

PROPOSAL FINAL PROJECT
DATA & AI ANALYTICS GRADXPERT
BITHEALTH
STUDY CASE BUSINESS



OLEH
KELOMPOK 2:
Alwan Abdurrahman
Berliana Fitria Dewi
Muhammad Fakhrian Abimanyu
Ryandino
Satriya Fauzan Adhim

DAFTAR ISI

DAFTAR ISI.....	II
BAB I LATAR BELAKANG	1
RUMUSAN MASALAH	3
TUJUAN PENELITIAN	4
BAB II DATASET.....	5
BAB III METODOLOGI.....	7
A. AWS Setup.....	8
B. Data Normalization	13
C. Data Transformation	20
D. Machine Learning	28
E. Model Deployment	50
F. Data Visualization.....	56
BAB IV HASIL DAN PEMBAHASAN	64
BUSINESS INSIGHTS	69
MACHINE LEARNING	75
BAB V KESIMPULAN.....	80

BAB I

LATAR BELAKANG

Transformasi digital telah berperan penting dalam berbagai industri, tidak terkecuali industri kesehatan. Banyak rumah sakit berupaya melakukan transformasi digital dengan melakukan optimalisasi teknologi dengan harapan data-data yang ada di rumah sakit dapat terekam dengan baik sekaligus meningkatkan layanan yang berkualitas bagi pasien.

Data-data digital di rumah sakit didapatkan dari perangkat yang terhubung satu sama lain antar bagian dalam sebuah rumah sakit. Kumpulan data tersebut didapatkan dari perjalanan paling awal sebuah pasien di rumah sakit dimulai dari administrasi, hasil pemeriksaan, pengawasan dalam rawat inap maupun rawat jalan sampai ke proses paling akhir yaitu pembayaran dan pasien kembali pulih.

Pemanfaatan *Analytic Insight* terhadap data atau *big data* memiliki potensi besar dalam bisnis di mana semua data yang ada terhubung satu sama lain dan dapat memberikan *insight* serta manfaat seperti *health tracking*, menganalisa dan memprediksi, menyediakan layanan yang lebih sesuai dengan masing-masing pasien, diagnosa yang lebih efektif, dan kemajuan dalam *telemedicine* serta masih banyak manfaat lainnya.

Big data sendiri pada rumah sakit terdiri dari koordinasi pelayanan kesehatan, pemberdayaan pasien, obat yang disesuaikan dengan kebutuhan pasien, dan reformasi pembayaran. Implementasi pemanfaatan *big data* ini menghasilkan kemudahan baik bagi operasional rumah sakit maupun pengalaman pasien rumah sakit.

Data kesehatan sangat dicari seperti ladang emas yang baru, di mana data-data yang ada dan telah diolah dengan baik (*data ingestion*, *data cleansing*, *data mining*, *reporting & visualization* dari data mentah) dapat menjadi informasi yang dapat digunakan oleh pengguna informasi dalam pengambilan keputusan melalui proses

digitalisasi data. Hal ini terwujud dengan adanya *data warehouse* yang merupakan jenis sistem manajemen data yang dirancang untuk mengaktifkan dan mendukung aktivitas *business intelligence* (BI), terutama analitik.

Manfaat yang didapatkan dalam menggunakan *data warehouse* adalah mengintegrasikan seluruh data yang ada dan juga melakukan *queries* serta analisis yang seringkali berisi data historis dalam jumlah besar. Selain itu, pemusatan data ini dapat memberikan akses ke berbagai departemen di sebuah rumah sakit dengan mudah sehingga rumah sakit memiliki satu sumber terpercaya dan dapat memaksimalkan nilai dari data tersebut.

Digitalisasi yang terjadi ini mendukung tercapainya efisiensi pelayanan yang tidak perlu berulang-ulang dan meminimalisir risiko terhadap pasien itu sendiri. Pada akhirnya, jika diterapkan prinsip dasar teknologi kesehatan pada setiap fasilitas kesehatan seperti rumah sakit, nantinya manfaat dari digitalisasi ini tidak hanya akan dirasakan oleh masyarakat yang menerima layanan, tetapi akan dirasakan juga oleh pihak rumah sakit sebagai pihak yang memberikan pelayanan maupun regulasi secara umum.

Pengolahan data yang baik berguna untuk memaksimalkan sistem *analytics* dan *maturity* suatu model bisnis yang dimiliki, dalam hal ini yaitu rumah sakit. Memperbaiki di level prosedur dan diagnostik serta layanan kritis di rumah sakit sampai melakukan prediksi baik pasien masuk ataupun keluar, dan hal lain yang mempengaruhi sistem keputusan. Melalui hal ini diharapkan dapat tercapainya sistem informasi digital yang cerdas dan terpadu (*Smart Hospital*).

RUMUSAN MASALAH

Proposal ini bertujuan agar data keuntungan beberapa cabang rumah sakit, jumlah pasien yang berkunjung, dan tingkat kepuasan pasien dapat digunakan untuk menentukan indeks prestasi rumah sakit sehingga pihak manajemen rumah sakit dapat mengidentifikasi cabang dengan indeks prestasi di atas atau di bawah batas, melakukan evaluasi yang diperlukan terhadap kinerja rumah sakit, dan meningkatkan tingkat kepuasan pasien di setiap cabang rumah sakit.

TUJUAN PENELITIAN

Tujuan penelitian adalah:

1. Menentukan cabang Rumah Sakit Mulia yang memiliki *index performance* di atas atau di bawah rata-rata 70 %.
2. Memberikan *insight* untuk cabang Rumah Sakit Mulia sesuai dengan *index performancenya*.
3. Visualisasi hasil nilai per cabang Rumah Sakit Mulia.
4. Visualisasi *insight* untuk setiap cabang Rumah Sakit Mulia.
5. Membuat *machine learning* untuk prediksi dari *score index performance* di masa yang akan datang.

BAB II

DATASET

Dataset yang kami gunakan ialah [Hospital data.csv](#). Dataset ini berisi tentang *historical data* pasien rumah sakit dengan periode 2020 – 2024 dan terdiri dari 9473 transaksi yang tercatat. Berikut merupakan kolom-kolom yang terdapat dari dataset yang kami gunakan beserta deskripsi dari tiap-tiap kolom.

Column	Description			
ID	ID penjualan (<i>Invoice</i>)			
Date IN	Tanggal pasien masuk rumah sakit			
Date OUT	Tanggal pasien keluar rumah sakit			
Branch	Cabang rumah sakit: - RSMA : Rumah Sakit Mulia Anggrek - RSMD : Rumah Sakit Mulia Duri - RSMS : Rumah Sakit Mulia Simatupang			
Name	Nama pasien			
Age	Usia pasien			
Gender	Gender pasien			
Hospital Care	Tipe perawatan pasien			
Room	Tipe kamar: - VIP : Rp. 300.000/malam - Kelas 1 : Rp. 250.000/malam - Kelas 2 : Rp. 200.000/malam - Kelas 3 : Rp. 150.000/malam			
Doctor	Tipe doktor: - Bedah : Rp. 300.000/kunjungan - Gigi : Rp. 300.000/kunjungan - Kandungan : Rp. 300.000/kunjungan - Penyakit dalam : Rp. 300.000/kunjungan - Umum : Rp. 200.000/kunjungan			
Surgery	Tipe tindakan/operasi: - Kecil : Rp. 4.000.000 - Besar : Rp. 8.000.000 - Kusus : Rp. 15.000.000			
Lab	Tipe uji laboratorium: - Hematologi : Rp. 90.000 - Kimia Darah : Rp. 195.000 - Rontgen : Rp. 150.000 - Serologi : Rp. 200.000 - Urinalisa : Rp. 80.000			
Drug Types	Tipe obat: - Antibiotik : Rp. 75.000 - Pereda nyeri : Rp. 50.000			

	- Umum : Rp. 40.000 - Vitamin : Rp. 110.000
Drug Brands	Merek-merek obat berdasarkan tipenya
Drug Quantity	Jumlah strip/botol yang perlu dikonsumsi oleh pasien selama masa penyembuhan
Food	Harga makanan untuk pasien rawat inap setiap hari (harga sudah termasuk sarapan, makan siang, dan malam)
Admin	Biaya administrasi rumah sakit (<i>fix price</i> /pasien)
COGS	Cost of Good Sold Rumah sakit untuk setiap pasien
Payment	Tipe pembayaran pasien
Review	Review dari pasien

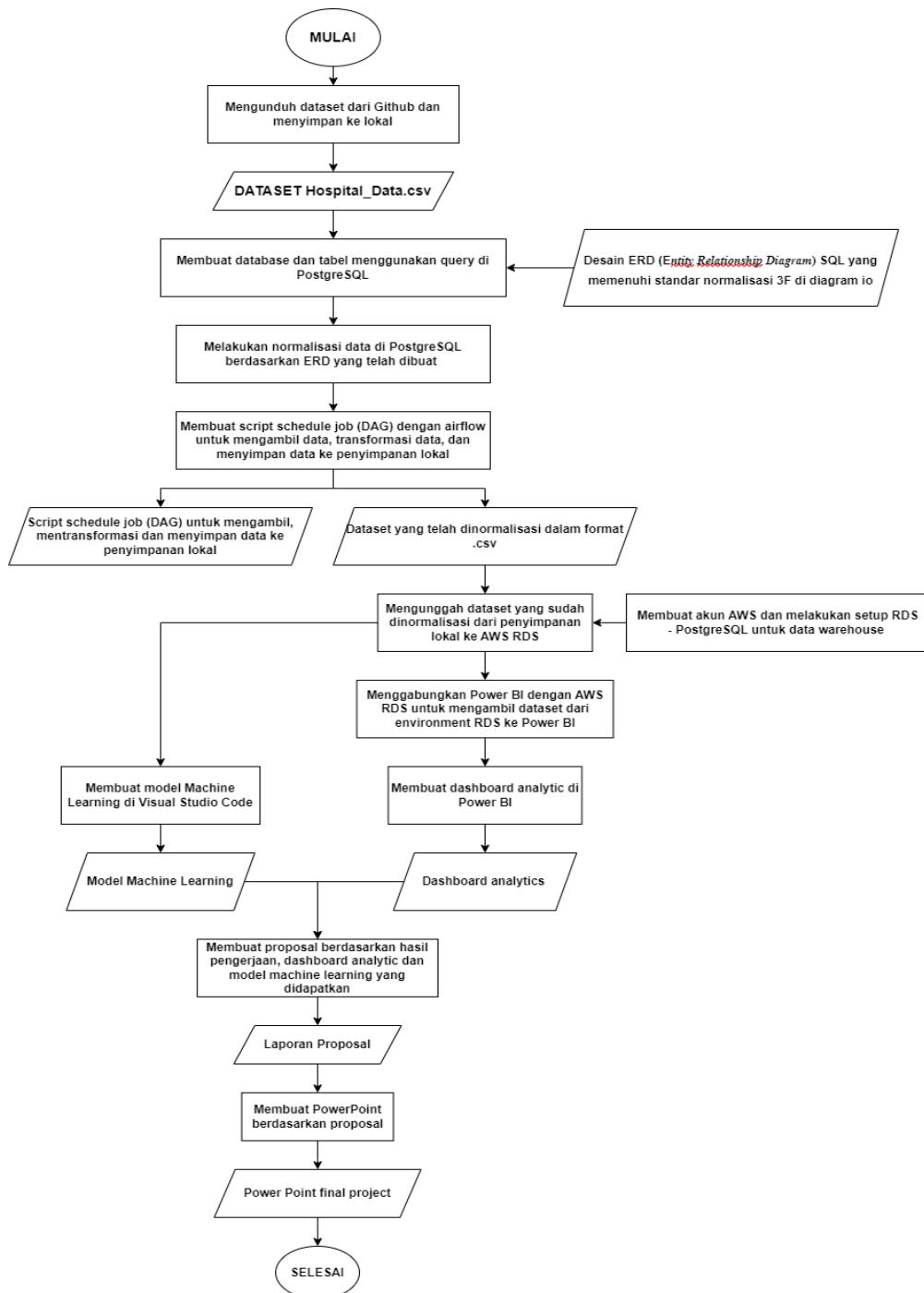
Asumsi yang kami gunakan dalam mengolah *dataset* ini ialah:

- Dokter setiap hari melakukan 1 kali kunjungan untuk pasien rawat inap.
- Pasien Rawat Jalan dianggap bertemu dengan dokter 1 kali (kunjungan).
- Tindakan (operasi) hanya dilakukan 1 kali selama pasien rawat inap.
Kecuali ada pasien yang sama datang lagi di hari yang berbeda dan statusnya rawat inap kembali.
- Seluruh hasil test laboratorium pada dataset kali ini adalah positif dan hanya dilakukan 1x baik untuk pasien rawat jalan, maupun rawat inap.
- Setiap pasien hanya membutuhkan 1 jenis obat per kasus penyakit.
- Kategori obat umum maksudnya adalah paracetamol.
- Setiap merek obat yang kategorinya sama, memiliki harga yang sama.
- Setiap pasien rawat inap, membutuhkan infus dengan harga Rp. 165.000/hari.
- Pemberian obat untuk pasien rawat inap dilakukan hanya sekali selama durasi rawat inap.
- Durasi rawat dihitung berdasarkan jumlah hari, bukan jumlah malam.

BAB III

METODOLOGI

Diagram alir pengolahan data secara keseluruhan pada penelitian ini adalah sebagai berikut:



Berikut merupakan langkah-langkah yang kami lakukan dalam melakukan pengolahan data dari penelitian kami:

A. AWS Setup

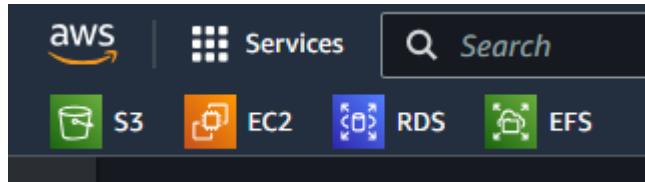
Database pada proyek ini menggunakan *relational database* yang disediakan oleh Amazon Web Service yaitu AWS *Amazon Relational Database Service* (RDS). Adapun beberapa alasan kenapa Amazon RDS dipilih sebagai tempat penyimpanan data, yaitu: (1) memberikan performa yang tinggi untuk proses data masuk dan keluar *database*, (2) tidak diperlukan pengaturan infrastruktur yang rumit seperti penggunaan *database* di *local*. Berikut merupakan langkah-langkah dalam men-setup AWS *Amazon Relational Database Service* (RDS):

- 1) *Login* melalui SSO AWS Portal melalui link di bawah ini dan pilih *role AdministratorAccess*.

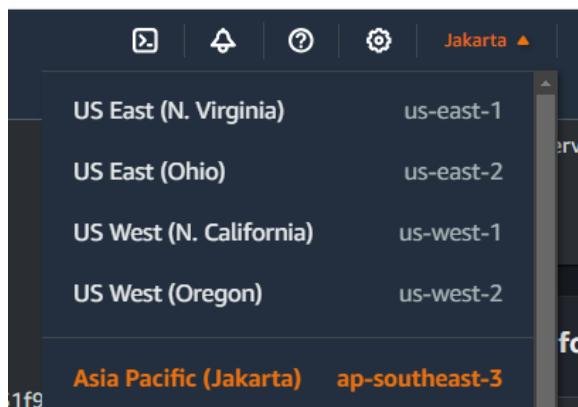
<https://d-966757453d.awsapps.com/start/#/?tab=accounts>

The screenshot shows the AWS Access Portal interface. At the top, there are two tabs: "Accounts" (which is highlighted in blue) and "Applications". Below the tabs, the title "AWS accounts (1)" is displayed. A search bar with the placeholder "Filter accounts by name, ID, or email address" is present. Under the search bar, a single account entry is shown: "satriya.fauzan" with a small orange cube icon. To the right of the account name, it shows the ID "533266956028 | satriya.fauzan" and a redacted email address. Below the account entry, there are two links: "AdministratorAccess | Access keys" and "DatabaseAdministrator | Access keys".

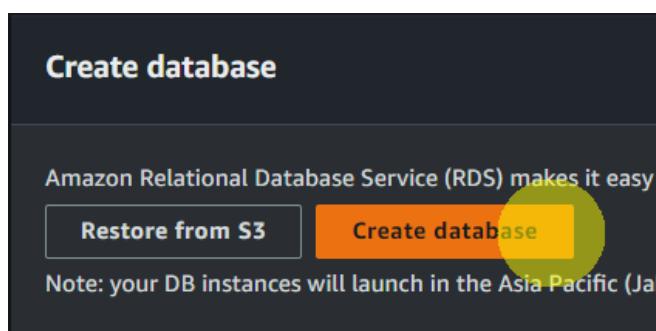
- 2) Buka halaman Amazon RDS pada pojok kanan atas



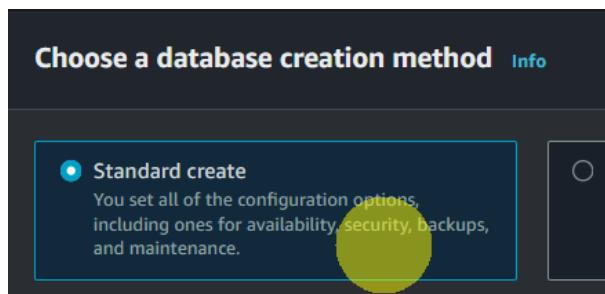
- 3) Sebelum pembuatan *database instance* pastikan region yang terpilih adalah *region* Jakarta (ap-southeast-3)



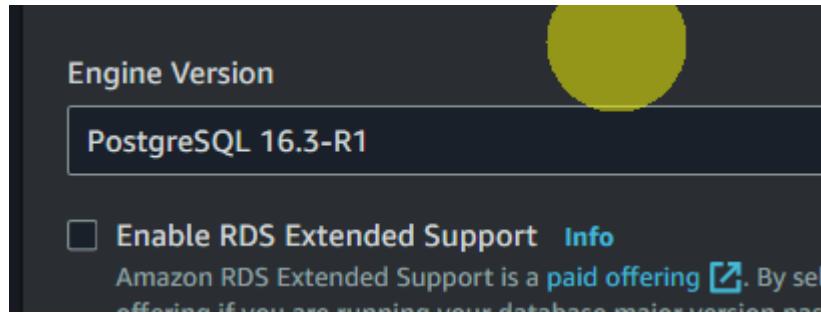
- 4) Scroll ke bawah sampai menemukan tombol “*create database*”



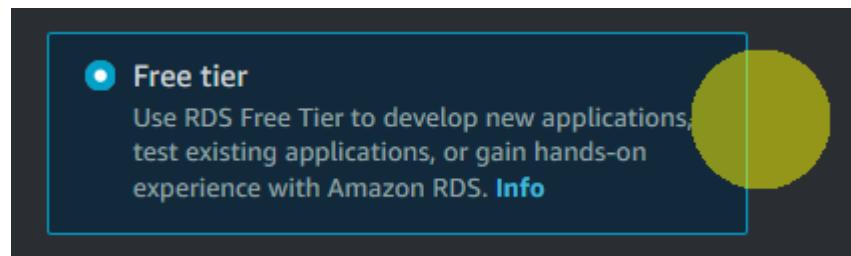
- 5) Pada *creation method* pilih *standard create*



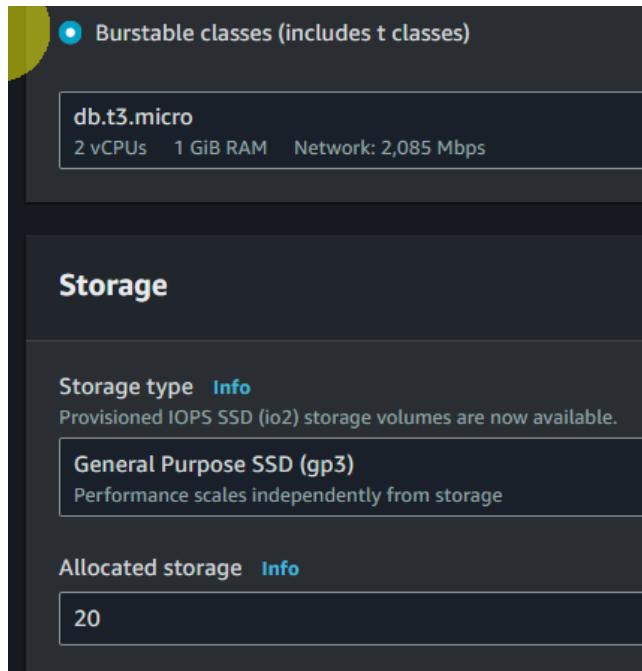
- 6) Pada bagian *engine type* pilih PostgreSQL. Kemudian versi yang dipakai adalah versi terbaru yaitu PostgreSQL 16.3-R1 dan pada bagian RDS *extended support* dibiarkan saja tidak tercentang.



- 7) Pada bagian *templates* dipilih "Free Tier". Database tipe ini sudah cukup untuk menampung data final project. Pada tipe *free tier* AWS memberikan 750 Jam dengan tipe instance t2.micro yang dibekali dengan *storage* sebesar 20GB untuk penyimpanan utama, serta 20GB untuk *bakcup database*.



- 8) Pada bagian *DB instance identifier*, berikan nama "final-project-db-instance."
- 9) Pilih *self-managed* pada opsi *credentials management*. Kemudian berikan *username* dan *password* untuk *master username*. Berikan *default* yaitu postgres dan postgres. Kedua kredensial ini dapat diganti nantinya saat database sudah berjalan.
- 10) Pada bagian *DB instance class* pilih tipe db.t3.micro dan berikan *storage* GP3 dengan ukuran 20GB



- 11) Pada bagian *connectivity* dibuat sama seperti gambar di bawah ini. *Public Access* diaktifkan karena *database* ini akan diakses melalui jaringan di luar infrastruktur AWS. Namun untuk *standard* keamanan *database* tidak boleh diakses *public*.

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Network type [Info](#)
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

IPv4
Your resources can communicate only over the IPv4 addressing protocol.

Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-078cd37b5c431f982)
3 Subnets, 3 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default-vpc-078cd37b5c431f982
3 Subnets, 3 Availability Zones

12) Pada bagian VPC pilih existing VPC yang telah dibuat sebelumnya. Kemudian pilih *availability zone* dimana *database* ini akan disimpan, yaitu *ap-southeast-3b*. Untuk bagian *certificate authority* biarkan pada opsi *default*.

13) Pada *database authentication* pilih *password authentication*. Sehingga saat akses database digunakan *master username* dan *password* yang telah dibuat sebelumnya.

Database authentication

Database authentication options [Info](#)

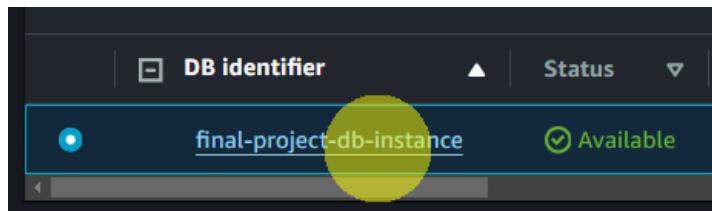
Password authentication
Authenticates using database passwords.

14) Pada bagian yang tidak disebutkan, dapat dibiarkan seperti bawaannya tanpa ada perubahan sedikit pun.

15) Langkah terakhir adalah mengecek perkiraan biaya yang ditagihkan pada bagian *estimated monthly cost*. Jika sudah sesuai klik pada *create database*.

Estimated Monthly costs	
DB instance	20.44 USD
Storage	2.76 USD
Total	23.20 USD

16) Jika *database* sudah aktif dan dapat digunakan pada bagian status akan berubah menjadi “*available*”. Kemudian klik pada bagian *DB identifier*



17) Setelah *database* aktif, cek pada *tab connectivity & security*, kemudian lihat bagian *endpoint & port*. Kedua parameter inilah yang digunakan untuk membuat koneksi ke database AWS RDS.

Connectivity & security	
Connectivity & security	
Endpoint	Network
final-project-db-instance.cvky8gakaf1d.ap-southeast-3.rds.amazonaws.com	Available
Port	VPC
5432	vpc-078cd
	Subnet
	default

B. Data Normalization

Sebelum data dapat digunakan, tahapan yang harus dilakukan ialah menormalisasi dan membersihkan data. Tahapan normalisasi dimulai dengan:

- 1) Mendesain ERD (*Entity Relationship Diagram*) yang memenuhi standar 3F dengan program diagram io. Dataset *raw* akan dibagi menjadi beberapa tabel atau entitas. Hospital_Trx sebagai entitas utama yang berisikan kolom *primary_key* id transaksi pasien dan *foreign_key* id dari entitas-entitas lainnya. Entitas lain memiliki hubungan *one-to-many* dengan entitas Hospital_Trx. Berikut adalah desain ERD yang digunakan.



- 2) Desain ERD yang sudah final dapat di-export ke dalam format PostgreSQL, query dari desain ERD kemudian dijalankan di aplikasi PostgreSQL untuk menciptakan tabel dan database sesuai dengan desain yang telah kita ciptakan. Pertama, yaitu membuat tabel transaksi HospitalTrx untuk menampung *raw data* dari file csv yang telah diunduh dari GitHub.

```
CREATE TABLE "HospitalTrx" (
    ID INTEGER PRIMARY KEY,
```

```

Date_IN DATE,
Date_OUT DATE,
Branch VARCHAR(50),
Name VARCHAR(100),
Age VARCHAR(5),
Gender VARCHAR(15),
Hospital_Care VARCHAR(50),
Room VARCHAR(20),
Doctor VARCHAR(100),
Surgery VARCHAR(50),
Lab VARCHAR(50),
Drug_Types VARCHAR(50),
Drug_Brands VARCHAR(50),
Drug_Qty VARCHAR(10),
Food VARCHAR(10),
Admin VARCHAR(10),
COGS VARCHAR(20),
Payment VARCHAR(20),
Review VARCHAR(20)
);

```

- 3) Setelah itu, kita menggunakan *syntax COPY* untuk *load* data dari penyimpanan lokal ke tabel transaksi HospitalTrx.

```

COPY "HospitalTrx" (ID, Date_IN, Date_OUT, Branch, Name, Age, Gender,
Hospital_Care, Room, Doctor, Surgery, Lab, Drug_Types, Drug_Brands,
Drug_Qty, Food, Admin, COGS, Payment, Review) FROM 'C:\Users\Berliana
Fitria          Dewi\Desktop\Bithealth\Training           Hacktiv8\Final
Project\Hospital_data.csv' DELIMITER ',' CSV HEADER;

```

- 4) Selanjutnya yaitu membuat tabel-tabel *master* untuk menyimpan nilai-nilai unik dari atribut-atribut yang berbeda. Tabel-tabel ini membantu menormalisasi data dengan menghilangkan redundansi.

Tabel Doctor:

```

CREATE TABLE "Doctor" (
"doctor_id" int PRIMARY KEY,
"doctor_type" varchar,
"doctor_price" money
);
INSERT INTO "Doctor" (doctor_id,doctor_type,doctor_price)
VALUES
(1, 'Bedah', '300000'),
(2, 'Gigi', '300000'),
(3, 'Kandungan', '300000'),
(4, 'Penyakit Dalam', '300000'),
(5, 'Umum', '200000');

```

Tabel Surgery:

```

CREATE TABLE "Surgery" (
"surgery_id" int PRIMARY KEY,
"surgery_type" varchar,

```

```

    "surgery_price" money
);
INSERT INTO "Surgery" (surgery_id,surgery_type,surgery_price)
VALUES
    (1, 'Kecil', '4000000'),
    (2, 'Besar', '8000000'),
    (3, 'Kusus', '15000000');

```

Tabel Lab:

```

CREATE TABLE "Lab" (
    "lab_id" int PRIMARY KEY,
    "lab_name" varchar,
    "lab_price" money
);
INSERT INTO "Lab" (lab_id,lab_name,lab_price)
VALUES
    (1, 'Hematologi', '90000'),
    (2, 'Kimia Darah', '195000'),
    (3, 'Rontgen', '150000'),
    (4, 'Serologi', '200000'),
    (5, 'Urinalisa', '80000');

```

Tabel Room:

```

CREATE TABLE "Room" (
    "room_id" int PRIMARY KEY,
    "room_type" varchar,
    "room_price" money,
    "food_price" money
);
INSERT INTO "Room" (room_id,room_type,room_price,food_price)
VALUES
    (1, 'VIP', '300000', '150000'),
    (2, 'Kelas 1', '250000', '110000'),
    (3, 'Kelas 2', '200000', '80000'),
    (4, 'Kelas 3', '150000', '50000');

```

Tabel Drugs:

```

CREATE TABLE "Drugs" (
    "drug_id" int PRIMARY KEY,
    "drug_brand" varchar,
    "drug_type_id" int
);
INSERT INTO "Drugs" (drug_id,drug_brand,drug_type_id)
VALUES
    (1, 'Amoxicillin', 1),
    (2, 'Azithromycin', 1),
    (3, 'Blackmores', 4),
    (4, 'Calpol', 3),
    (5, 'Ciprofloxacin', 1),
    (6, 'Diclofenac', 2),
    (7, 'Enervon-C', 4),
    (8, 'Holland & Barrett', 4),

```

```
(9, 'Naproxen', 2),
(10, 'Panadol', 3),
(11, 'Paramex', 3),
(12, 'Tramadol', 2);
```

Tabel DrugType:

```
CREATE TABLE "DrugType" (
    "drug_type_id" int PRIMARY KEY,
    "drug_type" varchar,
    "drug_price" money
);
INSERT INTO "DrugType" (drug_type_id,drug_type,drug_price)
VALUES
    (1, 'Antibiotik', '75000'),
    (2, 'Pereda Nyeri', '50000'),
    (3, 'Umum', '40000'),
    (4, 'Vitamin', '110000');
```

Tabel Patient:

```
CREATE TABLE "Patient" (
    "patient_id" SERIAL PRIMARY KEY,
    "patient_name" varchar,
    "gender" varchar,
    "age" int
);
INSERT INTO "Patient" (patient_name, gender, age)
SELECT DISTINCT name, gender, CAST(Age AS INT)
FROM "HospitalTrx";
```

Tabel Payment:

```
CREATE TABLE "Payment" (
    "payment_id" int PRIMARY KEY,
    "payment_name" varchar
);
INSERT INTO "Payment" (payment_id,payment_name)
VALUES
    (1, 'Asuransi'),
    (2, 'Pribadi');
```

Tabel Review:

```
CREATE TABLE "Review" (
    "review_id" int PRIMARY KEY,
    "review_name" varchar
);
INSERT INTO "Review" (review_id,review_name)
VALUES
    (1, 'Sangat Tidak Puas'),
    (2, 'Tidak Puas'),
    (3, 'Netral'),
```

```
(4, 'Puas'),
(5, 'Sangat Puas');
```

Tabel HospitalCare:

```
CREATE TABLE "HospitalCare" (
    "hospitalcare_id" int PRIMARY KEY,
    "hospital_care" varchar,
    "infusion_cost" money
);
INSERT INTO "HospitalCare"
(hospitalcare_id,hospital_care,infusion_cost)
VALUES
(1, 'Rawat Inap', '165000'),
(2, 'Rawat Jalan', '0');
```

Tabel Branch:

```
CREATE TABLE "Branch" (
    "branch_id" int PRIMARY KEY,
    "branch_name" varchar
);
INSERT INTO "Branch" (branch_id,branch_name)
VALUES
(1, 'RSMA'),
(2, 'RSMD'),
(3, 'RSMS');
```

- 5) Setelah membuat tabel-tabel *master* dan memasukkan data dari HospitalTrx, langkah selanjutnya yaitu dengan mengubah beberapa kolom di HospitalTrx agar menggunakan ID referensi dari tabel-tabel *master*.

```
ALTER TABLE "HospitalTrx"
RENAME COLUMN id TO id_trx;
-----
UPDATE "HospitalTrx" AS h
SET branch = br.branch_id
FROM "Branch" AS br
WHERE h.branch = br.branch_name;

ALTER TABLE "HospitalTrx"
RENAME COLUMN branch TO id_branch;
-----
UPDATE "HospitalTrx" AS h
SET hospital_care = hc.hospitalcare_id
FROM "HospitalCare" AS hc
WHERE h.hospital_care = hc.hospital_care;
ALTER TABLE "HospitalTrx"
RENAME COLUMN hospital_care TO id_hospital_care;
-----
SELECT * FROM "HospitalTrx";
UPDATE "HospitalTrx" AS h
```

```

SET name = p.patient_id
FROM "Patient" AS p
WHERE h.name = p.patient_name AND h.Gender = p.gender;

ALTER TABLE "HospitalTrx"
RENAME COLUMN name TO id_patient;
-----
UPDATE "HospitalTrx" AS h
SET room = r.room_id
FROM "Room" AS r
WHERE h.room = r.room_type;

ALTER TABLE "HospitalTrx"
RENAME COLUMN room TO id_room;
-----
UPDATE "HospitalTrx" AS h
SET doctor = d.doctor_id
FROM "Doctor" AS d
WHERE h.doctor = d.doctor_type;

ALTER TABLE "HospitalTrx"
RENAME COLUMN doctor TO id_doctor;
-----
UPDATE "HospitalTrx" AS h
SET payment = p.payment_id
FROM "Payment" AS p
WHERE h.payment = p.payment_name;

ALTER TABLE "HospitalTrx"
RENAME COLUMN payment TO id_payment;
-----
UPDATE "HospitalTrx" AS h
SET surgery = s.surgery_id
FROM "Surgery" AS s
WHERE h.surgery = s.surgery_type;

ALTER TABLE "HospitalTrx"
RENAME COLUMN surgery TO id_surgery;
-----
UPDATE "HospitalTrx" AS h
SET lab = l.lab_id
FROM "Lab" AS l
WHERE h.lab = l.lab_name;

ALTER TABLE "HospitalTrx"
RENAME COLUMN lab TO id_lab;
-----
UPDATE "HospitalTrx" AS h
SET drug_brands = d.drug_id
FROM "Drugs" AS d
WHERE h.drug_brands = d.drug_brand;

ALTER TABLE "HospitalTrx"
RENAME COLUMN drug_brands TO id_drug;
-----
UPDATE "HospitalTrx" AS h
SET review = r.review_id

```

```

FROM "Review" AS r
WHERE h.review = r.review_name;

ALTER TABLE "HospitalTrx"
RENAME COLUMN review TO id_review;
-----
ALTER TABLE "HospitalTrx"
DROP age,
DROP gender,
DROP drug_types,
DROP food;

ALTER TABLE "HospitalTrx"
RENAME COLUMN admin TO admin_price;

```

C. Data Transformation

Data transformation dilakukan secara otomatis melalui *scheduler airflow*.

Langkah-langkah yang dilakukan yaitu sebagai berikut.

- 1) *Import library* yang diperlukan

```

from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.postgres.hooks.postgres import PostgresHook
from datetime import datetime
import pandas as pd

```

- 2) *Load* data dari PostgreSQL

```

def fetch_and_convert_to_dataframe(**kwargs):
    table_names = table_names = [
        "HospitalTrx",
        "Doctor",
        "Surgery",
        "Lab",
        "Room",
        "Drugs",
        "DrugType",
        "Patient",
        "Payment",
        "Review",
        "HospitalCare",
        "Branch"
    ]

    # Menggunakan PostgresHook untuk melakukan koneksi ke
    # PostgreSQL
    hook = PostgresHook(postgres_conn_id='postgres_aws')

```

```

# Inisialisasi kamus kosong untuk menyimpan DataFrame hasil
dfs_dict = {}

for table_name in table_names:
    # Eksekusi query SQL untuk tabel yang diinginkan
    query = f'SELECT * FROM "{table_name}";'

    try:
        conn = hook.get_conn()
        cursor = conn.cursor()
        cursor.execute(query)
        result = cursor.fetchall()

        # Buat DataFrame dari hasil query
        columns = [desc[0] for desc in cursor.description]
        df = pd.DataFrame(result, columns=columns)

        # Tambahkan DataFrame ke kamus dengan nama tabel
        sebagai kunci
        dfs_dict[table_name] = df
    except Exception as e:
        print(f"Gagal mengambil data dari tabel {table_name}: {e}")

return dfs_dict

```

Fungsi `fetch_and_convert_to_dataframe` bertujuan untuk mengambil tabel-tabel yang telah dinormalisasi dari PostgreSQL dan mengonversinya menjadi DataFrame. Langkah pertama adalah membuat koneksi ke database menggunakan PostgresHook. Setelah koneksi berhasil, fungsi ini akan mengeksekusi *query* SQL untuk setiap tabel yang diinginkan, kemudian mengonversi hasilnya menjadi DataFrame. DataFrame yang dihasilkan disimpan dalam sebuah dictionary dengan nama tabel sebagai *key*. Dengan cara ini, tabel-tabel tersebut dapat dengan mudah diakses dan digunakan untuk proses selanjutnya.

3) Melakukan *data cleaning*

```

# Fungsi untuk mengubah tipe kolom ke numerik
def clean_numeric_columns(df, columns):
    for col in columns:

```

```

df[col] = pd.to_numeric(df[col], errors='coerce')

# Fungsi untuk melakukan transformasi pada data money
def clean_price_columns(df, columns):
    for col in columns:
        df[col] = pd.to_numeric(
            df[col].str.replace('$', '').str.replace(',', ''),
            '').str.replace('.00', '').
            errors='coerce'
        ).fillna(0)

# Fungsi untuk melakukan cleaning
def cleaning_data(**kwargs):
    df = kwargs['ti'].xcom_pull(task_ids='fetch_data_and_convert')

    try:
        df_trx = df['HospitalTrx']
        df_trx.replace('-', None, inplace=True)
        numeric_columns_trx = [
            'admin_price', 'drug_qty', 'cogs', 'id_branch',
            'id_patient',
            'id_hospital_care', 'id_room', 'id_doctor',
            'id_surgery',
            'id_lab', 'id_drug', 'id_payment', 'id_review'
        ]
        clean_numeric_columns(df_trx, numeric_columns_trx)
        df['HospitalTrx'] = df_trx

        # Mengubah kata kusus menjadi khusus
        df_surgery = df['Surgery']
        df_surgery['surgery_type'] =
        df_surgery['surgery_type'].replace('Kusus', 'Khusus')

        clean_price_columns(df_surgery, ['surgery_price'])
        df['Surgery'] = df_surgery

        df_doctor = df['Doctor']
        clean_price_columns(df_doctor, ['doctor_price'])
        df['Doctor'] = df_doctor

        df_lab = df['Lab']
        clean_price_columns(df_lab, ['lab_price'])
        df['Lab'] = df_lab

        df_room = df['Room']
    
```

```

    clean_price_columns(df_room, ['room_price', 'food_price'])
    df['Room'] = df_room

    df_drug = df['DrugType']
    clean_price_columns(df_drug, ['drug_price'])
    df['DrugType'] = df_drug

    df_care = df['HospitalCare']
    clean_price_columns(df_care, ['infusion_cost'])
    df['HospitalCare'] = df_care

    return df

except Exception as e:
    print(f"Error saat cleaning data: {str(e)}")
    raise e

```

Fungsi `cleaning_data` bertujuan untuk membersihkan dan merapikan data yang telah diambil dari PostgreSQL dan dikonversi menjadi DataFrame. Fungsi ini memuat hasil dari tugas sebelumnya, lalu melakukan beberapa proses *cleaning* pada kolom-kolom tertentu di berbagai tabel. Pada tabel `HospitalTrx`, nilai '-' diganti dengan `None` dan kolom numerik dilakukan transformasi menggunakan fungsi `clean_numeric_columns`. Tabel lain seperti `Surgery`, `Doctor`, `Lab`, `Room`, `DrugType`, dan `HospitalCare` mengalami pembersihan kolom harga menggunakan fungsi `clean_price_columns` untuk menghapus simbol \$, koma, dan '.00'. Setelah semua pembersihan dilakukan, DataFrame yang sudah diperbaiki dikembalikan untuk digunakan dalam proses selanjutnya.

4) Menambahkan kolom durasi rawat

```

def add_duration_column(**kwargs):
    df = kwargs['ti'].xcom_pull(task_ids='cleaning_data')      #
    Mendapatkan hasil dari task sebelumnya

    try:
        # Mengubah tipe data kolom 'Date IN' dan 'Date OUT' menjadi
        datetime
        df_hos = df['HospitalTrx']
        df_hos['date_in'] = pd.to_datetime(df_hos['date_in'])
        df_hos['date_out'] = pd.to_datetime(df_hos['date_out'])
    
```

```

# Menambahkan kolom Durasi Rawat
df_hos['Durasi_Rawat'] = df_hos['date_out'] -
df_hos['date_in']
df_hos['Durasi_Rawat'] = df_hos['Durasi_Rawat'] +
pd.Timedelta(days=1)

df[ 'HospitalTrx' ] = df_hos
return df
except Exception as e:
    print(f"Error saat menambahkan kolom durasi rawat:
{str(e)}")
    raise e

```

Fungsi `add_duration_column` bertujuan untuk menambahkan kolom durasi rawat pada tabel `HospitalTrx`. Fungsi ini memuat DataFrame yang sudah dibersihkan dari tugas sebelumnya. Pertama, tipe data kolom 'date_in' dan 'date_out' diubah menjadi tipe datetime untuk memudahkan perhitungan durasi. Kemudian, kolom 'Durasi_Rawat' ditambahkan dengan menghitung selisih antara 'date_out' dan 'date_in', ditambah satu hari untuk memperhitungkan hari masuk sebagai bagian dari durasi perawatan. Hasil modifikasi disimpan kembali ke dalam DataFrame, dan DataFrame yang telah diperbarui dikembalikan untuk digunakan lebih lanjut.

5) Menghitung *revenue* dan profit

```

def hitung_total_revenue(**kwargs):
    df = kwargs['ti'].xcom_pull(task_ids='add_duration_column')

    try:
        # Kolom-kolom untuk keperluan merge
        merge_keys = ['HospitalTrx', 'HospitalCare', 'Room',
'Doctor', 'Surgery', 'Lab', 'Drugs', 'DrugType', 'Branch',
'Patient', 'Payment', 'Review']
        merge_on_keys = ['id_hospital_care', 'id_room',
'id_doctor', 'id_surgery', 'id_lab', 'id_drug', 'drug_type_id',
'id_branch', 'id_patient', 'id_payment', 'id_review']
        right_on_keys = ['hospitalcare_id', 'room_id', 'doctor_id',
'surgery_id', 'lab_id', 'drug_id', 'drug_type_id', 'branch_id',
'patient_id', 'payment_id', 'review_id']

        final_df = df['HospitalTrx'].copy()
    
```

```

# Merge semua dataframe yang sudah dilist
    for left_key, right_key, df_key in zip(merge_on_keys,
right_on_keys, merge_keys[1:]):
        final_df = pd.merge(final_df, df[df_key],
left_on=left_key, right_on=right_key, how='left')

        # Menghilangkan kata 'days' pada kolom durasi rawat
        if 'Durasi_Rawat' in final_df:
            final_df['Durasi_Rawat'] =
final_df['Durasi_Rawat'].dt.days

        # Mengisi nilai NaN
        price_columns = ['room_price', 'food_price',
'doctor_price', 'surgery_price', 'lab_price', 'drug_price',
'admin_price']
        quantity_columns = ['drug_qty']

        for col in price_columns + quantity_columns:
            if col in final_df.columns:
                final_df[col] = final_df[col].fillna(0)

        # Menghitung revenue
        final_df['revenue'] = (
            (final_df['infusion_cost'] * final_df['Durasi_Rawat'])
+
            (final_df['room_price'] * final_df['Durasi_Rawat']) +
            (final_df['food_price'] * final_df['Durasi_Rawat']) +
            (final_df['doctor_price'] * final_df['Durasi_Rawat']) +
            final_df['surgery_price'] +
            final_df['lab_price'] +
            (final_df['drug_price'] * final_df['drug_qty']) +
            final_df['admin_price']
        )

        # Menghitung profit
        final_df['profit'] = final_df['revenue'] - final_df['cogs']

        # Drop kolom yang tidak diperlukan
        columns_to_drop = merge_on_keys + right_on_keys
        final_df.drop(columns=columns_to_drop, inplace=True)

    return final_df

except Exception as e:
    print(f"Error saat menghitung revenue: {str(e)}")

```

```
raise e
```

Fungsi `hitung_total_revenue` bertujuan untuk menghitung total biaya yang dikeluarkan oleh masing-masing pasien. Pertama-tama, fungsi ini menggabungkan data dari berbagai tabel menggunakan operasi merge. Setelah itu, durasi perawatan dihitung dalam hari, dan nilai-nilai yang hilang pada kolom harga dan jumlah diisi dengan nol. Fungsi ini kemudian menghitung total revenue (`revenue`) dengan menjumlahkan berbagai biaya seperti biaya infus, kamar, makanan, dokter, operasi, laboratorium, obat, dan administrasi. Keuntungan (`profit`) dihitung sebagai selisih antara total pendapatan yang diperoleh rumah sakit (*revenue*) dan harga pokok penjualan (COGS).

6) Menyimpan dataframe dalam bentuk csv

```
# Fungsi untuk menyimpan DataFrame ke file CSV
def save_to_csv(csv_file_path, **kwargs):
    df = kwargs['ti'].xcom_pull(task_ids='hitung_total_revenue')
    try:
        # Simpan DataFrame ke file CSV
        df.to_csv(csv_file_path, index=False)

        print(f"Data telah disimpan ke: {csv_file_path}")
    except Exception as e:
        print(f"Error saat menyimpan data ke CSV: {str(e)}")
        raise e
```

Fungsi `save_to_csv` bertujuan untuk menyimpan DataFrame yang telah dilakukan proses cleaning ke dalam file CSV. Fungsi ini memuat DataFrame hasil dari tugas sebelumnya, lalu mencoba menyimpannya ke path CSV yang ditentukan. Jika proses penyimpanan berhasil, fungsi akan mencetak pesan konfirmasi bahwa data telah disimpan. Namun, jika terjadi kesalahan selama proses penyimpanan, fungsi akan menangkap dan mencetak pesan *error*.

7) Setting DAG

```
default_args = {
    'owner': 'tim2Final',
    'depends_on_past': False,
    'start_date': datetime(2024, 4, 1),
}
```

```
with DAG('FinalProject_DAG', schedule_interval = '@daily',
         default_args = default_args,
         catchup=False) as dag:
```

Kode tersebut mendefinisikan pengaturan dasar dan jadwal untuk sebuah Directed Acyclic Graph (DAG) dalam Apache Airflow. DAG ini diberi nama 'FinalProject_DAG' dan dijadwalkan untuk berjalan setiap hari (@daily).

8) Pembuatan Task

```
# Task untuk mengambil data dan mengonversi ke DataFrame
fetch_data_task = PythonOperator(
    task_id='fetch_data_and_convert',
    python_callable=fetch_and_convert_to_dataframe,
    dag=dag
)

# Task untuk melakukan cleaning data
cleaning_data_task = PythonOperator(
    task_id='cleaning_data',
    python_callable=cleaning_data,
    dag=dag
)

# Operator untuk menambahkan kolom durasi rawat
add_duration_task = PythonOperator(
    task_id='add_duration_column',
    python_callable=add_duration_column,
    provide_context=True,
    dag=dag
)

# Operator untuk menghitung total revenue
hitung_total_revenue_task = PythonOperator(
    task_id='hitung_total_revenue',
    python_callable=hitung_total_revenue,
    provide_context=True,
    dag=dag
)

# Operator untuk menyimpan DataFrame ke file CSV
save_to_csv_task = PythonOperator(
    task_id='save_to_csv',
    python_callable=save_to_csv,
    provide_context=True,
```

```

        op_kwargs={'csv_file_path':
        '/opt/airflow/dags/cleaned_data.csv'}, # Path untuk menyimpan file
CSV
        dag=dag
)

```

Di sini, kita mendefinisikan sebuah *task* yang diberi nama `cleaning_data_task`. *Task* ini akan menjalankan fungsi `cleaning_data` ketika dijalankan dalam alur kerja yang diatur oleh `dag`. Pembuatan *task* yang serupa juga dilakukan pada fungsi-fungsi yang lain, seperti `fetch_and_convert_to_dataframe`, `add_duration_column`, `hitung_total_cost`, dan `save_to_csv`.

9) Task Dependency

```

# Definisikan urutan task (task dependency)
fetch_data_task >> cleaning_data_task >> add_duration_task >>
hitung_total_revenue_task >> save_to_csv_task

```

Task dependency memastikan bahwa *task* tertentu hanya akan dijalankan setelah *task* lain selesai sehingga alur kerja dapat berjalan dengan teratur. Urutan *task* yang akan dilakukan yaitu *fetch data*, *cleaning data*, *add duration*, *cost calculation*, dan *save to csv*.

D. Machine Learning

Pada bagian ini, kita akan membahas penerapan *machine learning* untuk memprediksi indeks performa rumah sakit. Pemodelan ini menggunakan pendekatan regresi untuk mendapatkan prediksi yang akurat berdasarkan data keuntungan (*profit*), jumlah pasien yang berkunjung, dan tingkat kepuasan pasien. Berikut adalah langkah-langkah yang akan dilakukan dalam proses pembuatan model *machine learning*:

a. Persiapan Data:

- Pengumpulan dan pembersihan data.
- Pembagian data menjadi set pelatihan dan set pengujian.

b. Pemilihan Model Regresi:

- Untuk memulai, kami akan menggunakan empat model regresi berikut:

- Linear Regression (`linreg`)
 - Support Vector Regressor (`svr`)
 - K-Neighbors Regressor (`knr`)
 - Decision Tree Regressor (`dectree`)
- c. Training dan Evaluasi Model Regresi:
- Melatih masing-masing model dengan data pelatihan.
 - Evaluasi performa model berdasarkan metrik tertentu (misalnya, MSE, RMSE, R-squared).
- d. Implementasi Model Boosting:
- Setelah evaluasi model regresi, kami akan menggunakan model boosting untuk meningkatkan performa prediksi. Model boosting yang akan digunakan adalah:
 - XGBoost Classifier (`xgb`)
 - LightGBM Classifier (`lgbm`)
- e. Hyperparameter Tuning:
- Melakukan *tuning hyperparameter* untuk setiap model menggunakan teknik seperti *Grid Search* atau *Random Search*.
 - Memilih model terbaik berdasarkan hasil *tuning* yang optimal.
- f. Visualisasi Hasil:
- Visualisasi performa masing-masing cabang rumah sakit berdasarkan prediksi indeks performa.

Langkah-langkah di atas akan membantu dalam menentukan model terbaik untuk memprediksi indeks performa rumah sakit dengan akurasi tinggi, yang selanjutnya dapat digunakan untuk memberikan rekomendasi peningkatan kinerja bagi cabang-cabang rumah sakit.

Berikut merupakan detail dari langkah-langkah yang dilakukan dalam pembuatan *machine learning*.

1. Import Library

Pada tahap ini merupakan tahapan *mengimport library* yang akan digunakan dalam pembuatan *machine learning*, berikut merupakan *library* yang digunakan.

```
pip install feature-engine xgboost lightgbm phik

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import OneHotEncoder, MinMaxScaler,
StandardScaler
from sklearn.model_selection import train_test_split,
cross_val_score, RandomizedSearchCV, GridSearchCV
from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score, make_scorer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,
GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.feature_selection import SelectKBest, f_classif,
f_regression
import joblib

from feature_engine.outliers import Winsorizer
from statsmodels.stats.outliers_influence import variance_inflation_factor
from xgboost import XGBClassifier, XGBRegressor
from lightgbm import LGBMClassifier, LGBMRegressor
from phik import phik_matrix

import warnings
import joblib

warnings.filterwarnings('ignore')
```

2. Data Loading

Pada tahap ini dilakukan *load data* yang akan digunakan dalam pembuatan *machine learning*, data yang digunakan telah disiapkan dalam *local computer* yang selanjutnya akan diimpor dengan menggunakan kode berikut.

```
data = pd.read_csv("cleaned_data.csv")
data.head()
```

3. Pre-Processing Data

Tahap *preprocessing* data bertujuan untuk membersihkan dan mengubah raw data ke dalam format yang sesuai untuk analisis dan model pelatihan. Langkah ini penting untuk memastikan kualitas data, mengurangi *noise*, mengatasi *missing values*, mengonversi tipe data, dan menormalkan fitur sehingga model dapat belajar dengan lebih efektif dan menghasilkan prediksi yang akurat. Adapun yang dilakukan pada langkah data *preprocessing*, yaitu sebagai berikut.

a) Pemeriksaan *missing* data pada *dataset*

Dilakukan pengecekan *missing* data pada *dataset*, untuk diputuskan apakah diperlukan penanganan terhadap *missing* data yang ada atau tidak.

```
# Memeriksa missing data
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum() / data.isnull().count() * 100).sort_values(ascending=False)
missing_data_df = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data_df.head(20)

# Menghitung total baris yang memiliki setidaknya satu nilai yang hilang
total_rows_with_missing = data.isnull().any(axis=1).sum()

print("Total baris dengan missing value:", total_rows_with_missing)
```

b) Pemeriksaan duplikat pada *dataset*

Data duplikat dapat menimbulkan bias pada model yang akan dihasilkan sehingga perlu dilakukan pemeriksaan duplikat sebelum dilakukan *training* pada *dataset*.

```

data.duplicated().sum()

dfeda = data
dfeda

dfeda.info()

print (dfeda)

```

- c) Mengonversi kolom `date_out` menjadi tipe data *datetime* dan menambahkan kolom bulan dan tahun

```

# Mengonversi kolom date_out menjadi tipe data datetime
dfeda['date_out'] = pd.to_datetime(dfeda['date_out'])

# Menambahkan kolom bulan dan tahun
dfeda['month'] = dfeda['date_out'].dt.month
dfeda['year'] = dfeda['date_out'].dt.year

```

- d) Memetakan nilai *review* ke skala yang diinginkan dan melakukan konversi nilai *review* menjadi skala numerik untuk memudahkan analisis statistik.

```

# Memetakan nilai-nilai review ke skala yang diinginkan
review_mapping = {
    'Sangat Tidak Puas': 1,
    'Tidak Puas': 2,
    'Cukup Puas': 3,
    'Puas': 4,
    'Sangat Puas': 5
}
dfeda['review_value'] = dfeda['review_name'].map(review_mapping)
dfeda['name_gender_age'] = dfeda['patient_name'] + '_' + dfeda['gender'] + '_' + dfeda['age'].astype(str)

```

- e) Melakukan normalisasi untuk kolom jumlah pasien menggunakan MinMaxScaler.

```

# Inisialisasi MinMaxScaler
scaler = MinMaxScaler()

# Menghitung nilai minimum dan maksimum dari jumlah_pasien
min_value = new_df['jumlah_pasien'].min()
max_value = new_df['jumlah_pasien'].max()

```

```

# Normalisasi nilai jumlah_pasien
new_df['norm_jumlah_pasien'] = 
scaler.fit_transform(new_df[['jumlah_pasien']])

# Menampilkan dataframe baru
new_df

```

- f) Mengelompokkan dan menghitung `total_pasien`, `revenue`, dan `profit`.

```

# Mengelompokkan dan menghitung total_pasien, total_revenue, dan
total_profit
new_df = dfeda.groupby(['branch_name', 'month', 'year']).agg(
    jumlah_pasien=('name_gender_age', 'nunique'),
    avg_review=('review_value', 'mean'),
    cogs=('cogs', 'sum'),
    total_revenue=('revenue', 'sum'),
    total_profit=('profit', 'sum')
).reset_index()

```

- g) Normalisasi data menggunakan MinMaxScaler

```

# Inisialisasi MinMaxScaler
min_value = new_df['avg_review'].min()
max_value = new_df['avg_review'].max()

# Normalisasi nilai avg_review
new_df['norm_avg_review'] = 
scaler.fit_transform(new_df[['avg_review']])

```

Normalisasi dilakukan pada kolom jumlah_pasien, total profit, dan average review menggunakan *MinMaxScaler*, yang mengubah nilai-nilai dalam kolom tersebut ke rentang 0-1. Normalisasi ini penting untuk memastikan bahwa semua fitur berada pada skala yang sama, sehingga model *machine learning* dapat berfungsi lebih baik dan menghindari masalah yang disebabkan oleh perbedaan skala fitur.

- h) Menghitung *score*

```

new_df['score'] = (new_df['norm_jumlah_pasien']*0.2) +
(new_df['norm_total_profit']*0.45) +
(new_df['norm_avg_review']*0.35)
print(new_df)

```

Score adalah indeks performa yang nantinya akan diprediksi oleh model. Untuk menghitung *score*, dilakukan pembobotan pada masing-masing indikator kemudian dijumlahkan untuk mendapatkan nilai *score* akhir.

4. Exploratory Data Analysis (EDA)

a) Perhitungan Korelasi

Perhitungan korelasi membantu untuk memahami hubungan antar fitur pada *dataset* sehingga dapat membantu memilih fitur yang akan digunakan pada model.

```
cols = ['branch_name', 'month', 'year', 'jumlah_pasien',  
'avg_review', 'cogs', 'total_cost', 'total_revenue', 'score']  
  
# Menghitung matriks korelasi Phik  
correlation_matrix =  
new_df[cols].phik_matrix(interval_cols=['month',  
'year',  
'jumlah_pasien',  
'avg_review',  
'cogs',  
'total_cost',  
'total_revenue',  
'score'])  
  
# Create heatmap from correlation_matrix temp  
plt.figure(figsize=(10, 8))  
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')  
plt.title('Phik Correlation Heatmap Temp')  
plt.show()
```

5. Feature Engineering

a) Data Splitting

Data *splitting* dilakukan dengan membagi *dataset* menjadi *training set* (data yang digunakan untuk pelatihan) dan *testing set* (data yang digunakan untuk pengujian).

```
# Defining X and y  
X = dfml.drop(['score'],axis=1)  
y = pd.DataFrame(dfml['score'])  
  
# Split between Train-Set, and Test-Set  
  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=0.2, random_state=42)
```

b) *Scaling* dan *Encoding*

Scaling digunakan untuk mengubah skala fitur numerik sehingga memiliki rentang yang sama, metode *scaling* yang digunakan ialah MinMaxScaling. *Encoding* digunakan untuk mengubah fitur kategori menjadi format numerik yang bisa digunakan oleh algoritma *machine learning*.

- *Scaling*

```
# Mengidentifikasi fitur numerik
numeric_features = ['month', 'year', 'jumlah_pasien', 'avg_review',
'cogs', 'total_cost', 'total_revenue']
numeric_transformer = StandardScaler()

# Membuat transformer untuk fitur numerik dan kategori
categorical_features = ['branch_name']
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Apply Standard Scaling pada fitur numerik di train set
X_train[numeric_features] =
    numeric_transformer.fit_transform(X_train[numeric_features])

# Gunakan parameter yang di-fit dari train set untuk transformasi
# test set
X_test[numeric_features] =
    numeric_transformer.transform(X_test[numeric_features])
```

- *Encoding*

```
# Mengidentifikasi fitur numerik dan kategori
numeric_features = ['month', 'year', 'jumlah_pasien', 'avg_review',
'cogs', 'total_cost', 'total_revenue']
categorical_features = ['branch_name']

# Membuat transformer untuk fitur numerik dan kategori
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Apply OneHotEncoding pada fitur kategori di train set
X_train_encoded =
    pd.DataFrame(categorical_transformer.fit_transform(X_train[categorical_features]).toarray(),
                columns=categorical_transformer.get_feature_names_out(categorical_features))
```

```

X_test_encoded = pd.DataFrame(categorical_transformer.transform(X_test[categorical_features]).toarray(),
                               columns=categorical_transformer.get_feature_names_out(categorical_features))

```

Selanjutnya, yaitu menggabungkan data yang telah di-*scaling* dan di-*encoding* dengan *code* berikut.

```

# Menggabungkan kembali data yang telah di-encode dengan data numerik
# yang telah di-scale
X_train = X_train.reset_index(drop=True)
X_train = pd.concat([X_train[numeric_features], X_train_encoded.reset_index(drop=True)], axis=1)

X_test = X_test.reset_index(drop=True)
X_test = pd.concat([X_test[numeric_features], X_test_encoded.reset_index(drop=True)], axis=1)

```

c) *Data Distribution*, *script* di bawah ini digunakan untuk menganalisis distribusi variabel dalam *dataset*, mengidentifikasi batas normal untuk *outlier*, dan menghitung persentase *outlier* di ujung kanan dan kiri distribusi.

```

def diagnostic_plots(dfml, variable):
    # Define figure size
    plt.figure(figsize=(16, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(dfml[variable], bins=20)
    plt.title('Histogram')

    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(y=dfml[variable])
    plt.title('Boxplot')

    plt.show()

def find_normal_boundaries(dfml, var):
    upper_boundary = dfml[var].mean() + 3 * dfml[var].std()
    lower_boundary = dfml[var].mean() - 3 * dfml[var].std()

```

```

        return upper_boundary, lower_boundary

def percentage_outlier(dfml, var, upper_boundary, lower_boundary):
    outlier_right_tail = (len(dfml[dfml[var] > upper_boundary]) / len(dfml) * 100)
    outlier_left_tail = (len(dfml[dfml[var] < lower_boundary]) / len(dfml) * 100)
    return outlier_right_tail, outlier_left_tail

variables = X_train

for variable in variables:
    print(f"\nVariable: {variable}")
    diagnostic_plots(X_train, variable)
    print('Skewness Value:', X_train[variable].skew())

    upper_boundary, lower_boundary = find_normal_boundaries(X_train, variable)
    print('Upper Boundary:', upper_boundary)
    print('Lower Boundary:', lower_boundary)

    outlier_right_tail, outlier_left_tail = percentage_outlier(X_train, variable, upper_boundary, lower_boundary)
    print(f"% right end outliers: {outlier_right_tail}")
    print(f"% left end outliers: {outlier_left_tail}")

    if (X_train[variable].skew() <= -0.5) | (X_train[variable].skew() >= 0.5):
        print('Skewed Distribution')
        print('*'*20)
    else:
        print('Normal Distribution')
        print('*'*20)

```

- d) *Outliers Handling*, script berikut ini bertujuan untuk melakukan analisis outlier berdasarkan distribusi data dalam kolom tertentu. Metode yang digunakan untuk mendekripsi *outlier* ada dua, yaitu: berdasarkan *mean* dan standar deviasi untuk distribusi yang mendekati normal (*skewness* antara -0.5 dan 0.5), serta berdasarkan *interquartile range* (IQR) untuk distribusi yang miring.

```
def outlier_analysis(dfml, col):
```

```

skewness = dfml[col].skew()
if skewness >= -0.5 and skewness <= 0.5:
    upper = dfml[col].mean() + 3 * dfml[col].std()
    lower = dfml[col].mean() - 3 * dfml[col].std()
else:
    Q1 = dfml[col].quantile(0.25)
    Q3 = dfml[col].quantile(0.75)
    IQR = Q3 - Q1

    upper = Q3 + (1.5 * IQR)
    lower = Q1 - (1.5 * IQR)

no_outliers = dfml[(dfml[col] >= lower) & (dfml[col] <= upper)]
outliers = dfml[(dfml[col] > upper) | (dfml[col] < lower)]

return outliers, no_outliers

columns = ['month', 'year', 'jumlah_pasien', 'avg_review', 'cogs',
'total_cost', 'total_revenue']

for col in columns:
    outliers, no_outliers = outlier_analysis(X_train, col)

    if len(outliers) > 0:
        print(f'Column name: {col}')
        print('Count of outliers:', len(outliers))
        print('Percentage of outliers:', (len(outliers) / len(X_train)) * 100, '%')
    else:
        print(f'Column name: {col}')
        print('No outliers found')

    print('-' * 20)

```

```

# capping outlier if exist and in large proportion
wins = Winsorizer(capping_method='iqr', tail='both', fold=1.5)
wins.fit(X_train)
X_train = wins.transform(X_train)

num = X_train
n = len(num.columns)
sns.set(font_scale=1)
fig, ax = plt.subplots(n, 1, figsize=(30, 45))
for i, col in enumerate(num.columns):
    sns.boxplot(ax=ax[i], data=X_train[col], width=0.50)

```

```

    ax[i].set_title(f'{col} in train set - after outlier handling')
plt.show()

```

- e) *Feature Selection* bertujuan untuk memilih fitur terbaik dari *dataset* menggunakan metode SelectKBest dengan fungsi skor f_regression.

```

X_train_features = X_train.copy()

# Inisialisasi Selector
selector = SelectKBest(score_func=f_regression, k='all')

# Fit dan Transform Data
X_train_selected = selector.fit_transform(X_train, y_train)

# Melihat Skor Feature
feature_scores = selector.scores_

# Pemilihan Fitur Terbaik
selected_features = X_train.columns[selector.get_support()]

# Menampilkan hasil
print("Feature Scores:")
for feature, score in zip(X_train.columns, feature_scores):
    print(f"{feature}: {score}")

print("\nSelected Features:")
print(selected_features)

```

- VIF For *Multicolinearity*, menghitung *Variance Inflation Factor* (VIF) untuk memeriksa multikolinearitas di antara variabel prediktor dalam *dataset*.

```

# Ambil hanya variabel prediktor yang dipilih
X = X_train_features[['jumlah_pasien', 'avg_review', 'cogs',
'total_cost', 'total_revenue']]

# Hitung VIF untuk setiap variabel prediktor
vif = pd.DataFrame()
vif["Variable"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

print(vif)
# Ambil hanya variabel prediktor yang dipilih

```

```

X = X_train_features[['jumlah_pasien', 'avg_review', 'total_cost',
'total_revenue']]

# Hitung VIF untuk setiap variabel prediktor
vif = pd.DataFrame()
vif[ "Variable" ] = X.columns
vif[ "VIF" ] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

print(vif)

```

Kemudian memfilter data *training* dan *test* menggunakan fitur yang telah dipilih.

```

selected_features = ['jumlah_pasien', 'avg_review', 'total_cost',
'total_revenue']

# Filter data train dan test menggunakan fitur yang telah dipilih
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]

```

6. Model Definition

Selanjutnya yaitu mendefinisikan model-model yang akan digunakan. Pada tugas ini akan digunakan 4 model sebagai perbandingan, yaitu model Linear Regression, SVR (Support Vector Regression), K-Neighbors Regressor, Decision Tree Regressor.

```

# Define Linear regression
linreg = LinearRegression()
svr = SVR()
knr = KNeighborsRegressor()
dectree = DecisionTreeRegressor()

```

7. Model Training

- Masukan *dictionary* untuk menyimpan model

```

models = {
    "Linear Regression": linreg,
    "Support Vector Regressor": svr,
    "K-Neighbors Regressor": knr,
    "Decision Tree Regressor": dectree}

```

- b) Masukan fungsi untuk melatih dan mengevaluasi model menggunakan *Cross Validation*

```

def cross_val_evaluate(models, X, y, cv=5):
    results = {}
    for name, model in models.items():
        # Cross-validation
        scores = cross_val_score(model, X, y, cv=cv,
scoring='neg_mean_squared_error')
        mse_scores = -scores # Negate to get positive MSE
        rmse_scores = np.sqrt(mse_scores)

        scores = cross_val_score(model, X, y, cv=cv,
scoring='neg_mean_absolute_error')
        mae_scores = -scores # Negate to get positive MAE

        scores = cross_val_score(model, X, y, cv=cv, scoring='r2')
        r2_scores = scores

        results[name] = {
            'MSE': {
                'mean': mse_scores.mean()
            },
            'RMSE': {
                'mean': rmse_scores.mean()
            },
            'MAE': {
                'mean': mae_scores.mean()
            },
            'R22      Mean:')
        print(f'{name}      Cross-Validation      Mean:')


    return results

```

```
# Perform cross-validation
results = cross_val_evaluate(models, X_train_selected, y_train,
cv=5)
```

- c) Pilih model terbaik berdasarkan hasil *cross-validation* dengan nilai MAE terendah

```
best_model_name = min(results, key=lambda x:
np.mean(results[x]['MAE']['mean']))
best_model = models[best_model_name]
print(f"Best Model: {best_model_name}")
```

- d) Pilih model terbaik berdasarkan hasil *cross-validation* dengan nilai MSE terendah

```
best_model_name = min(results, key=lambda x:
np.mean(results[x]['MSE']['mean']))
best_model = models[best_model_name]
print(f"Best Model: {best_model_name}")
```

- e) Pilih model terbaik berdasarkan hasil *cross-validation* dengan nilai RMSE terendah

```
best_model_name = min(results, key=lambda x:
np.mean(results[x]['RMSE']['mean']))
best_model = models[best_model_name]
print(f"Best Model: {best_model_name}")
```

- f) Evaluasi model terbaik pada *test* set

```
best_model.fit(X_train_selected, y_train)
y_test_pred = best_model.predict(X_test_selected)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
mae_test = mean_absolute_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)

print("Test Set Evaluation:")
print(f"Test MSE: {mse_test}")
print(f"Test RMSE: {rmse_test}")
print(f"Test MAE: {mae_test}")
print(f"Test R2: {r2_test}")
```

8. Model Boosting

- a) Lakukan model *boosting* dengan *gradient boosting regresor*

```

# Initialize the model
gbr = GradientBoostingRegressor(random_state=0)

# Train the model
gbr.fit(X_train_selected, y_train)

# Predict on the test set
y_test_pred_gbr = gbr.predict(X_test_selected)

# Evaluate the model
rmse_gbr = np.sqrt(mean_squared_error(y_test, y_test_pred_gbr))
mae_gbr = mean_absolute_error(y_test, y_test_pred_gbr)
r2_gbr = r2_score(y_test, y_test_pred_gbr)

print(f'Gradient Boosting Test RMSE: {rmse_gbr}')
print(f'Gradient Boosting Test MAE: {mae_gbr}')
print(f'Gradient Boosting Test R2: {r2_gbr}')

```

b) Lakukan model *boosting* dengan *XGB Regressor*

```

# Initialize the model
xgbr = XGBRegressor(random_state=0)

# Train the model
xgbr.fit(X_train_selected, y_train)

# Predict on the validation set
y_test_pred_xgbr = xgbr.predict(X_test_selected)

# Evaluate the model
rmse_xgbr = np.sqrt(mean_squared_error(y_test, y_test_pred_xgbr))
mae_xgbr = mean_absolute_error(y_test, y_test_pred_xgbr)
r2_xgbr = r2_score(y_test, y_test_pred_xgbr)

print(f'XGBoost Validation RMSE: {rmse_xgbr}')
print(f'XGBoost Validation MAE: {mae_xgbr}')
print(f'XGBoost Validation R2: {r2_xgbr}')

```

c) Lakukan model *boosting* dengan *LGBM Regressor*

```

# Initialize the model
lgbmr = LGBMRegressor(random_state=0, verbose=-1)

# Train the model
lgbmr.fit(X_train_selected, y_train)

```

```

# Predict on the validation set
y_test_pred_lgbmr = lgbmr.predict(X_test_selected)

# Evaluate the model
rmse_lgbmr = np.sqrt(mean_squared_error(y_test, y_test_pred_lgbmr))
mae_lgbmr = mean_absolute_error(y_test, y_test_pred_lgbmr)
r2_lgbmr = r2_score(y_test, y_test_pred_lgbmr)

print(f'LightGBM Validation RMSE: {rmse_lgbmr}')
print(f'LightGBM Validation MAE: {mae_lgbmr}')
print(f'LightGBM Validation R2: {r2_lgbmr}')

```

d) Membandingkan model terbaik dengan *boosted model*

```

# Dictionary to hold test results
test_results_boosting = {
    "Gradient Boosting": {"RMSE": rmse_gbr, "MAE": mae_gbr, "R2": r2_gbr},
    "XGBoost": {"RMSE": rmse_xgbr, "MAE": mae_xgbr, "R2": r2_xgbr},
    "LightGBM": {"RMSE": rmse_lgbmr, "MAE": mae_lgbmr, "R2": r2_lgbmr}
}

# Select the best model based on test RMSE
best_model_name_boosting = min(test_results_boosting, key=lambda x: test_results_boosting[x]["RMSE"])
best_model_boosting = {
    "Gradient Boosting": gbr,
    "XGBoost": xgbr,
    "LightGBM": lgbmr
}[best_model_name_boosting]

print(f'Best Boosting Model: {best_model_name_boosting}')

# Evaluate the best model on the test set
y_test_pred_boosting = best_model_boosting.predict(X_test_selected)
rmse_test_boosting = np.sqrt(mean_squared_error(y_test, y_test_pred_boosting))
mae_test_boosting = mean_absolute_error(y_test, y_test_pred_boosting)
r2_test_boosting = r2_score(y_test, y_test_pred_boosting)

print(f'Test RMSE: {rmse_test_boosting}')
print(f'Test MAE: {mae_test_boosting}')

```

```
print(f'Test R2: {r2_test_boosting}')
```

9. Hyper Parameter Tuning

- a) *Tuning* dan evaluasi model *LinearRegression* menggunakan *GridSearchcv*

```
# Define the parameter grid for GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.8, 0.9, 1.0]
}

# Initialize the GradientBoostingRegressor
gbr = GradientBoostingRegressor(random_state=0)

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=gbr, param_grid=param_grid,
                           scoring='neg_mean_absolute_error', cv=5,
                           n_jobs=-1, verbose=2)

# Fit GridSearchCV on the training data
grid_search.fit(X_train_selected, y_train)

# Get the best parameters and the best estimator
best_params = grid_search.best_params_
best_gbr = grid_search.best_estimator_

print(f'Best Parameters: {best_params}')

# Evaluate the best model on the training set
y_train_pred_best_gbr = best_gbr.predict(X_train_selected)
rmse_train_best_gbr = np.sqrt(mean_squared_error(y_train,
y_train_pred_best_gbr))
mae_train_best_gbr = mean_absolute_error(y_train,
y_train_pred_best_gbr)
r2_train_best_gbr = r2_score(y_train, y_train_pred_best_gbr)

print(f'training RMSE: {rmse_train_best_gbr}')
print(f'training MAE: {mae_train_best_gbr}')
print(f'training R2: {r2_train_best_gbr}')
```

```

# Evaluate the best model on the test set
y_test_pred_best_gbr = best_gbr.predict(X_test_selected)
rmse_test_best_gbr      = np.sqrt(mean_squared_error(y_test,
y_test_pred_best_gbr))
mae_test_best_gbr       = mean_absolute_error(y_test,
y_test_pred_best_gbr)
r2_test_best_gbr = r2_score(y_test, y_test_pred_best_gbr)

print(f'Test RMSE: {rmse_test_best_gbr}')
print(f'Test MAE: {mae_test_best_gbr}')
print(f'Test R2: {r2_test_best_gbr}')

```

- b) Tuning dan evaluasi model *GradientBoostingRegressor* menggunakan *RandomizedSearchcv*

```

Define the parameter grid for RandomizedSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.8, 0.9, 1.0]
}

# Initialize the GradientBoostingRegressor
gbr = GradientBoostingRegressor(random_state=0)

# Initialize GridSearchCV
random_search = RandomizedSearchCV(estimator=gbr,
param_distributions=param_grid,
n_iter=100,
scoring='neg_mean_absolute_error',
cv=5, n_jobs=-1, random_state=0,
verbose=2)

# Fit GridSearchCV on the training data
random_search.fit(X_train_selected, y_train)

# Get the best parameters and the best estimator
best_params = random_search.best_params_
best_gbr_random = random_search.best_estimator_

print(f'Best Parameters: {best_params}')

```

```

# Evaluate the best model on the training set
y_train_pred_best_gbr_random =
best_gbr_random.predict(X_train_selected)
rmse_train_best_gbr_random = np.sqrt(mean_squared_error(y_train,
y_train_pred_best_gbr_random))
mae_train_best_gbr_random = mean_absolute_error(y_train,
y_train_pred_best_gbr_random)
r2_train_best_gbr_random = r2_score(y_train,
y_train_pred_best_gbr_random)

print(f'training RMSE: {rmse_train_best_gbr_random}')
print(f'training MAE: {mae_train_best_gbr_random}')
print(f'training R2: {r2_train_best_gbr_random}')

# Evaluate the best model on the test set
y_test_pred_best_gbr_random =
best_gbr_random.predict(X_test_selected)
rmse_test_best_gbr_random = np.sqrt(mean_squared_error(y_test,
y_test_pred_best_gbr_random))
mae_test_best_gbr_random = mean_absolute_error(y_test,
y_test_pred_best_gbr_random)
r2_test_best_gbr_random = r2_score(y_test,
y_test_pred_best_gbr_random)

print(f'Test RMSE: {rmse_test_best_gbr_random}')
print(f'Test MAE: {mae_test_best_gbr_random}')
print(f'Test R2: {r2_test_best_gbr_random}')

```

10. Pemilihan Model

Memilih model yang memiliki hasil *test* dan *training* terbaik sebagai model *machine learning*.

11. Model Saving

- a) Simpan model ke dalam format

```

import pickle
model_filename = "model_linear.pkl"
with open(model_filename, 'wb') as file:
    pickle.dump(best_model, file)

```

a. Pipeline

- a) Pipeline dari *machine learning* untuk *pre-processing data* dan melatih model *Linear Regression*

```
data = pd.read_csv("cleaned_data.csv")
# Mengonversi kolom date_out menjadi tipe data datetime
dfeda['date_out'] = pd.to_datetime(dfeda['date_out'])

# Menambahkan kolom bulan dan tahun
dfeda['month'] = dfeda['date_out'].dt.month
dfeda['year'] = dfeda['date_out'].dt.year

# Memetakan nilai-nilai review ke skala yang diinginkan
review_mapping = {
    'Sangat Tidak Puas': 1,
    'Tidak Puas': 2,
    'Cukup Puas': 3,
    'Puas': 4,
    'Sangat Puas': 5
}
dfeda['review_value'] = dfeda['review_name'].map(review_mapping)
dfeda['name_gender_age'] = dfeda['patient_name'] + '_' + dfeda['gender'] + '_' + dfeda['age'].astype(str)
# Mengelompokkan dan menghitung total_pasien, total_cost, dan total_revenue
new_df = dfeda.groupby(['branch_name', 'month', 'year']).agg(
    jumlah_pasien=('name_gender_age', 'nunique'),
    avg_review=('review_value', 'mean'),
    cogs=('cogs', 'sum'),
    total_cost=('total_cost', 'sum'),
    total_revenue=('revenue', 'sum')
).reset_index()

# Normalisasi menggunakan MinMaxScaler
scaler_jumlah_pasien = MinMaxScaler()
scaler_total_revenue = MinMaxScaler()
scaler_avg_review = MinMaxScaler()

# Fit dan transformasi setiap fitur
new_df['norm_jumlah_pasien'] =
scaler_jumlah_pasien.fit_transform(new_df[['jumlah_pasien']])
new_df['norm_total_revenue'] =
scaler_total_revenue.fit_transform(new_df[['total_revenue']])
new_df['norm_avg_review'] =
scaler_avg_review.fit_transform(new_df[['avg_review']])
```

```

new_df['score'] = (new_df['norm_jumlah_pasien']*0.2) +
(new_df['norm_total_revenue']*0.45) +
(new_df['norm_avg_review']*0.35)

# Menentukan fitur dan target
X = new_df[['jumlah_pasien', 'avg_review', 'total_cost',
'total_revenue']]
y = new_df['score']

# Pipeline untuk preprocessing
# Fitur numerik: melakukan standard scaling
numeric_features = ['jumlah_pasien', 'avg_review', 'total_cost',
'total_revenue']
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

# 8. Pipeline utama: preprocessing dan model
pipeline = Pipeline(steps=[
    ('preprocessor', numeric_transformer),
    ('feature_selection', SelectFromModel(LinearRegression())),
    ('polynomial_features', PolynomialFeatures(degree=2)),
    ('regressor', LinearRegression())
])

# 9. Split data menjadi train dan test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 10. Training pipeline pada data train
pipeline.fit(X_train, y_train)

# 11. Evaluasi model
y_pred = pipeline.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'Mean Absolute Error: {mae}')
print(f'R-squared: {r2}')

```

```
# 12. Simpan model ke file pickle
model_filename = "model_pipeline.pkl"
with open(model_filename, 'wb') as file:
    pickle.dump(pipeline, file)
```

E. Model Deployment

Setelah memperoleh model *machine learning* terbaik, langkah berikutnya adalah mengimplementasikan model tersebut dengan melakukan *deployment*. Proses *deployment* bertujuan untuk menyediakan solusi yang dapat diakses secara *real-time*, memberikan manfaat langsung dalam meningkatkan efisiensi operasional, pengambilan keputusan, dan pengalaman pengguna. Proses *deployment* model menggunakan *streamlit*.

```
app.py
import streamlit as st
import predict_FP as predict_FP
import predict as predict
import ml_FP as ml

navigation = st.sidebar.selectbox('Select Page:', ('Form Page', 'File Upload Page', 'Chatbot Page'))

if navigation == 'Form Page':
    predict_FP.run()
elif navigation == 'File Upload Page':
    predict.run()
elif navigation == 'Chatbot Page':
    ml.run()
```

Kode `app.py` tersebut adalah aplikasi *Streamlit* sederhana yang memiliki *sidebar* untuk navigasi antara halaman-halaman berikut:

- 'Form Page': Memanggil fungsi `predict_FP.run()` untuk menampilkan halaman formulir dan melakukan prediksi.
- 'File Upload Page': Memanggil fungsi `predict.run()` untuk menampilkan halaman unggah file dan melakukan proses prediksi.
- 'Chatbot Page': Memanggil fungsi `ml.run()` untuk menampilkan halaman chatbot.

Pengguna memilih halaman yang diinginkan melalui pilihan pada sidebar, dan aplikasi akan mengarahkan mereka ke halaman sesuai pilihan tersebut.

predict_FP.py

```
import streamlit as st
import pandas as pd
import pickle
```

```

import calendar

# Load model
model_filename = "model_pipeline.pkl"
with open(model_filename, 'rb') as file:
    pipeline = pickle.load(file)

# Fungsi untuk melakukan prediksi
def predict_score(jumlah_pasien, avg_review, revenue, cogs):
    # Buat DataFrame dari input user
    profit = revenue - cogs
    input_data = pd.DataFrame({
        'jumlah_pasien': [jumlah_pasien],
        'avg_review': [avg_review],
        'total_revenue': [revenue],
        'total_profit': [profit]
    })
    print(input_data)
    # Lakukan prediksi menggunakan pipeline
    predicted_score = pipeline.predict(input_data)[0]

    return predicted_score

def run():
    # Interface Streamlit
    st.title('Hospital Performance Index Score Prediction 💼')

    # Form input dari user
    branch_name = st.selectbox("Branch", ["RSMA", "RSMS", "RSMD"])
    month = st.number_input('Month', min_value=1, max_value=12,
                           step=1)
    year = st.number_input('Year', min_value=2000, max_value=2100,
                          step=1)
    jumlah_pasien = st.number_input('Number of Patient', step=1)
    avg_review = st.number_input('Average of Review', min_value=1.0,
                                max_value=5.0, step = 0.1)
    total_revenue = st.number_input('Total Revenue', min_value=0)
    cogs = st.number_input('Total COGS', min_value=0)

    if st.button('Predict Score'):
        # Lakukan prediksi
        predicted_score = predict_score(jumlah_pasien, avg_review,
                                         total_revenue, cogs)
        # print(predicted_score)
        # Tampilkan hasil prediksi

```

```

if month == 12:
    month = 0
    year = year+1
month_name = calendar.month_name[month+1]

if predicted_score>1: predicted_score=1
elif predicted_score<0: predicted_score=0
st.write(f'Performance score for {branch_name} in {month_name} {year} is: {predicted_score*100:.2f}')

if __name__ == '__main__':
    run()

```

Code `predict_FP.py` adalah kode untuk prediksi yang berbentuk *form*. Pengguna diarahkan untuk mengisi *field-field* yang disediakan. Setelah terisi dan diklik *button* prediksi, model akan melakukan prediksi berdasarkan parameter-parameter yang dibutuhkan dan menghasilkan *performance index score* untuk bulan selanjutnya.

predict.py

```

import streamlit as st
import pandas as pd
import joblib
from sklearn.preprocessing import MinMaxScaler

# Load the preprocessing function
def preprocess_data(df):
    # Konversi kolom date_out menjadi tipe data datetime
    df['date_out'] = pd.to_datetime(df['date_out'])

    # Menambahkan kolom bulan dan tahun
    df['month'] = df['date_out'].dt.month
    df['year'] = df['date_out'].dt.year

    # Memetakan nilai-nilai review ke skala yang diinginkan
    review_mapping = {
        'Sangat Tidak Puas': 1,
        'Tidak Puas': 2,
        'Cukup Puas': 3,
        'Puas': 4,
        'Sangat Puas': 5
    }
    df['review_value'] = df['review_name'].map(review_mapping)

```

```

df['name_gender_age'] = df['patient_name'] + '_' + df['gender']
+ '_' + df['age'].astype(str)

# Mengelompokkan dan menghitung total_pasien, revenue, dan profit
new_df = df.groupby(['branch_name', 'month', 'year']).agg(
    jumlah_pasien=('name_gender_age', 'count'),
    avg_review=('review_value', 'mean'),
    cogs=('cogs', 'sum'),
    total_revenue=('revenue', 'sum'),
    total_profit=('profit', 'sum')
).reset_index()

# Normalisasi kolom jumlah_pasien, total_revenue, dan avg_review
scaler = MinMaxScaler()
new_df[['jumlah_pasien',
'total_revenue','total_profit','avg_review']] =
scaler.fit_transform(new_df[['jumlah_pasien',
'total_revenue','total_profit','avg_review']])
new_df = new_df[['jumlah_pasien','avg_review',
'total_revenue','total_profit']]
return new_df

def predict_score(data):
    # Load the pipeline from .pkl file
    loaded_pipeline = joblib.load('model_pipeline.pkl')

    # Preprocess the input data
    processed_data = preprocess_data(data)

    # Make predictions using the loaded pipeline
    predictions = loaded_pipeline.predict(processed_data)

    return predictions

def run():
    st.title('Hospital Performance Index Score Prediction')

    # Upload CSV file
    uploaded_file = st.file_uploader("Upload CSV file",
type=['csv'])

    if uploaded_file is not None:
        # Read the uploaded CSV file
        data = pd.read_csv(uploaded_file)

```

```

# Show the uploaded data
st.write('Uploaded Data:')
st.write(data)

# Make predictions
predictions = predict_score(data)

# Show predictions
st.write('Predictions:')
st.write(predictions)

if __name__ == '__main__':
    run()

```

Kode `predict.py` melakukan prediksi berdasarkan *file* yang di-upload. *File* berbentuk `*.csv` dan terdiri atas data per hari dari transaksi rumah sakit. Jadi, dengan adanya *deployment* ini, *user* tidak perlu menghitung sendiri nilai rata-rata per bulan dari masing-masing indikatornya. *User* hanya perlu mengunggah *file csv* berisi transaksi harian yang minimal terdiri dari kolom `date_out`, `patient_name`, `age`, `gender`, `review_name`, `branch_name`, `revenue`, `profit`, dan `cogs`.

ml_fp.py

```

import os
import streamlit as st
import pandas as pd
from pandasai import SmartDataframe
from pandasai.callbacks import BaseCallback
from pandasai.llm import OpenAI
from pandasai.responses.response_parser import ResponseParser


class StreamlitCallback(BaseCallback):
    def __init__(self, container) -> None:
        """Initialize callback handler."""
        self.container = container

    def on_code(self, response: str):
        self.container.code(response)

```

```

class StreamlitResponse(ResponseParser):
    def __init__(self, context) -> None:
        super().__init__(context)

    def format_dataframe(self, result):
        st.dataframe(result["value"])
        return

    def format_plot(self, result):
        st.image(result["value"])
        return

    def format_other(self, result):
        st.write(result["value"])
        return

    def load_data(path):
        df = pd.read_csv(path)
        return df

st.write("# Chat with Mulia Hospital Dataset 📈")

df = load_data('cleaned_data.csv')

with st.expander("👁️ Dataframe Preview"):
    st.write(df.tail(3))

query = st.text_area("👤 Ask me!")
container = st.container()

if query:
    llm = OpenAI(
        api_token=os.environ["OPENAI_API_KEY"],
        model="gpt-3.5-turbo-16k"
    )
    query_engine = SmartDataframe(
        df,
        config={
            "llm": llm,
            "response_parser": StreamlitResponse,
        },
    )

```

```

answer = query_engine.chat(query)
if answer != None:
    st.write("Sorry, I can't answer the question yet")

```

Kode tersebut adalah kode untuk halaman *chatbot*, model diberikan file `cleaned_data.csv` untuk dapat dipelajari sehingga *user* dapat lebih mudah mendapatkan informasi yang diinginkan dari `cleaned_data.csv`.

F. Data Visualization

- a. Menentukan *metrics* yang ingin dicari, visualisasi yang ingin ditampilkan berdasarkan data yang dimiliki, dan desain *dashboard* yang ingin ditampilkan.
- b. Membuat desain *background* Power BI di Canva.
- c. Mengimpor data dari AWS RDS ke Power BI.
- d. Mengecek data yang sudah diimpor apakah sesuai dan sudah bersih.
- e. Menghubungkan *relationship* antara satu tabel dengan tabel lainnya sesuai dengan diagram ERD yang telah dibuat.
- f. Mengimpor *background* yang telah dibuat di Canva ke dalam Power BI dan tentukan ukuran dari setiap halaman.
- g. Membuat visualisasi berdasarkan konsep yang dibuat di awal
- h. Menambahkan *measurement* untuk *metrics* yang dicari menggunakan fitur DAX di Power BI
 - **Date** merupakan *table measurement* untuk menggabungkan rentang waktu *date in* dan *date out* di table HospitalTrx supaya periode waktu yang digunakan selaras.

```

Date =
VAR MinDate = MINX(
    UNION(
        SELECTCOLUMNS('HospitalTrx', "Date",
'HospitalTrx'[date_in]),
        SELECTCOLUMNS('HospitalTrx', "Date",
'HospitalTrx'[date_out])
    ),
    [Date]
)
VAR MaxDate = MAXX(
    UNION(
        SELECTCOLUMNS('HospitalTrx', "Date",
'HospitalTrx'[date_in]),

```

```

        SELECTCOLUMNS('HospitalTrx', "Date",
'HospitalTrx'[date_out])
),
[Date]
)
)
RETURN
ADDCOLUMNS(
CALENDAR(MinDate, MaxDate),
"Year", YEAR([Date]),
"Quarter", "Q" & FORMAT(QUARTER([Date]), "0"),
"Month", FORMAT([Date], "MMMM"),
"Month Number", MONTH([Date]),
"Weekday", FORMAT([Date], "dddd"),
"Day", DAY([Date])
)
)

```

Penjelasan syntax:

MinDate:

Variabel ini menghitung tanggal minimum dari kolom 'date_in' dan 'date_out' pada tabel 'HospitalTrx' menggunakan fungsi MINX. Fungsi ini membuat tabel sementara dengan menggabungkan dua kolom tanggal menggunakan fungsi UNION dan kemudian mengekstrak tanggal minimum atau tanggal paling awal.

MaxDate:

Variabel ini menghitung tanggal maksimum dengan cara yang sama seperti MinDate, namun mengekstrak tanggal maksimum atau tanggal paling akhir.

ADDCOLUMNS:

Fungsi ini membuat tabel baru dengan menambahkan kolom-kolom yang dihitung ke tanggal-tanggal yang dihasilkan oleh fungsi CALENDAR. Setiap baris dalam tabel yang dihasilkan mewakili sebuah tanggal.

CALENDAR (MinDate, MaxDate):

Fungsi ini menghasilkan tabel dengan rentang tanggal dari MinDate hingga MaxDate.

"Year":

Fungsi ini bertujuan untuk menghitung tahun untuk setiap tanggal menggunakan fungsi YEAR.

"Quarter":

Fungsi ini bertujuan untuk menghitung kuartal setiap tanggal menggunakan fungsi QUARTER dan FORMAT.

"Month":

Fungsi ini bertujuan untuk menghitung nama bulan untuk setiap tanggal menggunakan fungsi FORMAT.

"Month Number":

Fungsi ini bertujuan untuk menghitung nomor bulan untuk setiap tanggal menggunakan fungsi MONTH.

"Weekday":

Fungsi ini bertujuan untuk menghitung nama hari dalam minggu untuk setiap tanggal menggunakan fungsi FORMAT.

"Day":

Fungsi ini bertujuan untuk menghitung hari dalam bulan untuk setiap tanggal menggunakan fungsi DAY.

- *Age Group*

```
Age_Group =  
SWITCH(  
    TRUE(),  
    cleaned_data[age] >= 0 && cleaned_data[age] <= 18, "0-18",  
    cleaned_data[age] >= 19 && cleaned_data[age] <= 35, "19-  
    35",  
    cleaned_data[age] >= 36 && cleaned_data[age] <= 50, "36-  
    50",  
    cleaned_data[age] >= 51 && cleaned_data[age] <= 65, "51-  
    65",  
    cleaned_data[age] >= 66, "66+",  
    "Unknown"  
)
```

Penjelasan syntax:

Ekspresi DAX ini mengategorikan pasien ke dalam kelompok umur yang berbeda berdasarkan usia. Ekspresi ini menggunakan fungsi SWITCH(TRUE(), ...) untuk mengevaluasi setiap kondisi secara berurutan dan mengembalikan kelompok umur yang sesuai ketika suatu kondisi terpenuhi. Sebagai contoh, apabila seorang pasien berumur 12, maka pasien tersebut akan masuk dalam kelompok usia 0-18. Adapun jika tidak ada kondisi yang terpenuhi, ekspresi ini akan mengembalikan nilai "Unknown".

- *Age Group Versi 2*

```

Age_Group_V2 =
SWITCH(
    TRUE(),
    cleaned_data[age]    >=  0   &&  cleaned_data[age]    <=  18,
"Pediatric",
    cleaned_data[age] >= 19 && cleaned_data[age] <= 65, "Adult",
"Geriatric"
)

```

Penjelasan syntax:

Cara kerja dari ekspresi ini sama seperti ekspresi Age Group sebelumnya, hanya saja klasifikasi dibagi menjadi tiga golongan, yaitu “Pediatric” untuk pasien anak-anak, “Adult” untuk pasien dewasa, dan “Geriatric” untuk pasien lansia.

- **Average Length of Stay (ALOS)** merupakan rata-rata lama waktu inap dari pasien rawat inap.

```

Average Length of Stay (ALOS) =
VAR AvgLengthOfStay =
    AVERAGEX (
        FILTER (
            'cleaned_data',
            NOT ISBLANK('cleaned_data'[room_type])
        ),
        'cleaned_data'[durasi_rawat]
    )
RETURN
    FORMAT(AvgLengthOfStay, "0") & " " &
    IF(AvgLengthOfStay > 1, "Days", "1 Day")

```

Penjelasan syntax:

Ekspresi DAX ini berjalan dengan cara memfilter data hanya untuk menyertakan baris di mana value dari *room_type* tidaklah kosong. Selanjutnya, ekspresi ini menghitung rata-rata dari kolom *durasi_rawat* pada data yang sudah difilter dan mengembalikan hasil rata-rata dengan format "n Days" jika hasil lebih dari 1, atau "1 Day" jika hasil adalah 1 atau kurang.

- **Average Visit** merupakan rata-rata jumlah kunjungan yang dilakukan seorang pasien.

```

Avg Visit =
VAR TotalVisits = COUNTROWS('HospitalTrx')

```

```

VAR UniquePatients = COUNTROWS(DISTINCT('HospitalTrx'[id_patient]))
VAR AvgVisits = DIVIDE(TotalVisits, UniquePatients)
VAR RoundedVisits = CEILING(AvgVisits, 1)
RETURN
IF (
    RoundedVisits = 1,
    "1 Time",
    FORMAT(RoundedVisits, "0") & " Times"
)

```

Penjelasan syntax:

Query DAX ini menghitung rata-rata jumlah kunjungan pasien dengan mencari jumlah baris di tabel HospitalTrx dan jumlah baris pasien dengan nomor ID pasien secara unik. Setelah kedua nilai ini diperoleh, dilakukan pembagian yang hasilnya dibulatkan ke angka terdekat dalam format *string*. Apabila hasil dari rata-rata adalah satu (1), maka hasilnya akan mengembalikan nilai "1 Time". Adapun jika hasil rata-ratanya lebih dari 1, maka akan mengembalikan nilai n atau angka yang telah dibulatkan, diikuti dengan kata "Times" di belakangnya.

- **Average Revenue per User** merupakan rata-rata *revenue* yang didapatkan dari seorang pasien.

```

Avg Revenue per User =
SUM(cleaned_data[Total_Revenue]) / COUNTROWS ( DISTINCT (
'HospitalTrx'[id_patient] ) )

```

Penjelasan syntax:

DAX query ini berjalan dengan cara menjumlahkan total *revenue* dari tabel *cleaned_data*, menghitung jumlah nomor ID pasien secara unik, kemudian membagi total *revenue* dengan jumlah ID pasien secara unik.

- **Net Profit Margin** merupakan persentase dari *Total Profit (Gross Profit)* dibagi oleh *Total Revenue*.

```

Net Profit Margin (NPM) =
SUM(cleaned_data[Total Profit (Gross)]) /
SUM(cleaned_data[Total_Revenue])

```

Penjelasan syntax:

Query DAX ini bertujuan untuk menghitung NPM dengan membagi total gross profit dengan total revenue di tabel *cleaned_data*.

- **Patient Satisfaction Score** merupakan nilai skor kepuasan pasien selama menjalani perawatan di rumah sakit. Adapun rentang skor 1-5, di mana skor 1 menunjukkan “sangat tidak puas” dan 5 menunjukkan “sangat puas”.

```
Patient Satisfaction Score =
DIVIDE (
    CALCULATE (
        COUNTROWS ( cleaned_data ),
        FILTER ( HospitalTrx, HospitalTrx[id_review] >= 4 )
    ),
    CALCULATE ( COUNTROWS ( 'cleaned_data' ) )
)
```

Penjelasan syntax:

DAX query ini menghitung Skor Kepuasan Pasien dengan cara membagi jumlah *review* positif (*rating* yang bernilai 4 atau lebih) dengan total jumlah *review*.

- **Normalized Patient Satisfaction Score** merupakan normalisasi dari skor kepuasaan pasien.

```
Normalized_Patient_Satisfaction_Score =
VAR MaxScore = 5
VAR AvgScore =
    AVERAGEX(
        cleaned_data,
        SWITCH(
            TRUE(),
            cleaned_data[review_name] = "Sangat Tidak Puas", 1,
            cleaned_data[review_name] = "Tidak Puas", 2,
            cleaned_data[review_name] = "Netral", 3,
            cleaned_data[review_name] = "Puas", 4,
            cleaned_data[review_name] = "Sangat Puas", 5
        )
    )
RETURN
    DIVIDE(
        AvgScore,
        MaxScore
    )
```

Penjelasan syntax:

VAR MaxScore:

Variabel ini mendefinisikan skor maksimum yang mungkin. Dalam kasus ini, variabel telah di-set nilainya ke 5, karena hal tersebut merupakan nilai maksimum dalam skor *review* yang ada (5 - “Sangat Puas”).

VAR AvgScore = AVERAGEX(cleaned_data, SWITCH(...)):

Variabel ini menghitung skor rata-rata dengan mengulang setiap baris dalam tabel 'cleaned_data' dan memberikan nilai numerik berdasarkan kolom 'review_name' dengan menggunakan fungsi SWITCH.

DIVIDE (AvgSkor, MaxSkor):

Fungsi ini membagi skor rata-rata (AvgSkor) dengan skor maksimum (MaxSkor) untuk menormalkan skor antara 0 dan 1.

- *Normalized Profit* merupakan normalisasi dari total profit

```
Normalized_Profit =
VAR TotalProfit = SUM(cleaned_data[total_revenue])
VAR MaxProfit = CALCULATE(
    MAXX(
        SUMMARIZE(cleaned_data, cleaned_data[branch_name],
        "BranchRevenue", SUM(cleaned_data[total_revenue])),
        [BranchRevenue]
    )
)
RETURN
DIVIDE(TotalProfit, MaxProfit)
```

Penjelasan syntax:

VAR TotalProfit = SUM(cleaned_data[Total Profit (Gross)]):

Variabel ini menghitung total gross profit dengan menjumlahkan semua nilai dalam kolom 'Total Profit (Gross)' dari tabel 'cleaned_data'.

VAR MaxProfit = CALCULATE(MAXX(SUMMARIZE(...))):

Variabel ini menghitung profit maksimum di antara semua cabang.

SUMMARIZE mengelompokkan data berdasarkan 'branch_name' dan menghitung total gross profit untuk setiap cabang.

MAXX mengembalikan nilai maksimum dari data yang telah diringkas, yang mewakili profit maksimum di antara semua cabang.

CALCULATE mengubah konteks perhitungan untuk menghapus semua filter yang diterapkan pada tabel 'cleaned_data'.

DIVIDE (TotalProfit, MaxProfit):

Fungsi membagi total gross profit setiap cabang (TotalProfit) dengan gross profit maksimum di antara semua cabang (MaxProfit).

- *Normalized Total Patients* merupakan normalisasi dari total patients.

```

Normalized_Total_Patients =
VAR TotalPatients = COUNT(cleaned_data[id_trx])
VAR MaxPatients = CALCULATE(
    MAXX(
        SUMMARIZE(cleaned_data, cleaned_data[branch_name],
        "BranchTotal", COUNT(cleaned_data[id_trx])),
        [BranchTotal]
    )
)
RETURN
DIVIDE(TotalPatients, MaxPatients)

```

Penjelasan syntax:

VAR TotalTransactions = COUNT(cleaned_data[id_trx]):

Variabel ini menghitung total jumlah transaksi (atau pasien) dengan menghitung nilai-nilai unik dalam kolom 'id_trx' dari tabel 'cleaned_data'.

**VAR MaxTransactions =
CALCULATE(MAXX(SUMMARIZE(...))):**

SUMMARIZE mengelompokkan data berdasarkan 'branch_name' dan menghitung jumlah transaksi untuk setiap cabang.

MAXX mengembalikan nilai maksimum dari data yang telah diringkas, yang mewakili jumlah maksimum transaksi di antara semua cabang.

CALCULATE mengubah konteks perhitungan untuk menghapus semua filter yang diterapkan pada tabel 'cleaned_data'.

DIVIDE (TotalTransactions, MaxTransactions):

Fungsi ini membagi total jumlah transaksi setiap cabang (TotalTransactions) dengan jumlah maksimum transaksi di antara semua cabang (MaxTransactions).

- **Hospital Performance Index** merupakan kinerja rumah sakit yang didapatkan dari data total pasien, total revenue, dan tingkat kepuasan pasien yang telah dinormalisasi dan diberi pembobotan.

```

Hospital_Performance_Index =
(0.2 * [Normalized_Total_Patients]) +
(0.45 * [Normalized_Profit]) +
(0.35 * [Normalized_Patient_Satisfaction_Score])

```

- i. Menarik *business insights* dan informasi-informasi yang relevan.

BAB IV

HASIL DAN PEMBAHASAN

Berikut merupakan *dataset* dari setiap tabel yang telah dinormalisasi dan memenuhi standar 3F:

a) Tabel HospitalTrx

	id_trx [PK] integer	date_in date	date_out date	id_branch character varying	id_patient character varying	id_hospital_ci character varying	id_room character varying	id_doctor character varying	id_surgery character varying	id_lab character varying	id_drug character varying	drug_qty character varying	admin_price character varying	cogs character varying	id_payment character varying	id_review character varying
1	10001	2021-04-30	2021-05-01	1	7644	1	1	3	1	-	12	2	50000	6692139	1	2
2	10006	2020-05-08	2020-05-12	2	9315	1	3	3	2	-	5	3	50000	10422346	2	2
3	10013	2022-10-11	2022-10-14	1	6450	1	4	4	2	5	8	5	50000	10025324	2	5
4	10015	2020-11-01	2020-11-02	1	540	1	4	4	3	1	9	4	50000	13612562	1	2
5	10031	2021-11-25	2021-11-27	3	2988	1	3	3	1	1	5	4	50000	790956	1	1
6	10033	2023-06-25	2023-06-26	1	916	1	4	5	-	5	10	5	50000	3497738	1	1
7	10046	2022-08-17	2022-08-18	2	6360	1	3	1	-	-	12	4	50000	4527478	1	4
8	10042	2020-06-01	2020-06-04	2	3603	1	3	3	-	3	3	1	50000	3523173	1	3
9	10063	2023-01-15	2023-01-19	1	4117	1	3	5	1	4	1	1	50000	4983097	1	3
10	10079	2023-12-05	2023-12-07	3	6133	1	1	3	3	-	2	3	50000	15350976	1	5
11	10082	2020-03-31	2020-04-05	3	7091	1	2	2	-	3	4	2	50000	4552100	1	4
12	10091	2023-02-10	2023-02-13	2	1347	1	1	3	2	3	9	4	50000	9957408	1	3
13	10109	2021-01-10	2021-01-12	1	6766	1	2	5	2	-	9	4	50000	11860847	1	2
14	10110	2020-05-08	2020-05-09	1	639	1	1	4	1	-	5	3	50000	7868821	2	3
15	10119	2023-02-22	2023-02-26	3	306	1	4	3	1	5	10	4	50000	6804282	1	2
16	10128	2022-02-17	2022-02-22	1	5945	1	3	2	2	-	7	2	50000	9661205	1	5
17	10131	2020-12-26	2020-12-27	3	6003	1	4	2	3	-	3	3	50000	15554048	1	5
18	10143	2022-06-06	2022-06-10	2	597	1	3	4	1	2	9	2	50000	6529135	2	5
19	10147	2021-03-13	2021-03-14	2	2348	1	1	2	2	3	7	4	50000	11597162	1	3
20	10153	2020-08-20	2020-08-22	3	7927	1	3	2	1	3	4	1	50000	4278761	2	4
21	10165	2023-01-09	2023-01-09	3	2896	2	-	4	-	3	9	2	50000	3135903	2	1

b) Tabel Branch

	branch_id [PK] integer	branch_name character varying
1	1	RSMA
2	2	RSMD
3	3	RSMS

c) Tabel Doctor

	doctor_id [PK] integer	doctor_type character varying	doctor_price money
1	1	Bedah	\$300,000.00
2	2	Gigi	\$300,000.00
3	3	Kandungan	\$300,000.00
4	4	Penyakit Dalam	\$300,000.00
5	5	Umum	\$200,000.00

d) Tabel DrugType

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations. Below the toolbar is a table titled 'drugs'. The table has four columns: 'drug_type_id' (PK integer), 'drug_type' (character varying), 'drug_price' (money), and a fourth column which appears to be a timestamp or date field. There are four rows of data:

	drug_type_id [PK] integer	drug_type character varying	drug_price money
1	1	Antibiotik	\$75,000.00
2	2	Pereda Nyeri	\$50,000.00
3	3	Umum	\$40,000.00
4	4	Vitamin	\$110,000.00

e) Tabel Drugs

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations. Below the toolbar is a table titled 'DrugDetails'. The table has three columns: 'drug_id' (PK integer), 'drug_brand' (character varying), and 'drug_type_id' (integer). There are twelve rows of data:

	drug_id [PK] integer	drug_brand character varying	drug_type_id integer
1	1	Amoxicillin	1
2	2	Azithromycin	1
3	3	Blackmores	4
4	4	Calpol	3
5	5	Ciprofloxacin	1
6	6	Diclofenac	2
7	7	Enervon-C	4
8	8	Holland & Barrett	4
9	9	Naproxen	2
10	10	Panadol	3
11	11	Paramex	3
12	12	Tramadol	2

f) Tabel HospitalCare

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations. Below the toolbar is a table titled 'HospitalCare'. The table has three columns: 'hospitalcare_id' (PK integer), 'hospital_care' (character varying), and 'infusion_cost' (money). There are two rows of data:

	hospitalcare_id [PK] integer	hospital_care character varying	infusion_cost money
1	1	Rawat Inap	\$165,000.00
2	2	Rawat Jalan	\$0.00

g) Tabel Lab

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations. Below the toolbar is a table titled 'Lab'. The table has three columns: 'lab_id' (PK integer), 'lab_name' (character varying), and 'lab_price' (money). There are five rows of data:

	lab_id [PK] integer	lab_name character varying	lab_price money
1	1	Hematologi	\$90,000.00
2	2	Kimia Darah	\$195,000.00
3	3	Rontgen	\$150,000.00
4	4	Serologi	\$200,000.00
5	5	Urinalisa	\$80,000.00

h) Tabel Patient

	patient_id [PK] integer	patient_name character varying	gender character varying	age integer
1	1	Budi Sitompul, M.Ak	Laki-laki	57
2	2	Zelda Padmasari	Laki-laki	17
3	3	Ratna Uwais, S.Kom	Perempuan	42
4	4	Iriana Kurniawan, S.Ked	Perempuan	38
5	5	Ade Winarsih	Perempuan	30
6	6	Darijan Nasyiah	Perempuan	77
7	7	R.A. Dian Nuraini, S.Sos	Laki-laki	42
8	8	Drs. Michelle Sinaga, M.F...	Laki-laki	59
9	9	Drs. Aris Hakim, S.Psi	Perempuan	75
10	10	Kayla Simbolon	Perempuan	27
11	11	Prayoga Permadi	Perempuan	30
12	12	Jati Utama	Perempuan	48
13	13	Fitria Widiastuti	Laki-laki	79
14	14	Hj. Putri Melani, M.Ak	Perempuan	51
15	15	Queen Mulyani	Laki-laki	44

i) Tabel Payment

	payment_id [PK] integer	payment_name character varying
1	1	Asuransi
2	2	Pribadi

j) Tabel Review

	review_id [PK] integer	review_name character varying
1	1	Sangat Tidak Puas
2	2	Tidak Puas
3	3	Netral
4	4	Puas
5	5	Sangat Puas

k) Tabel Room

	room_id [PK] integer	room_type character varying	room_price money	food_price money
1	1	VIP	\$300,000.00	\$150,000.00
2	2	Kelas 1	\$250,000.00	\$110,000.00
3	3	Kelas 2	\$200,000.00	\$80,000.00
4	4	Kelas 3	\$150,000.00	\$50,000.00

I) Tabel Surgery

	surgery_id [PK] integer	surgery_type character varying	surgery_price money
1	1	Kecil	\$4,000,000.00
2	2	Besar	\$8,000,000.00
3	3	Kusus	\$15,000,000.00

m) Tabel cleaned_data

	id_trx [PK] integer	date_in date	date_out date	drug_qty integer	admin_price double precis	cogs double precis	durasi_rawat integer	hospital_care character var	infusion_cost double precis	room_type character var	room_price double precis	food_price double precis	doctor_type character var	doctor_price double precis	surgery_type character var	surgery_price double precis	lab_name character var
1	10001	2021-04-30	2021-05-01	2	50000	6692139	2	Rawat Inap	165000	VIP	300000	150000	Kandungan	300000	Kecil	4000000	[null]
2	10006	2020-05-08	2020-05-12	3	50000	10422346	5	Rawat Inap	165000	Kelas 2	200000	80000	Kandungan	300000	Besar	8000000	[null]
3	10013	2022-10-11	2022-10-14	5	50000	10025324	4	Rawat Inap	165000	Kelas 3	150000	50000	Penyakit ...	300000	Besar	8000000	Urinalisa
4	10015	2020-11-01	2020-11-02	4	50000	13612562	2	Rawat Inap	165000	Kelas 3	150000	50000	Penyakit ...	300000	Khusus	15000000	Hematolo...
5	10031	2021-11-25	2021-11-27	4	50000	7990956	3	Rawat Inap	165000	Kelas 2	200000	80000	Kandungan	300000	Kecil	4000000	Hematolo...
6	10033	2023-06-25	2023-06-26	5	50000	3497738	2	Rawat Inap	165000	Kelas 3	150000	50000	Umum	200000	[null]	0	Urinalisa
7	10040	2022-08-17	2022-08-18	4	50000	4527478	2	Rawat Inap	165000	Kelas 2	200000	80000	Bedah	300000	[null]	0	[null]
8	10042	2020-06-01	2020-06-04	1	50000	3523173	4	Rawat Inap	165000	Kelas 2	200000	80000	Kandungan	300000	[null]	0	Rontgen
9	10063	2023-01-15	2023-01-19	1	50000	4983097	5	Rawat Inap	165000	Kelas 2	200000	80000	Umum	200000	Kecil	4000000	Serologi
10	10079	2023-12-05	2023-12-07	3	50000	15350976	3	Rawat Inap	165000	VIP	300000	150000	Kandungan	300000	Khusus	15000000	[null]
11	10082	2020-03-31	2020-04-05	2	50000	4552100	6	Rawat Inap	165000	Kelas 1	250000	110000	Gigi	300000	[null]	0	Rontgen
12	10091	2023-02-10	2023-02-13	4	50000	9957408	4	Rawat Inap	165000	VIP	300000	150000	Kandungan	300000	Besar	8000000	Rontgen
13	10109	2021-01-10	2021-01-12	4	50000	11860847	3	Rawat Inap	165000	Kelas 1	250000	110000	Umum	200000	Besar	8000000	[null]
14	10110	2020-05-08	2020-05-09	3	50000	7868821	2	Rawat Inap	165000	VIP	300000	150000	Penyakit ...	300000	Kecil	4000000	[null]
15	10119	2023-02-22	2023-02-26	4	50000	6804282	5	Rawat Inap	165000	Kelas 3	150000	50000	Kandungan	300000	Kecil	4000000	Urinalisa
16	10128	2022-02-17	2022-02-22	2	50000	9661205	6	Rawat Inap	165000	Kelas 2	200000	80000	Gigi	300000	Besar	8000000	[null]
17	10131	2020-12-26	2020-12-27	3	50000	15554048	2	Rawat Inap	165000	Kelas 3	150000	50000	Gigi	300000	Khusus	15000000	[null]
18	10143	2022-06-06	2022-06-10	2	50000	6529135	5	Rawat Inap	165000	Kelas 2	200000	80000	Penyakit ...	300000	Kecil	4000000	Kimia Dar...
19	10147	2021-03-13	2021-03-14	4	50000	11597162	2	Rawat Inap	165000	VIP	300000	150000	Gigi	300000	Besar	8000000	Rontgen
20	10153	2020-08-20	2020-08-22	1	50000	4278761	3	Rawat Inap	165000	Kelas 2	200000	80000	Gigi	300000	Kecil	4000000	Rontgen
21	10165	2023-01-09	2023-01-09	2	50000	3135093	1	Rawat Jl	n	Rawat Jl	n	n	Penyakit ...	500000	[null]	0	Rontgen

	lab_price double precis	drug_brand character var	drug_type character var	drug_price double precis	branch_name character varyin	patient_name character varyin	gender character var	age integer	payment_name character varyin	review_name character vary	total_cost double precis	revenue double precis
0	Tramadol	Pereda N...	Analgesik	50000	RSMA	Ella Habibi	Perempuan	49	Asuransi	Tidak Puas	5980000	-712139
0	Ciproflox...	Antibiotik	Antibiotik	75000	RSMD	Siti Samosir...	Laki-laki	23	Pribadi	Tidak Puas	12000000	1577654
80000	Holland &... Vitamin	Vitamin	Vitamin	110000	RSMA	Cut Ellis Pra...	Perempuan	49	Asuransi	Sangat Pu...	11340000	1314676
90000	Naproxen	Pereda N...	Analgesik	50000	RSMA	Jasmin Lail...	Perempuan	76	Asuransi	Tidak Puas	16670000	3057438
90000	Ciproflox...	Antibiotik	Antibiotik	75000	RSMS	Ellis Widodo...	Perempuan	39	Asuransi	Sangat Tid...	6675000	-1315956
80000	Panadol	Umum	Analgesik	40000	RSMA	Rika Januar	Perempuan	34	Asuransi	Sangat Tid...	1460000	-2037738
0	Tramadol	Pereda N...	Analgesik	50000	RSMD	Titi Astuti	Perempuan	37	Asuransi	Puas	1740000	-2787478
150000	Blackmorn...	Vitamin	Vitamin	110000	RSMD	Cut Anita Ku...	Perempuan	21	Asuransi	Netral	3290000	-233173
200000	Amoxicillin	Antibiotik	Antibiotik	75000	RSMA	Wage Nasyl...	Laki-laki	28	Asuransi	Netral	7550000	2566903
0	Azithrom...	Antibiotik	Antibiotik	75000	RSMS	Janet Safitri	Laki-laki	56	Asuransi	Sangat Pu...	18020000	2669024
150000	Calpol	Umum	Analgesik	40000	RSMS	Cengkir Usa...	Laki-laki	33	Asuransi	Puas	5230000	677900
150000	Naproxen	Pereda N...	Analgesik	50000	RSMD	Hafshah Sa...	Perempuan	66	Asuransi	Netral	12060000	2102592
0	Naproxen	Pereda N...	Analgesik	50000	RSMA	Candrakant...	Laki-laki	30	Asuransi	Tidak Puas	10425000	-1435847
0	Ciproflox...	Antibiotik	Antibiotik	75000	RSMA	Hardi Sapto...	Perempuan	66	Pribadi	Netral	6105000	-1763821
80000	Panadol	Umum	Analgesik	40000	RSMS	Wadi Usamah	Laki-laki	58	Asuransi	Tidak Puas	7615000	810718
0	Enervon-C	Vitamin	Vitamin	110000	RSMA	Amelia Hary...	Laki-laki	46	Asuransi	Sangat Pu...	12740000	3078795
0	Blackmorn...	Vitamin	Vitamin	110000	RSMS	Sutan Gada ...	Laki-laki	25	Asuransi	Sangat Pu...	16710000	1155952
195000	Naproxen	Pereda N...	Analgesik	50000	RSMD	Ir. Ayu Sapu...	Perempuan	43	Pribadi	Sangat Pu...	8070000	1540865
150000	Enervon-C	Vitamin	Vitamin	110000	RSMD	Liman Laksi...	Laki-laki	57	Asuransi	Netral	10470000	-1127162
150000	Calpol	Umum	Analgesik	40000	RSMS	Drs. Labuh ...	Perempuan	63	Pribadi	Puas	6475000	2196239
150000	Naproxen	Pereda N...	Analgesik	50000	RSMS	Dr. Gangsar ...	Laki-laki	69	Pribadi	Sangat Tid...	600000	-2535903

n) Tabel patient_records

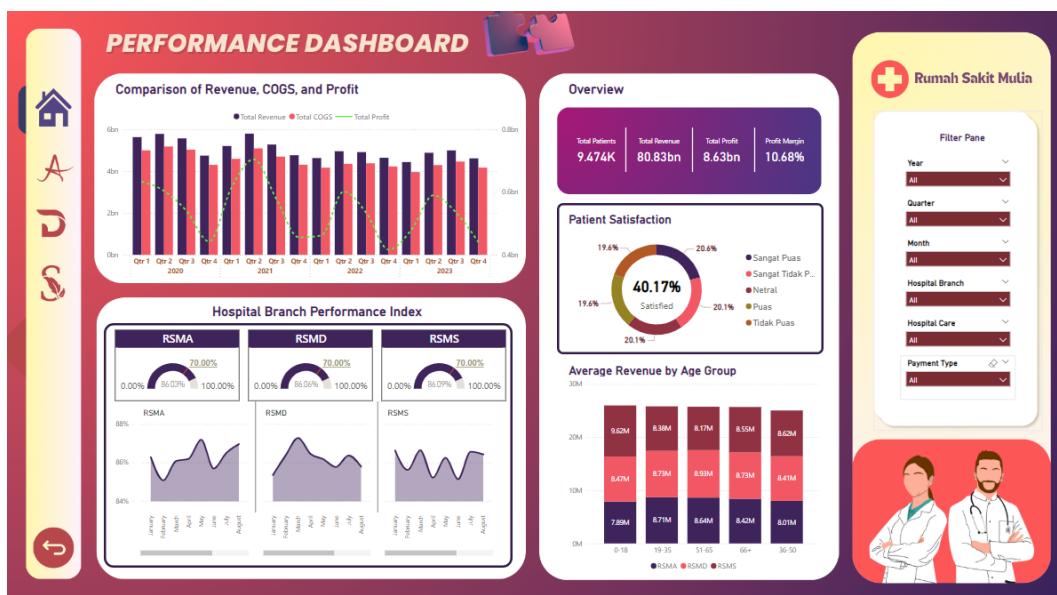
	id [PK] character varying (10)	date_in character varying (10)	date_out character varying (10)	branch character var	name character varying (255)	age integer	gender character varyin	hospital_care character varyin	room character var	doctor character varying (10)	surgery character vari	lab character varyin	drug_types character varyin	drug_brands character varying (10)
1	10001	30/04/2021	01/05/2021	RSMA	Ella Habil	49	Perempuan	Rawat Inap	VIP	Kandungan	Kecil	-	Pereda Nyeri	Tramadol
2	10006	08/05/2020	12/05/2020	RSMD	Siti Samosir, M.M.	23	Laki-laki	Rawat Inap	Kelas 2	Kandungan	Besar	-	Antibiotik	Ciprofloxacin
3	10013	11/10/2022	14/10/2022	RSMA	Cut Ellis Pradipta, S.E.	49	Perempuan	Rawat Inap	Kelas 3	Penyakit Dalam	Besar	Urinalisa	Vitamin	Holland & Barret
4	10015	01/11/2020	02/11/2020	RSMA	Jasmin Lailasari, S.T.	76	Perempuan	Rawat Inap	Kelas 3	Penyakit Dalam	Kusus	Hematologi	Pereda Nyeri	Naproxen
5	10031	25/11/2021	27/11/2021	RSMA	Ellis Widodo, M.Kom.	39	Perempuan	Rawat Inap	Kelas 2	Kandungan	Kecil	Hematologi	Antibiotik	Ciprofloxacin
6	10033	25/06/2023	26/06/2023	RSMA	Rika Januar	34	Perempuan	Rawat Inap	Kelas 3	Umum	-	Urinalisa	Umum	Panadol
7	10040	17/08/2022	18/08/2022	RSMD	Titi Astuti	37	Perempuan	Rawat Inap	Kelas 2	Beda	-	-	Pereda Nyeri	Tramadol
8	10042	01/06/2020	04/06/2020	RSMD	Cut Anita Kusumo, S.E.	21	Perempuan	Rawat Inap	Kelas 2	Kandungan	-	Rontgen	Vitamin	Blackmores
9	10063	15/01/2023	19/01/2023	RSMA	Wage Nayidah	28	Laki-laki	Rawat Inap	Kelas 2	Umum	Kecil	Serologi	Antibiotik	Amoxicillin
10	10079	05/12/2023	07/12/2023	RSMS	Janet Safrri	56	Laki-laki	Rawat Inap	VIP	Kandungan	Kusus	-	Antibiotik	Azithromycin
11	10082	31/03/2020	05/04/2020	RSMS	Cengkr Usamah	33	Laki-laki	Rawat Inap	Kelas 1	Gigi	-	Rontgen	Umum	Calpol
12	10091	10/02/2023	13/02/2023	RSMD	Hafshah Samosir	66	Perempuan	Rawat Inap	VIP	Kandungan	Besar	Rontgen	Pereda Nyeri	Naproxen
13	10109	10/01/2021	12/01/2021	RSMA	Candrakanta Putra	30	Laki-laki	Rawat Inap	Kelas 1	Umum	Besar	-	Pereda Nyeri	Naproxen
14	10110	08/05/2020	09/05/2020	RSMA	Hardi Saptono	66	Perempuan	Rawat Inap	VIP	Penyakit Dalam	Kecil	-	Antibiotik	Ciprofloxacin
15	10119	22/02/2023	26/02/2023	RSMS	Wadi Usamah	58	Laki-laki	Rawat Inap	Kelas 3	Kandungan	Kecil	Urinalisa	Umum	Panadol
16	10128	17/02/2022	22/02/2022	RSMA	Amelia Haryanto	46	Laki-laki	Rawat Inap	Kelas 2	Gigi	Besar	-	Vitamin	Enervon-C
17	10131	26/12/2020	27/12/2020	RSMS	Sutan Gada Saputra	25	Laki-laki	Rawat Inap	Kelas 3	Gigi	Kusus	-	Vitamin	Blackmores
18	10143	06/06/2022	10/06/2022	RSMD	Ir. Ayu Saputra, S.E.	43	Perempuan	Rawat Inap	Kelas 2	Penyakit Dalam	Kecil	Kimia Darah	Pereda Nyeri	Naproxen
19	10147	13/03/2021	14/03/2021	RSMD	Liman Lakista	57	Laki-laki	Rawat Inap	VIP	Gigi	Besar	Rontgen	Vitamin	Enervon-C
20	10153	20/08/2020	22/08/2020	RSMS	Drs. Labuh Firmansyah, M.Kom.	63	Perempuan	Rawat Inap	Kelas 2	Gigi	Kecil	Rontgen	Umum	Calpol
21	10165	09/01/2023	09/01/2023	RSMS	Dr. Gangsar Waskita, S.H.	69	Laki-laki	Rawat Jalan	-	Penyakit Dalam	-	Rontgen	Pereda Nyeri	Naproxen

drug_qty character var	food character var	admin character var	cogs character var	payment character var	review character varying (2)
2	150000	50000	6692139	Asuransi	Tidak Puas
3	80000	50000	10422346	Pribadi	Tidak Puas
5	50000	50000	10025324	Pribadi	Sangat Puas
4	50000	50000	13612562	Asuransi	Tidak Puas
4	80000	50000	7990956	Asuransi	Sangat Tidak Puas
5	50000	50000	3497738	Asuransi	Sangat Tidak Puas
4	80000	50000	4527478	Asuransi	Puas
1	80000	50000	3523173	Asuransi	Netral
1	80000	50000	4983097	Asuransi	Netral
3	150000	50000	15350976	Asuransi	Sangat Puas
2	110000	50000	4552100	Asuransi	Puas
4	150000	50000	9957408	Asuransi	Netral
4	110000	50000	11860847	Asuransi	Tidak Puas
3	150000	50000	7868821	Pribadi	Netral
4	50000	50000	6804282	Asuransi	Tidak Puas
2	80000	50000	9661205	Asuransi	Sangat Puas
3	50000	50000	15554048	Asuransi	Sangat Puas
2	80000	50000	6529135	Pribadi	Sangat Puas
4	150000	50000	11597162	Asuransi	Netral
1	80000	50000	4278761	Pribadi	Puas
2	-	50000	3135903	Pribadi	Sangat Tidak Puas

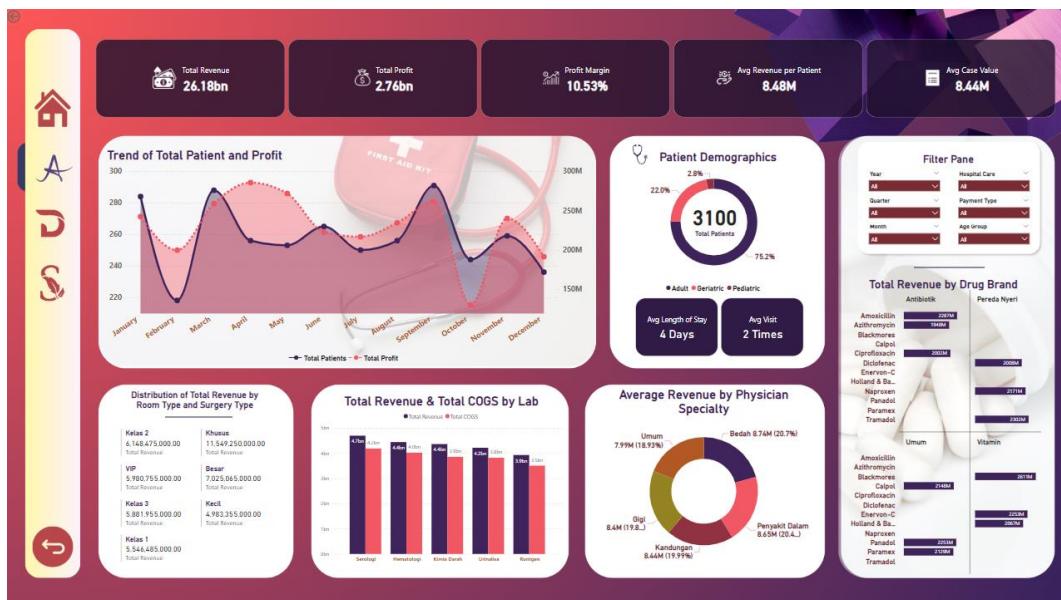
BUSINESS INSIGHTS

Tahapan untuk mendapatkan *business insights* kami lakukan dengan visualisasi pada *dashboard* menggunakan Power BI. *Dashboard* yang telah kami buat dapat dilihat pada [link berikut ini](#) (drive) dan [ini](#) (report).

Visualisasi di *dashboard* kami bagi menjadi empat halaman, halaman yang pertama ialah “*Home*” berisikan rangkuman *metrics* dari ketiga cabang Rumah Sakit, dan tiga halaman lainnya merupakan visualisasi yang lebih detail dari masing-masing cabang Rumah Sakit.



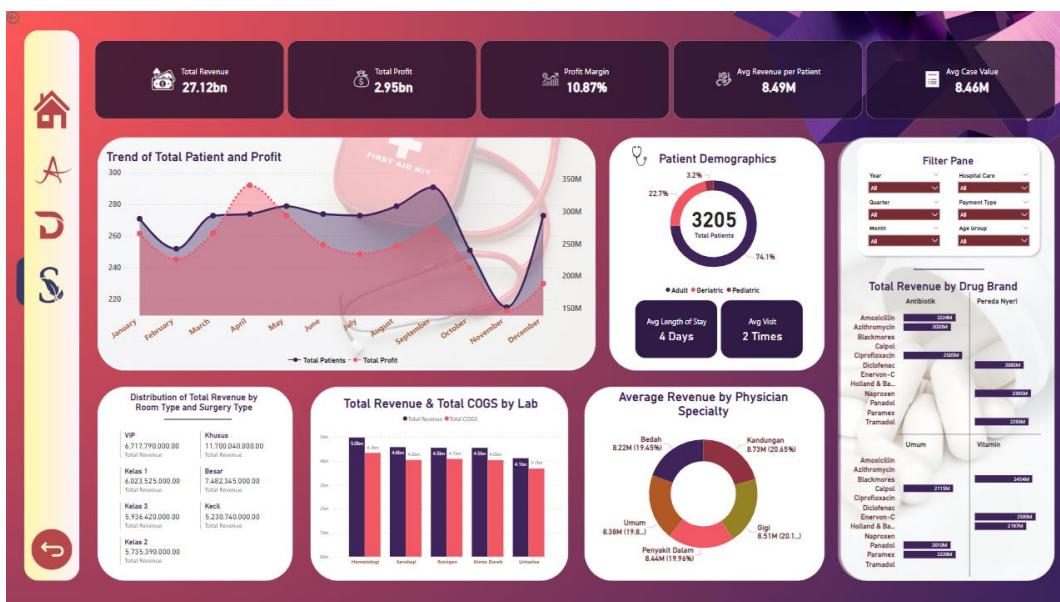
Gambar Page Home



Gambar Page Rumah Sakit Mulia Anggrek (RSMA)



Gambar Page Rumah Sakit Mulia Duri (RSMD)



Gambar Page Rumah Sakit Mulia Simatupang (RSMS)

Business insights atau informasi yang kami dapatkan berdasarkan *dashboard* ialah:

1. Selama periode 2020 – 2024 Rumah Sakit Mulia melayani 9474 pasien, mendapatkan *revenue* 80.81 Miliar rupiah, dengan *profit* 8.63 Miliar rupiah, dan *profit margin* 10.68%. Dengan rincian, cabang RSMA menyumbang *revenue* 26.16 Miliar rupiah dan *profit* 2.76 Miliar rupiah dari total pasien 3100 orang. Cabang RSMS menyumbang *revenue* 27.12 Miliar rupiah dan *profit* 2.95 Miliar rupiah dari total pasien 3205 orang. Cabang RSMD menyumbang *revenue* 27.53 Miliar rupiah dan *profit* 2.92 Miliar rupiah dari total pasien 3169 orang.
2. Secara menyeluruh, *revenue* dan *profit* terbesar dari Rumah Sakit Mulia didapatkan pada kuartal kedua tahun 2021 dengan nominal sebesar 5.8 Miliar rupiah dan 705 Juta rupiah.
3. Melihat dari segi keuntungan, RSMS mengalami penurunan signifikan sebesar 19.58% (setara dengan Rp 45.947.447) selama lima bulan, dimulai pada Juli 2020. RSMD dan RSMA mengalami penurunan total keuntungan yang dimulai pada bulan Juni 2020, dengan penurunan sebesar 14.15% (setara dengan Rp 31.572.291) selama enam bulan.
4. Total Biaya mulai mengalami penurunan pada bulan April 2022, dengan penurunan sebesar 1.40% (Rp 60.921.724) selama 4 kuartal. Total Biaya mengalami penurunan yang signifikan dari Rp 4.591.106.975 menjadi Rp

4.307.551.695 dengan penurunan yang paling besar terjadi pada periode Januari 2021 hingga Oktober 2021.

5. Semua cabang Rumah Sakit Mulia menunjukkan *profit margin* yang lebih dari 10%. Hal ini mengindikasikan manajemen biaya yang tergolong baik dan dapat membuka peluang untuk lebih mengoptimalkan pengeluaran operasional dan investasi pada teknologi medis yang dapat meningkatkan efisiensi dan kualitas pelayanan.
6. Secara menyeluruh, mayoritas pasien tidak merasa puas dengan pelayanan yang didapatkan. Hal ini terlihat pada tingkat kepuasan pasien yang berada pada angka 40.17%. Adapun jika di-*breakdown* secara per cabang rumah sakit, Rumah Sakit Mulia Anggrek (RSMA) menempati posisi terendah dengan tingkat kepuasan pasien 39.71%, Rumah Sakit Mulia Simatupang (RSMS) berada pada angka 40.25%, dan Rumah Sakit Mulia Duri (RSMD) menempati posisi tertinggi, yakni dengan angka 40.55%. Angka-angka yang diperoleh di sini masih tergolong cukup rendah, dan oleh karenanya perlu adanya peningkatan pelayan yang berfokus pada pengalaman pasien dan perbaikan fasilitas pelayanan di seluruh cabang Rumah Sakit Mulia.
7. Selama periode Januari hingga Desember 2020, cabang rumah sakit RSMS, RSMD, dan RSMA mengalami perbedaan pola dalam volume pasien dan margin keuntungan. RSMS mengalami sedikit peningkatan 0.74% pada jumlah pasien, namun menghadapi penurunan signifikan 29.12% pada keuntungan secara keseluruhan. Di sisi lain, RSMD mengalami pertumbuhan pada kedua indikator tersebut dengan peningkatan 3.20% pada total keuntungan dan 2.31% pada jumlah pasien. Sebaliknya, RSMA mengalami penurunan 21.02% pada total keuntungan dan penurunan 16.90% pada total pasien.
8. Selama periode 2020 – 2024, cabang rumah sakit RSMA mencatatkan *Average Revenue per Patient* sebesar 2.79 Juta rupiah dan *Average Case Value* 2.76 Juta rupiah. Pasien rawat inap yang berobat ke cabang RSMA rata-rata menginap selama 4 hari dan setiap pasien rata-rata berobat sebanyak 2 kali.
9. Untuk cabang rumah sakit RSMS pada periode 2020 – 2024, didapatkan *Average Revenue per Patient* sebesar 8.49 Juta rupiah dan *Average Case Value* 8.46 Juta rupiah. Pasien rawat inap yang berobat ke cabang RSMS rata-rata menginap selama 4 hari dan setiap pasien rata-rata berobat sebanyak 2 kali.

10. Sedangkan, pada cabang RSMD di periode yang sama, tercatat *Average Revenue per Patient* sebesar 8.70 Juta rupiah dan *Average Case Value* 8.69 Juta rupiah. Pasien rawat inap yang berobat ke cabang RSMS rata-rata menginap selama 4 hari dan setiap pasien rata-rata berobat sebanyak 2 kali.
11. Kelompok pasien dengan usia 0-18 tahun memberikan kontribusi besar terhadap pendapatan Rumah Sakit Mulia. Adapun pendapatan rata-rata tertinggi dihasilkan oleh RSMS, dengan rata-rata *revenue* sebesar 9.62 juta rupiah.
12. Selama beberapa tahun terakhir, Rumah Sakit Mulia menunjukkan pertumbuhan keuangan yang relatif stabil, meskipun terdapat penurunan dan kenaikan di beberapa kuartal.
13. Berdasarkan kinerja rumah sakit selama empat (4) tahun terakhir, RSMS memiliki kinerja terbaik dengan indeks kinerja atau indeks prestasi (86.09%), diikuti oleh RSMD (86.06%), dan RSMA (86.03%). Hal ini menunjukkan bahwa RSMS memiliki manajemen paling efisien dan efektif dibandingkan cabang lainnya. Adapun kinerja rumah sakit ini dinilai dari total pasien, total keuntungan (*profit*), dan tingkat kepuasan pasien.
14. Berdasarkan demografi pasien, mayoritas pasien di seluruh cabang rumah sakit adalah pasien dewasa dengan proporsi sekitar 74%-75%, disusul oleh pasien lansia sekitar 22%-23%, dan pediatri atau anak-anak sekitar 2%-3%.
15. Berdasarkan spesialisasi dokter, spesialisasi bedah menunjukkan pendapatan tertinggi di seluruh cabang rumah sakit. Optimalisasi kapasitas dokter perlu dilakukan untuk meningkatkan pelayanan, pendapatan rumah sakit, dan juga kepuasan pasien.
16. Berdasarkan layanan laboratorium, serologi dan hematologi menunjukkan kontribusi pendapatan yang tinggi di seluruh cabang rumah sakit. Optimalisasi dibidang ini juga dapat meningkatkan kualitas pelayanan rumah sakit.
17. Berdasarkan obat-obatan, merek obat Enervon-C, Azithromycin, dan Blackmores menunjukkan penggunaan yang konsisten di seluruh cabang rumah sakit. Adapun dalam segi pendapatan, obat-obatan ini memiliki kontribusi yang signifikan terhadap profitabilitas rumah sakit.
18. Secara menyeluruh, jenis operasi khusus menunjukkan pendapatan tertinggi dibanding jenis operasi lainnya, yakni sekitar 11 hingga 12 miliar rupiah. Adapun jika melihat berdasarkan jenis kamar, kamar kelas 2 dan VIP menunjukkan pendapatan tertinggi.

MACHINE LEARNING

Berikut merupakan pembahasan dari model *machine learning* yang telah dibuat.

File *machine learning* dapat diakses [di link berikut](#).

b. Model Training

Berdasarkan hasil yang diberikan, kita dapat membandingkan kinerja tiga model dengan metrik yang berbeda: RMSE (*Root Mean Squared Error*), MAE (*Mean Absolute Error*), dan R² (*R-squared*) pada set validasi dan uji. Berikut merupakan hasil dari ketiga model tersebut:

1. Base Model Linear Regression:
 - *Test* RMSE: 0.005640423298909966
 - *Test* MAE: 0.00470213683326266
 - *Test* R²: 0.9957070904648211
2. GridSearchCV:
 - *Training* RMSE: 0.0016639990514262552
 - *Training* MAE: 0.0012881417501793512
 - *Training* R²: 0.9998430878063516
 - *Test* RMSE: 0.012263859784308469
 - *Test* MAE: 0.009180273218964564
 - *Test* R²: 0.9797052972892438
3. RandomizedSearchCV:
 - *Training* MSE: 0.00205276827834728
 - *Training* MAE: 0.0016433329088276576
 - *Training* R²: 0.9997612021562718
 - *Test* RMSE: 0.011648308902925918
 - *Test* MAE: 0.009101478434303208
 - *Test* R²: 0.981691443720057

Model	Set	RMSE	MAE	R ²
Base Model Linear Regression	Training	0.0123	0.0065	0.9900
	Test	0.0056	0.0047	0.9957
GridSearchCV	Training	0.0017	0.0013	0.9998
	Test	0.0123	0.0092	0.9797
RandomizedSearchCV	Training	0.0021	0.0016	0.9998
	Test	0.0116	0.0091	0.9817

Dari hasil *test* di atas, dapat ditarik kesimpulan bahwa:

1. Base Model Linear Regression:

- Model ini memiliki performa yang cukup baik dengan nilai R² yang tinggi baik pada *training* maupun *test* set, yang menunjukkan bahwa model mampu menjelaskan variabilitas data dengan baik. Nilai RMSE dan MAE juga lebih rendah dibandingkan dengan model yang lain.
- *Overfitting Indicator*: Tidak ada indikasi overfitting yang jelas karena performa pada *training* dan *test* set relatif konsisten.

2. GridSearchCV:

- Model ini menunjukkan performa yang sangat baik pada *training* set dengan nilai R² yang mendekati 1 dan nilai *error* yang sangat rendah. Namun, performa menurun pada *test* set, dengan peningkatan RMSE dan MAE, serta penurunan R².
- *Overfitting Indicator*: Ada indikasi *overfitting* karena performa pada *training* set sangat baik dibandingkan dengan *test* set.

3. RandomizedSearchCV:

- Model RandomizedSearchCV juga menunjukkan kinerja yang sangat baik pada data *training*, namun sedikit lebih baik dibandingkan GridSearchCV pada data uji.
- *Overfitting Indicator*: Perbedaan antara metrik *training* dan *test* juga menunjukkan sedikit indikasi *overfitting*. Meskipun ada indikasi

overfitting, performa model ini lebih stabil dibandingkan GridSearchCV.

Sehingga, dapat ditarik kesimpulan bahwa “Base Model Linear Regression” adalah pilihan yang paling tepat untuk digunakan dalam memprediksi *score index performance* di masa yang akan datang karena kinerjanya yang stabil dan akurat baik pada data *training* maupun data uji.

c. Model Deployment

Model *deployment* digunakan untuk mencari prediksi dari *index performance* Rumah Sakit Mulia dimasa depan. Berikut adalah tampilan dari model *deployment* menggunakan streamlit. Hasil *deployment* dapat dilihat pada [link ini](#).

Form Page

The screenshot shows a Streamlit application interface. On the left, there is a sidebar with a dropdown menu labeled "Select Page:" containing the option "Form Page". The main area has a title "Hospital Performance Index Score Prediction" with a score of "100" in pink. Below the title are several input fields:

- Branch: RSMA
- Month: 1
- Year: 2022
- Number of Patient: 100
- Average of Review: 4.00
- Total Revenue: 10000000
- Total COGS: 8900000

At the bottom is a red-bordered button labeled "Predict Score". Below the button, a small note says "Performance score for RSMA in February 2022 is: 50.31".

Form page dapat digunakan dengan cara mengisi parameter yang ada pada *form page*, yaitu branch, month, year, number of patient, average of review, total revenue dan total cogs.

File Upload Page

The screenshot shows a user interface for uploading files. At the top, there is a dropdown menu labeled "Select Page" with "File Upload Page" selected. Below it is a section titled "Hospital Performance Index Score Prediction". A "Upload CSV file" button is present, with a "Browse files" button next to it. A file named "data11_2022.csv" is shown as uploaded, with a size of 16.9KB. The "Uploaded Data:" section displays a table with 10 rows of data. The columns are: id_trx, date_in, date_out, drug_qty, admin_price, cogs, Durasi_Rawat, and hospital_care. The last column contains icons representing different types of hospital admissions. The table data is as follows:

	id_trx	date_in	date_out	drug_qty	admin_price	cogs	Durasi_Rawat	hospital_care
0	85,370	2022-11-20	2022-11-23	4	50,000	14,967,274	4	Rawat Inap
1	61,019	2022-11-03	2022-11-06	5	50,000	15,593,680	4	Rawat Inap
2	33,260	2022-11-16	2022-11-20	4	50,000	15,411,718	5	Rawat Inap
3	11,415	2022-11-22	2022-11-26	3	50,000	5,841,815	5	Rawat Inap
4	50,965	2022-11-09	2022-11-09	1	50,000	1,996,622	1	Rawat Jalan
5	46,329	2022-11-13	2022-11-15	3	50,000	13,756,179	3	Rawat Inap
6	14,576	2022-11-01	2022-11-03	4	50,000	14,829,337	3	Rawat Inap
7	69,857	2022-11-08	2022-11-08	1	50,000	3,232,427	1	Rawat Jalan
8	14,084	2022-11-20	2022-11-22	2	50,000	10,988,782	3	Rawat Inap
9	86,019	2022-11-14	2022-11-17	4	50,000	7,754,246	4	Rawat Inap

Predictions:

value
0.5178

Untuk *file upload page*, kita dapat mengupload file berisikan data-data yang diperlukan oleh model *Machine learning* dan mendapatkan hasil prediksi *index performance*.

Chatbot Page

The screenshot shows a chatbot interface. At the top, it says "Chat with Mulia Hospital Dataset" and features a logo of a hospital building with a red cross. Below this, there is a button labeled "Ask me!". A message box contains the text "How many branches does Mulia Hospital have?". To the right of the message box is a green circular icon with a white letter "G". Below the message box, the response "Mulia Hospital has 3 branches." is displayed.

Dengan *chatbot page* kita dapat mengirimkan input berupa teks pertanyaan dalam bahasa Inggris, dan *chatbot page* akan memberikan output jawaban dalam bahasa

Inggris. Pertanyaan yang dapat dijawab oleh *chatbot page* terbatas pada data yang digunakan.

Berdasarkan hasil *deployment*, didapatkan hasil prediksi *index performance* pada bulan Februari 2024 untuk tiga cabang Rumah Sakit Mulia, yaitu sebagai berikut:

- RSMA: 55%
- RSMD: 71%
- RSMS: 65%

Jika dibandingkan dengan *index performance* ketiga cabang Rumah Sakit tersebut dari periode sebelumnya, maka terjadi penurunan nilai yang sangat signifikan. Hal ini dapat diakibatkan oleh data yang digunakan untuk memprediksi *index performance* pada bulan Februari 2024, hanya menggunakan data dari bulan Januari 2024 yang hanya memiliki sedikit transaksi. Sehingga, input data yang kurang bagus menghasilkan prediksi yang kurang bagus.

BAB V

KESIMPULAN

Kesimpulan yang kami dapatkan berdasarkan hasil penelitian kami ialah:

1. Setiap cabang Rumah Sakit Mulia memiliki *performance index* di atas rata-rata 70% dengan rincian, cabang RSMA memiliki *performance index* 86.03%, cabang RSMD memiliki *performance index* 86.06%, dan cabang RSMS memiliki *performance index* tertinggi sebesar 86.09 %. Adapun *performance index* ini dinilai dari total pasien, total keuntungan (*profit*), dan tingkat kepuasan pasien.
2. Selama periode 2020 – 2024 Rumah Sakit Mulia melayani 9474 pasien, mendapatkan *revenue* 80.81 Miliar rupiah, dengan *profit* 8.63 Miliar rupiah, dan *profit margin* 10.68 %
3. Ketiga cabang Rumah Sakit Mulia menunjukkan *profit margin* di atas 10%. Hal ini mengindikasikan manajemen keuangan yang sehat, sehingga membuka peluang untuk investasi pada teknologi medis yang dapat meningkatkan efisiensi dan kualitas pelayanan. Rumah Sakit Mulia juga dapat mengefisiensikan pengeluaran operasional dan menekan pengeluaran yang tidak diperlukan.
4. Berdasarkan data dari seluruh cabang Rumah Sakit Mulia, mayoritas pasien tidak merasa puas dengan pelayanan yang didapatkan. Hal ini terlihat pada tingkat kepuasan pasien yang berada diangka 40.17%. Oleh karena itu, perlu dilakukan peningkatan pelayan yang berfokus pada pengalaman pasien dan perbaikan fasilitas pelayanan atau penunjang operasional Rumah Sakit di seluruh cabang.
5. Berdasarkan model *Machine Learning* menggunakan *base model linear regression* dan *model deployment* pada striimlit, didapatkan prediksi *index performance* pada bulan Februari 2024 untuk ketiga cabang Rumah Sakit Mulia sebagai berikut, RSMA sebesar 55%, RSMD sebesar 71%, dan RSMS sebesar 65%.