FINAL PROJECT HACKTIV8

Transformasi Digital Kesehatan: Integrasi dan Analisis Data Rumah Sakit untuk Pengambilan Keputusan berbasis data

Disusun oleh

Arya Sangga Buana	Data Trainee
Nadya Novalina	Data Trainee
Rafi Athallah Seniang	Data Trainee
Rosita Laili Udhiah	Data Trainee

Cisauk, Tangerang, Banten 15345

Green Office Park 9

2024

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara daerah endemik untuk penyakit Demam Berdarah Dengue (DBD). Tercatat per 1 Maret 2024 terdapat hampir 16.000 kasus DBD di 213 Kabupaten/Kota di Indonesia dengan 124 kematian [p2p.kemkes.go.id]. Kasus DBD terbanyak tercatat terjadi di Tangerang, Bandung Barat, Kota Kendari, Subang, dan Lebak. Keadaan ini diperkirakan terus berlanjut sampai bulan April seiring dengan musim hujan setelah El nino. Meskipun DBD dapat disembuhkan, namun masyarakat perlu waspada kemungkinan komplikasi terjadinya Syok pada DBD atau istilah medisnya *Dengue Shock Syndrome* (DSS) yang bisa berujung kematian. Sehingga, diperlukan langkah untuk *monitoring* dan *tracking* pasien dengan penyakit Demam Berdarah Dengue (DBD) untuk mencegah timbulnya korban jiwa. Mengingat kasus DBD yang selalu terjadi di berbagai wilayah Indonesia, Bithealth sebagai jasa IT konsultan pada bidang *healthcare* mampu memberikan solusi melalui transformasi digital pada data digital rumah sakit sebagai upaya peningkatan kualitas layanan, meningkatkan sistem kesehatan, dan menciptakan lingkungan perawatan yang lebih baik dan lebih aman untuk masyarakat.

Data digital di rumah sakit diperoleh dari berbagai perangkat yang terhubung satu sama lain di seluruh bagian rumah sakit. Data ini mencakup seluruh perjalanan pasien mulai dari administrasi, hasil pemeriksaan, *monitoring* selama rawat inap maupun rawat jalan, hingga proses pembayaran dan pemulihan pasien. Sebagai contoh, PERSI (Perhimpunan Rumah Sakit Seluruh Indonesia) bekerja sama dengan BitHealth dalam sebuah *webinar* pada 16 Juli 2022 yang membahas pemanfaatan *analytics insight* untuk optimalisasi manajemen inventori di rumah sakit. Kolaborasi ini menunjukkan upaya nyata dalam edukasi dan implementasi teknologi untuk meningkatkan manajemen rumah sakit. Pentingnya digitalisasi di sektor kesehatan juga didorong oleh inisiatif pemerintah. Kementerian Kesehatan melalui Direktorat Jenderal Pelayanan Kesehatan mengadakan uji publik mengenai pelayanan kesehatan dengan teknologi informasi dan komunikasi pada 20 September 2023. Direktur Tata Kelola Pelayanan Kesehatan, dr. Sunarto, M.Kes., menyampaikan bahwa tujuan dari pengaturan ini adalah untuk memperluas akses pelayanan kesehatan, meningkatkan efisiensi, dan mengoptimalkan pengelolaan data

kesehatan. Dengan tele-kesehatan dan *telemedicine*, pelayanan kesehatan dapat dilakukan secara jarak jauh, meningkatkan akses dan menurunkan angka rujukan.

Pemanfaatan *big data* di rumah sakit memungkinkan koordinasi pelayanan kesehatan yang lebih baik, pemberdayaan pasien, dan penyesuaian obat sesuai kebutuhan pasien. Implementasi *big data* menghasilkan kemudahan dalam operasional rumah sakit serta meningkatkan pengalaman pasien. Data yang telah diolah melalui proses *data ingestion, data cleansing, data mining, reporting*, dan *visualization* dapat menjadi informasi yang berguna dalam pengambilan keputusan melalui proses digitalisasi data. Manfaat utama dari penggunaan *data warehouse* adalah integrasi seluruh data dan kemudahan dalam melakukan *queries* serta analisis historis dalam jumlah besar. Hal ini memberikan akses yang lebih mudah ke berbagai departemen di rumah sakit, memastikan satu sumber terpercaya dan memaksimalkan nilai dari data tersebut.

Akan tetapi, rumah sakit di era modern juga menghadapi berbagai tantangan dalam mengelola data dan informasi. Meskipun teknologi digital telah berkembang pesat, masih banyak rumah sakit yang belum sepenuhnya memanfaatkan potensi tersebut. Beberapa masalah utama yang dihadapi rumah sakit terkait dengan pengelolaan data dan informasi mencakup: (i) masih banyak rumah sakit masih menggunakan proses manual yang rentan terhadap kesalahan dan ketidakefisienan. Proses manual ini menghambat aliran informasi dan dapat menyebabkan keterlambatan dalam perawatan pasien serta peningkatan biaya operasional, (ii) sumber data yang terpisah di dalam rumah sakit menyulitkan untuk mendapatkan informasi yang terintegrasi, (iii) pasien dan manajemen seringkali kekurangan akses terhadap informasi yang diperlukan untuk membuat keputusan yang cepat dan tepat, (iv) diperlukan analisa data yang mendalam untuk membantu dalam pengambilan keputusan strategis dan meningkatkan perawatan pasien. Tanpa sistem yang memadai untuk mengolah dan menganalisis data, rumah sakit kesulitan dalam mengidentifikasi tren, mengevaluasi efektivitas perawatan, dan mengoptimalkan alokasi sumber daya.

Oleh karena itu, digitalisasi rumah sakit merupakan upaya penting dalam mendukung efisiensi pelayanan yang mengurangi risiko terhadap pasien. Dengan sistem monitoring yang terintegrasi, tenaga medis dapat memantau kondisi pasien secara real-time dan memberikan respons cepat jika ada perubahan kondisi yang memerlukan tindakan medis segera. Selain itu, pengelolaan data pasien yang terjamin keamanannya mengurangi risiko kesalahan medis akibat

informasi yang tidak akurat atau tidak lengkap. Secara keseluruhan, transformasi digital di industri kesehatan merupakan langkah penting menuju peningkatan kualitas layanan. Melalui transformasi digital diharapkan rumah sakit dapat menjadi lebih efisien, responsif, dan adaptif terhadap perubahan kebutuhan pasien dan perkembangan teknologi. Hal ini tidak hanya menguntungkan pasien, tetapi juga meningkatkan keseluruhan sistem kesehatan, menciptakan lingkungan perawatan yang lebih baik dan lebih aman, serta meningkatkan kualitas hidup masyarakat Indonesia pada bidang kesehatan.

Dengan semakin kompleksnya data yang harus diolah, penggunaan *Machine Learning* (ML) dan *Large Language Models* (LLM) menjadi sangat penting dalam mendukung transformasi digital di rumah sakit. *Machine Learning* memungkinkan analisis data yang lebih cepat dan akurat, seperti misalnya dalam memprediksi biaya perawatan pasien. Model ML dapat memproses berbagai variabel yang mempengaruhi biaya perawatan, seperti usia pasien, jenis kelamin, tipe perawatan, dan jenis obat, untuk menghasilkan estimasi biaya yang lebih tepat. Prediksi biaya yang akurat ini dapat membantu rumah sakit dalam mengelola anggaran dan sumber daya dengan lebih efisien, serta memberikan transparansi kepada pasien mengenai perkiraan biaya perawatan mereka.

Selain itu, teknologi LLM memungkinkan interaksi yang lebih alami dan intuitif antara sistem dan *user*. Dengan kemampuan LLM untuk memahami dan merespons pertanyaan dalam bahasa natural (bahasa non programming), *Users* dapat dengan mudah mengajukan pertanyaan mengenai data rumah sakit dan mendapatkan jawaban yang relevan. Hal ini sangat berguna dalam konteks rumah sakit, di mana dokter, perawat, dan manajemen seringkali memerlukan informasi cepat dan tepat untuk pengambilan keputusan. Implementasi ML dan LLM juga dapat mengurangi beban kerja administratif dengan mengotomatiskan proses analisis dan pelaporan. Dalam jangka panjang, penggunaan teknologi ini dapat meningkatkan efisiensi operasional rumah sakit, mengurangi biaya, dan meningkatkan kualitas perawatan pasien. Integrasi teknologi ML dan LLM tidak hanya menunjukkan inovasi dalam manajemen data rumah sakit, tetapi juga membuka potensi baru untuk pengembangan layanan kesehatan yang lebih cerdas dan responsif terhadap kebutuhan pasien.

1.2 Tujuan

Berdasarkan uraian latar belakang yang dijelaskan diatas, diperoleh tujuan dari *project* ini yaitu sebagai berikut:

1. Me*monitoring* dan men*tracking* pasien dengan penyakit Demam Berdarah Dengue (DBD) baik dengan pelayanan rawat jalan maupun rawat inap pada rumah sakit melalui pembuatan *dashboard analytics*.
2. Estimasi biaya pasien dengan penyakit Demam Berdarah Dengue (DBD) baik dengan pelayanan rawat jalan maupun rawat inap pada rumah sakit menggunakan model *machine learning.*

BAB II
DATA DAN PENGATURAN EKSPERIMEN

2.1 Data

Data yang digunakan dalam pengerjaan *project* ini merupakan data *historical* pasien dimulai dari administrasi, hasil pemeriksaan, *monitoring* dalam rawat inap maupun rawat jalan, *tracking* asuransi, nominal biaya, pembayaran dan pasien kembali pulih pada bulan Januari 2020 hingga Januari 2024. Data terkumpul sebanyak 9474 data pasien dengan spesifikasi penyakit dan nominal biaya yang berbeda-beda. Uraian mengenai atribut dan masing-masing deskripsinya pada data dijelaskan pada tabel berikut:

Tabel 2.1. Atribut dan Deskripsi Data

No.	Atribut	Deskripsi
1	ID	ID transaksi penjualan (*Invoice*)
2	Date IN	Tanggal pasien masuk rumah sakit
3	Date OUT	Tanggal pasien keluar rumah sakit
4	Branch	Cabang rumah sakit: - RSMA : Rumah Sakit Mulia Anggrek - RSMD : Rumah Sakit Mulia Duri - RSMS : Rumah Sakit Mulia Simatupang
5	Name	Nama pasien
6	Age	Usia pasien
7	Gender	Jenis kelamin pasien

8	Hospital Care	Tipe perawatan pasien terdiri dari rawat inap dan rawat jalan. Setiap pasien rawat inap, membutuhkan infus dengan harga Rp. 165.000/hari.
9	Room Types	<u>Tipe kamar:</u> - VIP : Rp. 300.000/malam - Kelas 1 : Rp. 250.000/malam - Kelas 2 : Rp. 200.000/malam - Kelas 3 : Rp. 150.000/malam
10	Doctor	<u>Tipe dokter:</u> - 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 <u>Asumsi:</u> - Dokter setiap hari melakukan 1 kali kunjungan untuk pasien rawat inap. - Pasien Rawat Jalan dianggap bertemu dengan dokter 1 kali (kunjungan).

11	Surgery	Tipe tindakan/operasi: - Kecil : Rp. 4.000.000 - Besar : Rp. 8.000.000 - Kusus : Rp. 15.000.000 Asumsi: - 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.
12	Laboratorium	Tipe uji laboratorium: - Hematologi : Rp. 90.000 - Kimia Darah : Rp. 195.000 - Rontgen : Rp. 150.000 - Serologi : Rp. 200.000 - Urinalisa : Rp. 80.000 Asumsi: - Seluruh hasil test laboratorium pada data adalah positif dan hanya dilakukan 1x baik untuk pasien rawat jalan, maupun rawat inap.

13	Drug Types	<u>Tipe obat:</u> - Antibiotik : Rp. 75.000 - Pereda nyeri : Rp. 50.000 - Umum : Rp. 40.000 - Vitamin : Rp. 110.000 <u>Asumsi:</u> - 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.
14	Drug Brands	Merek-merek obat berdasarkan tipenya.
15	Drug Quantity	Jumlah strip/botol yang perlu dikonsumsi oleh pasien selama masa penyembuhan.
16	Food Price	Harga makanan untuk pasien rawat inap setiap hari (harga sudah termasuk sarapan, makan siang, dan malam).
17	Admin	Biaya administrasi rumah sakit
18	COGS	*Cost of Good Sold* Rumah sakit untuk setiap pasien
19	Payment	Tipe pembayaran pasien terdiri dari pribadi dan asuransi.
20	Review	*Review* dari pasien terdiri dari sangat puas, puas, netral, tidak puas, sangat tidak puas.

Pada data diberikan asumsi untuk mengklasifikasi pasien dengan penyakit Demam Berdarah Dengue (DBD). Asumsi ini digunakan untuk menspesialisasikan pasien dengan penyakit DBD yang selanjutnya disebut sebagai data pasien dengan penyakit DBD dengan total sebanyak 539 data.

Tabel 2.2. Asumsi Pasien dengan Penyakit Demam Berdarah Dengue

No.	Atribut	Deskripsi
1	Doctor	Pasien dengan penyakit DBD diasumsikan ditangani oleh dokter dengan spesialisasi: - Umum - Penyakit Dalam
2	Laboratorium	Pasien dengan penyakit DBD diasumsikan melakukan uji tes pada laboratorium: - Serologi - Hematologi - Kimia Darah
3	Drugs Type	Pasien dengan penyakit DBD diasumsikan memperoleh obat dengan tipe: - Umum - Pereda Nyeri - Vitamin
4	Surgery	Pasien dengan penyakit DBD diasumsikan tidak membutuhkan penanganan tindakan operasi.

Seluruh data yang telah terkumpul selanjutnya diproses dari normalisasi dan dimasukkan ke dalam *database* PostgreSQL hingga dilakukan estimasi biaya menggunakan *machine learning* dimana diuraikan pada sub bab selanjutnya.

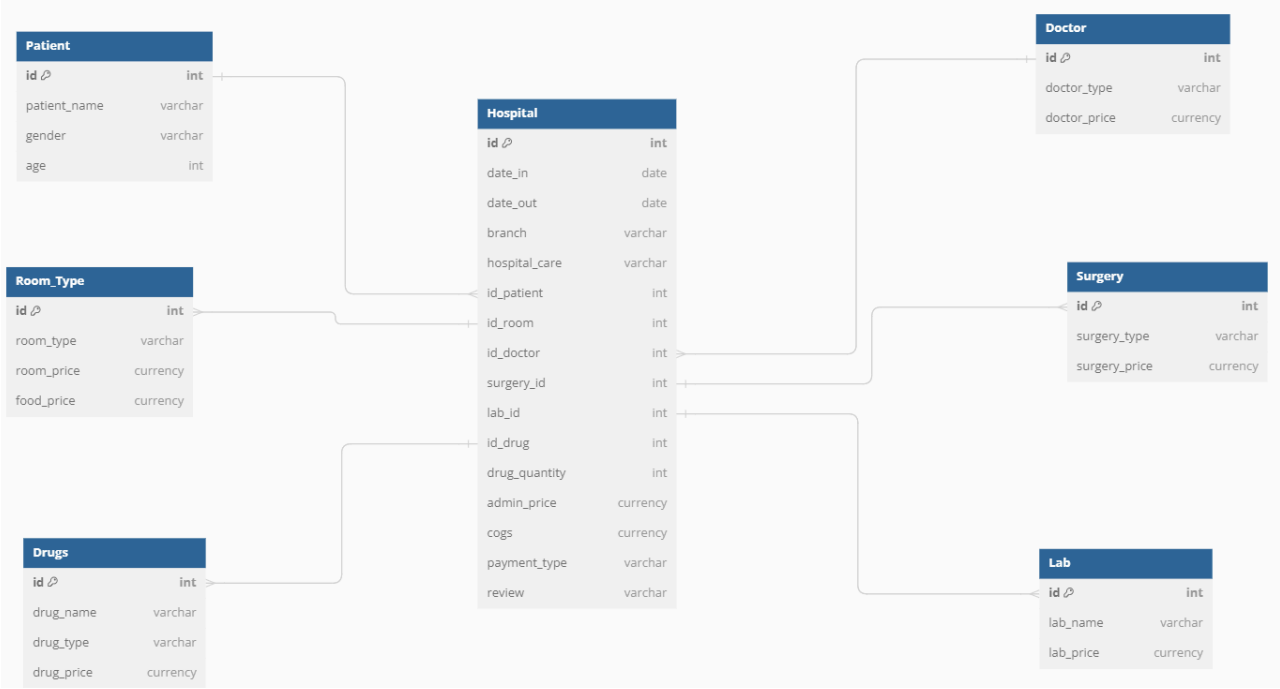
2.2 Detail Workflow

Pada sub bab ini, dijelaskan mengenai *workflow* dari pengerjaan *project* dalam melakukan pengolahan data.

2.2.1 Normalisasi Data

Normalisasi data adalah proses pengelompokan atribut data *source* yang membentuk entitas sederhana, nonredundant, fleksibel, dan mudah beradaptasi. Sehingga dapat dipastikan bahwa data yang masuk kedalam *database* yang dibuat berkualitas baik. *Database* yang digunakan pada *project* ini adalah PostgreSQL. Pada *project* ini, dilakukan normalisasi data 3NF yaitu proses normalisasi dengan menghilangkan ketergantungan transitif di antara atribut-atribut non-kunci. Artinya dalam tabel yang mengikuti 3NF, tidak boleh ada atribut bukan kunci yang bergantung pada atribut bukan kunci lainnya, yang pada gilirannya bergantung pada kunci utama. Sederhananya, semua atribut *non-primary key* harus bergantung langsung pada kunci primer, bukan secara tidak langsung melalui atribut *non-primary key* lainnya. Dengan demikian, 3NF memastikan bahwa redundansi diminimalkan dengan tetap menjaga kemudahan pembuatan kueri dan memfasilitasi manajemen basis data yang efisien. Berikut normalisasi data 3NF yang digambarkan pada *Entity Relationship Diagram* (ERD). Secara garis besar, tabel-tabel yang dibentuk pada ERD memuat data-data berikut:

- *Table Hospital*: Menyimpan informasi utama tentang perawatan pasien di rumah sakit, termasuk tanggal masuk dan keluar, cabang, jenis perawatan, dan detail lainnya.
- *Table Doctor*: Menyimpan informasi tentang jenis dokter dan harga layanan dokter.
- *Table Surgery*: Menyimpan informasi tentang jenis operasi dan harga operasi.
- *Table Lab*: Menyimpan informasi tentang jenis laboratorium dan harga tes laboratorium.
- *Table Room_Type*: Menyimpan informasi tentang jenis kamar di rumah sakit dan harga kamar serta makanan.
- *Table Drugs*: Menyimpan informasi tentang jenis obat, nama obat, dan harga obat.
- *Table Patient*: Menyimpan informasi tentang pasien, termasuk nama, jenis kelamin, dan usia.



Gambar 2.1 *Entity Relationship Diagram* (ERD)

Pada pengerjaan *project,* dalam melakukan normalisasi data digunakan bahasa pemrograman Python. Berikut diuraikan normalisasi data dengan Python di google colab. Pertama dilakukan *import library* yang diperlukan seperti *pandas* untuk membuat *dataframe* dari data *source*. Selanjutnya, dilakukan *mount* kepada penyimpanan *colab* untuk menghubungkan penyimpanan yang ada di *google drive*. Hal ini bertujuan untuk mengakses *data source* yang telah tersimpan di *google drive.* Kemudian, data *source* dibaca dan disimpan kedalam bentuk *dataframe* menggunakan *library pandas* dan fungsi *read_csv* yang ditunjukkan pada *code* berikut.

```
import pandas as pd
from google.colab import drive
drive.mount("/content/drive")
df =
pd.read_csv('/content/drive/MyDrive/gradexpert2024/final_project/Hospital_data
.csv')
```

Kedua, didefinisikan variabel baru bernama *patient* yang digunakan untuk membuat data tabel pasien yang berasal dari *data source* dengan hanya mengambil kolom *Name, Age,* dan *Gender*. Selanjutnya, dilakukan *drop_duplicates()* untuk menghapus data yang duplikat. Lalu, ditambahkan kolom *patient_id* untuk membuat id dari setiap pasien dan bersifat unik. Setelah itu,

dilakukan pengubahan format pada kolom *patient_id* menjadi integer menggunakan fungsi *astype()* dan hasil dari rangkaian proses tersebut disimpan kedalam *dataframe* dalam bentuk csv dengan fungsi *to_csv()*.

```
patient = df[['Name', 'Age', 'Gender']]
patient = patient.drop_duplicates()
patient['patient_id'] = range(1, len(patient) + 1)
patient['patient_id'] = patient['patient_id'].astype(int)
patient.to_csv('patient.csv', index = False)
```

Langkah selanjutnya adalah membuat tabel untuk *room* yang berisi *Room, Food, Room Price,* dan *room_id*. Pada *dataframe room* ini, diterapkan filter dan mengecualikan *room* yang bernilai "-". Lalu dilakukan sortir nilai kolom berdasarkan kolom *Room* dan menghapus data duplikat dan melakukan *reset_index* agar tidak error ketika melakukan *indexing*. Setelah itu, membuat kolom "*Room Price*" berdasarkan suatu kondisi yang sesuai berdasarkan deskripsi data untuk kolom tersebut. Pada tahap terakhir, dilakukan hal yang sama seperti tabel *room* untuk pembuatan tabel *drugs, doctor, surgery,* dan *lab* akan tetapi tetap disesuaikan dengan asumsi untuk masing-masing tabel berdasarkan deskripsi data yang telah dijelaskan sebelumnya.

Tabel *Room*

```
room = df[['Room', 'Food']]
room = room[room['Room'] != '-']
room = room.sort_values(by = 'Room')
room = room.drop_duplicates().reset_index()

for i in range(len(room)):
    if room.loc[i,'Room'] == 'Kelas 3':
        room.loc[i,'Room Price'] = 150000
    elif room.loc[i,'Room'] == 'Kelas 2':
        room.loc[i,'Room Price'] = 200000
    elif room.loc[i,'Room'] == 'Kelas 1':
        room.loc[i,'Room Price'] = 250000
    elif room.loc[i,'Room'] == 'VIP':
        room.loc[i,'Room Price'] = 300000
    else:
        room.loc[i,'Room Price'] = '-'
room['room_id'] = range(1, len(room) + 1)
room['room_id'] = room['room_id'].astype(int)
room.drop(columns = 'index', inplace = True)
room.to_csv('room.csv', index = False)
```

Tabel *Drugs*

```
drugs = df[['Drug Brands', 'Drug Types']]
drugs = drugs.drop_duplicates().reset_index()
for i in range(len(drugs)):
    if drugs.loc[i,'Drug Types'] == 'Antibiotik':
```

```
        drugs.loc[i,'Drug Price per Item'] = 75000
    if drugs.loc[i,'Drug Types'] == 'Pereda Nyeri':
        drugs.loc[i,'Drug Price per Item'] = 50000
    if drugs.loc[i,'Drug Types'] == 'Umum':
        drugs.loc[i,'Drug Price per Item'] = 40000
    if drugs.loc[i,'Drug Types'] == 'Vitamin':
        drugs.loc[i,'Drug Price per Item'] = 110000
drugs['drug_id'] = range(1, len(drugs) + 1)
drugs['drug_id'] = drugs['drug_id'].astype(int)
drugs.drop(columns = 'index', inplace = True)
drugs.to_csv('drugs.csv', index = False)
```

Tabel *Doctor*

```
doctor = df['Doctor']
doctor = doctor.drop_duplicates().reset_index()
for i in range(len(doctor)):
    if doctor.loc[i,'Doctor'] == 'Bedah':
        doctor.loc[i,'Doctor Price'] = 300000
    elif doctor.loc[i,'Doctor'] == 'Gigi':
        doctor.loc[i,'Doctor Price'] = 300000
    elif doctor.loc[i,'Doctor'] == 'Kandungan':
        doctor.loc[i,'Doctor Price'] = 300000
    elif doctor.loc[i,'Doctor'] == 'Penyakit Dalam':
        doctor.loc[i,'Doctor Price'] = 300000
    elif doctor.loc[i,'Doctor'] == 'Umum':
        doctor.loc[i,'Doctor Price'] = 250000
doctor['doctor_id'] = range(1, len(doctor) + 1)
doctor['doctor_id'] = doctor['doctor_id'].astype(int)
doctor.drop(columns = 'index', inplace = True)
doctor.to_csv('doctor.csv', index = False)
```

Tabel *Surgery*

```
surgery = df[['Surgery']]
surgery = surgery[surgery['Surgery'] != '-']
surgery = surgery.drop_duplicates().reset_index()
for i in range(len(surgery)):
    if surgery.loc[i,'Surgery'] == 'Kecil':
        surgery.loc[i,'Surgery Price'] = 4000000
    elif surgery.loc[i,'Surgery'] == 'Besar':
        surgery.loc[i,'Surgery Price'] = 8000000
    elif surgery.loc[i,'Surgery'] == 'Kusus':
        surgery.loc[i,'Surgery Price'] = 15000000
surgery['surgery_id'] = range(1, len(surgery) + 1)
surgery['surgery_id'] = surgery['surgery_id'].astype(int)
surgery.drop(columns = 'index', inplace = True)
surgery.to_csv('surgery.csv', index = False)
```

Tabel *Lab*

```
Lab = df[['Lab']]
Lab = Lab[Lab["Lab"] != '-']
Lab = Lab.drop_duplicates().reset_index()
for i in range(len(Lab)):
```

```
    if Lab.loc[i,'Lab'] == 'Hematologi':
       Lab.loc[i,'Lab Price'] = 90000
    elif Lab.loc[i,'Lab'] == 'Kimia Darah':
       Lab.loc[i,'Lab Price'] = 195000
    elif Lab.loc[i,'Lab'] == 'Rontgen':
       Lab.loc[i,'Lab Price'] = 150000
    elif Lab.loc[i,'Lab'] == 'Serologi':
       Lab.loc[i,'Lab Price'] = 200000
    elif Lab.loc[i,'Lab'] == 'Urinalisa':
       Lab.loc[i,'Lab Price'] = 80000
Lab['lab_id'] = range(1, len(Lab) + 1)
Lab['lab_id'] = Lab['lab_id'].astype(int)
Lab.drop(columns = 'index', inplace = True)
Lab.to_csv('lab.csv', index = False)
```

Selanjutnya, apabila semua tabel sudah dibuat maka dilakukan *join* pada setiap tabel untuk membuat suatu tabel akhir. Pada *project* ini, digunakan fungsi *merge* dan dilakukan satu persatu pada setiap tabel. Kemudian, dilakukan penghapusan kolom yang tidak diperlukan dengan *drop* dan langkah terakhir adalah menyimpan *dataframe* untuk tabel *hospital* menjadi csv.

```
df1 = df
df1 = df1.merge(patient, how = 'left', on = ['Name', 'Age', "Gender"])
df1 = df1.merge(room, how = 'left', on = 'Room')
df1 = df1.merge(drugs, how = 'left', on = 'Drug Brands')
df1 = df1.merge(doctor, how = 'left', on = 'Doctor')
df1 = df1.merge(surgery, how = 'left', on = 'Surgery')
df1 = df1.merge(Lab, how = 'left', on = 'Lab')
df1.drop(columns = ['Drug Types_x', 'Food_x', 'Food_y', 'Room Price','Drug
Types_y', 'Drug Price per Item', 'Doctor Price', 'Surgery Price', 'Lab
Price'], inplace = True)
df1.columns

df1.to_csv('hospital.csv', index = False)
```

Hasil akhir dari proses normalisasi data berupa *file* csv meliputi data *patient, room, drugs, doctor, surgery, lab* dan *hospital* dengan kolom yang sesuai pada *Entity Relationship Diagram* (ERD). data-data tersebut selanjutnya dilakukan proses *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

2.2.2 *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML)

DDL atau *Data Definition Language* adalah sebuah bagian dari bahasa SQL (*Structured Query Language*) yang digunakan untuk mengatur struktur dari *database* meliputi membuat, mengubah, dan menghapus objek *database* seperti tabel, indeks, dan *constraint*. Berbeda dengan DDL, *Data Manipulation Language* (DML) adalah bagian dari bahasa SQL (*Structured Query*

Language) yang digunakan untuk memanipulasi atau mengubah data yang sudah ada dalam sebuah *database* meliputi proses menambah, mengubah, dan menghapus data dari tabel yang sudah ada. Pada *project* ini, dilakukan pembuatan beberapa tabel pada *PostgreSQL* dengan *syntax CREATE* untuk membuat tabel baru beserta kolom-kolom sesuai dengan ERD dimana tergolong proses DDL dan *syntax COPY* yang digunakan untuk memasukkan *file* csv yang telah dihasilkan pada proses normalisasi ke dalam tabel-tabel yang sesuai dimana tergolong dalam proses DML. Berikut diuraikan proses pembuatan tabel dan menambahkan data pada masing-masing tabel.

Pada *query* dibawah ini, dilakukan pembuatan tabel *surgery* dengan kolom *surgery_id* bertipe integer sebagai *primary key*, *surgery* bertipe *varchar*, dan *surgery_price* bertipe integer. Selanjutnya, dilakukan proses menambah data pada tabel *surgery* yang berasal dari *file* csv *surgery* hasil proses normalisasi data yang berisikan id unik untuk *surgery,* nama tipe *surgery,* dan harga untuk masing-masing tipe *surgery*. Dikarenakan data yang ditambah pada tabel berformat csv, maka perlu ditambahkan *delimiter ','* dimana berarti bahwa tiap kolom dipisahkan dengan tanda koma (,) dan *CSV HEADER* yang berarti bahwa pada *file* baris pertama merupakan *header* dan bukan termasuk data *input.*

```
create table surgery (
             surgery_id int primary key,
             surgery varchar,
             surgery_price int
)

COPY surgery(surgery, surgery_price, surgery_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\surgery.csv'
DELIMITER ','
CSV HEADER
```

Kemudian, dilakukan pembuatan tabel *patient* dengan kolom *patient_id* bertipe integer sebagai *primary key*, *patient_name* bertipe *varchar*, *gender* bertipe *varchar,* dan *age* bertipe integer. Selanjutnya, dilakukan proses menambah data pada tabel *patient* yang berasal dari *file* csv *patient* hasil proses normalisasi data yang berisikan id unik untuk *patient,* nama pasien, usia pasien dan jenis kelamin pasien.

```
create table patient(
        patient_id int primary key,
        patient_name varchar(255),
        gender varchar(255),
        age int
)
```

```
COPY patient(patient_name, age, gender, id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\patient (1).csv'
DELIMITER ','
CSV HEADER
```

Proses dilanjutkan dengan pembuatan tabel *doctor* dengan kolom *doctor_id* bertipe integer sebagai *primary key*, *doctor* bertipe *varchar*, dan *doctor_price* bertipe integer. Selanjutnya, dilakukan proses menambah data pada tabel *doctor* yang berasal dari *file* csv *doctor* hasil proses normalisasi data yang berisikan id unik untuk *doctor,* tipe dokter berdasarkan spesialisasinya, dan harga untuk layanan masing-masing spesialisasi.

```
create table doctor (
            doctor_id int primary key,
            doctor varchar,
            doctor_price int
)

COPY doctor(doctor, doctor_price, doctor_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\doctor.csv'
DELIMITER ','
CSV HEADER
```

Setelah itu, dilakukan pembuatan tabel *drugs* dengan kolom *drug_id* bertipe integer sebagai *primary key*, *drug_brand* bertipe *varchar*, *drug_type* bertipe *varchar,* dan *drug_price* bertipe integer. Selanjutnya, dilakukan proses menambah data pada tabel *drugs* yang berasal dari *file* csv *drugs* hasil proses normalisasi data yang berisikan id unik untuk setiap tipe obat, nama brand obat, tipe obat, dan harga obat.

```
create table drugs(
            drug_id int primary key,
            drug_brand varchar,
            drug_type varchar,
            drug_price int
)

COPY drugs(drug_brand, drug_type, drug_price, drug_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\drugs.csv'
DELIMITER ','
CSV HEADER
```

Pada pembuatan tabel *lab* dengan kolom *lab_id* bertipe integer sebagai *primary key*, *lab* bertipe *varchar*, dan *lab_price* bertipe integer. Selanjutnya, dilakukan proses menambah data pada tabel *lab* yang berasal dari *file* csv *lab* hasil proses normalisasi data yang berisikan id unik untuk setiap bidang laboratorium, nama bidang laboratorium, dan harga untuk setiap layanan masing-masing bidang laboratorium.

```
create table lab (
            lab_id int primary key,
            lab varchar,
            lab_price int
)

COPY lab(lab, lab_price, lab_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\lab.csv'
DELIMITER ','
CSV HEADER
```

Proses selanjutnya yaitu pembuatan tabel *room* dengan kolom *room_id* bertipe integer sebagai *primary key*, *room* bertipe *varchar*, *food_price* bertipe integer, dan *room_price* bertipe *integer*. Selanjutnya, dilakukan proses menambah data pada tabel *room* yang berasal dari *file* csv *room* hasil proses normalisasi data yang berisikan id unik untuk setiap tipe ruangan, tipe ruangan, harga layanan makanan pada masing-masing tipe ruangan, dan harga layanan penggunaan masing-masing tipe ruangan.

```
create table room (
            room_id int primary key,
            room varchar,
            food_price int,
            room_price int
)

COPY room(room, food_price, room_price, room_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\room_new.csv'
DELIMITER ','
CSV HEADER
```

Proses terakhir adalah pembuatan tabel *hospital* dengan kolom *id* bertipe integer sebagai *primary key*, *date_ind* bertipe *date*, *date_out* bertipe *date*, *branch* bertipe *varchar*, *hospital_care* bertipe *varchar, drug_quantity* bertipe integer, *admin* bertipe integer, cogs bertipe integer, *payment* bertipe *varchar, review* bertipe *varchar,* dan beberapa *foreign_key* meliputi *patient_id, room_id, drug_id, doctor_id, surgery_id,* dan *lab_id*. Selanjutnya, dilakukan proses menambah data pada tabel *hospital* yang berasal dari *file* csv *hospital* hasil proses normalisasi data yang berisikan id unik untuk setiap transaksi, tanggal saat pasien masuk, tanggal saat pasien keluar, cabang rumah sakit, tipe layanan rumah sakit, kuantitas obat yang diberikan, biaya admin, biaya cogs, metode pembayaran, review oleh pasien, id unik pasien, id unik ruangan, id unik obat, id unik dokter, id unik layanan operasi, dan id unik layanan laboratorium.

```
create table hospital (
    id  int primary key,
    date_ind date,
```

```
    date_out date,
    branch varchar,
    hospital_care varchar,
    drug_quantity int,
    admin int,
    cogs int,
    payment varchar,
    review varchar,
    patient_id int,
    room_id int,
    drug_id int,
    doctor_id int,
    surgery_id int,
    lab_id int
)

COPY hospital(id, date_ind, date_out, branch, hospital_care, drug_quantity,
admin, cogs, payment, review, patient_id, room_id, drug_id, doctor_id,
surgery_id, lab_id)
FROM 'C:\Users\Arya Sangga Buana\Downloads\hospital_new.csv'
DELIMITER ','
CSV HEADER
```

2.2.3 *Data Cleaning dan Data Transforming*

Dalam melakukan Data *cleaning* dan *transforming* digunakan *apache airflow* dan *docker* untuk membuat *Extract Transform Load* (ETL). Dibawah ini diuraikan secara menyeluruh mengenai proses ETL yang meliputi *setup* Docker dan Airflow serta *Direct Acyclic Graph* (DAG) *pipeline*.

(i) Docker dan Airflow

Proses Pertama yang dilakukan pada *setup* Docker dan Airflow yaitu membuat dockerfile yang berisi *image* dari airflow agar airflow tersebut dapat di-*install* di *container*.

```
FROM apache/airflow:latest


USER root
RUN apt-get update && \
    apt-get -y install git && \
    apt-get clean


USER airflow
```

Pada dockerfile dilakukan pengaturan pada *image* yang akan dibuat untuk dilakukan *compose* pada *docker* nantinya. Baris pertama '*FROM apache/airflow:latest'* berarti bahwa mengambil

versi airflow terbaru untuk *image* yang akan dibuat, lalu dilanjutkan dengan '*USER root'* dimana untuk mengatur *user* dari *image airflow* menjadi *root* agar bisa memiliki akses/izin penuh pada terminal yang biasanya digunakan untuk meng-*install package*. Lalu dilanjutkan dengan '*RUN apt-get update'* yang merupakan instruksi untuk melakukan *update* versi terbaru dari setiap *package*. '*apt-get -y install git'* yang berfungsi untuk meng-*install* git dan terakhir adalah '*apt-get clean'* adalah untuk menghapus *cache* dari hasil *download* pada setiap *package* yang telah di-*install* sebelumnya. '*USER airflow'* merupakan proses mengembalikan *user* dari *root* menjadi *airflow*.

```
version: '3'

services:
  test-airflow:
    image: test-airflow:latest
    volumes:
      - ./airflow:/opt/airflow
    ports:
      - "8080:8080"
    command: airflow standalone
```

Setelah melakukan *build image* pada *dockerfile*, selanjutnya adalah membuat *file* yaml untuk melakukan *compose* pada *image* yang telah dibuat sebelumnya. Pada awal kode terdapat *version:"3"* merupakan deklarasi variabel untuk mengatur versi *docker* untuk melakukan *compose* pada *file* yaml. Selanjutnya, pada bagian *services* yang dimana disini *airflow2* adalah nama *container* yang akan dibuat. Lalu, memasukan *image* dari *image* yang sudah kita *build* yaitu *progress*. Selanjutnya, mengatur letak direktori dari *airflow* dengan *volumes* dan mengatur pada port berapa *airflow* akan bekerja. Kemudian, diberikan *command* pada *server airflow* yang akan dibuat dan dilakukan secara *standalone* yang berarti *webserver, database*, dan hal dilakukan *running* sekaligus.

(ii) DAG Pipeline

Setelah *container* pada *docker* telah terbentuk, proses selanjutnya yaitu membuat *Direct Acyclic Graph* (DAG) *pipeline* menggunakan bahasa pemrograman python.

```
import pandas as pd
import psycopg2
import re
from airflow import DAG
from datetime import datetime
from airflow.operators.python import PythonOperator
```

```
default_args={
    'start_date' : datetime(2024, 4, 29)
}

connection = psycopg2.connect(
   dbname = 'finpro1',
   user = 'postgres',
   password = 'nadya123',
   host = 'host.docker.internal',
   port = '5432'
)

cursor = connection.cursor()
```

Pertama dilakukan *import* dari *library* yang diperlukan seperti *pandas* untuk mengolah data yang dibaca menjadi *dataframe*, *psycopg*2 digunakan sebagai koneksi pada *postgreSQL* yang ada di lokal dengan *airflow* yang berada pada *container*. DAG digunakan untuk mengatur *task* dan *pipeline* yang ada di *airflow*. *Datetime* digunakan untuk mengatur atau mengubah format *string* menjadi format waktu yang sesuai. Lalu, *PythonOperator* digunakan sebagai mendefinisikan identitas *task* yang dibuat di dalam *airflow*. *Library re* merupakan *library* yang berfungsi untuk mendefinisikan pola *string*. Dideklarasikan *default_args* yang digunakan untuk mengatur mulai dari tanggal berapa DAG akan berjalan, lalu mengatur koneksi ke *postgreSQL* agar *airflow* dapat mengambil data dari *postgreSQL* dengan mengatur nama db, user, password, host dan port dari *postgreSQL local* yang berada pada *device*. Selanjutnya, akan dieksekusi beberapa perintah pada *database postgreSQL*, maka untuk melakukan hal tersebut perlu menggunakan *connection.cursor()* yang digunakan untuk dapat berinteraksi dengan *database postgreSQL*.

```
def doctor():
    cursor = connection.cursor()
    cursor.execute('''
                    select * from doctor
                   ''')
    result = cursor.fetchall()
    cursor.close()
    df_doctor = pd.DataFrame(result, columns = ['doctor_id', 'doctor',
'doctor_price'])

    return df_doctor
```

Selanjutnya, akan dibuat fungsi yang berisi mengambil data dari *postgreSQL* menjadi *dataframe*. *cursor.execute()* digunakan untuk mengeksekusi *query* dan *cursor.fetchall()* digunakan untuk mengambil dan mengumpulkan hasil *query* serta diubah menjadi *list.* Setelah itu, dilakukan *close* pada *cursor* agar me-*reset* atau menghapus hasil *query* yang terdapat pada *cursor*. Kemudian,

dilakukan proses mengubah data hasil *query* yang berbentuk *list* menjadi *dataframe* dengan *pd.DataFrame*.

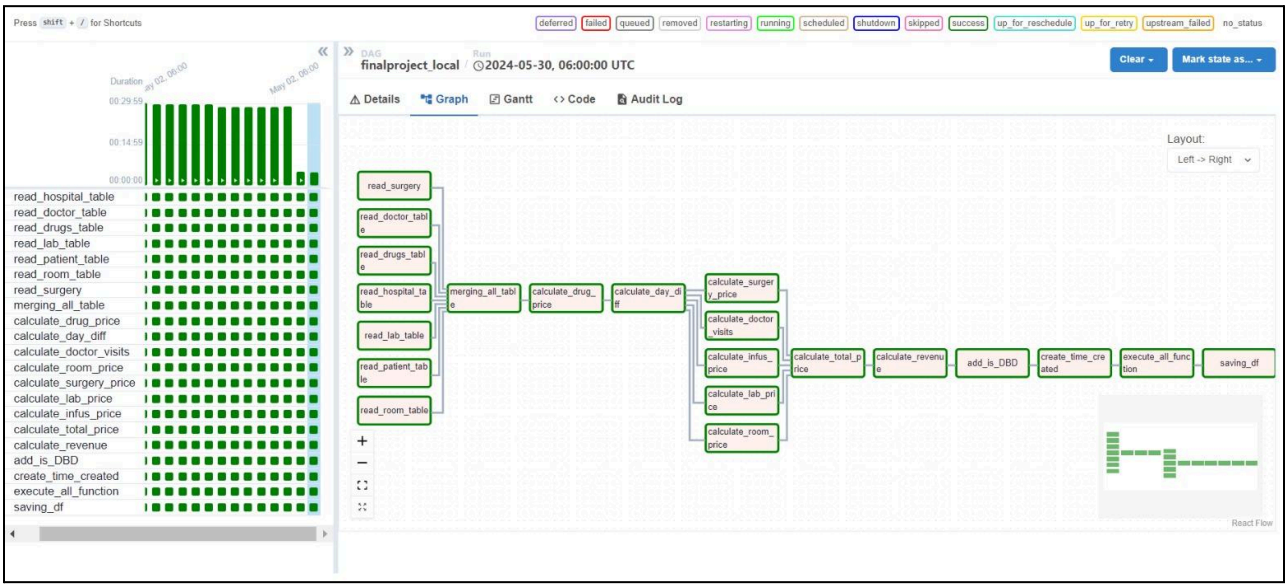
```
with DAG('finalproject_local',
         schedule_interval = '0 6 * * *',
         default_args = default_args,
         catchup = False,) as dag:

    read_table_hospital = PythonOperator(
        task_id = 'read_hospital_table',
        python_callable = hospital,
        provide_context = True,
    )


    read_table_doctor = PythonOperator(
        task_id = 'read_doctor_table',
        python_callable = doctor,
        provide_context = True,
    )


    read_table_drugs = PythonOperator(
        task_id = 'read_drugs_table',
        python_callable = drugs,
        provide_context = True,
    )


    read_table_lab = PythonOperator(
        task_id = 'read_lab_table',
        python_callable = lab,
        provide_context = True,
    )


    read_table_patient = PythonOperator(
        task_id = 'read_patient_table',
        python_callable = patient,
        provide_context = True,
    )


    read_table_room = PythonOperator(
        task_id = 'read_room_table',
        python_callable = room,
        provide_context = True,
    )


    read_table_surgery = PythonOperator(
        task_id = 'read_surgery',
        python_callable = surgery,
        provide_context = True,
    )
```

```
merge_df = PythonOperator(
    task_id = 'merging_all_table',
    python_callable = merge_hospital,
    provide_context = True,
)


calculate_drugs_price = PythonOperator(
    task_id = 'calculate_drug_price',
    python_callable = drugs_price,
    provide_context = True,
)


calculate_days_different = PythonOperator(
    task_id = 'calculate_day_diff',
    python_callable = days_diff,
    provide_context = True,
)


calculate_doctor_visit = PythonOperator(
    task_id = 'calculate_doctor_visits',
    python_callable = doctor_visit,
    provide_context = True,
)




calculate_room_price = PythonOperator(
    task_id = 'calculate_room_price',
    python_callable = room_price_total,
    provide_context = True,
)


calculate_surgery_price = PythonOperator(
    task_id = 'calculate_surgery_price',
    python_callable = surgery_price_total,
    provide_context = True,
)


calculate_lab_price = PythonOperator(
    task_id = 'calculate_lab_price',
    python_callable = lab_price_total,
    provide_context = True,
)


calculate_infus_price = PythonOperator(
    task_id = 'calculate_infus_price',
    python_callable = infus_price_total,
```

```
        provide_context = True,
    )


    calculate_total_price = PythonOperator(
        task_id = 'calculate_total_price',
        python_callable = total_price_all,
        provide_context = True,
    )




    calculate_revenue = PythonOperator(
        task_id = 'calculate_revenue',
        python_callable = total_revenue,
        provide_context = True,
    )


    add_is_DBD = PythonOperator(
        task_id = 'add_is_DBD',
        python_callable = is_DBD,
        provide_context = True,
    )


    create_time_created = PythonOperator(
        task_id = 'create_time_created',
        python_callable = time_created,
        provide_context = True,
    )


    execution = PythonOperator(
        task_id = 'execute_all_function',
        python_callable = execute,
        provide_context = True,
    )


    saving_df = PythonOperator(
        task_id = 'saving_df',
        python_callable = save_local_csv,
        provide_context = True,
    )


[read_table_hospital,
read_table_doctor,
read_table_drugs,
read_table_lab,
read_table_patient,
read_table_room,
read_table_surgery] >> merge_df  >> calculate_drugs_price >>
calculate_days_different >> [calculate_doctor_visit,
```

```
calculate_room_price,
calculate_surgery_price,
calculate_lab_price,
calculate_infus_price] >> calculate_total_price >> calculate_revenue >>
add_is_DBD >> create_time_created >> execution >> saving_df
```

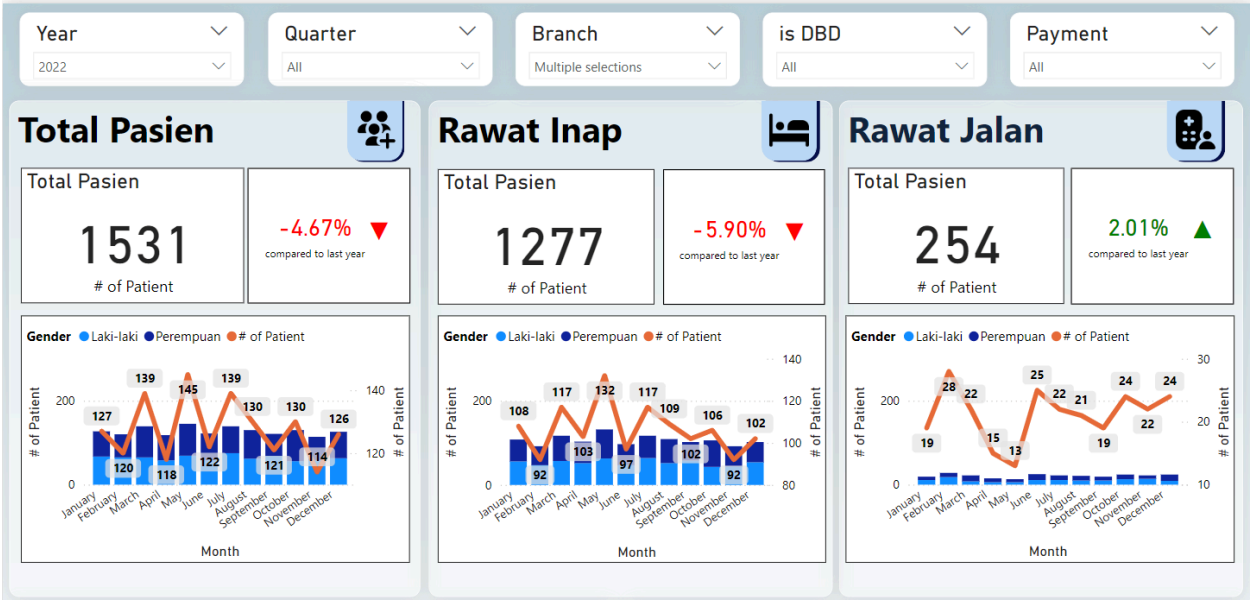
Setelah membuat koneksi dan membuat fungsi dari masing-masing *task*, maka selanjutnya adalah membuat DAG *pipeline*. Pada DAG, perlu dimasukan beberapa parameter seperti nama *file* DAG yang muncul pada *airflow* dan *schedule interval* yang digunakan sebagai penjadwalan atau *scheduling* dari DAG yang dimana disini menggunakan sintaks bernama *cronjob*. Kemudian terdapat *default_args* yang berisi *start_date* yang sebelumnya dibuat dan terakhir adalah *catchup* merupakan variabel yang mengatur apakah DAG perlu melakukan *backfill* apabila ada penjadwalan yang terlewat pada interval waktu yang telah ditentukan. Selain itu dilakukan pelabelan pada setiap *task*. Hal ini penting, dikarenakan dalam membuat *pipeline* diperlukan nama atau label dari setiap masing-masing *task*. Pada *project* ini, digunakan *PythonOperator* yang berisi parameter seperti *task_id* sebagai label yang akan muncul ketika akan membuat pipeline, lalu *python callable* yang merupakan parameter berisi fungsi yang telah dibuat sebelumnya, dan *provide_context* adalah suatu parameter agar DAG dapat mengakses variabel metadata seperti waktu eksekusi dan *task_id*. Pada akhir kode, *pipeline* disusun sesuai dengan urutan-urutan yang telah tentukan secara seri maupun paralel. Berikut ditunjukkan proses berjalannya DAG pada *webserver airflow* pada Gambar 2.2. Hasil akhir dari *job schedule* ini yaitu data berformat csv yang telah di *cleaning* dan *transforming*.



Gambar 2.2 *Direct Acyclic Graph* (DAG) *pipeline*

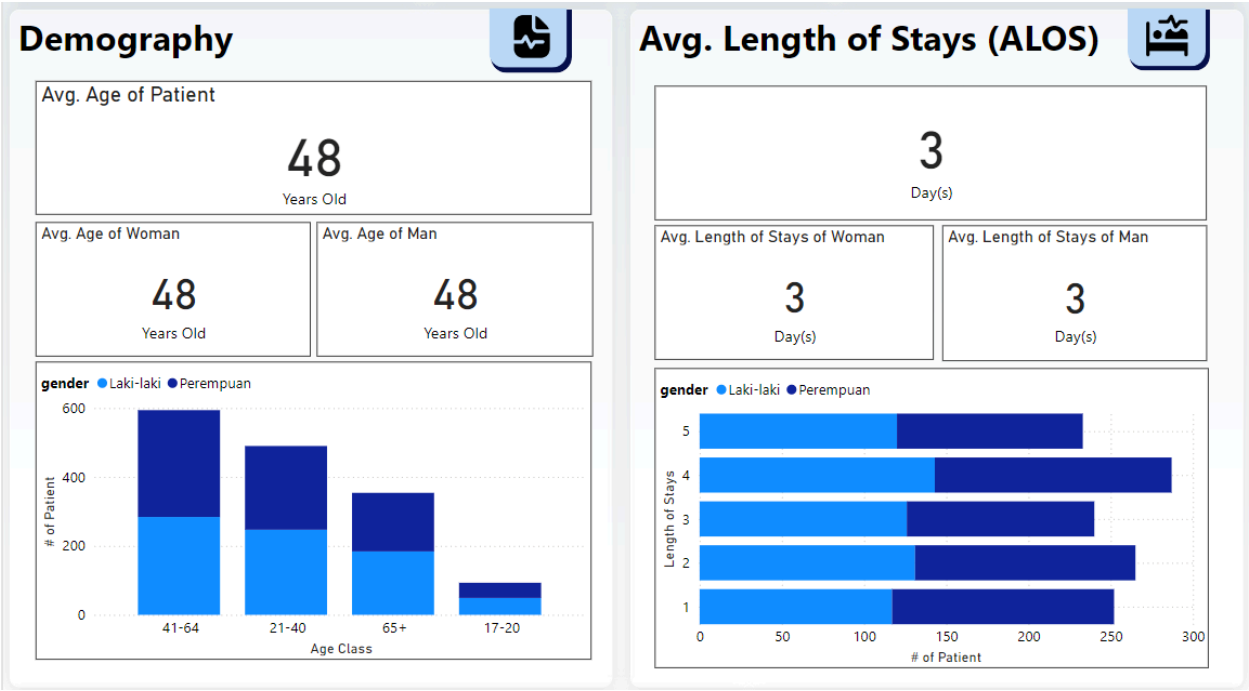
2.2.4 *Data Visualization*

Data yang telah di *cleaning* dan *transforming* selanjutnya divisualisasikan dalam bentuk *dashboard analytics* melalui *tools* Power BI. *Data source* yang digunakan berasal dari *local* dimana merupakan hasil dari proses *jobs schedule* yang telah dilakukan sebelumnya. Secara keseluruhan, *dashboard* ini digunakan untuk me-*monitoring* dan men-*tracking* pasien dengan penyakit Demam Berdarah Dengue (DBD) baik dengan pelayanan rawat jalan maupun rawat inap pada suatu rumah sakit mulai dari segi pertumbuhan jumlah pasien, persebaran demografi berdasarkan umur dan jenis kelamin pasien, rata-rata lama menginap pada pasien yang diberikan pelayanan rawat inap, persebaran jumlah pasien untuk setiap bidang spesialisasi dokter, laboratorium, tipe obat yang diberikan, tipe spesialisasi operasi hingga *financial* dan kepuasan pasien meliputi trend rata-rata *revenue* rumah sakit dan *total amount* pasien untuk setiap bulan dan persentase jumlah pasien untuk setiap *rating review* yang diberikan. Berikut diuraikan secara detail mengenai *dashboard analytics* menggunakan data pasien keseluruhan. Filter yang digunakan dalam *dashboard* dan bersifat interaktif meliputi tahun, quarter, cabang rumah sakit, indikasi pasien DBD atau tidak, dan metode pembayaran berupa *dropdown*. Pada menu teratas, terdapat visualisasi berupa jumlah pasien disertai dengan *growth* jika dibandingkan dengan tahun sebelumnya dan grafik kombinasi *bar chart* dan *line chart* untuk mengetahui trend jumlah pasien pada setiap bulannya jika ditinjau dari jenis kelamin. Visualisasi ini dapat memberikan informasi mengenai jumlah pasien secara lengkap, perbandingan antara pasien rawat inap dan rawat jalan sehingga pihak manajemen rumah sakit dapat mengetahui secara cepat dan tepat bagaimana pertumbuhan jumlah pasien dari tahun ke tahun dan dapat membantu dalam pengambilan keputusan suatu rumah sakit.



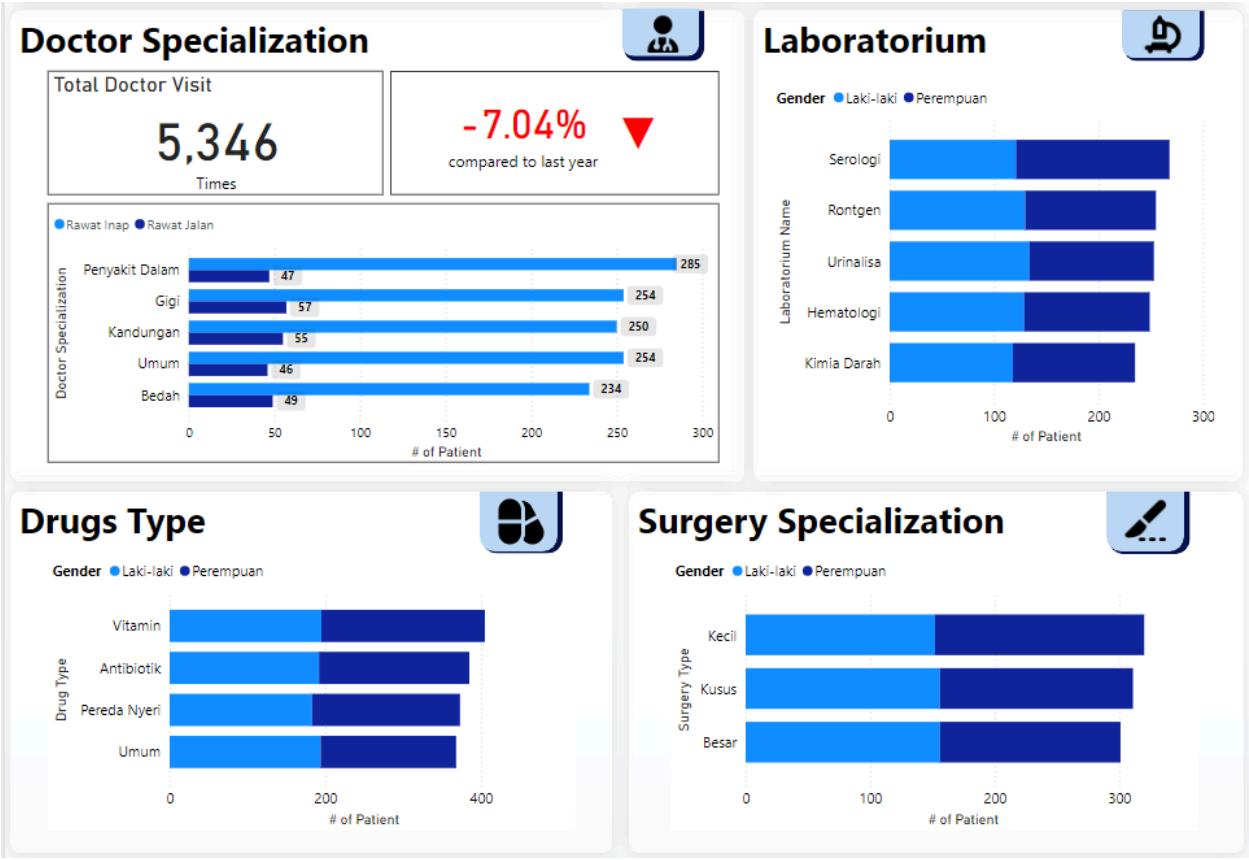
Gambar 2.3 *Mockup Dashboard Analytics* Trend Jumlah Pasien

Selain itu, pada *dashboard* juga terdapat persebaran demografi berdasarkan umur dan jenis kelamin pasien dan rata-rata lama menginap pada pasien yang diberikan pelayanan rawat inap. Visualisasi demografi ini bertujuan untuk mengetahui berapakah rata-rata umur pasien yang terkena penyakit DBD atau tidak (dapat dikostumisasi melalui filter) dan bagaimana distribusi kelompok umur pasien baik yang berjenis kelamin laki-laki dan perempuan. Pada visualisasi rata-rata lama menginap pada pasien rawat inap dapat memberikan informasi dan *insight* berapakah rata-rata lama menginap pada pasien rawat inap yang terkena penyakit DBD atau tidak dan bagaimana persebarannya jika didasarkan pada kelas hari lama menginap.



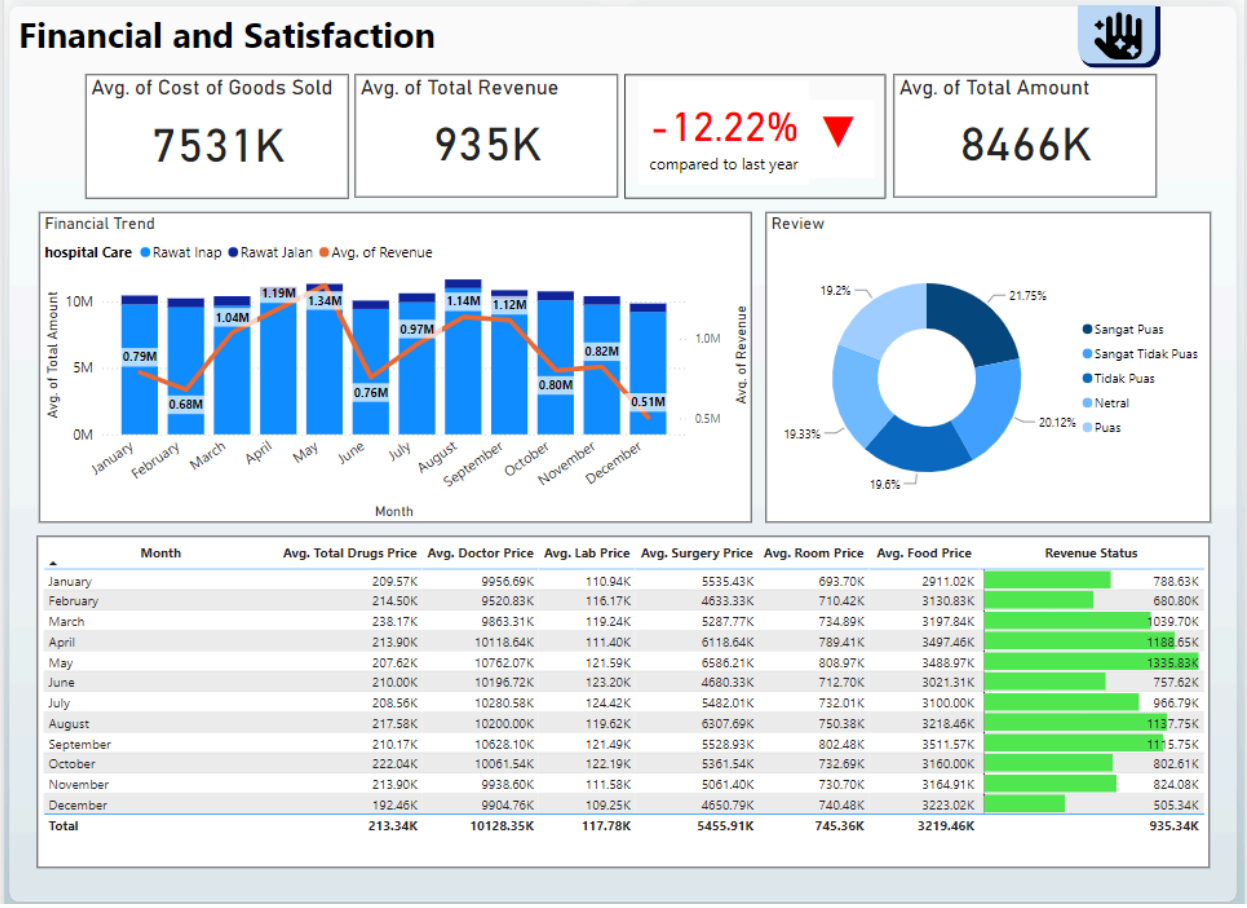
Gambar 2.3 *Mockup Dashboard Analytics* Demografi dan *Avg. Length of Stays*

Fitur yang tidak kalah pentingnya dalam melakukan *monitoring* dan *tracking* pasien pada suatu rumah sakit yaitu persebaran jumlah pasien untuk setiap bidang spesialisasi dokter, laboratorium, tipe obat yang diberikan dan tipe spesialisasi operasi. Masing-masing menu divisualisasikan menggunakan *stacked bar chart* untuk mengetahui jumlah pasien laki-laki dan perempuan jika berdasarkan pada masing-masing tipe layanan spesialisasi yang diberikan. Khusus pada menu *doctor specialization,* diberikan tambahan berupa *total doctor visit* beserta *growth*-nya jika dibandingkan dengan tahun sebelumnya yang bertujuan untuk memberikan informasi berapa total pelayanan dokter untuk seluruh bidang spesialisasi yang telah diberikan pada tahun tersebut. Fitur-fitur ini tentunya dapat memberikan kemudahan bagi manajemen rumah sakit dalam memberikan keputusan terkait manajemen pelayanan rumah sakit, seperti yaitu peningkatan jumlah dokter di departemen tertentu, penambahan jumlah kamar untuk pasien rawat inap, dan lain sebagainya. Disisi lain, jika ditinjau dari segi kualitas pelayanan dan masyarakat sebagai contoh pada Gambar 2.4, jenis spesialisasi dokter yang paling banyak dikunjungi oleh pasien adalah penyakit dalam sehingga dapat juga diadakan solusi pencegahan atau meminimalisir semakin meningkatnya penyakit dalam melalui penyuluhan maupun pemeriksaan dini pada masyarakat wilayah sekitar. Tentu saja, hal ini bermanfaat untuk meningkatkan kualitas hidup dibidang kesehatan.



Gambar 2.4 *Mockup Dashboard Analytics Doctor Specialization, Laboratorium, Drugs Type,* dan *surgery Specialization*

Fitur yang terakhir pada *dashboard* yaitu menu *financial and satisfaction.* Menu ini berisikan trend rata-rata *revenue* rumah sakit dan *total amount* pasien untuk setiap bulan serta persentase jumlah pasien untuk setiap *rating review* yang diberikan. Manajemen rumah sakit juga perlu mengetahui bagaimana perkembangan finansial rumah sakit agar proses transaksi untuk melakukan pembelian dan peningkatan layanan rumah sakit tidak terhambat. Selain itu, diberikan fitur *revenue status* untuk setiap bulannya agar mengetahui apakah pada bulan tersebut rumah sakit mengalami *profit* atau mengalami kerugian. Pada menu *satisfaction* digunakan *pie chart* untuk mengetahui persentase *review* pelayanan rumah sakit yang telah diberikan oleh pasien. Hal ini bertujuan untuk mengetahui apakah layanan yang diberikan sudah mampu memenuhi kepuasan pasien, atau masih diperlukan pembenahan dan peningkatan agar pasien dapat merasa puas atas pelayanan yang telah diberikan.



Month	Avg. Total Drugs Price	Avg. Doctor Price	Avg. Lab Price	Avg. Surgery Price	Avg. Room Price	Avg. Food Price	Revenue Status
January	209.57K	9956.69K	110.94K	5535.43K	693.70K	2911.02K	788.63K
February	214.50K	9520.83K	116.17K	4633.33K	710.42K	3130.83K	680.80K
March	238.17K	9863.31K	119.24K	5287.77K	734.89K	3197.84K	1039.70K
April	213.90K	10118.64K	111.40K	6118.64K	789.41K	3497.46K	1188.65K
May	207.62K	10762.07K	121.59K	6586.21K	808.97K	3488.97K	1335.83K
June	210.00K	10196.72K	123.20K	4680.33K	712.70K	3021.31K	757.62K
July	208.56K	10280.58K	124.42K	5482.01K	732.01K	3100.00K	966.79K
August	217.58K	10200.00K	119.62K	6307.69K	750.38K	3218.46K	1137.75K
September	210.17K	10628.10K	121.49K	5528.93K	802.48K	3511.57K	1115.75K
October	222.04K	10061.54K	122.19K	5361.54K	732.69K	3160.00K	802.61K
November	213.90K	9938.60K	111.58K	5061.40K	730.70K	3164.91K	824.08K
December	192.46K	9904.76K	109.25K	4650.79K	740.48K	3223.02K	505.34K
Total	**213.34K**	**10128.35K**	**117.78K**	**5455.91K**	**745.36K**	**3219.46K**	**935.34K**

Gambar 2.5 *Mockup Dashboard Analytics Financial and Satisfaction*

Mockup secara keseluruhan dapat diakses pada link berikut [https://intip.in/DigitalKesehatanDashboard]. Secara keseluruhan, *Dashboard Analytics* pada *project* ini mampu memberikan informasi dalam melakukan *monitoring* dan *tracking* pasien dengan penyakit Demam Berdarah Dengue (DBD) atau tidak baik dengan pelayanan rawat jalan maupun rawat inap pada suatu rumah sakit mulai dari segi pertumbuhan jumlah pasien, persebaran demografi berdasarkan umur dan jenis kelamin pasien, rata-rata lama menginap pada pasien yang diberikan pelayanan rawat inap, persebaran jumlah pasien untuk setiap bidang spesialisasi dokter, laboratorium, tipe obat yang diberikan, tipe spesialisasi operasi hingga *financial* dan kepuasan pasien meliputi trend rata-rata *revenue* rumah sakit dan *total amount* pasien untuk setiap bulan dan persentase jumlah pasien untuk setiap *rating review* yang diberikan. Sehingga, dapat membantu pihak rumah sakit dalam melakukan pengambilan keputusan untuk melakukan peningkatan pelayanan rumah sakit dan bagi pihak pasien dapat

mengetahui serta mengantisipasi layanan bagaimana yang akan diperoleh jika mengidap penyakit Demam Berdarah Dengue (DBD) maupun penyakit yang lainnya.

2.2.5 *Machine learning* untuk Estimasi Biaya

Setelah data diproses melalui DAG (*Directed Acyclic Graph*) dan divisualisasikan dalam bentuk *dashboard analytics* menggunakan Power BI, langkah selanjutnya adalah menerapkan model *machine learning* untuk mengestimasi biaya perawatan pasien, khususnya pasien yang mengalami Demam Berdarah Dengue (DBD). Data yang telah dibersihkan dan ditransformasikan menyediakan dasar yang kuat untuk membangun model prediktif yang dapat membantu rumah sakit dalam manajemen keuangan dan perencanaan sumber daya.

Model *machine learning* ini dirancang untuk memprediksi biaya perawatan pasien dengan menggunakan berbagai fitur yang relevan, seperti usia, jenis kelamin, durasi rawat inap, jenis layanan medis yang diterima (rawat jalan atau rawat inap), spesialisasi dokter yang menangani, informasi asuransi, obat, dan data terkait lainnya. Semua data ini dianalisis terlebih dahulu sebelum digunakan dalam model. Dengan mengintegrasikan fitur-fitur tersebut setelah melalui proses analisis, model ini diharapkan dapat memberikan prediksi biaya akhir kepada pasien, membantu rumah sakit dalam memberikan estimasi biaya yang akurat. Ini tidak hanya memungkinkan pasien untuk merencanakan keuangan mereka dengan lebih baik, tetapi juga meningkatkan efisiensi manajemen sumber daya dan operasional rumah sakit. Dengan demikian, model ini menjadi alat yang berharga dalam manajemen keuangan dan pelayanan kesehatan.

Proses pengembangan model *machine learning* ini mengikuti pendekatan *data science life cycle* [1] yang terdiri dari beberapa tahapan, dimulai dari *business understanding* dan *data preparation*, di mana data dari DAG diambil dan disiapkan untuk analisis. Setelah itu, dilakukan *exploratory data analysis* untuk memahami karakteristik dan pola dalam data. Tahap berikutnya adalah *model development*, di mana model *machine learning* dikembangkan dan disesuaikan dengan data yang telah dipersiapkan. Setelah model selesai, dilakukan *model evaluation* untuk memastikan bahwa model memberikan hasil yang diharapkan. Langkah terakhir adalah *model deployment*, di mana model yang sudah dikembangkan diintegrasikan agar dapat digunakan dengan mudah.

(i) *Data Preparation*

Data dari file CSV yang telah melalui proses *cleaning* dan *transformation* dari DAG diimpor dan dibaca sebagai *dataframe* ke dalam *Jupyter Notebook* menggunakan *library pandas* dan fungsi *read_csv* seperti pada potongan kode dibawah ini. Tahap ini penting untuk mempersiapkan data agar siap untuk analisis lebih lanjut.

```
import pandas as pd
df = pd.read_csv('FP_team1_data_clean.csv')
```

(ii) *Exploratory Data Analysis*

Eksplorasi data analisis dilakukan untuk memahami karakteristik *dataset* yang telah dipersiapkan dan mengidentifikasi faktor-faktor yang dapat digunakan untuk memprediksi estimasi biaya perawatan penyakit DBD. Langkah-langkah ini mencakup melihat statistik deskriptif untuk mendapatkan gambaran umum tentang data, visualisasi data untuk mendeteksi pola dan hubungan antar variabel, serta identifikasi pola menarik yang dapat berkontribusi dalam pembangunan model prediktif. Analisis ini mencakup *univariate analysis, bivariate analysis*, dan *multivariate analysis*.

Untuk melihat karakteristik data dapat menggunakan *df.info()*, yang mengecek nilai yang hilang (*missing value*), dan mendeteksi duplikasi data. Ternyata terdapat data yang belum ditransformasi saat menjalankan DAG, sehingga terdapat nilai yang hilang. Contohnya, terdapat nilai *NaN* yang seharusnya menunjukkan bahwa tidak dilakukan operasi. Data ini perlu diubah terlebih dahulu agar analisis lebih mudah dilakukan. Setelah itu, kolom-kolom data dikelompokkan berdasarkan tipe datanya untuk mempermudah analisis. Kolom-kolom ini dikelompokkan menjadi *categorical*, *numerical*, dan *temporal*. Kode yang digunakan dapat dilihat di bawah ini:

```
# Fill missing values in categorical columns (forgot to change it in dags)
categorical_cols = ["room_type", "surgery", "lab"]
df[categorical_cols] = df[categorical_cols].fillna("Tidak Digunakan")

# Fill missing values in numerical columns
numerical_cols = ["food_price", "room_price", "surgery_price", "lab_price"]
df[numerical_cols] = df[numerical_cols].fillna(0)

# Grouping columns by data type
cats = []
nums = []
temps = []
```

```
for column, dtype in df.dtypes.items():
    if dtype == 'object' or dtype == 'bool':
        cats.append(column)
    elif dtype == 'int64' or dtype == 'float64':
        nums.append(column)
    elif dtype == 'datetime64[ns]':
        temps.append(column)

# Display the grouped columns
print("Categorical columns:", cats)
print("Numerical columns:", nums)
print("Temporal columns:", temps)
```

Selanjutnya, analisis statistik deskriptif dilakukan pada setiap grup tipe data menggunakan fungsi *.describe()*. Untuk analisis univariat, digunakan visualisasi distribusi plot. Analisis bivariat dilakukan untuk melihat hubungan antara fitur-fitur dengan target total_amount, menggunakan *scatter plot* untuk kolom numerik dan *box plo*t untuk kolom kategorikal, dengan bantuan matplotlib dan seaborn untuk visualisasi. Analisis multivariat juga dilakukan untuk melihat korelasi antar fitur. Untuk data numerik, digunakan *correlation matrix* atau Pearson dengan fungsi .corr(), sedangkan untuk data kategorikal, digunakan chi-square terhadap fitur target menggunakan library scipy, karena Pearson atau correlation tidak bisa diterapkan pada data kategorikal. Hal ini dilakukan untuk memahami fitur-fitur pada *dataframe* dan menentukan fitur yang akan digunakan untuk membuat model prediktif biaya akhir yang akan dibayarkan pasien DBD.

(iii) *Feature Engineering* dan *Selection*

Feature engineering melibatkan pembuatan dan transformasi fitur untuk meningkatkan kinerja model. Teknik-teknik yang digunakan termasuk pembuatan fitur baru atau transformasi fitur, seperti mentransformasi fitur *age* menjadi *age group* untuk kesederhanaan. Fitur kategorikal diubah menjadi numerik menggunakan teknik seperti One-Hot Encoding dan Label Encoding. Selain itu, *feature selection* dilakukan untuk memilih fitur-fitur yang paling relevan berdasarkan EDA yang telah dilakukan. Standarisasi diterapkan pada data final agar berada pada skala yang sama menggunakan MinMaxScaler dari *library* sklearn, yang mengubah nilai fitur sehingga berada dalam rentang 0 hingga 1 [2]. Terakhir, data dipisahkan menjadi set pelatihan dan set pengujian dengan rasio 80% untuk pelatihan dan 20% untuk pengujian untuk menguji kinerja model secara independen.

```
# Define age groups
age_bins = [0, 18, 30, 45, 60, float('inf')]
```

```
age_labels = ['0-18', '19-30', '31-45', '46-60', '60+']

# Assign age groups to the data
df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels,
right=False)
# Label encode (mapping) age_group
age_mapping = {'0-18': 0, '19-30': 1, '31-45': 2, '46-60': 3, '60+': 4}
df['age_group'] = df['age_group'].replace(age_mapping).astype(int)

# Label encode (mapping) review
review_mapping = {'Sangat Tidak Puas': 0, 'Tidak Puas': 1, 'Netral': 2,
'Puas': 3, 'Sangat Puas': 4}
df['review_encoded'] = df['review'].replace(review_mapping).astype(int)

# Label encode (mapping) room_type
room_mapping = {'Tidak Digunakan': 0, 'Kelas 3': 1, 'Kelas 2': 2, 'Kelas 1':
3, 'VIP': 4}
df['room_type_encoded'] = df['room_type'].replace(room_mapping).astype(int)

# One-Hot Encoding
from sklearn.preprocessing import OneHotEncoder

onehot_cols = ['branch', 'hospital_care', 'payment', 'gender', 'drug_brand',
'drug_type', 'doctor', 'lab']

for col in onehot_cols:
    onehots = pd.get_dummies(df[col], prefix=col, dtype=int)  # Specify
dtype=int to convert to integers
    df = df.join(onehots)
    df = df.drop(col, axis=1)

#Feature Selection
final_df = df.drop(column_to_drop, axis=1)

# Separate features and target
X = final_df.drop('total_amount', axis=1)
y = final_df['total_amount']

# Initialize the MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_mms = mms.fit_transform(X)

# splitting data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_mms, y, test_size=0.2,
random_state=42)
```

(iv) ***Model Development***

Setelah tahap *Exploratory Data Analysis* (EDA) dan melakukan *feature engineering*, *feature selection*, serta standarisasi pada fitur yang akan digunakan, langkah selanjutnya adalah mengembangkan model *machine learning*. Model yang digunakan termasuk Linear Regression,

Decision Tree, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), Random Forest, dan XGBoost.

Linear Regression dipilih karena kesederhanaan dan efisiensi dalam menangani hubungan linier [3]. Decision Tree digunakan karena kemampuannya menangani data non-linear dan interpretasi yang mudah [4]. K-Nearest Neighbors (KNN) efektif untuk data dengan distribusi tidak teratur dan prediksi non-linear [5]. Support Vector Regression (SVR) handal dalam menangani data berdimensi tinggi dan hubungan non-linear [6]. Random Forest, sebagai model *ensemble*, meningkatkan akurasi prediksi dan ketahanan terhadap *overfitting* [7]. XGBoost dipilih karena efisiensi komputasi, kemampuan mengatasi *overfitting*, dan performa unggulnya [8].

Pemilihan model dengan jenis yang berbeda dilakukan untuk mengevaluasi kinerja masing-masing model dan menentukan model yang paling cocok untuk prediksi total_amount, yang merupakan masalah regresi. Dengan membandingkan berbagai model, diharapkan dapat dipilih model yang memberikan hasil terbaik berdasarkan metrik evaluasi yang digunakan.

(v) *Model Evaluation*

Setelah model dikembangkan, tahap berikutnya adalah melakukan evaluasi untuk memastikan bahwa model memberikan hasil yang diharapkan. Teknik evaluasi yang digunakan termasuk *cross-validation* untuk menguji model pada berbagai subset data, serta metrik seperti R-squared, yang mengukur seberapa baik model cocok dengan data, Mean Absolute Error (MAE), yang memberikan estimasi rata-rata dari kesalahan prediksi, Mean Squared Error (MSE), yang mengukur rata-rata dari kuadrat kesalahan prediksi, Root Mean Squared Error (RMSE), yang merupakan akar kuadrat dari MSE dan memberikan ukuran dari seberapa jauh prediksi model dari nilai sebenarnya, serta Mean Absolute Percentage Error (MAPE), yang mengukur persentase rata-rata dari kesalahan prediksi relatif terhadap nilai sebenarnya [9]. Metrik-metrik ini dievaluasi menggunakan fungsi-fungsi dari *library sklearn.metrics*, seperti r2_score, mean_absolute_error, mean_squared_error, mean_squared_error, mean_squared_error, dan mean_absolute_percentage_error. Metrik-metrik ini membantu dalam mengukur akurasi dan efisiensi prediktif model secara komprehensif.

(vi) *Model Deployment*

Langkah terakhir adalah *model deployment*, di mana model yang sudah dikembangkan di *deploy* menggunakan streamlit sebagai demo untuk digunakan dalam prediksi biaya perawatan

pasien secara real-time yang memungkinkan akses mudah dan cepat bagi staf rumah sakit dan pasien untuk mendapatkan estimasi biaya yang akurat bagi pasien DBD.

2.2.6 *Deployment*

Setelah model *machine learning* dan LLM (*Large Language Model*) dikembangkan dan dievaluasi, langkah penting selanjutnya adalah melakukan *deployment* model tersebut pada *Streamlit Community Cloud*. Ini memungkinkan *end user* untuk berinteraksi dengan model dan data secara mudah melalui *interface web yang user friendly,* Dengan menggunakan Streamlit, aplikasi dapat di-*deploy* dengan cepat dan efisien tanpa memerlukan pengaturan server yang kompleks.

(i) *Streamlit App Setup*

Streamlit dipilih karena kemudahan dan kesederhanaannya dalam membuat aplikasi web interaktif. Deployment melibatkan pembuatan aplikasi web interaktif dimana pengguna dapat memasukkan detail pasien untuk mendapatkan prediksi biaya dan melakukan query terhadap data menggunakan bahasa alami.

(ii) CSS untuk Peningkatan *Estetika*

Untuk meningkatkan pengalaman pengguna, diterapkan CSS khusus untuk menata aplikasi. Ini mencakup pengaturan warna latar belakang, penyesuaian tampilan tombol, dan penataan kolom input agar terlihat profesional dan kohesif.

```
'''css
body {
    font-family: Arial, sans-serif;
    background-color: #f5f5f5;
    color: #333;
    margin: 0;
    padding: 0;
    overflow-x: hidden;
}

.stApp {
    max-width: 100%;
    margin: 0;
    padding: 5px 10px;
}

h1, h2, h3 {
    color: #004c97;
    font-size: 1.5em;
}

.stButton>button {
    background-color: #004c97;
```

```
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s, transform 0.2s;
    font-size: 1em;
}

.stButton>button:hover {
    background-color: #003366;
    transform: scale(1.05);
}
'''
and continue….
```

Snippet CSS ini menunjukkan set up basic untuk menata tampilan body dan container utama app, memastikan *interface* yang bersih dan *user friendly.*

(iii) Integrasi Model Machine Learning

Fitur utama dari app ini menggunakan model XGBoost yang memprediksi total biaya berdasarkan input pengguna. Model ini diload bersama dengan *scaler* dan nama kolom untuk memproses data input. Fungsi ini memastikan bahwa model dan resource terkait dimuat secara efisien, dan juga memanfaatkan *caching* Streamlit untuk mengoptimalkan performa.

```
Python
'''
@st.cache_resource
def load_ml_model():
    model = joblib.load('XGBoost_best_model.pkl')
    scaler = joblib.load('minmax_scaler.pkl')
    column_names = joblib.load('column_names.pkl')
    return model, scaler, column_names

'''
```

Fungsi ini memastikan bahwa model dan *resource* terkait dimuat secara efisien, ini juga memanfaatkan *caching* Streamlit untuk mengoptimalkan performa.

(iv) Penanganan Data Input

App ini memproses berbagai detail pasien melalui widget interaktif. Data tersebut diproses dan disiapkan untuk prediksi oleh model. Widget interaktif ini memungkinkan pengguna untuk memasukkan informasi seperti nama pasien, kelompok umur, jenis kelamin, cabang rumah sakit, metode pembayaran, dan informasi lainnya yang relevan.

```
Python
'''
# Fungsi untuk kalkulasi days_diff untuk rawat inap
def calculate_days_diff(date_in, date_out):
    if date_in >= date_out:
        st.error("Tanggal keluar harus setelah tanggal masuk.")
        return None
    return (date_out - date_in).days + 1

# Fungsi untuk ML prediction page
def ml_prediction_page():
    model, scaler, column_names = load_ml_model()

    st.write("## Prediksi Biaya Total dengan ML")

    # Input patient name
    patient_name = st.text_input('Nama Pasien:', value="", help="Masukkan nama
pasien")


    # Organize input ke sections tertentu
    st.write("### Informasi Pasien")
    age_group = st.selectbox('Umur:', ['0-18', '19-30', '31-45', '46-60',
'60+'], help="Pilih kelompok umur pasien")
    age_mapping = {'0-18': 0, '19-30': 1, '31-45': 2, '46-60': 3, '60+': 4}
    age_group_value = age_mapping[age_group]
    gender = st.selectbox('Jenis kelamin:', ['Laki-laki', 'Perempuan'],
help="Pilih jenis kelamin pasien")
    gender_value = 1 if gender == 'Laki-laki' else 0
    branch = st.selectbox('Cabang RS Siloam:', ['RSMA', 'RSMD', 'RSMS'],
help="Pilih cabang rumah sakit")
    branch_values = {'branch_RSMA': 0, 'branch_RSMD': 0, 'branch_RSMS': 0}
    branch_values[branch] = 1
    payment = st.selectbox('Metode Pembayaran:', ['Asuransi', 'Pribadi'],
help="Pilih metode pembayaran")
    payment_value = 1 if payment == 'Asuransi' else 0

    st.write("### Informasi Perawatan")
    hospital_care = st.radio('Tipe perawatan rumah sakit:', ['Rawat Inap',
'Rawat Jalan'], help="Pilih tipe perawatan")
    if hospital_care == 'Rawat Inap':
        date_in = st.date_input('Tanggal Masuk', value=datetime.today(),
help="Pilih tanggal masuk")
        date_out = st.date_input('Tanggal Keluar', value=datetime.today(),
help="Pilih tanggal keluar")
        days_diff = calculate_days_diff(date_in, date_out)
    else:
        days_diff = 0.0

    room_type_encoded = st.selectbox('Tipe Kamar:', ['Tidak Digunakan', 'Kelas
3', 'Kelas 2', 'Kelas 1', 'VIP'], help="Pilih tipe kamar")
    room_mapping = {'Tidak Digunakan': 0, 'Kelas 3': 1, 'Kelas 2': 2, 'Kelas
1': 3, 'VIP': 4}
    room_type_value = room_mapping[room_type_encoded]
    hospital_care_value = 1 if hospital_care == 'Rawat Inap' else 0
```

```
    st.write("### Informasi Obat dan Dokter")
    drug_brand = st.selectbox('Merk Obat:', ['Blackmores', 'Calpol',
'Diclofenac', 'Enervon-C', 'Holland & Barrett',
                                             'Naproxen', 'Panadol', 'Paramex',
'Tramadol'], help="Pilih merk obat")
    drug_brand_values = {f'drug_brand_{brand}': 0 for brand in ['Blackmores',
'Calpol', 'Diclofenac', 'Enervon-C',
                                                                'Holland &
Barrett', 'Naproxen', 'Panadol', 'Paramex',
                                                                'Tramadol']}
    drug_brand_values[f'drug_brand_{drug_brand}'] = 1
    drug_type = st.selectbox('Tipe obat:', ['Pereda Nyeri', 'Umum',
'Vitamin'], help="Pilih tipe obat")
    drug_type_values = {f'drug_type_{type}': 0 for type in ['Pereda Nyeri',
'Umum', 'Vitamin']}
    drug_type_values[f'drug_type_{drug_type}'] = 1
    drug_quantity = st.number_input('Jumlah Obat:', value=1, help="Masukkan
jumlah obat")
    doctor = st.selectbox('Dokter:', ['Penyakit Dalam', 'Umum'], help="Pilih
dokter")
    doctor_values = {f'doctor_{doc}': 0 for doc in ['Penyakit Dalam', 'Umum']}
    doctor_values[f'doctor_{doctor}'] = 1
    lab = st.selectbox('Uji Lab:', ['Hematologi', 'Kimia Darah', 'Serologi'],
help="Pilih uji lab")
    lab_values = {f'lab_{test}': 0 for test in ['Hematologi', 'Kimia Darah',
'Serologi']}
    lab_values[f'lab_{lab}'] = 1

    # Collect input datas
    new_data = {
        'drug_quantity': drug_quantity,
        'days_diff': days_diff,
        'age_group': age_group_value,
        'room_type_encoded': room_type_value,
        'branch_RSMA': branch_values['branch_RSMA'],
        'branch_RSMD': branch_values['branch_RSMD'],
        'branch_RSMS': branch_values['branch_RSMS'],
        'hospital_care_Rawat Inap': hospital_care_value,
        'payment_Asuransi': payment_value,
        'gender_Laki-laki': gender_value,
        **drug_brand_values,
        **drug_type_values,
        **doctor_values,
        **lab_values
    }

    # Convert ke DataFrame
    new_input_df = pd.DataFrame([new_data])
    new_input_df = new_input_df.reindex(columns=column_names, fill_value=0)

'''
```

Snippet ini menunjukkan bagaimana input pengguna dikumpulkan, termasuk nama pasien, kelompok umur, jenis kelamin, cabang rumah sakit, dan metode pembayaran. Input yang

dipilih akan dipetakan ke suatu value yang dapat diproses oleh model ML. Data ini kemudian digunakan untuk memproses prediksi biaya.

(v) Prediksi dan Output

Setelah input dikumpulkan, data tersebut diproses dan diskalakan sebelum dimasukkan ke model untuk prediksi. Biaya yang diprediksi kemudian ditampilkan kepada user.

```
Python
'''
    # Convert ke DataFrame
    new_input_df = pd.DataFrame([new_data])
    new_input_df = new_input_df.reindex(columns=column_names, fill_value=0)

    # Scale input data
    new_input_scaled = scaler.transform(new_input_df)

    # Predict
    if st.button("Predict"):
        if days_diff is not None:
            predictions = model.predict(new_input_scaled)
            formatted_prediction = f"Rp {predictions[0]:,.0f}"
            patient_name_display = patient_name if patient_name else "ini"
            st.write(f"Biaya yang pasien {patient_name_display} perlu bayar
ialah sebanyak <b>{formatted_prediction}</b>", unsafe_allow_html=True)
        else:
            st.write("Pastikan input tanggal valid, lalu coba lagi!")



'''
```

Snippet kode ini mengilustrasikan bagaimana prediksi dijalankan dan ditampilkan berdasarkan input pengguna. Dengan menekan tombol "Predict", prediksi biaya akan dihitung dan ditampilkan dalam format yang mudah dibaca.

(vi) Integrasi LLM untuk Query Data

Selain *machine learning*, kami juga mencoba fitur tambahan yaitu mengintegrasikan dan memanfaatkan LLM untuk *query* data transaksi pasien yang dapat lebih mudah dilakukan oleh *user*. Integrasi LLM memungkinkan pengguna untuk melakukan query terhadap data transaksi pasien menggunakan natural language (bahasa non programming). Fitur ini memanfaatkan model OpenAI GPT-3.5 untuk memproses instruksi *query* user menjadi *query library* panda yang diproses dan dijalankan oleh *library* pandasai.

```
Python
'''
```

```
from pandasai import SmartDataframe
from pandasai.llm import OpenAI
from pandasai.connectors import PandasConnector
from pandasai.helpers.openai_info import get_openai_callback

# Initialize OpenAI LLM
llm_t0 = OpenAI(api_token=os.getenv("OPENAI_API_KEY"), temperature=0, seed=26)

# Load dataset
@st.cache_data
def load_data():
    return pd.read_csv('mini_data_rev.csv')

df = load_data()

# Create PandasConnector and SmartDataframe
connector = PandasConnector({"original_df": df})
sdf = SmartDataframe(connector, config={"llm": llm_t0, "conversational":
False})

# User query input
user_query = st.text_input("Kamu dapat query dan bertanya mengenai data rumah
sakit disini", key="user_query")

if st.button("Kirim"):
    if user_query:
        try:
            with get_openai_callback() as cb:
                response = sdf.chat(user_query)
                st.write("**Jawaban:**")
                st.write(response)
                st.write(f"Tokens terpakai: {cb.total_tokens}")
                st.write(f"Total Cost dalam (USD): ${cb.total_cost:.6f}")
        except Exception as e:
            if "context_length_exceeded" in str(e):
                st.error("Pertanyaan Anda terlalu panjang. Harap perpendek
pertanyaan Anda dan coba lagi.")
            else:
                st.error(f"Terjadi kesalahan: {e}")
    else:
        st.error("Masukkan pertanyaan terlebih dahulu.")

# Function untuk display image
def display_image(image_path, width=300, center=False):
    if center:
        st.image(image_path, width=width, use_column_width='auto')
    else:
        st.image(image_path, width=width)

'''
```

(vii) Penambahan halaman beranda dan bantuan

App Streamlit ini juga mencakup halaman beranda dan bantuan untuk memudahkan pengguna dalam memahami tujuan app dan cara menggunakannya. Halaman beranda berisi informasi umum tentang project kami, sedangkan halaman bantuan memberikan panduan penggunaan fitur-fitur app yang dideploy.

```
Python
'''
def homepage():
    st.title("Deploy App Final Project Kelompok 01 Hactiv8")

    # Display main page
    main_logo_path = 'Website_Logo_Bithealth_.png'
    display_image(main_logo_path, width=300)

    st.write("""
    **Disusun oleh:**
    - Arya Sangga Buana (Data Trainee)
    - Nadya Novahina (Data Trainee)
    - Rafi Athallah Seniang (Data Trainee)
    - Rosita Laili Udhiah (Data Trainee)
    """)

    st.write("""
    ### Tentang Proyek Ini
    App ini merupakan bagian dari Final Project kami di program training

continue…..
'''
```

```
Python
'''
def help_page():
    st.write("""
    ## Bantuan
    Di halaman ini, Anda dapat menemukan panduan dan tips untuk menggunakan
fitur Tanya BitAI serta Prediksi Biaya Total dengan Machine Learning.

    # Machine Learning
    **Petunjuk Penggunaan Prediksi Biaya Total**:
    - Masukkan informasi pasien dan perawatan yang diminta pada form yang
tersedia.
    - Pastikan semua input diisi dengan benar dan lengkap.
    - Tekan tombol "Predict" untuk mendapatkan estimasi biaya perawatan.
    - Jika terdapat kesalahan atau input tidak valid, periksa kembali data
yang dimasukkan.

    **Contoh Input untuk Prediksi Biaya Total**:
    - Nama Pasien: "Budi"
    - Umur: "19-30"
    - Jenis Kelamin: "Laki-laki"
```

```
    - Cabang RS Siloam: "RSMA"

continue…..
'''
```

BAB III
ANALISIS HASIL DAN PEMBAHASAN

3.1 *Analytics Insight* Dari *Dashboard Analytics*

Jika ditinjau pada data, sekitar 6% dari total pasien keseluruhan terindikasi mengidap penyakit DBD dengan pertumbuhan fluktuatif pada setiap tahunnya. Terjadi kenaikan pada jumlah pasien DBD pada tahun 2022 jika dibandingkan dengan tahun 2021. Untuk setiap tahun, hampir 60% pasien DBD ditangani dengan rawat inap. Jika ditinjau dari segi medis, pasien DBD direkomendasikan untuk ditangani dengan rawat inap disebabkan beberapa hal, yaitu: (i) Pasien DBD membutuhkan pemantauan ketat terhadap tanda-tanda vital seperti tekanan darah, suhu tubuh, dan tanda-tanda syok *dengue*, (ii) pemberian infus adalah cara yang efektif untuk memberikan cairan yang diperlukan secara cepat dan tepat guna mencegah atau mengatasi dehidrasi, dan (iii) perawatan inap memungkinkan penanganan cepat terhadap komplikasi yang mungkin timbul, seperti pendarahan internal, gangguan organ, atau syok. Penanganan segera dan tepat dapat menyelamatkan nyawa pasien. Jika berdasarkan data demografi, rata-rata umur pasien DBD baik yang berjenis kelamin perempuan maupun laki-laki berada pada rentang 48 hingga 53 tahun dengan rata-rata lama menginap selama 3 hari.

Sebesar 52% pasien DBD cenderung ditangani oleh dokter umum. Akan tetapi jika ditinjau berdasarkan jenis pelayanan, pasien rawat jalan lebih banyak ditangani oleh dokter penyakit dalam sedangkan pada pasien dengan pelayanan rawat inap cenderung melakukan perawatan pada dokter umum. Biaya yang harus dikeluarkan saat ditangani oleh dokter penyakit dalam lebih besar jika dibandingkan dengan dokter umum. Sehingga jika ditinjau dari segi total pengeluaran yang perlu dibayarkan oleh pasien, pasien yang ditangani oleh dokter penyakit dalam dapat memberikan *revenue* rumah sakit lebih banyak jika dibandingkan dengan pasien yang ditangani oleh dokter umum. Disisi lain, rata-rata *total amount* dari transaksi pasien DBD rawat jalan sebesar Rp276.556 dimana lebih kecil daripada pasien DBD rawat inap dengan rata-rata sebesar Rp1.103.020. Hal ini dipengaruhi juga dengan banyaknya kunjungan yang dilakukan oleh dokter dimana pada rawat jalan, dokter hanya berkunjung sekali sehingga biaya yang dikeluarkan lebih terjangkau. Pasien DBD rawat inap juga harus mengeluarkan biaya untuk

pemberian infus, layanan kamar dengan rata-rata biaya sebesar Rp909.400 dan pelayanan makanan dengan rata-rata biaya Rp394.832.

Selanjutnya jika ditinjau pada penanganan laboratorium, pasien DBD lebih dominan melakukan pengecekkan pada laboratorium bidang Kimia Darah dengan sebanyak 38% pasien dari total seluruh pasien yang mengidap penyakit DBD. Disamping itu, sebanyak 33% pasien DBD melakukan pengecekkan pada laboratorium bidang Hematologi dan sebanyak 29% pasien DBD pada laboratorium bidang Serologi. Dari ketiga laboratorium yang melayani pengecekkan penyakit DBD, laboratorium bidang Serologi memerlukan biaya yang paling mahal dan laboratorium bidang Hematologi memerlukan biaya yang paling murah. Sehingga, dapat diasumsikan bahwa pasien cenderung memilih melakukan pengecekkan pada laboratorium dengan biaya yang tidak terlalu mahal. Hal ini didukung dengan data rata-rata biaya pengeluaran dari transaksi pasien DBD untuk melakukan pengecekkan pada laboratorium membutuhkan biaya sebesar Rp163.460.

Kemudian, pada transaksi pemberian dan pembelian obat, sebesar 36% pasien DBD diberikan obat umum, 34% pasien DBD diberikan obat pereda nyeri, dan 30% pasien DBD diberikan obat vitamin. Pembelian tipe obat umum memerlukan biaya yang paling terjangkau dan tipe obat vitamin memerlukan biaya yang paling mahal jika dibandingkan dengan ketiga tipe obat yang diberikan untuk pasien DBD. Hal ini menunjukkan bahwa pasien DBD dominan memilih tipe obat umum dikarenakan harganya yang terjangkau dan tentunya dapat memberikan khasiat untuk menyembuhkan penyakit DBD. Pada data, rata-rata biaya pengeluaran dari transaksi pasien DBD untuk melakukan pembelian obat membutuhkan biaya sebesar Rp190.540.

Secara gambaran umum, pasien DBD dengan tipe pelayanan rawat inap memerlukan lebih banyak biaya dan memberikan lebih banyak *revenue* pada rumah sakit jika dibandingkan dengan pasien DBD dengan rawat jalan. Jika ditinjau secara keseluruhan dan berdasarkan rata-rata total biaya pengeluaran pada data, untuk pasien DBD rawat inap dapat memperoleh seluruh pelayanan mulai dari perawatan oleh dokter hingga pemberian obat membutuhkan biaya sebesar Rp2.763.202 dengan *profit* rumah sakit yang dihasilkan sebesar Rp520.130. Sedangkan, untuk pasien DBD rawat jalan membutuhkan biaya sebesar Rp627.776 dengan kerugian rumah sakit sebesar Rp2.357.180. Hal ini menunjukkan adanya perbedaan yang cukup jauh antara biaya operasional yang harus dibayarkan oleh rumah sakit dan pendapatan yang diterima. Jika ditinjau dari metode pembayaran yang digunakan, pada pasien DBD rawat inap 51% menggunakan

pembayaran pribadi dan 49% menggunakan asuransi. Disisi lain, seluruh pasien DBD rawat jalan menggunakan metode pembayaran pribadi.

3.2 *Machine Learning* untuk Estimasi Biaya

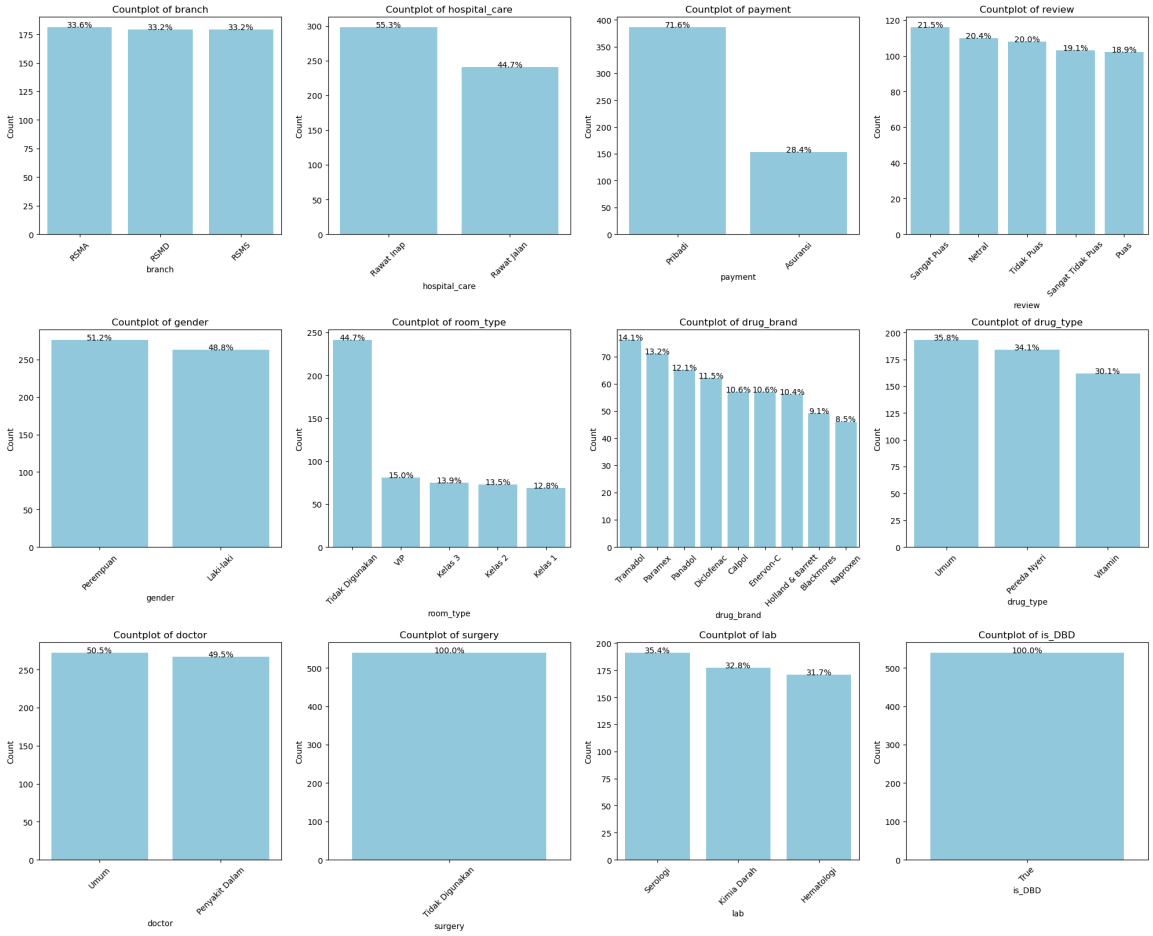
3.2.1 *Exploratory Data Analysis*

Analisis statistik deskriptif menunjukkan variasi luas dalam biaya dan penggunaan layanan medis untuk pasien DBD. Usia pasien rata-rata sekitar 50 tahun. Jumlah obat rata-rata sekitar 3 unit, dan biaya administrasi selalu 50.000. Biaya barang yang dijual (COGS) rata-rata sekitar 3.000.000. Harga kamar, obat, dan kunjungan dokter rata-rata masing-masing sekitar 125.000, 65.000, dan 733.000. Durasi perawatan baik rawat inap maupun rawat jalan rata-rata sekitar 2 hari dengan maksimum 6 hari. Total biaya perawatan rata-rata sekitar 2.200.000. Pendapatan sering negatif, menunjukkan perbedaan antara biaya yang ditagih dan pendapatan yang diterima. Pada kolom kategorikal, cabang rumah sakit RSMA paling banyak tercatat (181 entri). Mayoritas pasien menerima rawat inap (298 entri) dan pembayaran pribadi paling umum (386 entri). Ulasan "Sangat Puas" terbanyak (116 entri), dan mayoritas pasien perempuan (276 entri). Tipe kamar paling umum "Tidak Digunakan" (241 entri). Merek obat Tramadol paling sering digunakan (76 entri), dan tipe obat "Umum" paling umum (193 entri). Dokter yang paling banyak menangani pasien adalah "Umum" (272 entri). Pasien tidak menjalani operasi dan layanan lab paling banyak serologi (191 entri). Total data sebanyak 539 menunjukkan keragaman besar dalam biaya dan layanan. Pada kolom temporal, tanggal masuk pasien berkisar dari 1 Januari 2020 hingga 21 Desember 2023, dengan rata-rata 23 Januari 2022. Tanggal keluar berkisar dari 1 Januari 2020 hingga 25 Desember 2023, dengan rata-rata 25 Januari 2022.

Analisis lanjutan meliputi univariate, bivariate, dan multivariate analysis. Fitur dengan banyak nilai unik seperti nama pasien dan data ID tidak dimasukkan dalam analisis lanjutan, fokus pada fitur lebih informatif seperti jenis ruangan, obat, dokter, operasi, dan lab untuk memudahkan analisis dan visualisasi.

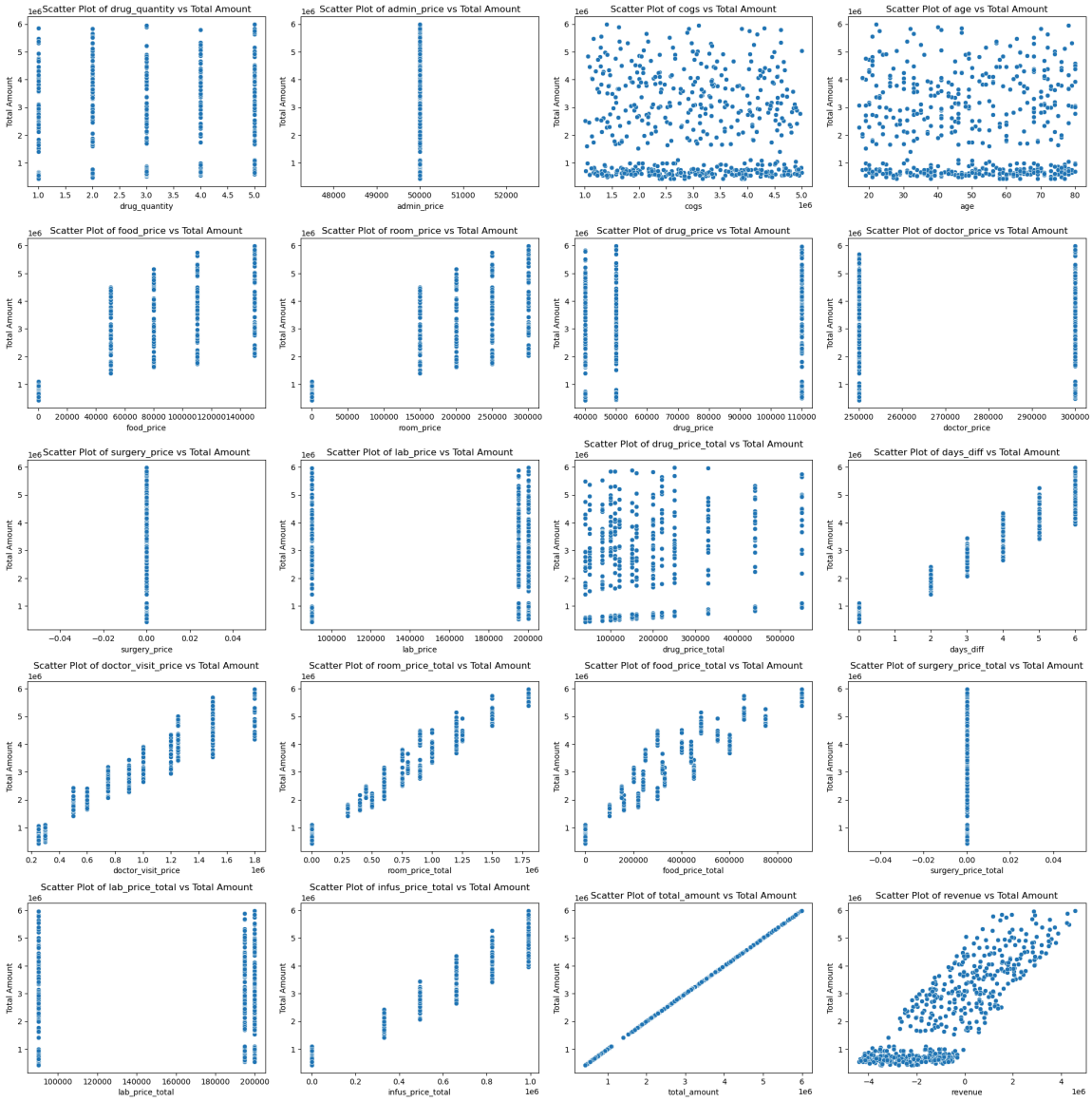
Gambar 3.1 Distribusi kolom numerikal

Berdasarkan gambar 3.1, distribusi kolom numerikal menunjukkan variasi yang signifikan. Distribusi drug_quantity hampir merata, sedangkan admin_price tetap konstan di 50.000. cogs dan age memiliki distribusi hampir seragam. food_price, room_price, dan drug_price menunjukkan variasi besar. Distribusi days_diff dan total_amount menunjukkan rentang luas, dengan beberapa pasien memiliki durasi dan biaya jauh di atas rata-rata. revenue cenderung lebih normal, menunjukkan variasi lebih seimbang. Analisis ini menggambarkan variasi signifikan dalam biaya dan penggunaan layanan medis untuk pasien DBD.



Gambar 3.2 Distribusi kolom kategorikal

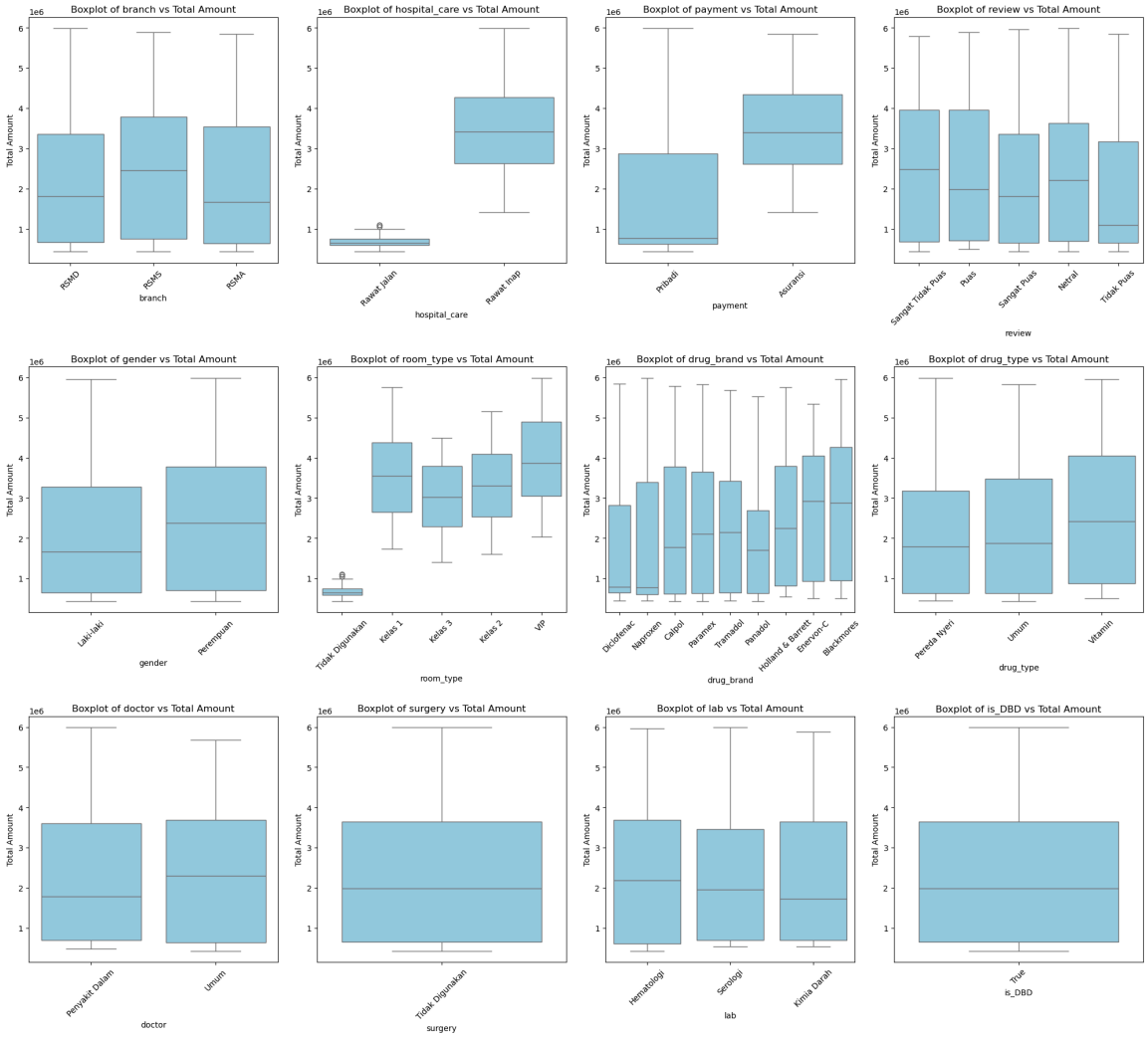
Gambar 3.2 menunjukkan distribusi kolom kategorikal. Cabang rumah sakit tersebar merata antara RSMA, RSYKD, dan RSMG. Mayoritas pasien menerima perawatan rawat inap (55,3%) dan menggunakan pembayaran pribadi (71,6%). Ulasan pasien bervariasi, dengan "Sangat Puas" paling banyak diisi (21,5%). Distribusi gender hampir seimbang antara perempuan (51,2%) dan laki-laki (48,8%). Tipe kamar yang paling umum adalah "Tidak Digunakan" (44,7%), diikuti oleh VIP dan kelas 3. Merek obat yang sering digunakan adalah Tramadol (14,1%), dengan tipe obat umum mendominasi (35,6%). Mayoritas pasien dirawat oleh dokter umum (50,5%) dan tidak menjalani operasi. Jenis lab yang sering digunakan adalah serologi (35,4%). Data ini menunjukkan distribusi seimbang dan variasi dalam jenis layanan yang diterima pasien DBD.



Gambar 3.3 Scatter Plot Kolom Numerikal Terhadap Total Amount

Berdasarkan *scatter plot* pada gambar 3.3, kolom numerikal seperti cogs, age, dan drug_quantity menunjukkan distribusi yang menyebar tanpa hubungan jelas dengan total_amount. admin_price dan surgery_price tidak menunjukkan variasi. Kolom seperti drug_price, days_diff, food_price, room_price, lab_price, dan infus_price_total juga menunjukkan distribusi menyebar, menandakan hubungan yang tidak konsisten dengan total_amount. Namun, terdapat hubungan linear antara total_amount dan kolom seperti day_diff, doctor_visit_price, room_price_total, food_price_total, infus_price_total, serta revenue, meskipun ada beberapa variasi. Analisis ini menunjukkan bahwa biaya seperti lama perawatan,

kunjungan dokter dan total harga kamar memiliki pengaruh signifikan terhadap total biaya perawatan pasien DBD.



Gambar 3.4 Box Plot Kolom Kategorikal terhadap Total Amount untuk Analisis Bivariat

Berdasarkan *boxplot* pada gambar 3.4 antara kolom kategorikal dan total_amount, pasien rawat inap dan metode pembayaran asuransi cenderung memiliki total_amount lebih tinggi. Tidak ada perbedaan signifikan dalam total_amount berdasarkan ulasan, gender, dan dokter. Tipe kamar VIP dan kelas 1 memiliki total_amount lebih tinggi. Distribusi total_amount berdasarkan drug_brand, drug_type, dan jenis lab tidak menunjukkan perbedaan signifikan. Analisis ini menunjukkan bahwa tipe perawatan, metode pembayaran, dan tipe kamar memiliki pengaruh terbesar terhadap total biaya perawatan pasien DBD.

Berdasarkan hasil korelasi, data terkait harga memiliki korelasi yang sangat tinggi dengan total_amount, terutama doctor_visit_price (0.98), room_price_total (0.94), food_price_total

(0.87), dan infus_price_total (0.83). Sebaliknya, beberapa kolom menunjukkan korelasi yang sangat rendah dengan total_amount, seperti admin_price (0.02), surgery_price (-0.03), dan doctor_price (0.02). Hasil uji Chi-square menunjukkan bahwa kolom hospital_care, payment, dan room_type memiliki p-value yang sangat kecil, menandakan hubungan signifikan dengan total_amount. Sebaliknya, kolom review dan is_DBD memiliki p-value yang cukup besar, menunjukkan tidak adanya hubungan signifikan dengan total_amount. Analisis ini menunjukkan bahwa jenis perawatan rumah sakit, metode pembayaran, dan tipe kamar memiliki pengaruh signifikan terhadap total biaya perawatan pasien DBD, sedangkan review dan status DBD tidak memiliki pengaruh signifikan.

3.2.2 *Feature Engineering dan Selection*

Feature engineering yang digunakan adalah mentransformasi fitur age menjadi age group untuk untuk mempermudah pemahaman dan analisis. Fitur kategorikal diubah menjadi bentuk numerik menggunakan teknik seperti One-Hot Encoding dan Label Encoding sehingga memungkinkan model untuk memproses informasi tersebut dengan baik. Pemilihan fitur dilakukan melalui analisis dan korelasi terhadap data target untuk memastikan hanya fitur-fitur yang paling berpengaruh yang digunakan dalam model. Fitur-fitur yang redundan, seperti fitur yang berkaitan dengan harga, tidak dimasukkan karena dapat digantikan oleh fitur lain yang tetap merepresentasikan informasi tersebut, seperti jenis dokter, jenis obat, dan lain sebagainya. Sehingga fitur yang digunakan adalah days_diff, hospital_care, room_type, payment, drug_type, branch, drug_brand, drug_quantity, gender, age_group, lab, doctor. Standarisasi yang digunakan adalah MinMaxScaler dari library sklearn, yang mengubah nilai fitur sehingga berada dalam rentang 0 hingga 1. Hal ini penting bagi beberapa algoritma machine learning agar beroperasi secara optimal. Terakhir, data dipisahkan menjadi set pelatihan dan set pengujian dengan rasio 80% untuk pelatihan dan 20% untuk pengujian untuk menguji kinerja model secara independen dan mencegah overfitting.

3.2.3 *Model Training dan Evaluation*

Model machine learning yang telah dikembangkan dievaluasi menggunakan berbagai metrik untuk memastikan akurasi dan efisiensinya. Teknik evaluasi meliputi cross-validation dan beberapa metrik seperti R-squared (R^2), Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), dan Mean Absolute Percentage Error (MAPE).

Evaluasi dilakukan dengan menggunakan library sklearn.metrics untuk menghitung metrik-metrik tersebut.

Tabel 3.1 Hasil Evaluasi Model Machine Learning

Model	CV Score	R2		MSE		MAE		RMSE		MAPE		Waktu prediksi (s)
		train	test	train	test	train	test	train	test	train	test	
Linear Regression	0.995	0.995	0.996	12237119100.96	12089829811.13	83477.84	79958.8	110621.51	109953.76	5.58%	4.82%	0.000s
Decision Tree Regression	0.988	1.000	0.991	0.00	24836805555.56	0.00	92731.48	0.00	157596.97	0.00%	3.37%	0.000s
KNN Regression	0.696	0.816	0.726	467740603248.26	784699925925.93	475512.76	551851.85	683915.64	885832.90	24.07%	22.11%	0.031s
SVR Regression	-0.062	-0.034	-0.118	2624624097547.16	3203017045014.53	1403116.60	1565263.02	1620069.16	1789697.47	107.78%	104.08%	0.016s
Random Forest Regression	0.993	0.999	0.996	1993289559.16	10683982361.11	25899.30	64321.30	44646.27	103363.35	0.99%	2.46%	0.019s
XGBoost Regression	0.997	1.00	0.998	45496.85	6102962261.32	142.44	43720.06	213.30	78121.46	0.02%	1.42%	0.000s

Berdasarkan Tabel 3.1, model Linear Regression menunjukkan performa baik dengan nilai R^2 tinggi dan kesalahan prediksi rendah pada data train dan test. Waktu prediksi sangat cepat, membuat model ini efisien. Decision Tree Regression mengalami overfitting pada data train, namun performa pada data test masih cukup baik. KNN Regression menunjukkan performa kurang baik dengan error tinggi dan R^2 rendah, meskipun waktu prediksi cukup cepat. SVR Regression memiliki performa buruk dengan R^2 negatif dan error sangat tinggi, sehingga tidak cocok untuk dataset ini. Random Forest Regression menunjukkan performa sangat baik dengan nilai R^2 tinggi dan error rendah pada data train dan test, serta waktu prediksi cepat. XGBoost Regression menunjukkan performa terbaik di antara semua model dengan nilai R^2 sangat tinggi dan error sangat rendah. Waktu prediksi yang sangat cepat juga menambah keunggulan model ini. Selanjutnya, model terbaik yaitu XGBoost Regression akan dilakukan hyperparameter tuning menggunakan RandomSearch untuk mengetahui apakah akan terdapat peningkatan skor evaluasi.

Tabel 3.2 Perbandingan Evaluasi Model Terbaik tanpa dan dengan Hyperparamater Tuning

Model	CV Score	R2		MSE		MAE		RMSE		MAPE		Waktu prediksi (s)
		train	test	train	test	train	test	train	test	train	test	
XGBoost Regression	0.997	1.00	0.998	45496.85	6102962261.32	142.44	43720.06	213.30	78121.46	0.02%	1.42%	0.000s

XGBoost Regression (Tuned)	0.997	0.999	0.999	148423314.15	1926154664.63	8966.73	27319.43	12182.91	43887.98	0.57%	1.14%	0.000s

Tabel 3.2 memberikan perbandingan evaluasi model XGBoost Regression sebelum dan sesudah dilakukan hyperparameter tuning. Sebelum tuning, model memiliki nilai R^2 yang sangat tinggi (0.998 untuk data train dan 1.00 untuk data test) dan MSE yang sangat rendah pada data train (45496.85) namun tinggi pada data test (6102962261.32), mengindikasikan kemungkinan overfitting. Setelah tuning, nilai R^2 tetap sangat tinggi (0.999 untuk data train dan test), namun MSE pada data test menurun signifikan menjadi 1926154664.63, menunjukkan bahwa model menjadi lebih mampu generalisasi atau memberikan prediksi yang lebih akurat pada data yang belum pernah dilihat sebelumnya. Selain itu, MAE dan RMSE pada data test menurun dari 43720.06 dan 78121.46 menjadi 27319.43 dan 43887.98, menandakan peningkatan akurasi prediksi. MAPE juga menurun dari 1.42% menjadi 1.14%, menunjukkan penurunan dalam kesalahan relatif prediksi. Waktu prediksi tetap efisien pada 0 detik dalam kedua kondisi, menunjukkan bahwa tuning tidak mempengaruhi efisiensi waktu model. Secara keseluruhan, hyperparameter tuning membuat model XGBoost Regression lebih stabil dan akurat, mengurangi overfitting dan meningkatkan kemampuan generalisasi, sehingga lebih ideal untuk prediksi data di masa mendatang.

3.3 *Deployment*

Dalam subbab ini, akan dibahas hasil dari *deployment model machine learning* dan LLM yang telah dilakukan pada *Streamlit Community Cloud*. Proses *deployment* ini bertujuan untuk memastikan bahwa model dan aplikasi dapat diakses dan digunakan oleh *end user* dengan mudah dan efisien.

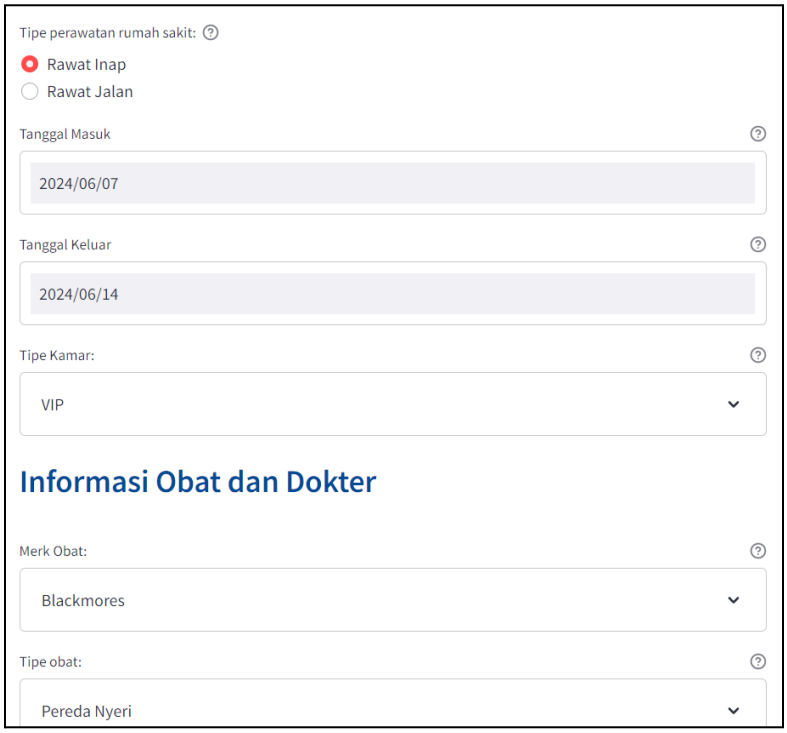
3.3.1 Hasil Deployment *Model Machine Learning*

Setelah model machine learning berhasil di-deploy, aplikasi Streamlit memungkinkan user untuk memasukkan berbagai informasi tentang pasien dan mendapatkan prediksi biaya perawatan secara real-time. Berikut adalah cara user menggunakan aplikasi dan penjelasan mengenai hasil yang diperoleh dari proses deployment ini.

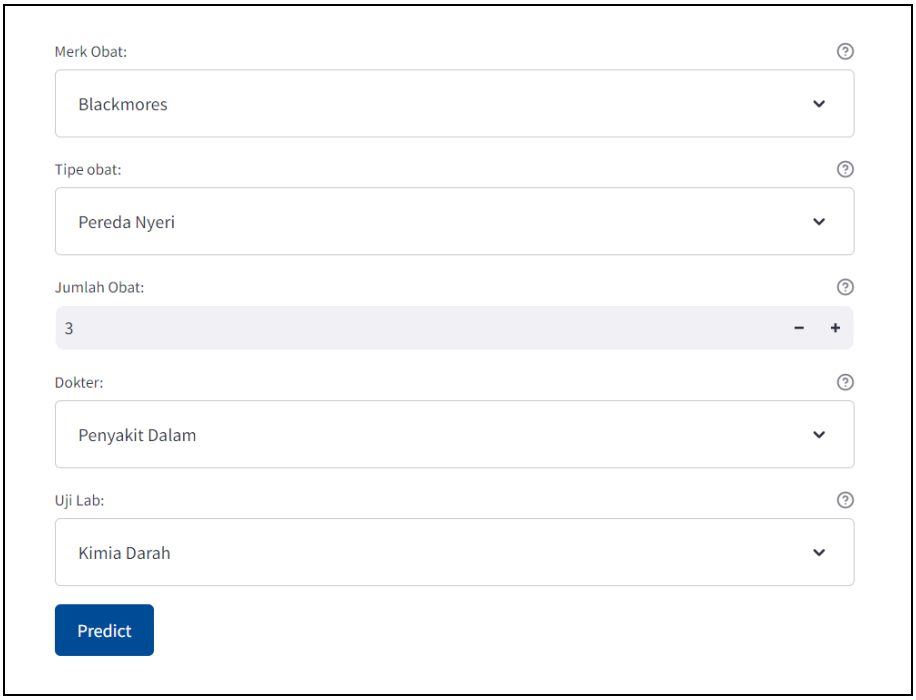
(i) Tampilan Utama dan Input Data

Pada halaman utama aplikasi, *user* dapat melihat form input yang interaktif. *user* diminta untuk memasukkan informasi seperti nama pasien, kelompok umur, jenis kelamin, cabang rumah sakit, metode pembayaran, dan detail perawatan lainnya. Tampilan web dapat dilihat pada Gambar 3.5.

Gambar 3.5 Tampilan Web App page input data untuk ML



Gambar 3.6 Tampilan Web App page input data untuk ML con't



Gambar 3.7 Tampilan Web App page input data untuk ML con't 2

Form input dirancang untuk mudah digunakan dengan *dropdown* menu dan *input fields* yang jelas. Ini memungkinkan user untuk memasukkan data dengan cepat dan akurat.

(ii) Proses Prediksi dan Hasil Output

Setelah semua data dimasukkan, *user* dapat menekan tombol "Predict" untuk mendapatkan estimasi biaya perawatan. Model XGBoost yang telah di-*deploy* akan memproses data input tersebut dan memberikan prediksi biaya yang akurat. Hasil prediksi ditampilkan secara jelas di halaman yang sama, memungkinkan *user* untuk melihat total biaya yang perlu dibayarkan.

Gambar 3.8 Hasil prediksi biaya dengan Machine Learning

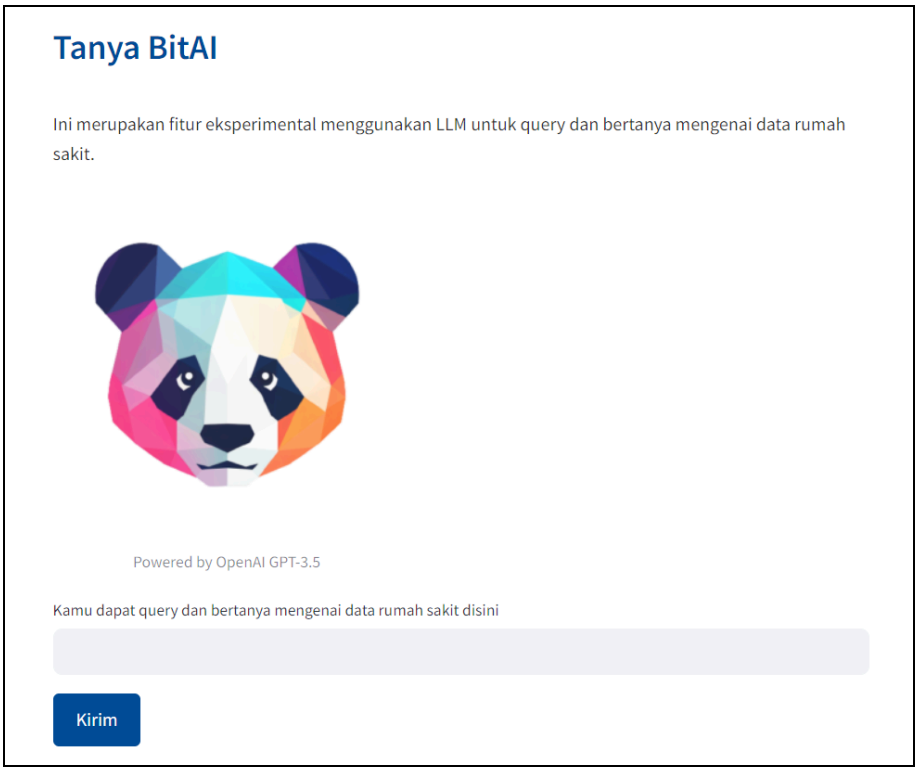
Output prediksi ditampilkan dalam format yang mudah dibaca dan memberikan informasi yang lengkap kepada *user* tentang biaya yang diestimasi berdasarkan data yang mereka masukkan. Misalnya, hasil prediksi dapat menunjukkan bahwa biaya perawatan pasien "**Rafi**" adalah "**Rp 5,857,033**".

3.3.2 Hasil Integrasi LLM untuk Query Data

Integrasi LLM (*Large Language Model*) dengan bantuan OpenAI GPT-3.5 memungkinkan *user* untuk melakukan *query* terhadap data transaksi pasien menggunakan *natural language* (bahasa non programming seperti bahasa inggris atau indonesia). Fitur ini sangat berguna untuk memperoleh informasi dan analisis yang lebih mendalam tanpa perlu menulis kode SQL yang kompleks.

(i) Tampilan Halaman Query Data

Halaman query data dirancang untuk menerima input dalam bentuk teks. *user* dapat mengetik pertanyaan mereka terkait data transaksi pasien dan menerima jawaban yang relevan dari sistem.

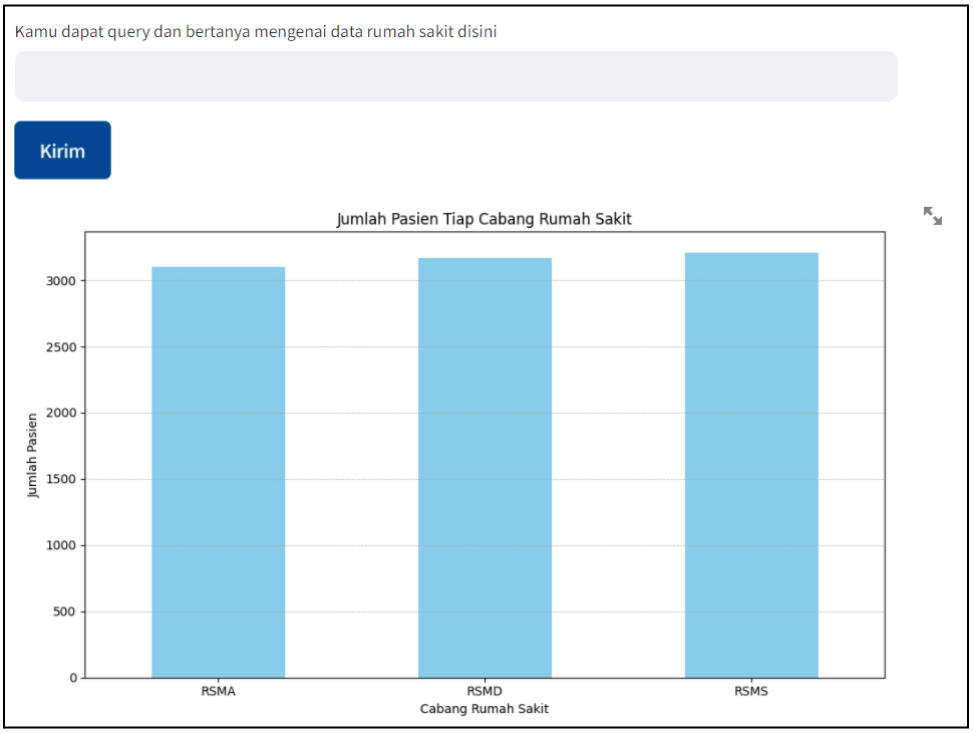


Gambar 3.9 Tampilan Fitur LLM BitAI

(ii) Proses Query dan Output Jawaban

Setelah *user* mengetik pertanyaan dan menekan tombol "Kirim", sistem akan memproses pertanyaan tersebut menggunakan LLM dan mengubahnya menjadi *query* yang dapat dijalankan oleh *library* pandasai. Hasil *query* kemudian ditampilkan di halaman yang sama.

Gambar 3.10 Tampilan Output LLM string



Gambar 3.11 Tampilan Output LLM Bar Chart

Jawaban yang diberikan oleh sistem ditampilkan dalam format yang jelas dan mudah dimengerti. Misalnya, *user* dapat bertanya "Berapa pasien rawat inap pada bulan Desember 2023?" dan sistem akan memberikan jawaban yang spesifik berdasarkan data yang tersedia. Atau user juga dapat meminta jawaban analisis data dalam bentuk grafik atau *chart* terkait.

3.3.3 Penambahan Halaman Beranda dan Bantuan

Selain fitur utama prediksi biaya dan query data, aplikasi ini juga mencakup halaman beranda dan bantuan untuk memudahkan user dalam memahami tujuan aplikasi dan cara menggunakannya.

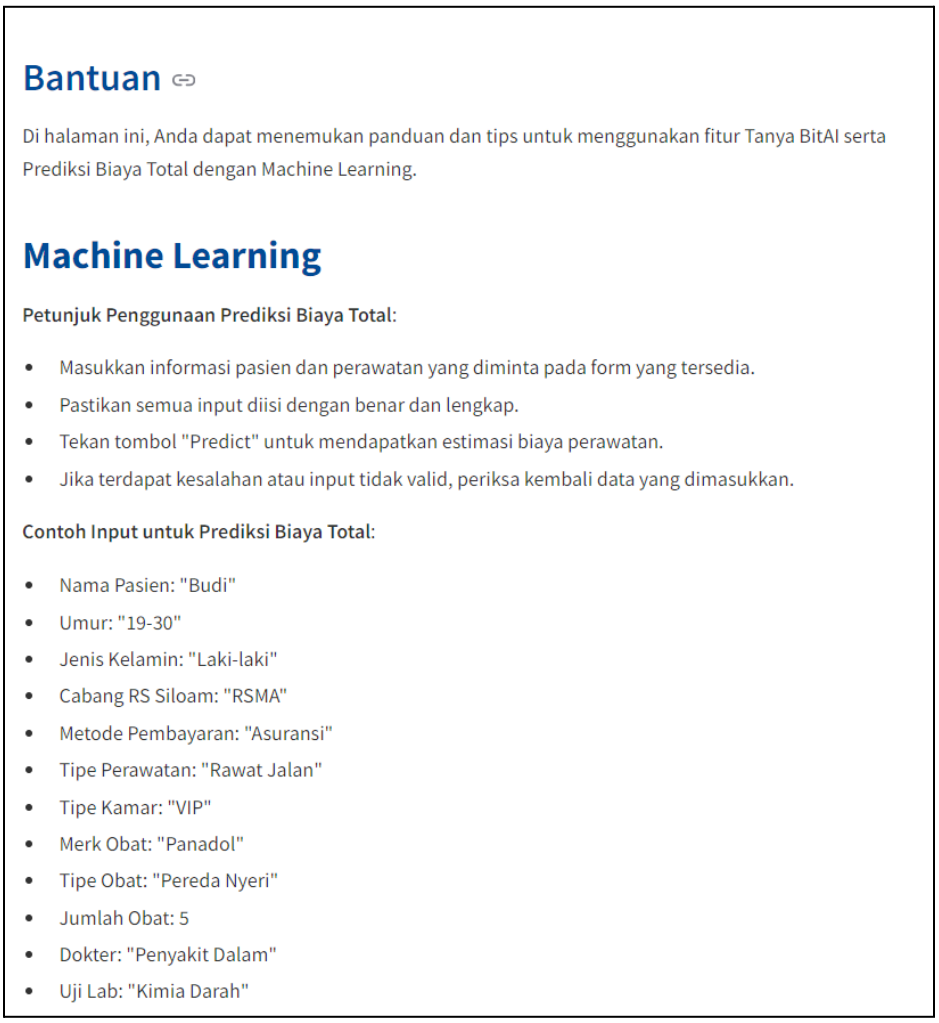
(i) Halaman Beranda

Halaman beranda memberikan informasi umum tentang *project* ini, termasuk tujuan dan tim yang terlibat. Ini membantu user untuk memahami konteks dan manfaat dari aplikasi ini.

Gambar 3.11 Tampilan Page Beranda

(ii) Halaman Bantuan

Halaman bantuan menyediakan panduan dan tips untuk menggunakan fitur-fitur yang ada dalam aplikasi. Ini termasuk instruksi langkah demi langkah untuk melakukan prediksi biaya dan query data, serta contoh input yang dapat digunakan.



Gambar 3.12 Tampilan Page Bantuan

Secara keseluruhan, deployment aplikasi ini memastikan bahwa model *machine learning* dan LLM dapat digunakan dengan mudah oleh user untuk mendapatkan prediksi biaya dan melakukan *query* data dengan efisien. Antarmuka yang user friendly dan fitur tambahan seperti halaman beranda dan bantuan meningkatkan pengalaman *user* secara keseluruhan.

BAB IV
KESIMPULAN

Berdasarkan hasil analisis dan pembahasan yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

1. Analisis ini menunjukkan bahwa pasien DBD dengan rawat inap memerlukan biaya lebih besar (rata-rata Rp2.763.202) dan memberikan lebih banyak *revenue* (*profit* Rp520.130) dibandingkan dengan pasien rawat jalan (rata-rata Rp627.776 dan kerugian Rp2.357.180), dengan mayoritas rawat inap ditangani oleh dokter umum dan rawat jalan oleh dokter penyakit dalam. Pasien rawat inap menggunakan asuransi (49%) dan pembayaran pribadi (51%), sedangkan seluruh pasien rawat jalan menggunakan pembayaran pribadi.
2. Penerapan model machine learning yang tepat, seperti XGBoost Regression, memungkinkan rumah sakit untuk mengoptimalkan prediksi biaya perawatan. Hal ini pada akhirnya dapat meningkatkan efisiensi dan kualitas layanan medis, serta membantu pasien dalam mengestimasi biaya yang akan dibayarkan.

REFERENSI

[1] A. Joel, “Data Science Life Cycle: Detailed Explanation [2023],” *www.odinschool.com*, Oct. 20, 2022. https://www.odinschool.com/blog/data-science-life-cycle-detailed-explanation-2023

[2] N. Gogia, “Why Scaling is Important in Machine Learning?,” *Analytics Vidhya*, Dec. 04, 2019. https://medium.com/analytics-vidhya/why-scaling-is-important-in-machine-learning-aee5781d161a

[3] K. Mali, “Linear Regression | Everything you need to Know about Linear Regression,” *Analytics Vidhya*, Oct. 04, 2021. https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/

[4] geeksforgeeks, “Pros and Cons of Decision Tree Regression in Machine Learning,” *GeeksforGeeks*, Feb. 15, 2024. https://www.geeksforgeeks.org/pros-and-cons-of-decision-tree-regression-in-machine-learning/

[5] A. Singh, “KNN algorithm: Introduction to K-Nearest Neighbors Algorithm for Regression,” *Analytics Vidhya*, Aug. 22, 2018. https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/#:~:text=As%20we%20saw%20above%2C%20the

[6] A. Sethi, “Support Vector Regression Tutorial for Machine Learning,” *Analytics Vidhya*, Mar. 27, 2020. https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/#:~:text=SVR%20offers%20greater%20flexibility%20and (accessed Jun. 07, 2024).

[7] E. R. Sruthi, “Random Forest | Introduction to Random Forest Algorithm,” *Analytics Vidhya*, Jun. 17, 2021. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[8] Analytics Vidhya, “Understanding the Math behind the XGBoost Algorithm,” *Analytics Vidhya*, Sep. 06, 2018. https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

[9] M. Padhma, “Evaluation Metric for Regression Models,” *Analytics Vidhya*, Oct. 28, 2021. https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/