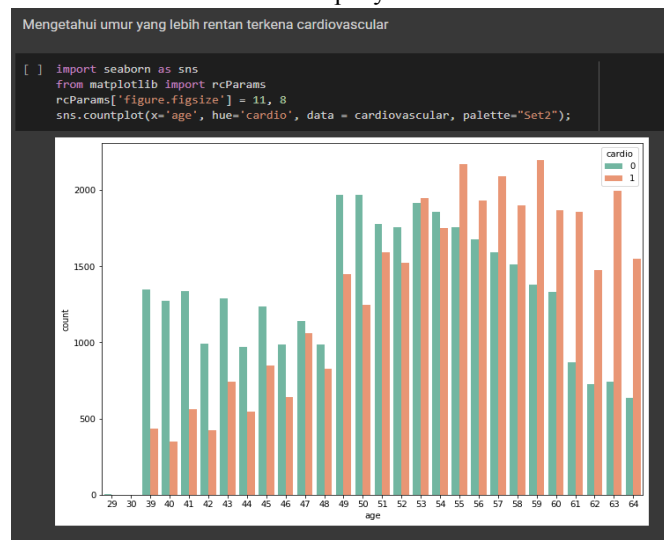
# Statistic Overview
cardiovascular.describe()

```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	49972.419900	52.840671	1.349571	164.359229	74.205690	128.817286	96.630414	1.365871	1.226457	0.088129	0.053771	0.803729	0.499700
std	28851.302333	6.766774	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.225568	0.397179	0.500003
min	0.000000	29.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	25006.750000	48.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
50%	50001.500000	53.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
75%	74889.250000	58.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000	1.000000	1.000000
max	99999.000000	64.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000

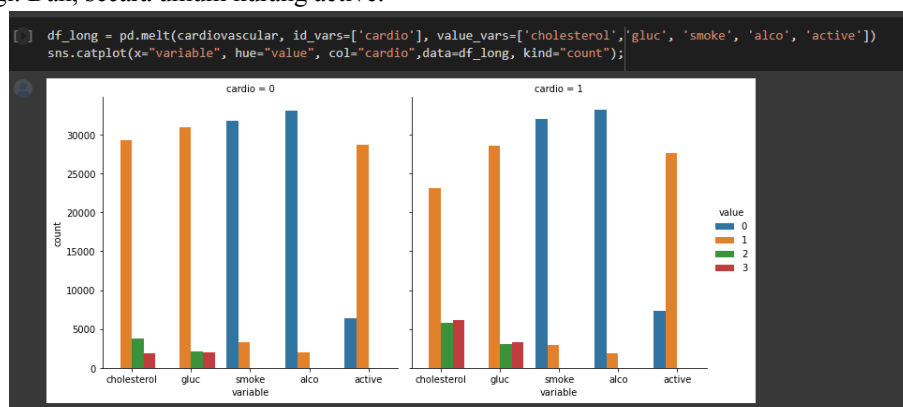
Gambar 33 Menampilkan Statistik dari Data

- Mengetahui umur yang rentan terhadap cardiovascular
Orang berusia 55 tahun keatas lebih rentan terkena penyakit cardiovascular



Gambar 34 Menampilkan Visualisasi Umur yang Rentan terhadap Cardiovascular

- Bivariate Analysis
Berfungsi untuk membagi variabel kategorik berdasarkan kelas target
 - Pasien dengan penyakit cardiovascular memiliki kadar kolesterol dan glukosa darah yang lebih tinggi. Dan, secara umum kurang active.



Gambar 35 Visualisasi Hubungan Cardiovascular dengan kolesterol, glucose, dan active

- Pada kolom gender terdapat 2 kategori yaitu 0 dan 1. Untuk mengetahui maksud kategori tersebut kita hitung rata-rata tinggi badan per jenis kelamin. dan mami mengasumsikan bahwa rata-rata pria lebih tinggi daripada wanita.
 - Tinggi rata-rata untuk jenis kelamin "2" lebih besar daripada jenis kelamin "1", oleh karena itu "1" berarti wanita.

```
cardiovascular.groupby('gender')['height'].mean()
gender
1    161.355612
2    169.947895
Name: height, dtype: float64
```

Gambar 36 Rata-rata gender untuk mengetahui kategori wanita atau pria dari gender tersebut

- Jumlah masing-masing pria dan wanita dalam dataset

```
[ ] cardiovascular['gender'].value_counts()
1    45530
2    24470
Name: gender, dtype: int64
```

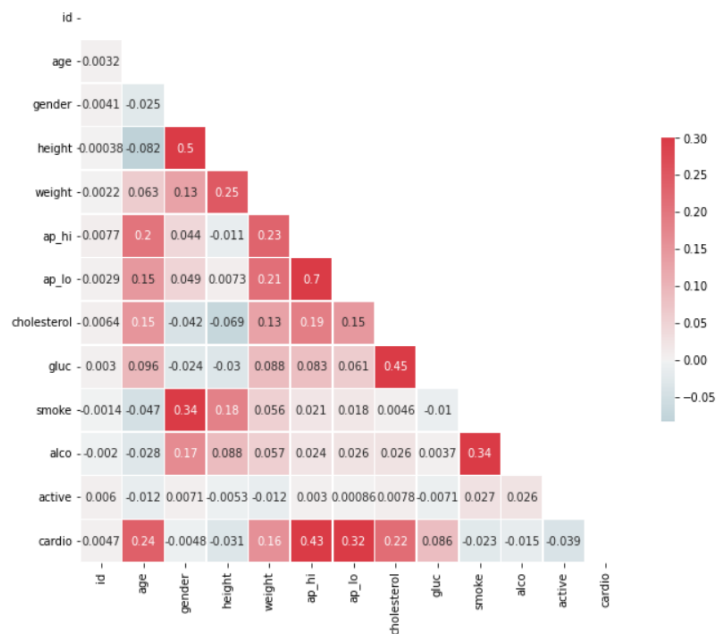
Gambar 37 Jumlah Pria dan Wanita dalam Dataset

- Bagaimana kelas target didistribusikan di antara pria dan wanita.

```
pd.crosstab(cardiovascular['cardio'],cardiovascular['gender'],normalize=True)
gender    1    2
cardio
0    0.327343  0.172957
1    0.323086  0.176614
```

Gambar 38 Distribusi Kelas Target antara pria dan Wanita

- Usia dan kolesterol memiliki pengaruh yang signifikan, tetapi tidak berkorelasi terlalu tinggi dengan kelas target



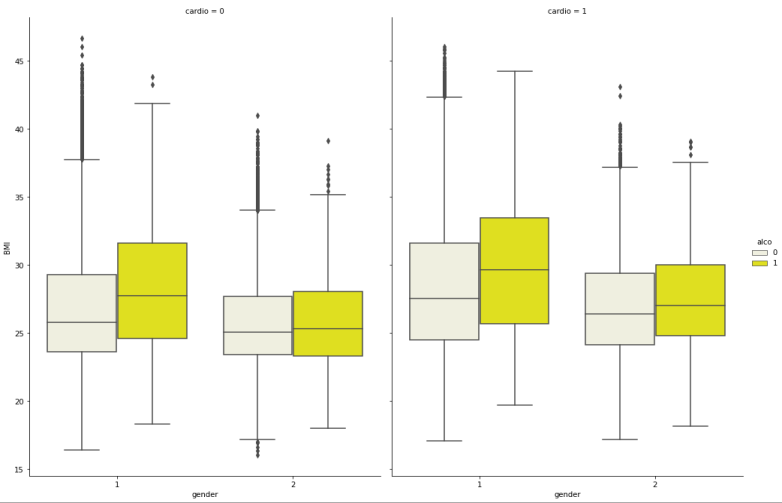
Gambar 39 Visualisasi Korelasi dari Setiap Fitur

- BMI

Mari kita buat fitur baru - Body Mass Index (BMI) :

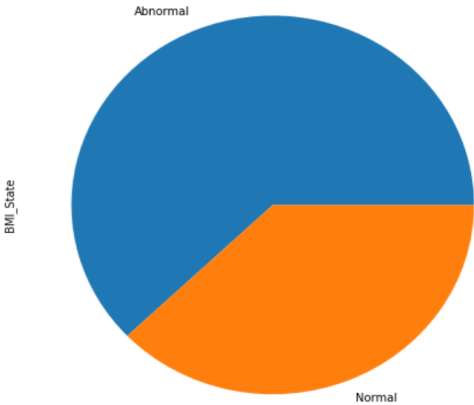
$BMI = \frac{mass(kg)}{height^2(m)}$, dan bandingkan rata-rata BMI orang sehat dengan rata-rata BMI orang sakit. Nilai BMI normal dikatakan dari 18,5 hingga 25.

- Wanita peminum alkohol memiliki resiko CVD lebih tinggi daripada pria peminum alkohol berdasarkan BMI mereka.



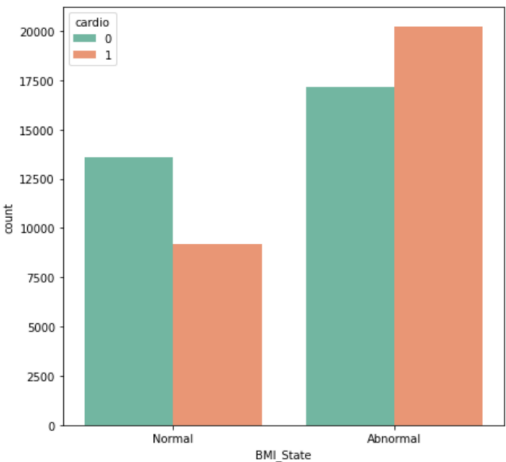
Gambar 40 Visualisasi Hubungan Alcohol dengan BMI dan Cardiovascular

b. Sebagian besar pasien memiliki BMI tidak normal.



Gambar 41 Visualisasi BMI Normal dan Abnormal dalam dataset

c. Dapat diketahui bahwa orang yang memiliki BMI normal kurang rentan terhadap penyakit cardiovascular dan orang dengan BMI abnormal mengalami peningkatan penyakit cardiovascular.



Gambar 42 Hubungan BMI_State (Normal dan Abnormal) dengan Cardiovascular

4. MODELING DAN DASHBOARD

Tahap selanjutnya adalah melakukan modelling sekaligus pengujian dan juga pembuatan dashboard. Kami menggunakan modelling klasifikasi yaitu Random Forest. Algoritma ini digunakan pada klasifikasi data dalam jumlah yang besar. Klasifikasi random forest dilakukan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki. Penggunaan pohon (tree) yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan random forest diambil berdasarkan hasil voting dari tree yang terbentuk.

Selanjutnya adalah pembuatan dashboard dengan menggunakan tool Google Data Studio. Tool ini adalah data visualization tool yang tidak hanya merupakan salah satu yang paling unggul saat ini, tetapi juga gratis. Dengan menggunakan Google Data Studio, kita bisa membuat interaktif dashboard serta menyusun laporan data yang tampil keren serta mudah dimengerti. Selain itu, Google Data Studio juga sangat mudah dibagikan.

4.1. Modelling dan Pengujian Algoritma Random Forest

4.1.1 Modelling Algoritma Random Forest

Sebelum dilakukannya pemodelan maka terlebih dahulu dataset tersebut diubah bentuknya dalam bentuk array. sehingga bisa dilakukan membuat variabel x dan juga variabel y.

```
[ ] cardioEDCleaning.columns

Index(['Unnamed: 0', 'id', 'age', 'gender', 'height', 'weight', 'ap_hi',
      'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio',
      'BMI', 'BMI_State'],
      dtype='object')

[ ] cvd = cardioEDCleaning[['age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol',
                           'gluc', 'smoke', 'alco', 'active', 'BMI', 'cardio']]

[ ] X = cvd.loc[:, cvd.columns != 'cardio']
    y = cvd.loc[:, cvd.columns == 'cardio']
```

Gambar 43 Mengetahui Kolom dan menggunakan kolom cardio sebagai data target

Selanjutnya adalah membagi data menjadi data test dan data train dengan rasio 80:20

```
import pandas as pd
import numpy as np
import sklearn as sk
from sklearn.model_selection import train_test_split
import sklearn.ensemble as skens

# split data into test and train set (Rasio : 80% : 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
columns = X_train.columns

[ ] X_train.shape
(48113, 12)

[ ] X_test.shape
(12029, 12)

[ ] y_train.shape
(48113, 1)

[ ] y_test.shape
(12029, 1)
```

Gambar 44 Melakukan Split Data antara data train dan data test

Selanjutnya kita melakukan modelling dengan menggunakan library sklearn

```
# build 10 random trees
rf_model = skens.RandomForestClassifier(n_estimators = 10, oob_score=True, criterion = 'entropy')
rf_model.fit(X_train, y_train)

RandomForestClassifier(criterion='entropy', n_estimators=10, oob_score=True)
```

Gambar 45 Melakukan build 10 random trees

4.1.2 Pengujian Algoritma Random Forest

Untuk menguji model tersebut, diperlukan sebuah penilaian yang berupa tingkat akurasi, tingkat presisi, recall dan juga f1-score. Pengujian ini dilakukan dengan menggunakan classification report. Hasil dari Pengujian Algoritma Random Forest adalah sebagai berikut

```
from sklearn.metrics import classification_report
print(classification_report(y_test['cardio'], y_test['predicted_rf']))
```

	precision	recall	f1-score	support
0	0.68	0.74	0.71	6164
1	0.70	0.64	0.67	5865
accuracy			0.69	12029
macro avg	0.69	0.69	0.69	12029
weighted avg	0.69	0.69	0.69	12029

```
[ ] from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test['cardio'], y_test['predicted_rf'])
cm

array([[4551, 1613],
       [2136, 3729]])
```

Gambar 46 Hasil Classification Report Random Forest Classifier

Pada gambar 46 dilihat bahwa Algoritma Random Forest memiliki akurasi yang cukup baik yakni senilai 69% dengan nilai presisi sebanyak 70%, recall sebanyak 64% dan f1-score sebanyak 0,67.

4.2. Dashboard

Google Data Studio bisa diakses melalui link [Dashboarding & Data Visualization Tools - Google Data Studio](#). Secara umum, untuk memulai kamu hanya perlu mengklik menu Reports dan Data Sources untuk menghubungkan laporan dengan sumber data pilihan.

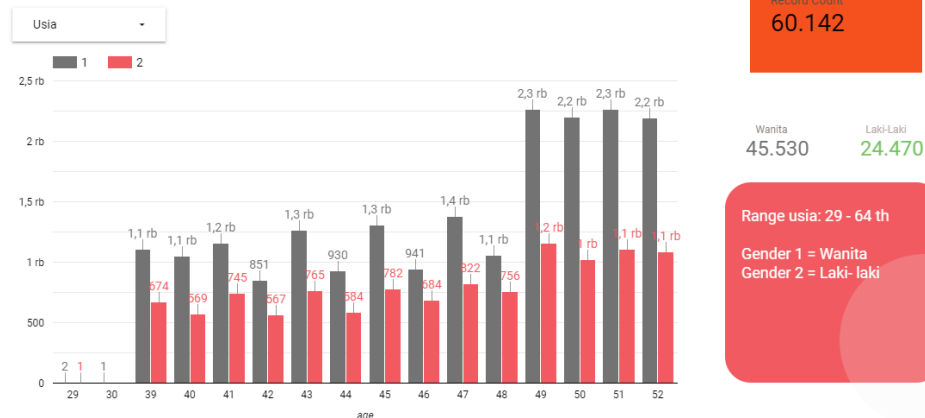
Lalu, tinggal klik tanda panah yang ada di pojok kanan bawah halaman untuk mulai mengerjakan laporan baru di Google Data Studio. Dashboard cardiovascular ini dapat diakses pada link berikut <https://datastudio.google.com/reporting/76b49300-5f0e-4583-9ac3-f4554cd241c4>.

Dashboard Cardiovascular

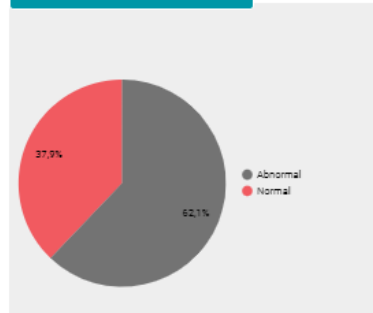


Data ini merupakan data yang diambil dari pemeriksaan medis, informasi faktual, dan informasi yang didapat dari pasien. Data ini digunakan untuk melihat hubungan antara informasi yang telah diberikan dengan cardiovascular

HASIL EDA

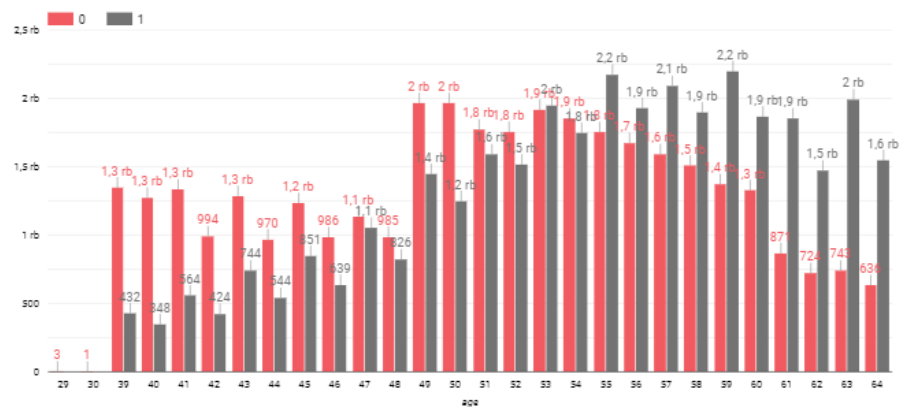
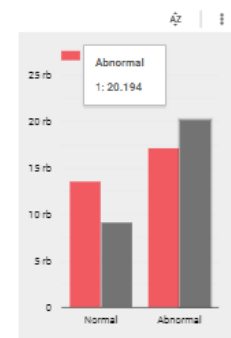


BMI State



Body Mass Index (BMI) adalah metode penghitungan mudah yang dapat memberikan informasi dasar terhadap masalah berat badan. BMI ini didapatkan melalui perhitungan: $BMI = \text{mass}(\text{kg}) / \text{height}^2(\text{m})$

Rata-rata Nilai BMI normal dikatakan dari 18,5 hingga 25.



Dapat diamati bahwa orang yang berusia di atas 55 tahun lebih rentan terkena CVD.

HASIL MODELING				
Accuracy with Random Forrest Classifier: 68.92%				
Classification Report				
	precision	recall	f1-score	support
0	0.68	0.74	0.71	6164
1	0.70	0.64	0.67	5865
accuracy			0.69	12029
macro avg	0.69	0.69	0.69	12029
weighted avg	0.69	0.69	0.69	12029

Gambar 47 Hasil Dashboard Cardiovascular

KESIMPULAN



Kardiovaskular merupakan masalah pada jantung atau pembuluh darah dimana terjadi penyempitan pada pembuluh darah sehingga menimbulkan berbagai penyakit mematikan. Penderita kardiovaskular harus cepat ditangani segera karena penyakit ini melibatkan jantung sebagai organ vital yang jika bermasalah akan menimbulkan penyakit mematikan. Meskipun tergolong umum, penyakit ini merupakan penyakit penyebab kematian terbesar di dunia. Selain itu, penyakit kardiovaskular ini masih belum bisa disembuhkan secara total.


Pada pengujian yang telah dilakukan diatas terdapat beberapa faktor yang dapat mempengaruhi seseorang lebih rentan terkena penyakit kardiovaskular ini antara lain faktor usia dan jumlah kolesterol dalam darah. Tekanan darah juga berperan pada penyakit kardiovaskular ini, tekanan darah yang tinggi memiliki resiko yang lebih tinggi pula dan seringkali berujung pada penyakit stroke. Selain itu, orang yang memiliki *Body Mass Index* (BMI) yang diluar normal juga memiliki resiko tinggi terkena penyakit kardiovaskular.

Hasil dari perhitungan menggunakan algoritma klasifikasi *Random Forest Classifier* ini menghasilkan akurasi yang cukup baik yaitu sebesar 69 persen, dengan nilai presisi sebanyak 70 persen, *recall* sebanyak 64 persen dan *f1-score* sebesar 0.67.

DAFTAR PUSTAKA

- Ortiz, L.G., Rodriguez, J.L.R., Simon, S.M., Guillaumet, J., Marti, R., Conde, C.A., Sanchez, E.R., Fernandez, J.A.M., Blanes, R.R., Marcos, M.A.G. (2016). Vascular Structure and Function and Their Relationship with Health-related Quality of Life in the MARK study. *BMC Cardiovascular Disorders*. 16 (95):1-20.
- Plale, B., & Kouper, I. (2017). The centrality of data: data lifecycle and data pipelines. In *Data analytics for intelligent transportation systems* (pp. 91-111). Elsevier.
- Krzyzanowski, P., 2015. Paul Krzyzanowski's Site. [Online] Tersedia di: <https://www.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html> [Diakses 26 Desember 2021].

	<p>Dyah Ayu Wulandari</p> <p>Peran dalam project (ETL dan Scheduling, EDA), Laporan (Metodologi Penelitian : Instalasi dan konfigurasi Docker, Persiapan Data, ETL dan Scheduling).</p> <p>No whatsapp: 082229487114</p>
	<p>Peran dalam project : EDA, Model Evaluation, Laporan : EDA, Modelling, Dashboard</p> <p>No whatsapp: 081232202236</p>

	<p>Anggasta Aji Azhari</p> <p>Peran dalam project: Dashboard, Modelling, Laporan (Abstrak, Pendahuluan, tinjauan pustaka)</p> <p>No whatsapp: 085536563254</p>
---	--