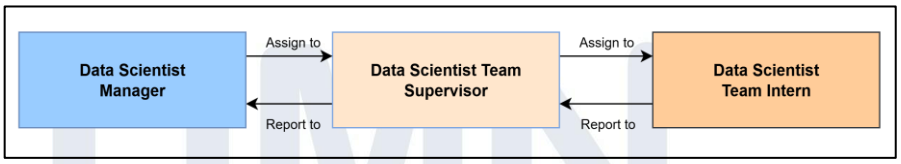
BAB III

PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

3.1.1 Kedudukan

Selama pelaksanaan kerja magang di PT Global Loyalty Indonesia, penulis menempati posisi sebagai *intern data scientist*. Tugas utama dalam peran ini adalah mengolah data mentah menjadi informasi yang bernilai guna mendukung pengambilan keputusan dan pertumbuhan perusahaan, serta menginterpretasikan hasil analisis data dari berbagai sumber untuk memberikan rekomendasi strategis. Alur koordinasi dalam tim data *scientist* ditampilkan pada Gambar 3.1. Proses penugasan dimulai dari *Data Scientist Manager*, Bapak Denny Desanleon Yuwono, kemudian diteruskan kepada supervisor, Bapak Ivan Nur Amanda, sebelum akhirnya diberikan kepada intern data scientist.

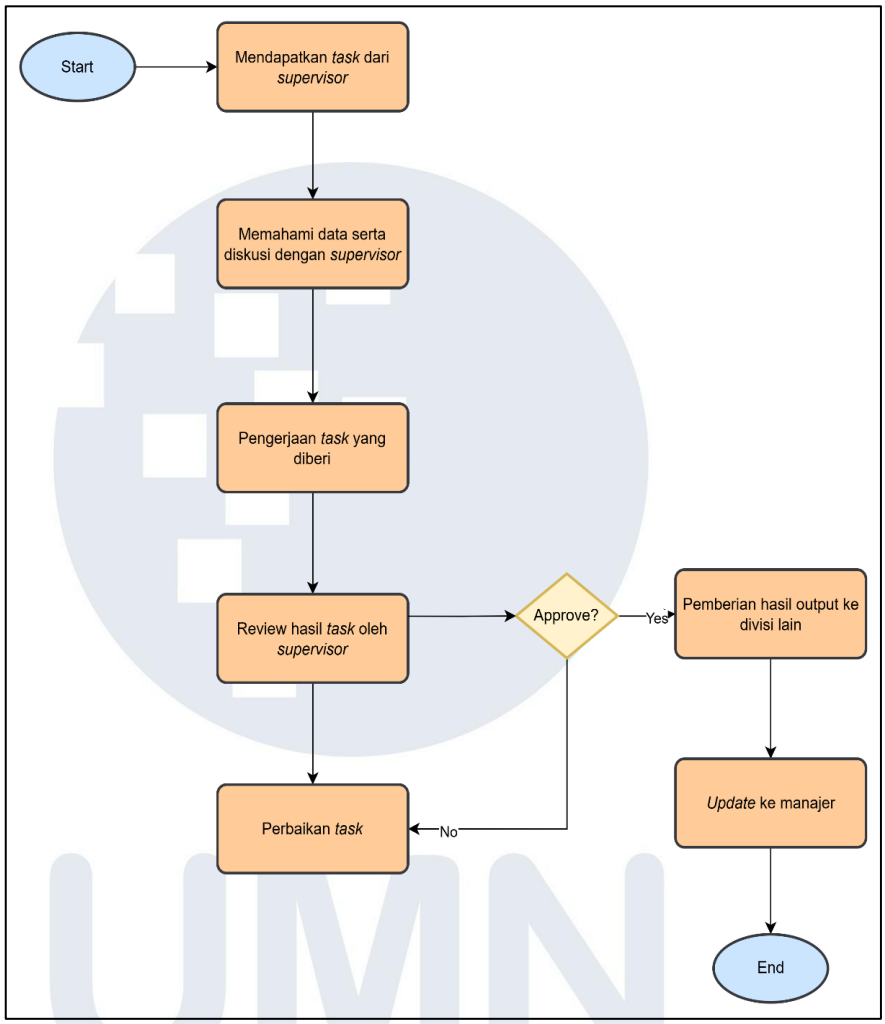


Gambar 3.1 Alur Koordinasi Tim *Data Science*

3.1.2 Koordinasi

Proses pelaporan pekerjaan dilakukan secara berjenjang dari intern *data scientist* kepada supervisor, yang kemudian diteruskan kepada *data scientist manager*. Koordinasi dan komunikasi dalam tim didukung oleh penggunaan platform daring seperti Trello. Platform ini dimanfaatkan untuk memperbarui perkembangan tugas yang sedang dikerjakan, yang selanjutnya akan dibahas dan dievaluasi bersama dalam pertemuan mingguan setiap hari Senin. Alur pelaksanaan kegiatan magang secara keseluruhan dapat dilihat pada

Gambar 3.2, yang menggambarkan secara rinci tahapan pengerjaan suatu tugas oleh *intern data scientist.*



Gambar 3.2 Alur Pelaksanaan Kerja Magang

Alur pelaksanaan kerja magang diawali dengan pemberian tugas oleh supervisor. Tugas yang diberikan dapat berupa data mentah maupun *query* yang relevan untuk mendukung proses pengerjaan. Selanjutnya, dilakukan pemahaman terhadap data yang tersedia, disertai diskusi dengan *supervisor* untuk memperjelas ruang lingkup tugas. Setelah itu, proses pengerjaan dimulai. Hasil pekerjaan kemudian disampaikan kepada supervisor untuk ditinjau. Apabila hasil kerja disetujui, *output* tersebut akan diteruskan kepada divisi

terkait yang membutuhkan, dan *supervisor* akan melaporkan perkembangan kepada manajer. Namun, apabila hasil belum memenuhi standar, intern data *scientist* diminta untuk melakukan perbaikan sebelum kembali melalui proses *review*.

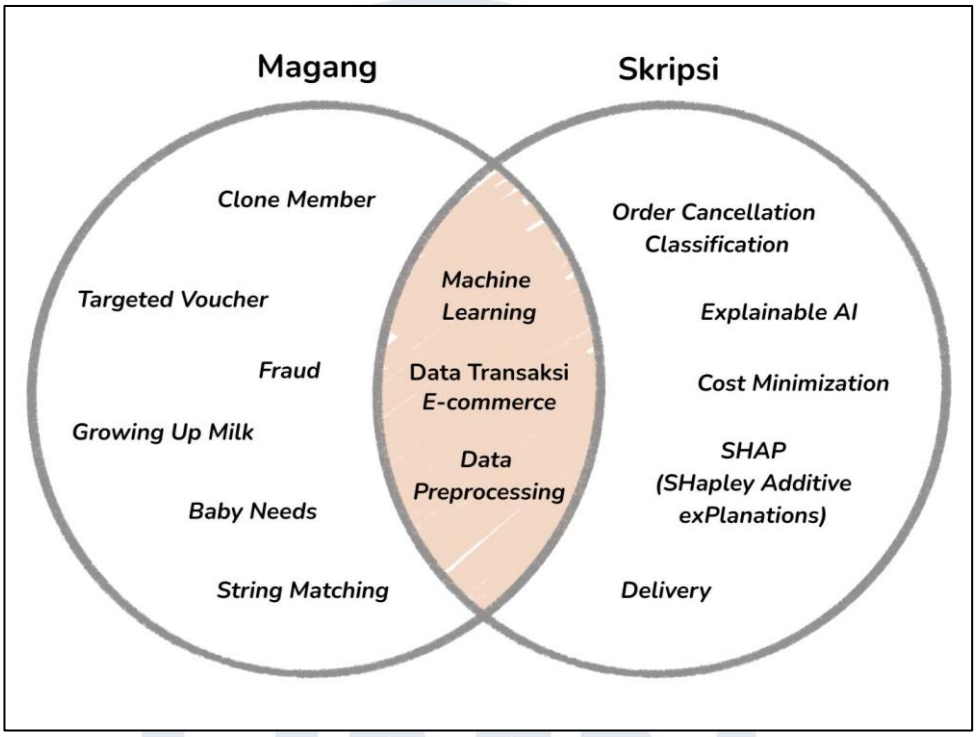
3.2 Tugas yang Dilakukan

Fokus pekerjaan selama masa magang adalah analisis *clone member*, yaitu akun ganda yang berpotensi disalahgunakan untuk memperoleh keuntungan berlebih dari program loyalitas. Permasalahan *clone member* menjadi tantangan penting dalam bisnis ritel digital karena dapat menyebabkan kerugian finansial akibat penyalahgunaan promo khusus, menurunkan efektivitas strategi pemasaran, serta mengganggu kualitas data pelanggan. Analisis dilakukan dengan memahami pola perilaku pengguna yang mencurigakan, mengidentifikasi indikasi akun ganda, serta menghasilkan temuan yang mendukung peningkatan validitas data pelanggan. Data yang lebih bersih dan akurat memungkinkan strategi pemasaran berjalan lebih tepat sasaran serta mendukung pengambilan keputusan bisnis yang efektif.

Selain itu, keterlibatan juga mencakup pengerjaan program *targeted voucher*, yaitu program distribusi *voucher* yang ditujukan secara personal berdasarkan aktivitas *member* di aplikasi Alfagift. Program ini bertujuan untuk meningkatkan loyalitas pelanggan dan mendorong pertumbuhan transaksi melalui pendekatan pemasaran yang relevan dengan kebutuhan masing-masing segmen. Tahapan pekerjaan meliputi eksplorasi dan pemahaman data, pembangunan model prediktif, pengolahan data hasil model, hingga integrasi *output* untuk menghasilkan daftar *member* sasaran yang diperbarui setiap bulan. Program *targeted voucher* difokuskan pada produk tagI (produk eksklusif Alfagift) yang mencakup kategori susu pertumbuhan dan *baby needs*.

Pengerjaan tidak berhenti pada distribusi *voucher*, tetapi juga mencakup analisis lanjutan terhadap efektivitas program. Evaluasi dilakukan

dengan mengukur tingkat penukaran *voucher* (*redemption rate*), menganalisis perubahan pola belanja pelanggan setelah menerima *voucher*, serta menilai dampaknya terhadap penjualan produk yang ditargetkan. Hasil evaluasi dijadikan dasar perbaikan strategi distribusi pada periode berikutnya agar efektivitas program terus meningkat.



Gambar 3.3 Diagram Venn Perbedaan Aspek Magang dan Skripsi

Gambar 3.3 memperlihatkan hubungan antara kegiatan magang dan penelitian skripsi yang memiliki keterkaitan dalam ruang lingkup akademis maupun praktis. Pada sisi kiri, kegiatan magang berfokus pada analisis *clone member, targeted voucher*, dan pencegahan *fraud* dalam konteks bisnis ritel digital. Sementara itu, sisi kanan menggambarkan fokus penelitian skripsi yang menekankan pengembangan model *machine learning* untuk order *cancellation classification* dengan pendekatan Explainable AI (SHAP) dan tujuan *cost minimization*. Bagian tengah menunjukkan area irisan yang mencerminkan kesamaan keduanya, yaitu penggunaan data transaksi *e-commerce*, proses *data preprocessing*, serta penerapan *machine learning*.

3.3 Uraian Pelaksanaan Kerja

Tabel 3.1 Rincian kegiatan serta waktu pelaksanaan selama masa magang di PT Global Loyalty Indonesia dapat dilihat pada Tabel 3.1. Kegiatan diawali dengan pemahaman terhadap proyek *clone member*, pengenalan sistem kerja tim data, serta proses penyiapan lingkungan analisis seperti Jupyter Notebook dan SQL Server yang digunakan dalam aktivitas harian. Tahap berikutnya difokuskan pada proses *monitoring* hasil program *targeted voucher* periode Mei 2025 dan dilanjutkan dengan *data preparation* serta pembangunan model *machine learning* untuk memprediksi penerima *voucher* periode Juli 2025. Setelah model terbentuk, dilakukan pengolahan *output* serta analisis *best time to buy* untuk menentukan waktu pembelian paling optimal bagi pelanggan. Selanjutnya, kegiatan berfokus pada eksplorasi data *clone member* dan implementasi berbagai metode *similarity detection* menggunakan *library* seperti FuzzyWuzzy, serta pendekatan berbasis lokasi (*longitude-latitude*) dan *secure device* ID. Setelah seluruh pendekatan diuji, dilakukan proses data *cleaning* untuk memastikan hasil deteksi memiliki kualitas yang baik dan siap digunakan. Pada tahap berikutnya, dilakukan pemodelan serta pengolahan *output* untuk mendukung program *targeted voucher* periode September dan Oktober 2025. Kegiatan ini juga mencakup proses penggabungan data *member* sasaran, analisis hasil prediksi, serta pemantauan tingkat *redeem voucher* guna mengevaluasi efektivitas strategi promosi dan memahami perilaku pelanggan dalam program loyalitas Alfagift.

Tabel 3.1 Tugas dan Uraian Kerja Magang

No.	Pekerjaan	Waktu Pengerjaan
1	Pengenalan dan pemahaman proyek *clone member.*	2 Juni – 5 Juni (Minggu 1)
2	*Monitoring targeted voucher redeemed* Mei 2025.	9 Juni – 13 Juni (Minggu 2)
3	Data *preparation* dan modeling untuk *voucher targeted* Juli 2025.	9 Juni – 20 Februari (Minggu 2-3)

No.	Pekerjaan	Waktu Pengerjaan
4	Pengolahan *output* model dan *best time to buy* untuk *voucher targeted* Juli 2025.	16 Februari – 26 Maret (Minggu 3-4)
5	Eksplorasi data *clone member* dan *library* untuk *similarity detection.*	30 Juni – 11 Juli (Minggu 5-6)
6	*Similarity detection* dengan FuzzyWuzz.	14 Juli – 18 Juli (Minggu 7)
7	*Similarity detection* dengan *longitude* dan *latitude*.	21 Juli – 31 Juli (Minggu 8)
8	*Similarity detection* dengan *secure device id*.	1 Agustus – 8 Agustus (Minggu 9)
9	Data *cleaning* hasil *similarity detection*	11 Agustus – 15 Agustus (Minggu 10)
10	Modeling dan pengolahan *output* model dan penggabungan *list member* untuk *voucher targeted* September 2025.	18 Agustus – 29 Agustus (Minggu 10-12)
11	Monitoring hasil program *voucher targeted* September 2025	1 September – 12 September (Minggu 13-14)
12	Modeling dan pengolahan *output* model dan penggabungan *list member* untuk *voucher targeted* Oktober 2025.	15 September – 30 September (Minggu 15-17)

3.3.1 Proses Pelaksanaan

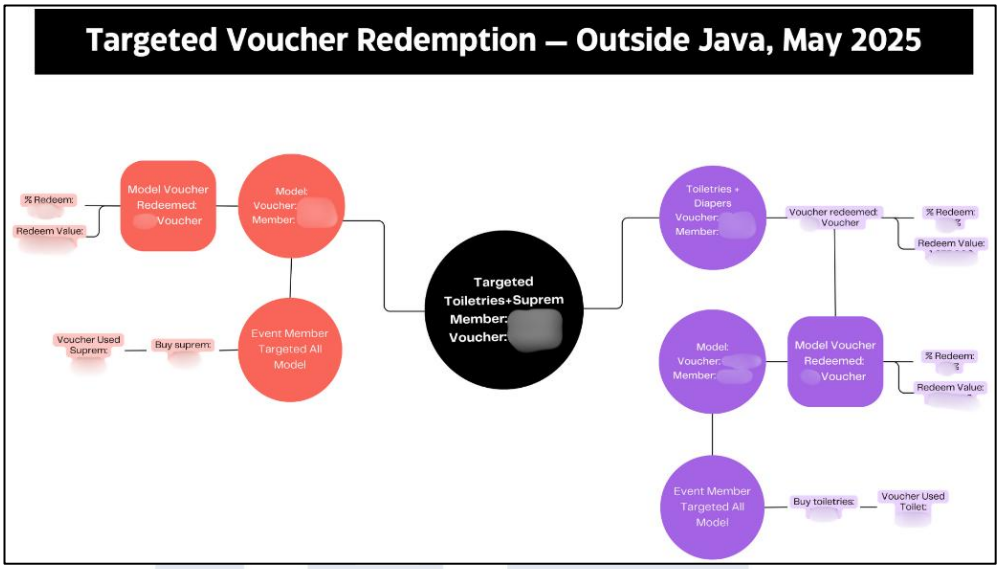
3.3.1.1 Pengenalan dan Pemahaman Proyek *Clone Member*

Tahap awal pengerjaan difokuskan pada pemahaman konsep proyek *Clone Member* beserta ruang lingkup pekerjaannya. Proyek ini diperkenalkan melalui penjelasan alur bisnis yang mendasari kebutuhan analisis *clone member* serta peran pentingnya bagi strategi perusahaan. Penjelasan mencakup definisi *clone member*, dampak keberadaannya terhadap kualitas data pelanggan, serta tujuan yang ingin dicapai melalui proyek ini, seperti peningkatan keakuratan data, identifikasi pola duplikasi, hingga mendukung efektivitas program pemasaran yang ditargetkan. Pemahaman proyek didukung dengan penjelasan mengenai data yang digunakan, sumber data yang relevan, serta kendala yang sering muncul seperti inkonsistensi atau

redundansi data. Akses terhadap *dataset* diberikan melalui Google BigQuery, dan langkah awal dilakukan dengan eksplorasi sederhana untuk memahami struktur tabel, atribut, serta distribusi data. Selanjutnya dilakukan diskusi bersama mentor dan tim *data scientist* mengenai metodologi yang sesuai untuk mendeteksi *clone member* tersebut, seperti eksplorasi pola menggunakan atribut atau fitur tertentu, hingga *library* yang dapat membantu deteksi di tahap lanjutan.

3.3.1.2 Monitoring *targeted voucher redeemed* Mei 2025

Monitoring pada tahap ini difokuskan pada program *voucher targeted* kategori *baby needs* dan susu pertumbuhan bulan Mei 2025. *Voucher* yang di *redeem* selama periode Mei 2025 memperlihatkan sejauh mana program *targeted voucher* berjalan sesuai dengan tujuan yang telah ditetapkan. Proses *monitoring* dilakukan secara menyeluruh dengan memanfaatkan data transaksi yang tercatat pada sistem, sehingga dapat diperoleh gambaran mengenai partisipasi *member*, efektivitas distribusi *voucher*, serta perbandingan antara jumlah *voucher* yang dibagikan dengan jumlah *voucher* yang benar-benar digunakan. Analisis dilakukan melalui eksplorasi data di Jupyter menggunakan *library* pandas yang kemudian divisualisasikan menggunakan *website* Canva agar lebih mudah dipahami serta dapat dijadikan bahan evaluasi oleh manajer maupun divisi terkait. Hasil visualisasi monitor dapat dilihat pada gambar 3.4.



Gambar 3.4 Hasil monitor *targeted voucher redemption* Mei 2025

Monitoring mencakup distribusi *voucher* pada berbagai segmen *member*, *redemption rate* masing-masing kategori, serta *redeem value*. Hasil *monitoring* memberikan masukan mengenai tingkat *engagement* member terhadap program promosi yang dijalankan serta kontribusinya terhadap penjualan secara keseluruhan. Melalui tahapan *monitoring* ini, diperoleh pemahaman lebih mendalam mengenai dampak penerapan *targeted voucher*, baik dari sisi bisnis maupun perilaku konsumen. Visualisasi dalam bentuk *chart* digunakan juga untuk dipresentasikan saat *weekly meeting* untuk memberi *update* pada manajer dan divisi *data science*.

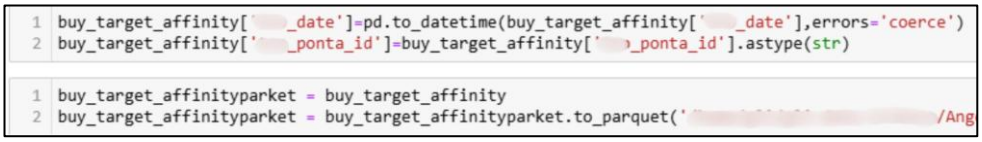
3.3.1.3 Data *preparation* dan modeling untuk *voucher targeted* Juli 2025

Setelah menyelesaikan *monitoring* terhadap program *targeted voucher* periode Mei 2025, dilakukan persiapan untuk *pelaksanaan* program *targeted voucher* berikutnya untuk periode Juli 2025. *Output* utama yang diperlukan untuk program tersebut adalah *list* nomor *member* yang nantinya akan diberi *voucher.* Program ini dibagi menjadi 2 kategori produk utama, yang pertama adalah susu pertumbuhan, dan yang kedua adalah *baby needs*. Tahapan awal untuk memperoleh *output member targeted* kategori susu pertumbuhan dimulai dengan proses *data preparation*, yaitu dengan

melakukan penarikan data yang dibutuhkan untuk pembangunan model *machine learning*. Data tersebut bersumber dari berbagai tabel yang tersimpan di Google BigQuery, kemudian ditarik dan diolah di Jupyter Notebook untuk proses persiapan dan pemodelan lebih lanjut.

Gambar 3.5 Menarik data *buy target affinity* Mei 2025

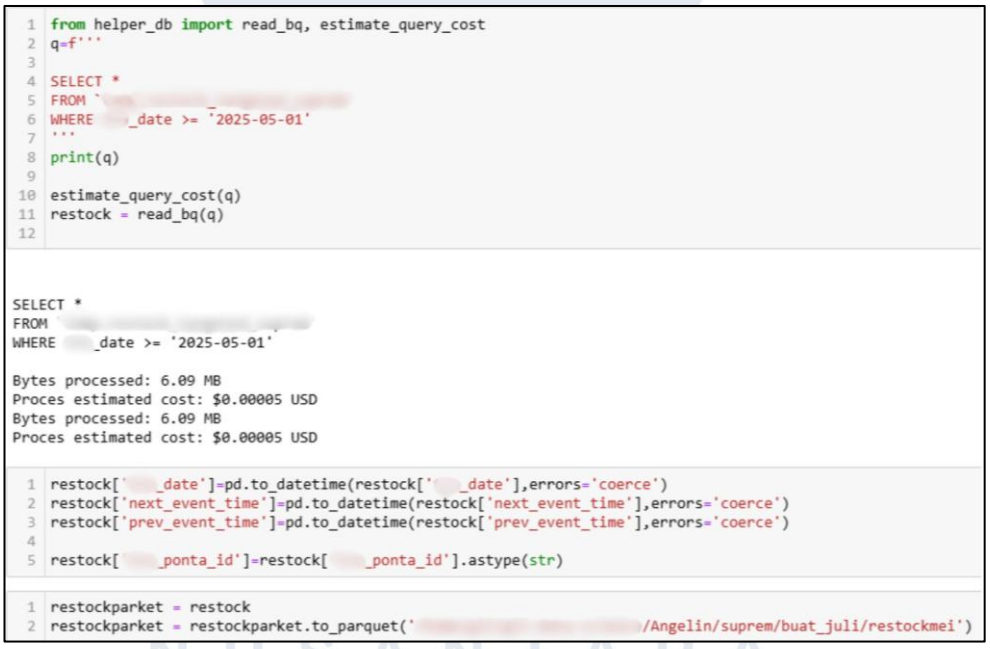
Gambar 3.5 menunjukkan penarikan data menggunakan *library* Python. Data pertama yang ditarik adalah tabel 'buy_target_affinity' bulan Mei 2025. Tabel 'buy_target_affinity' adalah tabel yang berisikan data pembelian produk *target* atau dalam hal ini adalah susu pertumbuhan, dan produk *affinity* atau produk-produk yang berkaitan dengan produk tertentu atau dalam kasus ini juga susu pertumbuhan. Data yang ditarik kemudian akan dimasukkan ke dalam *dataframe* agar selanjutnya dapat dengan mudah diolah.



```
buy_target_affinity[ _date']=pd.to_datetime(buy_target_affinity[ _date'],errors='coerce')
buy_target_affinity[ _ponta_id']=buy_target_affinity[ _ponta_id'].astype(str)

buy_target_affinityparket = buy_target_affinity
buy_target_affinityparket = buy_target_affinityparket.to_parquet(' /Ang
```

Gambar 3.6 Menyimpan *dataframe* 'buy_target_affinity'

Gambar 3.6 memperlihatkan proses penyimpanan *dataframe* 'buy_target_affinity'. Pada tahap ini, tipe data tanggal dikonversi ke format *datetime* agar lebih mudah digunakan untuk analisis berbasis waktu, sementara nomor *member* diubah menjadi *string* agar diperlakukan sebagai identitas unik dan tidak terbaca sebagai nilai numerik. Setelah itu, *dataframe* disimpan dalam format Parquet yang dipilih karena lebih efisien dalam hal ukuran penyimpanan serta lebih cepat dalam proses pembacaan data, sehingga memudahkan penggunaan data pada tahap analisis maupun pemodelan selanjutnya.



```
from helper_db import read_bq, estimate_query_cost
q=f'''

SELECT *
FROM `
WHERE  _date >= '2025-05-01'
'''
print(q)

estimate_query_cost(q)
restock = read_bq(q)


SELECT *
FROM
WHERE  _date >= '2025-05-01'

Bytes processed: 6.09 MB
Proces estimated cost: $0.00005 USD
Bytes processed: 6.09 MB
Proces estimated cost: $0.00005 USD

restock[ _date']=pd.to_datetime(restock[ _date'],errors='coerce')
restock['next_event_time']=pd.to_datetime(restock['next_event_time'],errors='coerce')
restock['prev_event_time']=pd.to_datetime(restock['prev_event_time'],errors='coerce')

restock[ _ponta_id']=restock[ _ponta_id'].astype(str)

restockparket = restock
restockparket = restockparket.to_parquet(' /Angelin/suprem/buat_juli/restockmei')
```

Gambar 3.7 Menarik data *restock* Mei 2025

Gambar 3.7 memperlihatkan proses penarikan data *restock* untuk periode Mei 2025. Data *restock* merupakan data transaksi *member* yang memperhitungkan perkiraan waktu produk habis hingga *member* melakukan

pembelian ulang, sehingga dapat menggambarkan apakah seorang *member* melakukan *restock* atau tidak. Sama seperti sebelumnya, data diambil melalui *query* SQL yang dijalankan di Google BigQuery, kemudian dipanggil menggunakan fungsi read_bq pada Python. Sebelum data diproses lebih lanjut, sistem terlebih dahulu menampilkan estimasi ukuran data yang diproses beserta estimasi biaya eksekusi *query*. Setelah data berhasil ditarik, dilakukan tahap data *preprocessing* berupa konversi tipe data pada beberapa kolom, misalnya kolom tanggal 'date', 'next_event_time', dan 'prev_event_time') yang diubah menjadi format *datetime*, serta kolom ID member yang diubah menjadi tipe *string* agar terbaca sebagai identitas unik. Data yang telah dibersihkan ini kemudian disimpan dalam format Parquet agar lebih efisien dan mudah digunakan pada analisis serta pemodelan selanjutnya.

Gambar 3.8 Menarik data *event* Mei 2025

Gambar 3.8 memperlihatkan proses penarikan data *event* untuk periode Mei 2025. Sama seperti proses pada data sebelumnya, dilakukan *query* di Google BigQuery dan pemanggilan menggunakan fungsi read_bq di Python, kemudian dilanjutkan dengan *data preprocessing* berupa konversi tipe data tanggal dan ID *member*. Berbeda dengan data sebelumnya, data

event ini berfokus pada aktivitas member terhadap produk target, yang terbagi menjadi dua jenis, yaitu ‘add_to_cart’ (*member* memasukkan produk target ke keranjang belanja) dan ‘view_product’ (*member* melihat serta membaca deskripsi produk target). Hasil akhir kemudian disimpan dalam format Parquet juga.

```
from helper_db import read_bq, estimate_query_cost
q=f'''

SELECT *
FROM `[redacted]`
WHERE tgl >= '2025-05-01'
'''
print(q)

estimate_query_cost(q)
trans = read_bq(q)
```

```
SELECT *
FROM `[redacted]`
WHERE tgl >= '2025-05-01'

Bytes processed: 5.98 GB
Proces estimated cost: $0.05044 USD
Bytes processed: 5.98 GB
Proces estimated cost: $0.05044 USD
```

```
trans['tgl'] = pd.to_datetime(trans['tgl'], errors='coerce')
trans['no_member']=trans['no_member'].astype(str)
```

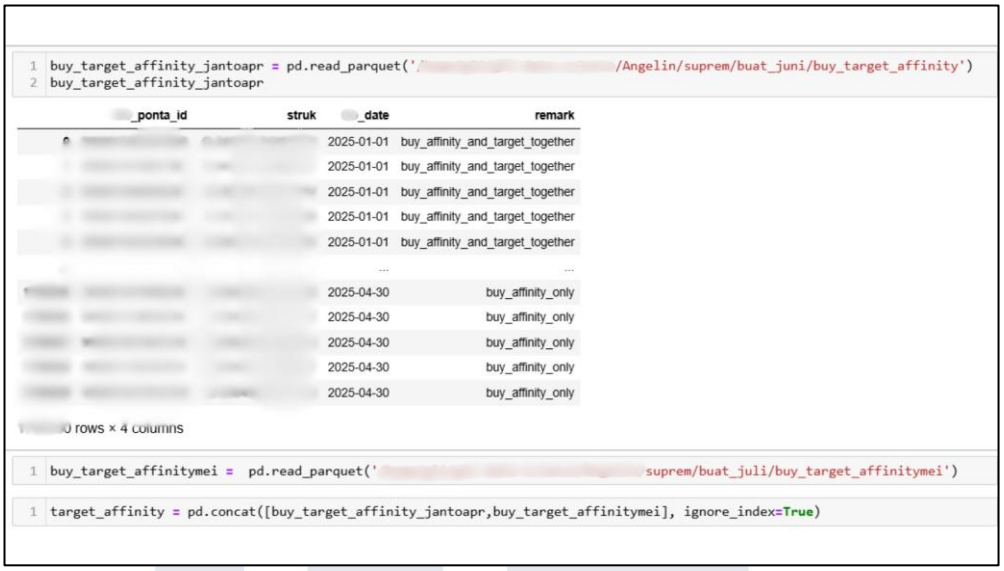
```
trans['tgl'].max()
```

```
Timestamp('2025-05-31 00:00:00')
```

```
transparket = trans
transparket = transparket.to_parquet('[redacted]/Angelin/suprem/buat_juni/transluarja
```

Gambar 3.9 Menarik data transaksi Mei 2025

Gambar 3.9 menunjukkan proses penarikan data transaksi bulan Mei 2025. Berbeda dengan tabel sebelumnya yang bersifat pendukung, data transaksi ini dijadikan tabel utama karena memuat keseluruhan aktivitas pembelian *member*. Informasi di dalamnya kemudian diproses dan disimpan dalam format Parquet agar siap digunakan pada tahap analisis dan pemodelan.



Gambar 3.10 Penggabungan data *buy target affinity* Januari hingga Mei 2025

Karena dalam pembangunan model *machine learning* diperlukan data historis yang cukup banyak agar pola perilaku *member* dapat dipelajari dengan lebih baik, maka data hasil penarikan dari bulan Mei 2025 tidak digunakan sendiri. Seperti yang dapat dilihat pada gambar 3.10, data tersebut digabungkan (*concat*) dengan data yang sama dari bulan Januari hingga April 2025, sehingga terbentuk satu *dataset* yang lebih komprehensif. Dengan cara ini, model dapat menangkap tren pembelian dan perilaku *affinity* secara lebih konsisten dari waktu ke waktu, sekaligus mengurangi risiko bias apabila hanya menggunakan data dari satu periode tertentu.

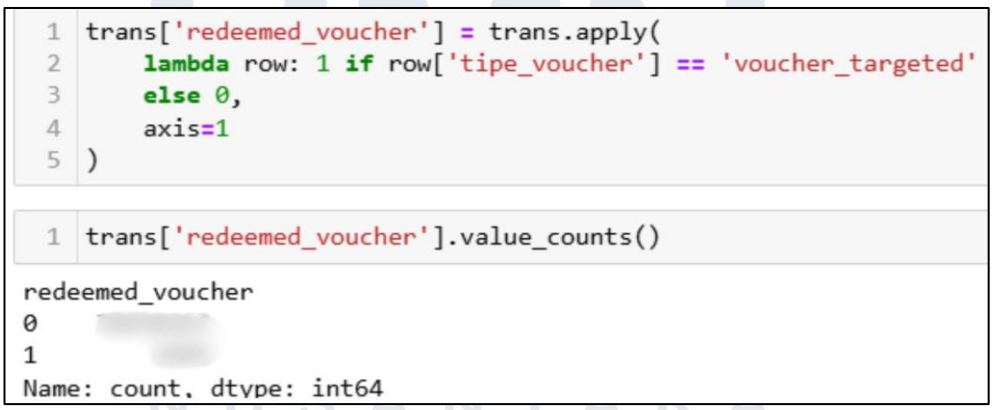
Gambar 3.11 Penggabungan data *restock* Januari hingga Mei 2025

Gambar 3.11 memperlihatkan proses penggabungan data *restock* periode Januari hingga Mei 2025. Data dari Januari-April yang telah disimpan dalam format Parquet dipanggil kembali bersama data bulan Mei, kemudian digabungkan menggunakan fungsi *concat* pada Python dengan opsi

ignore_index=True agar indeks baris tersusun ulang secara berurutan. Hasil penggabungan ini menghasilkan satu *dataframe* yang berisi catatan transaksi *restock*. Dengan data historis yang lebih panjang, analisis dan pembangunan model dapat dilakukan dengan lebih akurat.

Gambar 3.12 Penggabungan data trans Januari hingga Mei 2025

Gambar 3.12 menunjukkan penggabungan data transaksi periode Januari hingga Mei 2025. Proses ini dilakukan dengan cara memanggil data Januari-April serta data bulan Mei yang disimpan dalam format Parquet, kemudian keduanya digabungkan menggunakan fungsi pd.concat dengan parameter ignore_index=True agar indeks tersusun ulang dari awal. Penggabungan ini menghasilkan *dataset* transaksi yang lebih lengkap, sehingga pola perilaku belanja konsumen dapat dianalisis dengan cakupan historis yang lebih luas.



Gambar 3.13 Pembuatan kolom target untuk *targeted* Juli 2025

Gambar 3.13 memperlihatkan pembuatan kolom target ‘redeemed_voucher’. Kolom ini dibentuk dengan logika sederhana, yaitu memberi nilai 1 apabila tipe *voucher* yang digunakan adalah *voucher_targeted* dan 0 untuk lainnya. Hasilnya digunakan sebagai label

target dalam proses pemodelan, sehingga *model* dapat mempelajari perbedaan perilaku antara transaksi yang menukarkan *voucher targeted* dengan yang tidak.

```
#Total transaksi perember
trans['jumlah_transaksi'] = trans.groupby('no_member')['struk'].transform('nunique')

#Total transaksi suprem
growing_up = trans[trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM']

jumlah_transaksi_suprem = growing_up.groupby('no_member')['struk'].nunique().reset_index()
jumlah_transaksi_suprem.rename(columns={'struk': 'jumlah_transaksi_suprem'}, inplace=True)

trans = trans.merge(jumlah_transaksi_suprem, on='no_member', how='left')

trans['jumlah_transaksi_suprem'] = trans['jumlah_transaksi_suprem'].fillna(0).astype(int)

#Total sales member
trans['total_sales'] = trans.groupby('no_member')['sales'].transform('sum')

#Basket size semua transaksi
trans['basket_size'] = trans['total_sales'] / trans['jumlah_transaksi']

#total produk yang di purchased
trans['qty_purchased'] = trans.groupby('no_member')['qty'].transform('sum')
```

Gambar 3.14 *Feature engineer* untuk *targeted* Juli 2025

Gambar 3.14 memperlihatkan proses *feature engineering* Pada tahap ini, dibentuk sejumlah variabel turunan dari data transaksi, seperti jumlah transaksi per *member*, jumlah transaksi khusus kategori susu pertumbuhan atau *target*, serta total *sales* per *member*. Selain itu, dihitung pula *basket size*, yaitu nilai rata-rata pembelian per transaksi, dan total *quantity* produk yang dibeli. Fitur-fitur ini penting untuk merepresentasikan perilaku belanja konsumen secara lebih mendalam sehingga model dapat mengenali pola yang relevan dalam menilai kemungkinan *redeem voucher*.

```
#Pernah membeli produk growing up milk sebelumnya atau tidak
produk_kategori = ['GROWING UP MILK MAINSTREAM', 'GROWING UP MILK PREMIUM', 'GROWING UP MILK SUPER PREMIUM']

trans['purchased_gum'] = trans['descp_kat'].isin(produk_kategori)
trans['purchased_gum'] = trans['purchased_gum'].astype(int)
trans['purchased_gum'] = trans.groupby('no_member')['purchased_gum'].transform('max')

#Jumlah suprem yang dibeli
trans['qty_suprem'] = 0
mask_suprem = trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM'
trans.loc[mask_suprem, 'qty_suprem'] = trans.loc[mask_suprem, 'qty']
trans['qty_suprem'] = trans.groupby('no_member')['qty_suprem'].transform('sum')

# jumlah penggunaan voucher_others per member
voucher_counts = (
    trans[(trans['tipe_voucher'] == 'voucher_others') & (trans['nominal_voucher'] > 0)
          & (trans['retur']=='N')
         ]
    .groupby('no_member')
    .size()
)
trans['jumlah_voucher_others'] = trans['no_member'].map(voucher_counts).fillna(0).astype(int)
```

Gambar 3.15 *Feature engineer* untuk *targeted* Juli 2025

Gambar 3.15 memperlihatkan pembuatan fitur tambahan lainnya yang digunakan dalam pemodelan. Bedanya, pada bagian ini fitur yang dihasilkan lebih menekankan pada riwayat perilaku belanja *member*, seperti pembelian produk susu pertumbuhan dengan berbagai kategori (*mainstream*, *premium*, maupun *super premium*), total pembelian khusus untuk kategori *super premium*, serta pemanfaatan *voucher_others*. Fitur-fitur historis ini penting karena dapat merepresentasikan kecenderungan pola belanja maupun sensitivitas *member* terhadap promosi, sehingga model nantinya memiliki informasi yang lebih banyak untuk mengenali karakteristik tiap *member*.

```python
# Latency
def hitung_rata_rata_latency_per_tanggal(grup):

    tanggal_unik = grup['tgl'].drop_duplicates().sort_values()

    selisih_hari = tanggal_unik.diff().dt.days.dropna()

    # Hitung rata-rata latency dan bulatkan
    if len(selisih_hari) > 0:
        rata_rata = round(selisih_hari.mean())
    else:
        rata_rata = 0

    # Kembalikan nilai yang sama untuk semua baris dalam grup
    return pd.Series([rata_rata] * len(grup), index=grup.index)


trans['latency_suprem'] = (
    trans[(trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM') &
         (trans['retur']=='N')]
    .groupby('no_member', group_keys=False)
    .apply(hitung_rata_rata_latency_per_tanggal)
)
```

Gambar 3.16 *Feature engineer* untuk *targeted* Juli 2025

Gambar 16 menghitung rata-rata *latency* pembelian produk kategori *target* per *member*. Fungsi hitung_rata_rata_latency_per_tanggal pertama-tama mengambil daftar tanggal transaksi unik, kemudian menghitung selisih hari antar transaksi (interval pembelian). Dari selisih ini dihitung nilai rata-rata *latency*, yaitu rata-rata jarak waktu antar pembelian, yang kemudian dibulatkan agar lebih mudah dimengerti. Jika tidak ada selisih hari (misalnya hanya ada satu transaksi), maka nilai *latency* diberikan nol. Selanjutnya, fungsi ini diaplikasikan pada data yang telah difilter, yaitu hanya transaksi produk *target* dengan status bukan retur (retur = 'N'). Data kemudian dikelompokkan berdasarkan nomor *member* untuk memperoleh nilai rata-rata *latency* masing-masing, yang disimpan ke dalam kolom baru bernama latency_suprem. Dengan fitur ini, model dapat memahami seberapa sering *member* kembali membeli produk tersebut, sehingga bisa menjadi indikator penting untuk mengenali kebiasaan belanja dan potensi loyalitas pelanggan.

```
#  PLU suprem terakhir
suprem = trans[(trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM')&
               (trans['retur']=='N')]

last_troplu_suprem = (
    suprem.sort_values('tgl', ascending=False)
    .groupby('no_member')['  _PLU']
    .first()
    .astype(str)

)
trans['  _plu_suprem'] = trans['no_member'].map(last_t  plu_suprem)

#Tgl beli suprem terakhir
filtered = trans[trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM']

last_purchase = filtered.groupby('no_member')['tgl'].max().reset_index()
last_purchase.rename(columns={'tgl': 'Recent'}, inplace=True)

trans = trans.merge(last_purchase, on='no_member', how='left')
```

Gambar 3.17 *Feature engineer* untuk *targeted* Juli 2025

Gambar 3.17 menambahkan fitur-fitur yang mengidentifikasi produk terakhir yang dibeli oleh masing-masing *member* khusus pada kategori produk target. Pertama, kode mengekstraksi PLU terakhir berdasarkan tanggal transaksi yang paling baru, kemudian menyimpannya sebagai variabel plu_suprem. Selanjutnya, dicari juga tanggal pembelian terakhir produk tersebut (*Recent*) dengan melakukan agregasi maksimum pada kolom tgl. Informasi ini penting untuk melihat produk terakhir yang dipilih konsumen sekaligus kapan terakhir kali mereka membeli, sehingga bisa dipakai dalam analisis *targeted voucher* periode Juli 2025.

```
# Recency semua transaksi (hari sejak transaksi terakhir semua produk)
last_all = trans.groupby('no_member')['tgl'].max()
trans['recency_all'] = trans['no_member'].map(last_all)
trans['recency_all'] = (trans['tgl'].max() - trans['recency_all']).dt.days

# Recency pembelian GUM Premium & Mainstream (bukan Suprem)
gum_pm = trans[trans['descp_kat'].isin(['GROWING UP MILK MAINSTREAM', 'GROWING UP MILK PREMIUM'])]
last_gum_pm = gum_pm.groupby('no_member')['tgl'].max()
trans['recency_gum_pm'] = trans['no_member'].map(last_gum_pm)
trans['recency_gum_pm'] = (trans['tgl'].max() - trans['recency_gum_pm']).dt.days
trans['recency_gum_pm'] = trans['recency_gum_pm'].fillna(trans['tgl'].max().day)

# Recency suprem (dalam hari sejak transaksi terakhir suprem)
trans['recency_suprem'] = (trans['tgl'].max() - trans['Recent']).dt.days
# Isi dengan 0 jika belum pernah beli suprem
trans['recency_suprem'] = trans['recency_suprem'].fillna(0)
```

Gambar 3.18 *Feature engineer* tambahan untuk *targeted* Juli 2025

Gambar 3.18 menampilkan *feature engineering* yang menambahkan variabel *recency* untuk mengukur jarak hari sejak transaksi terakhir. Pertama, dihitung *recency* untuk semua produk (recency_all), lalu khusus untuk kategori *Growing Up Milk Premium & Mainstream* (recency_gum_pm), dan terakhir untuk produk *Super Premium* (recency_suprem). Dengan adanya fitur ini, model dapat memahami seberapa baru aktivitas pembelian member, sehingga membantu memprediksi kecenderungan mereka dalam *redeem voucher targeted*.

```
# Persentase pembelian GUM Suprem dibanding semua transaksi
jumlah_suprem_trans = trans[trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM'].groupby('no_member')['struk'].nunique()
jumlah_total_trans = trans.groupby('no_member')['struk'].nunique()
persen_suprem = (jumlah_suprem_trans / jumlah_total_trans).fillna(0)
trans['persen_trans_suprem'] = trans['no_member'].map(persen_suprem)

# Avg qty per transaksi GUM Suprem
avg_qty_suprem = trans[trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM'].groupby('no_member')
.apply(lambda df: df['qty'].sum() / df['struk'].nunique())
trans['avg_qty_per_trans_suprem'] = trans['no_member'].map(avg_qty_suprem).fillna(0)
```

Gambar 3.19 *Feature engineer* tambahan untuk *targeted* Juli 2025

Gambar 3.19 menunjukkan pembuatan fitur tambahan lainnya seperti persentase keseluruhan transaksi dengan transaksi susu pertumbuhan atau target juga ditambahkan (‘persen_trans_suprem’), yaitu rata-rata jumlah produk target yang dibeli per transaksi juga ditambahkan untuk memperkaya data.

```
# Ratio qty suprem vs total qty semua produk
total_qty = trans.groupby('no_member')['qty'].sum()
qty_suprem = trans[trans['descp_kat'] == 'GROWING UP MILK SUPER PREMIUM'].groupby('no_member')['qty'].sum()
ratio_qty_suprem = (qty_suprem / total_qty).fillna(0)
trans['ratio_qty_suprem'] = trans['no_member'].map(ratio_qty_suprem)

# Jumlah voucher_targeted di produk lain selain Suprem
voucher_targeted_non_suprem = trans[(trans['tipe_voucher'] == 'voucher_targeted') &
                                    (trans['descp_kat'] != 'GROWING UP MILK SUPER PREMIUM')].groupby('no_member').size()
trans['voucher_targeted_non_suprem'] = trans['no_member'].map(voucher_targeted_non_suprem).fillna(0).astype(int)


# Jumlah semua penggunaan voucher (semua tipe, tidak hanya targeted)
voucher_all = trans[trans['nominal_voucher'] > 0].groupby('no_member').size()
trans['jumlah_voucher_all'] = trans['no_member'].map(voucher_all).fillna(0).astype(int)
```

Gambar 3.20 *Feature engineer* tambahan untuk *targeted* Juli 2025

Pada gambar 3.20, ditunjukkan pembuatan fitur ‘rasio_qty_suprem’ atau rasio jumlah pembelian item target dibanding dengan seluruh jumlah item dibeli masing-masing *member.* Terdapat juga ‘voucher_targeted_non_suprem’ atau jumlah *voucher targeted* lainnya selain

produk target. Kemudian ‘jumlah_voucher_all’ atau jumlah semua *voucher* yang telah digunakan oleh *member* tersebut.

```
trans.columns
```

```
Index(['no_member', 'tgl', 'struk', '[redacted]_PLU', 'descp', 'kat', 'descp_kat',
       'payment', 'sales', 'qty', 'retur', 'tipe_voucher', 'nominal_voucher',
       'redeemed_voucher', 'jumlah_transaksi', 'jumlah_transaksi_suprem',
       'total_sales', 'basket_size', 'qty_purchased', 'purchased_gum',
       'qty_suprem', 'jumlah_voucher_others', 'latency_suprem',
       '[redacted]_plu_suprem', 'Recent', 'recency_all', 'recency_gum_pm',
       'recency_suprem', 'persen_trans_suprem', 'avg_qty_per_trans_suprem',
       'ratio_qty_suprem', 'voucher_targeted_non_suprem',
       'jumlah_voucher_all'],
      dtype='object')
```

```
trans.drop(columns=['struk','sales','tipe_voucher','retur','descp_kat',
                    '[redacted]_PLU','kat','descp','payment','qty','nominal_voucher','tgl'],inplace=True)
```

Gambar 3.21 Penghapusan kolom untuk *targeted* Juli 2025

Gambar 3.21 menampilkan kolom-kolom yang pada akhirnya ada dalam tabel trans dan proses *drop* atau menghapus kolom-kolom yang tidak dibutuhkan. Tabel trans yang ada sebelumnya itu datanya dalam level produk, atau per baris per produk, sedangkan fitur-fitur yang ditambahkan adalah per *member*, maka dari itu, kolom-kolom yang ada sebelumnya, selain *identifier* dihapus.

```
vp=event[event['[redacted]_EVENT_NAME']=='view_product']
atc=event[event['[redacted]_EVENT_NAME']=='add_to_cart']

targeted = trans

targeted['view_product'] = targeted['no_member'].isin(vp['[redacted]_member_id']).astype(int)

vp_sorted = vp.sort_values(by=['[redacted]_member_id', 'date_event'])

vp_last = vp_sorted.drop_duplicates(subset='[redacted]_member_id', keep='last')

targeted = targeted.merge(
    vp_last[['[redacted]_member_id', '[redacted]_plu']],
    how='left',
    left_on='no_member',
    right_on='[redacted]_member_id'
)
targeted = targeted.rename(columns={'[redacted]_plu': 'plu_vp'})
targeted = targeted.drop(columns=['[redacted]_member_id'])
targeted['plu_vp'] = targeted['plu_vp'].fillna(0)
```

Gambar 3.22 Penambahan kolom *event* untuk *targeted* Juli 2025

Setelah proses *feature engineering* dari data transaksi dan pembersihan kolom yang tidak relevan, tahap berikutnya adalah menambahkan fitur tambahan dari tabel *event* yang mencatat aktivitas interaksi pengguna di aplikasi seperti yang dapat dilihat pada gambar 3.22. Pada gambar ini, data *event* dipisahkan menjadi dua jenis utama, yaitu

tindakan melihat deskripsi produk (view_product) dan menambahkan produk ke keranjang (add_to_cart). Selanjutnya, dibuat kolom baru bernama ‘view_product’ pada tabel utama untuk menandai apakah seorang *member* pernah melihat suatu produk, sehingga model dapat memanfaatkan perilaku interaksi pengguna sebagai informasi tambahan dalam analisis *targeted* Juli 2025.

```
targeted['add_to_cart'] = targeted['no_member'].isin(atc['[illegible]_member_id']).astype(int)
```

```
atc_sorted = atc.sort_values(by=['[illegible]_member_id', 'date_event'])

atc_last = atc_sorted.drop_duplicates(subset='[illegible]_member_id', keep='last')

targeted = targeted.merge(
    atc_last[['[illegible]_member_id', '[illegible]_plu']],
    how='left',
    left_on='no_member',
    right_on='[illegible]_member_id'
)
targeted = targeted.rename(columns={'[illegible]_plu': 'plu_atc'})
targeted = targeted.drop(columns=['[illegible]_member_id'])
targeted['plu_atc'] = targeted['plu_atc'].fillna(0)
```

Gambar 3.23 Penambahan kolom *event* “add_to_cart” untuk *targeted* Juli 2025

Setelah kolom *view_product* ditambahkan pada tahap sebelumnya, proses selanjutnya adalah menambahkan kolom ‘add_to_cart’ seperti yang ditunjukkan pada gambar 3.23, yang menunjukkan apakah seorang *member* pernah menambahkan produk ke keranjang belanja di aplikasi. Langkah ini dilakukan dengan memanfaatkan data *event* yang memiliki tipe aktivitas “add_to_cart”. Data tersebut kemudian diurutkan berdasarkan waktu interaksi, diambil catatan terakhir untuk setiap *member*, dan digabungkan dengan tabel utama. Penambahan fitur ini memberikan informasi perilaku pembelian yang lebih dalam, karena pengguna yang menambahkan produk ke keranjang umumnya memiliki minat lebih tinggi untuk melakukan transaksi dibandingkan yang hanya melihat produk.

```
targeted['restock'] = targeted['no_member'].isin(restock['[redacted]_ponta_id']).astype(int)

restock_agg = restock.groupby('[redacted]_ponta_id').agg({
    'perkiraan_habis_hari_multiplied': 'mean',
    'selisih_waktu_next': 'median',
    '[redacted]_date': 'count'
}).reset_index().rename(columns={
    'perkiraan_habis_hari_multiplied': 'mean_perkiraan_habis_hari_multiplied',
    'selisih_waktu_next': 'median_selisih_waktu_next',
    '[redacted]_date': 'count_restock'
})

targeted = targeted.merge(
    restock_agg,
    how='left',
    left_on='no_member',
    right_on='[redacted]_ponta_id'
).drop(columns=['[redacted]_ponta_id'])

targeted['mean_perkiraan_habis_hari_multiplied'] = targeted['mean_perkiraan_habis_hari_multiplied'].fillna(0)
targeted['median_selisih_waktu_next'] = targeted['median_selisih_waktu_next'].fillna(0)
targeted['count_restock'] = targeted['count_restock'].fillna(0)
```

Gambar 3.24 Penambahan kolom “restock” untuk *targeted* Juli 2025

Gambar 3.24 memperlihatkan proses penambahan fitur ‘restock’ ke dalam *dataset targeted*. Fitur ini digunakan untuk mengidentifikasi perilaku pembelian ulang (*restock*) oleh *member*. Pada tahap ini, data *restock* diolah dengan melakukan agregasi berdasarkan ID *member* untuk menghitung beberapa metrik penting, yaitu rata-rata perkiraan waktu habisnya produk (mean_perkiraan_habis_hari_multiplied), median selisih waktu antar pembelian (median_selisih_waktu_next), serta jumlah total aktivitas *restock* (count_restock). Hasil agregasi kemudian digabungkan dengan tabel utama (*targeted*) menggunakan metode *left join*, sehingga setiap *member* memiliki informasi tambahan terkait frekuensi dan pola pembelian ulang mereka. Fitur ini berperan penting dalam analisis perilaku pelanggan jangka panjang serta dalam memprediksi kebutuhan *restock* di masa depan.

```
1  buy_affinity = target_affinity[target_affinity['remark']=='buy_affinity_only']

1  targeted['buy_affinity'] = targeted['no_member'].isin(buy_affinity['[redacted]_ponta_id']).astype(int)

1  buy_target_affinity = target_affinity[target_affinity['remark']=='buy_affinity_and_target_together']

1  targeted['buy_target_affinity'] = targeted['no_member'].isin(buy_target_affinity['[redacted]_ponta_id']).astype(int)
```

Gambar 3.25 Penambahan kolom afinitas pembelian pada data *targeted* Juli 2025

Gambar 3.25 menampilkan proses pembuatan dua fitur baru, yaitu buy_affinity dan buy_target_affinity, yang ditambahkan ke dalam *dataset targeted*. Kedua fitur ini dibuat berdasarkan hasil pemetaan dari tabel

target_affinity, yang berisi informasi hubungan pembelian antara produk afinitas dan produk target. Fitur buy_affinity menunjukkan apakah seorang *member* pernah membeli produk afinitas saja (buy_affinity_only), sedangkan buy_target_affinity menunjukkan apakah *member* tersebut pernah membeli produk afinitas dan produk target secara bersamaan (buy_affinity_and_target_together). Nilai pada kolom tersebut diubah menjadi tipe data integer (0 atau 1) untuk memudahkan analisis selanjutnya. Proses ini penting karena membantu mengidentifikasi pola perilaku pelanggan yang memiliki kecenderungan terhadap produk tertentu dan potensi keterkaitan pembelian antar kategori produk.

```
df_per_member = targeted.groupby('no_member').agg({
    'redeemed_voucher': 'max',
    'jumlah_transaksi': 'first',
    'jumlah_transaksi_suprem':'first',
    'total_sales': 'first',
    'basket_size': 'first',
    'purchased_gum': 'first',
    'jumlah_voucher_others': 'first',
    'qty_suprem': 'first',
    'qty_purchased': 'first',
    'latency_suprem': 'first',
    '    _plu_suprem':'first',
    'Recent':'first',

    'recency_all':'first',
    'recency_gum_pm':'first',
    'recency_suprem':'first',
    'persen_trans_suprem':'first',
    'avg_qty_per_trans_suprem':'first',
    'ratio_qty_suprem':'first',
    'voucher_targeted_non_suprem':'first',
    'jumlah_voucher_all':'first',

    'view_product' :'first',
    'add_to_cart':'first',
    'restock':'first',
    'mean_perkiraan_habis_hari_multiplied':'first',
    'median_selisih_waktu_next':'first',
    'count_restock':'first',
    'buy_affinity':'first',
    'buy_target_affinity':'first'
}).reset_index()

df_per_member
```

Gambar 3.26 Agregasi fitur per *member* untuk *targeted* Juli 2025

Gambar 3.26 menunjukkan proses agregasi data per *member* untuk membentuk *dataset* akhir yang siap digunakan dalam tahap pemodelan. Data

transaksi yang semula berada pada level baris per transaksi dikonversi menjadi satu baris per pelanggan menggunakan fungsi groupby('no_member'). Setiap kolom fitur yang relevan, seperti jumlah transaksi, total penjualan, ukuran keranjang (*basket size*), frekuensi pembelian produk *suprem*, recency, serta aktivitas *member* (*view_product*, *add_to_cart*, dan *restock*), diambil dengan metode agregasi seperti 'first' atau 'max' sesuai konteks datanya.
Langkah ini menghasilkan *dataframe* df_per_member, yang berisi representasi ringkas namun informatif dari perilaku masing-masing pelanggan.

```
df_per_member['latency_suprem']=df_per_member['latency_suprem'].fillna(0)
df_per_member['    _plu_suprem']=df_per_member['    _plu_suprem'].fillna(0)
df_per_member['Recent']=df_per_member['Recent'].fillna(0)

parq = df_per_member
parq['    _plu_suprem'] = parq['    _plu_suprem'].astype(str)
parq = parq.to_parquet('                                    /Angelin/suprem/buat_juli/transluarjawajantoaprFE')
```

Gambar 3.27 Tahap akhir pembersihan data dan penyimpanan dalam format Parquet untuk *targeted* Juli 2025

Gambar 3.27 memperlihatkan tahap akhir dalam proses pengolahan data, yaitu pembersihan nilai hilang (*missing values*) dan penyimpanan *dataset* dalam format efisien untuk analisis lanjutan. Pada bagian awal kode, kolom seperti latency_suprem, plu_suprem, dan Recent diisi menggunakan fungsi .fillna(0) agar tidak ada nilai kosong yang dapat mengganggu proses analisis atau pelatihan model. Selanjutnya, dataset df_per_member disalin ke variabel parq, dan tipe data kolom plu_suprem diubah menjadi *string* untuk memastikan konsistensi format. Langkah terakhir adalah menyimpan hasil olahan data ke dalam format Parquet, yaitu format penyimpanan kolom yang mendukung kompresi dan pemrosesan data berskala besar dengan lebih efisien.

```
from sklearn.preprocessing import LabelEncoder, StandardScaler

model = targeted_perbulan[[
    'no_member',
                  'redeemed_voucher',
                  'jumlah_transaksi',
                  'jumlah_transaksi_suprem',
                  'total_sales',
                  'basket_size',
                  'purchased_gum',
                  'jumlah_voucher_others',
                  'qty_suprem',
                  'qty_purchased',
                  'latency_suprem',
#                   '   _plu_suprem',
#                   'Recent',
                                       'recency_all',
                                       'recency_gum_pm',
                                       'recency_suprem',
                                       'persen_trans_suprem',
                                       'avg_qty_per_trans_suprem',
                  'ratio_qty_suprem',
                                       'jumlah_voucher_all',
                  'view_product',
#                   'plu_vp',
                  'add_to_cart',
#                   'plu_atc',
                  'restock',
                  'mean_perkiraan_habis_hari_multiplied',
                  'median_selisih_waktu_next',
                  'count_restock',
                  'buy_affinity',
                  'buy_target_affinity'
                    ]]
model.set_index('no_member',inplace=True)
model
```

Gambar 3.28 Pemilihan fitur dari tabel hasil *preprocessing* untuk pembangunan model untuk *targeted* Juli 2025

Gambar 3.28 menampilkan tahapan pemilihan fitur (*feature selection*) dari tabel yang telah melalui proses *data preprocessing*. Pada tahap ini, hanya variabel-variabel yang relevan dan memiliki kontribusi signifikan terhadap prediksi perilaku pelanggan yang dipertahankan. Fitur-fitur yang dipilih mencakup aspek demografis pelanggan, perilaku transaksi, serta interaksi promosi (misalnya penggunaan *voucher* atau promo tertentu).

```python
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from skl=earn.metrics import classification_report
from sklearn.impute import SimpleImputer
import seaborn as sns
import matplotlib.pyplot as plt

X = model.drop(columns=['redeemed_voucher'])
y = model['redeemed_voucher']

imputer = SimpleImputer(strategy='mean')
X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

smote = SMOTE(sampling_strategy=0.3, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_imputed, y)

print(y_resampled.value_counts(normalize=True))

sns.countplot(x=model['redeemed_voucher'])
plt.title("Distribusi redeemed_voucher")
plt.show()
```

```
redeemed_voucher
0    0.769232
1    0.230768
Name: proportion, dtype: float64
```

Gambar 3.29 Penggunaan teknik *oversampling* SMOTE untuk *targeted* Juli 2025

Gambar 3.29 menunjukkan proses penyeimbangan data menggunakan metode SMOTE (*Synthetic Minority Over-sampling Technique*). Sebelum dilakukan penyeimbangan, data menunjukkan ketidakseimbangan pada variabel target redeemed_voucher. Untuk mengatasi hal ini, terlebih dahulu dilakukan imputasi data menggunakan SimpleImputer dengan strategi rata-rata (*mean*) untuk mengisi nilai yang hilang pada variabel numerik. Selanjutnya, diterapkan metode SMOTE dengan rasio sampling 0.3 agar jumlah data pada kelas minoritas meningkat secara sintetis dan lebih seimbang terhadap kelas mayoritas. Hasil penyeimbangan kemudian divisualisasikan menggunakan *countplot* untuk menunjukkan distribusi baru dari variabel redeemed_voucher. Langkah ini penting agar model yang akan dibangun tidak bias terhadap kelas mayoritas dan mampu memberikan performa prediksi yang lebih akurat untuk kedua kelas.

```
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled
)

clf = RandomForestClassifier(
    max_depth=15,
    random_state=42
)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Gambar 3.30 Pembagian data dan pelatihan model untuk *targeted* Juli 2025

Gambar 3.30 menunjukkan proses pembagian data dan pelatihan model menggunakan algoritma Random Forest. Dataset hasil penyeimbangan dibagi menjadi dua bagian, yaitu data latih (70%) dan data uji (30%), menggunakan fungsi train_test_split dengan parameter *stratify* untuk memastikan proporsi kelas tetap seimbang di kedua sub set. Selanjutnya, model Random Forest dibangun dengan hyperparameter *max_depth* sebesar 15 dan *random_state* sebesar 42 untuk menjaga konsistensi hasil. Model kemudian dilatih menggunakan data latih (X_train, y_train) dan menghasilkan prediksi pada data uji (X_test).

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

y_train_pred = clf.predict(X_train)
train_acc = accuracy_score(y_train, y_train_pred)

y_test_pred = clf.predict(X_test)
test_acc = accuracy_score(y_test, y_test_pred)

print("Training Accuracy:", train_acc)
print("Testing Accuracy :", test_acc)

print("\nClassification Report (Test Data):dengan smote\n", classification_report(y_test, y_test_pred))
```

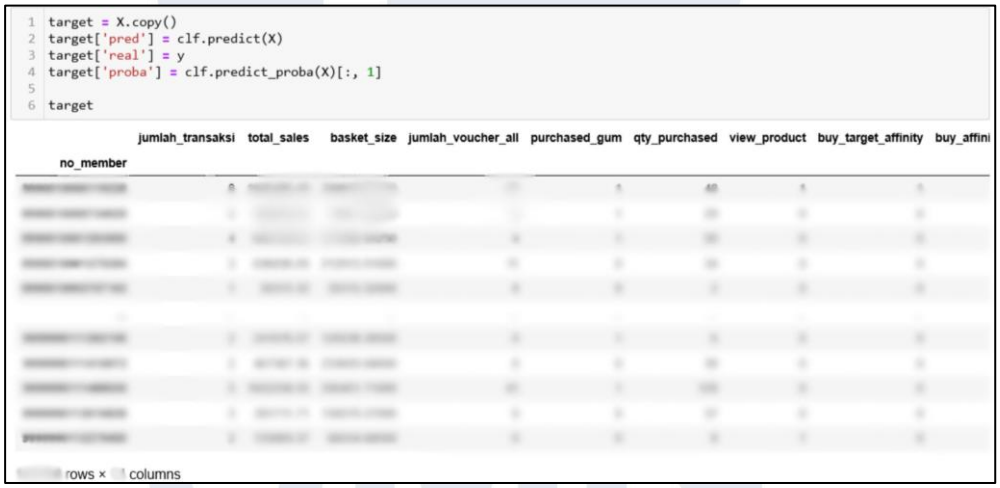
```
Training Accuracy: 0.9758858753714527
Testing Accuracy : 0.9720139092546818

Classification Report (Test Data):dengan smote
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     36722
           1       0.97      0.91      0.94     11016

    accuracy                           0.97     47738
```

Gambar 3.31 Evaluasi model Random Forest untuk *targeted* Juli 2025

Gambar 3.31 menunjukkan hasil evaluasi model Random Forest setelah dilakukan proses pelatihan menggunakan data yang telah diseimbangkan dengan teknik SMOTE. Model dievaluasi menggunakan

metrik *accuracy* dan *classification report* untuk melihat performa pada data latih dan data uji.

Hasil menunjukkan bahwa model memiliki akurasi pelatihan sebesar 97.59% dan akurasi pengujian sebesar 97.20%, menandakan bahwa model mampu melakukan generalisasi dengan baik tanpa mengalami overfitting yang signifikan. Berdasarkan laporan klasifikasi, nilai *precision*, *recall*, dan *F1-score* untuk kedua kelas (0 dan 1) relatif tinggi. Hal ini menunjukkan bahwa model cukup seimbang dalam mengidentifikasi pengguna yang menukarkan *voucher* (*redeemed*) maupun yang tidak, sehingga model dapat diandalkan untuk tahap analisis lebih lanjut dan penerapan strategi penargetan *voucher*.



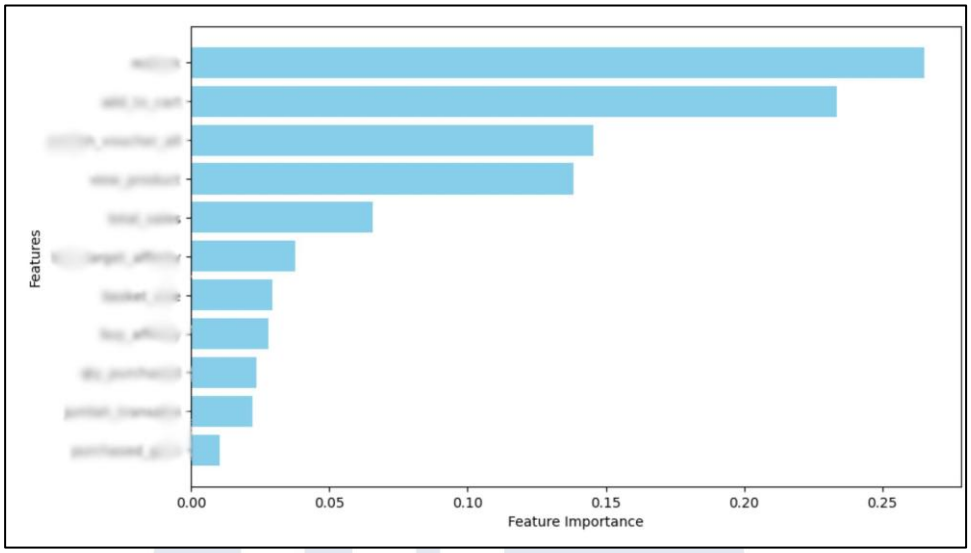
```
target = X.copy()
target['pred'] = clf.predict(X)
target['real'] = y
target['proba'] = clf.predict_proba(X)[:, 1]

target
```

no_member	jumlah_transaksi	total_sales	basket_size	jumlah_voucher_all	purchased_gum	qty_purchased	view_product	buy_target_affinity	buy_affini
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

[redacted] rows × [redacted] columns

Gambar 3.32 Penambahan kolom probabilitas untuk *targeted* Juli 2025

Gambar 3.32 menampilkan hasil *output* model. Tabel *output* model kemudian ditambahkan kolom probabilitas klasifikasi untuk kelas 1 (*redeem voucher*) yang dihasilkan oleh model. Probabilitas ini ditambahkan agar dapat mem-*filter member-member* yang memiliki kecenderungan untuk *redeem voucher* lebih tinggi..



Gambar 3.33 Feature Importance untuk *targeted* Juli 2025

Gambar 3.33 menampilkan hasil *feature importance* dari model yang digunakan untuk memprediksi perilaku *redeem voucher* oleh pelanggan. Grafik ini menggambarkan sejauh mana setiap fitur berkontribusi terhadap keputusan model dalam mengklasifikasikan apakah seorang pelanggan cenderung menukarkan *voucher* atau tidak.

```
model = toiletries[['_ponta_id',
                    'sales', 'qty', 'basket',
                    'sales_toileters', 'qty_toileters', 'basket_toileters',
                    'sales_gum', 'qty_gum', 'basket_gum',
                    'sales_affinity', 'qty_affinity', 'basket_affinity',
                    'struk', 'struk_toileters', 'struk_gum', 'struk_affinity',
#                     'voucher_used',
                    'redeemed_targeted', 'voucher_others_used',
                    'vp_all', 'atc_all', 'atc_and_buy_all',
                    'vp_target', 'atc_target', 'atc_and_buy_target',
                    'vp_affinity', 'atc_affinity', 'atc_and_buy_affinity',
                    'tbtd_delivery_city',

                    'purchased_gum', 'purchased_affinity', 'purchased_toiletries',
                   'atc_toiletries','atc_affinity',
                   'view_affinity', 'view_toiletries']]
model.set_index('_ponta_id', inplace=True)
model
```

_ponta_id	sales	qty	basket	sales_toileters	qty_toileters	basket_toileters	sales_gum	qty_gum	basket_gum	sales_affinity	...	atc_affinity

Gambar 3.34 Pemilihan fitur untuk model *targeted baby needs* Juli 2025

Gambar 3.34 menunjukkan proses pemilihan fitur yang digunakan untuk membangun model *targeted baby needs* pada Juli 2025. Pada tahap ini, dilakukan seleksi terhadap kolom-kolom yang relevan dari *dataset* hasil

preprocessing sebelumnya, kemudian disimpan ke dalam variabel baru bernama model. Fitur-fitur yang dipilih mencakup berbagai indikator perilaku pelanggan, seperti *sales*, *quantity*, dan *basket size* untuk beberapa kategori produk seperti *toiletries*, *gum*, dan *affinity*. Selain itu, variabel interaksi seperti *view*, *add to cart (atc)*, dan *purchased* juga disertakan untuk masing-masing kategori, bersama dengan fitur terkait penggunaan voucher dan lokasi pengiriman. Langkah terakhir adalah menetapkan kolom *ponta_id* sebagai *index*, sehingga setiap baris data dapat diidentifikasi secara unik berdasarkan ID pelanggan.

```
Training Accuracy: 0.9958792040708618
Testing Accuracy : 0.9933408893618617

Classification Report (Test Data):
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    266726
           1       0.98      1.00      0.99     80017

    accuracy                           0.99    346743
   macro avg       0.99      0.99      0.99    346743
weighted avg       0.99      0.99      0.99    346743
```

Gambar 3.35 Hasil evaluasi model *targeted baby needs* Juli 2025

Gambar 3.35 menunjukkan hasil evaluasi performa model *targeted baby needs* Juli 2025. Berdasarkan hasil yang ditampilkan, model menunjukkan kinerja yang sangat baik dengan *training accuracy* sebesar 99,59% dan *testing accuracy* sebesar 99,33%, yang menandakan kemampuan generalisasi model tergolong sangat tinggi. Nilai *precision*, *recall*, dan *f1-score* untuk kedua kelas juga sangat baik, masing-masing berada pada kisaran 0,98–1,00, menunjukkan bahwa model mampu mengklasifikasikan pelanggan dengan benar hampir tanpa kesalahan. Untuk kelas "1" (pelanggan yang menjadi target potensial), *recall* mencapai 1,00, yang berarti model

sangat efektif dalam mengenali seluruh pelanggan yang benar-benar termasuk kategori tersebut.

Gambar 3.36 Penambahan fitur ‘proba’ untuk *targeted baby needs* Juli 2025

Gambar 3.36 memperlihatkan proses seleksi *member* berdasarkan probabilitas hasil prediksi model klasifikasi. Pada tahap ini, *dataset* hasil prediksi disalin ke dalam variabel baru dan ditambahkan kolom ‘proba’, yang berisi nilai probabilitas untuk kelas 1, yaitu peluang bahwa seorang *member* termasuk ke dalam kategori *targeted*. Selanjutnya, dilakukan penyaringan terhadap *member* dengan nilai probabilitas di atas 0,5, atau 50%. Dengan demikian, hanya pelanggan yang memiliki kemungkinan tinggi untuk merespons promosi produk *toiletries* yang akan dimasukkan ke dalam kelompok *targeted*.

3.3.1.4 Pengolahan *output* model dan *best time to buy* untuk *voucher targeted* Juli 2025.

Setelah diperoleh dua keluaran (*output*) hasil prediksi model dengan target *redeem voucher*, tahap berikutnya adalah melakukan penggabungan dengan sumber data tambahan, yaitu tabel *best time to buy*. Tabel ini berisi perkiraan waktu terbaik bagi setiap *member* untuk melakukan pembelian ulang (*repurchase*) terhadap suatu produk. Dengan mengintegrasikan hasil model prediksi dan informasi dari tabel ‘Best Time to Buy’, proses penentuan

member targeted untuk program *voucher targeted* menjadi lebih tepat dan efektif. Pendekatan ini memungkinkan perusahaan untuk tidak hanya menargetkan pelanggan dengan potensi tinggi dalam penukaran *voucher*, tetapi juga menyesuaikan waktu pengiriman penawaran berdasarkan perilaku pembelian aktual masing-masing *member*, sehingga dapat meningkatkan peluang konversi dan efisiensi kampanye pemasaran.

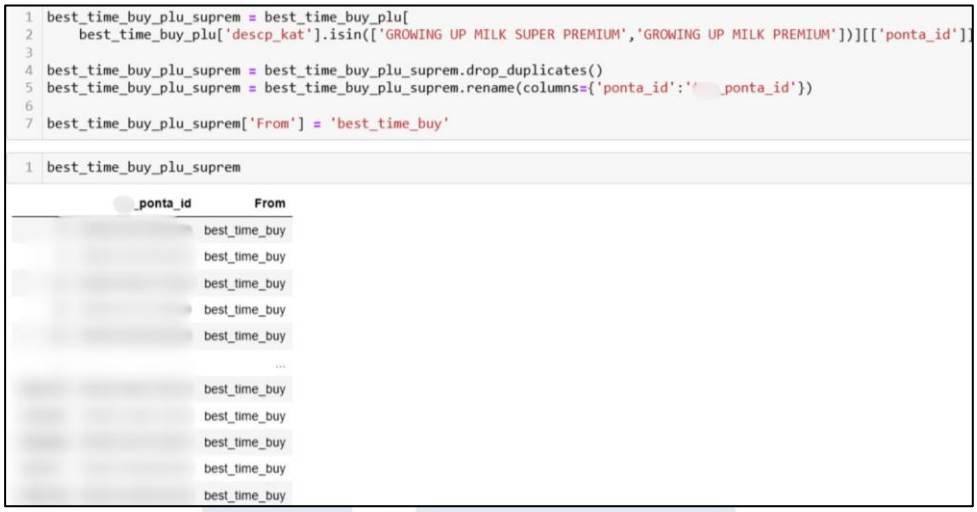
```
best_time_buy_plu = pd.read_parquet('                    Angelin/best_time_buy_juli_luar_jawa')
best_time_buy_plu=best_time_buy_plu[(best_time_buy_plu['tanggal_kembali']>='2025-07-01') &
                                   (best_time_buy_plu['tanggal_kembali']<='2025-07-31')]

best_time_buy_plu_toilet = best_time_buy_plu[best_time_buy_plu['descp_kat']=='PANTS DIAPERS'][['ponta_id']]

best_time_buy_plu_toilet = best_time_buy_plu_toilet.drop_duplicates()
best_time_buy_plu_toilet = best_time_buy_plu_toilet.rename(columns={'ponta_id':'    _ponta_id'})

best_time_buy_plu_toilet['From'] = 'best_time_buy'
```

```
best_time_buy_plu_toilet
```

_ponta_id	From
	best_time_buy
	best_time_buy
	best_time_buy
	best_time_buy
	best_time_buy
	...
	best_time_buy
	best_time_buy
	best_time_buy
	best_time_buy
	best_time_buy

Gambar 3.37 Penambahan *member* best time to buy untuk *targeted baby needs* Juli 2025

Gambar 3.37 menunjukkan proses penambahan *member* yang berasal dari tabel ‘best time to buy’ untuk kategori *baby needs* pada periode Juli 2025. Pada tahap ini, data diambil dari tabel sumber best_time_buy_juli_luar_jawa dan difilter berdasarkan rentang tanggal perkiraan pembelian ulang, yaitu antara 1 Juli 2025 hingga 31 Juli 2025. Selanjutnya, dilakukan pemilihan data dengan kategori produk “PANTS DIAPERS” sebagai representasi dari kebutuhan bayi. Data hasil penyaringan kemudian dibersihkan dari duplikasi dan disesuaikan nama kolom identitas *member*-nya agar konsisten dengan struktur data utama. Proses ini menghasilkan daftar *member* yang memiliki potensi tinggi untuk melakukan pembelian ulang produk bayi pada periode tersebut, yang kemudian ditandai dengan label sumber “best_time_buy” untuk integrasi ke dalam model *targeted voucher*.



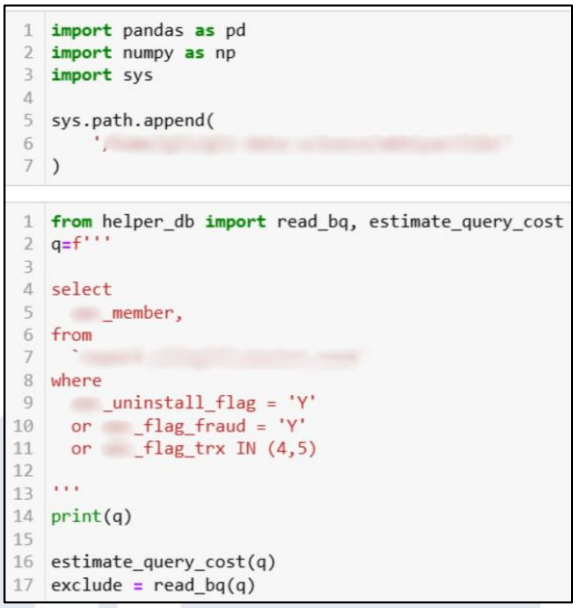
```
best_time_buy_plu_suprem = best_time_buy_plu[
    best_time_buy_plu['descp_kat'].isin(['GROWING UP MILK SUPER PREMIUM','GROWING UP MILK PREMIUM'])][['ponta_id']]

best_time_buy_plu_suprem = best_time_buy_plu_suprem.drop_duplicates()
best_time_buy_plu_suprem = best_time_buy_plu_suprem.rename(columns={'ponta_id':'[illegible]ponta_id'})

best_time_buy_plu_suprem['From'] = 'best_time_buy'
```

```
best_time_buy_plu_suprem
```

[illegible]ponta_id	From
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy
...	
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy
[illegible]	best_time_buy

Gambar 3.38 Penambahan *member best time to buy* untuk *targeted* susu pertumbuhan Juli 2025

Gambar 3.38 menunjukkan proses penambahan *member* yang berasal dari tabel ‘best time to buy’ untuk kategori produk *susu pertumbuhan* pada periode Juli 2025. Pada tahap ini, data diambil dari tabel best_time_buy_plu dan diseleksi berdasarkan kolom descp_kat yang berisi kategori ‘*GROWING UP MILK SUPER PREMIUM’* dan ‘*GROWING UP MILK PREMIUM’*. Setelah itu, data difilter untuk mengambil hanya kolom identitas *member* (ponta_id), kemudian dibersihkan dari duplikasi menggunakan fungsi drop_duplicates(). Nama kolom juga disesuaikan agar konsisten dengan struktur tabel utama melalui rename(), dan setiap *member* diberi label sumber 'best_time_buy' untuk menandakan asal data tersebut. Hasil akhir menghasilkan daftar *member* potensial yang diperkirakan akan melakukan pembelian susu pertumbuhan pada periode Juli 2025 untuk digunakan dalam proses *targeted voucher*.



```
import pandas as pd
import numpy as np
import sys

sys.path.append(
    '
)
```

```
from helper_db import read_bq, estimate_query_cost
q=f'''

select
   _member,
from
  `
where
   _uninstall_flag = 'Y'
  or _flag_fraud = 'Y'
  or _flag_trx IN (4,5)

'''
print(q)

estimate_query_cost(q)
exclude = read_bq(q)
```

Gambar 3.39 Filter *flag member* untuk *targeted* Juli 2025

Gambar 3.39 menunjukkan proses *exclude* terhadap *member* yang memiliki *flag* tertentu, seperti *uninstall flag, fraud flag*, maupun *flag* transaksi yang menandakan aktivitas tidak aktif (misalnya status transaksi 4 atau 5). Tahapan ini dilakukan sebagai bagian dari proses pembersihan data (data *cleaning*) untuk memastikan bahwa hanya *member* yang benar-benar aktif dan valid yang disertakan dalam tahap analisis berikutnya, terutama dalam penentuan target promosi. Dengan mengecualikan *member* yang memiliki indikasi perilaku tidak aktif, penipuan, atau sudah tidak menggunakan aplikasi, maka hasil analisis dan model yang dikembangkan menjadi lebih akurat, efisien, dan relevan terhadap tujuan bisnis. Langkah ini juga membantu mengoptimalkan penggunaan sumber daya promosi agar lebih tepat sasaran dan tidak dialokasikan kepada *member* yang berpotensi tidak memberikan respons terhadap kampanye yang dijalankan.

3.3.1.5 Eksplorasi data *clone member* dan *library* untuk *similarity detection.*

```
query = '''
SELECT [redacted]_member_id CREATED_BY,
 [redacted]name MEMBER_NAME,
 [redacted]_email EMAIL,
 tgl_daftar CREATED_DATE
 FROM [redacted]_check_clone`
where tgl_daftar >= '2024-01-01'
and [redacted]_email is not null and pmp_email <> ''
'''

df = helper_db.read_bq(query)
```

```
df.head()
```

	CREATED_BY	MEMBER_NAME	EMAIL	CREATED_DATE
0	[redacted]	[redacted]	[redacted]	[redacted]
1	[redacted]	[redacted]	[redacted]	[redacted]
2	[redacted]	[redacted]	[redacted]	[redacted]
3	[redacted]	[redacted]	[redacted]	[redacted]
4	[redacted]	[redacted]	[redacted]	[redacted]

Gambar 3.40 Eksplorasi data akun *member* untuk deteksi *clone member*

Gambar 3.40 menunjukkan proses awal eksplorasi data akun *member* yang digunakan untuk mendeteksi kemungkinan adanya *clone member* atau akun duplikat. Data ini merupakan hasil pemberian dari supervisor proyek, yang berisi informasi dasar seperti *member ID*, *member name*, *email*, serta *tanggal pendaftaran (created date)*. Tujuan dari tahap ini adalah untuk memahami struktur dan karakteristik data, sekaligus mengidentifikasi potensi fitur yang dapat digunakan dalam analisis deteksi duplikasi akun. Langkah pertama yang dilakukan adalah memanggil dan memuat data dari BigQuery menggunakan *library* helper_db yang telah disediakan oleh *supervisor*. Data kemudian diekstraksi ke dalam bentuk *DataFrame* dengan bantuan *library* pandas untuk mempermudah eksplorasi. Selain itu, *library* numpy juga digunakan untuk mendukung operasi numerik yang mungkin dibutuhkan dalam tahap pra-pemrosesan atau analisis lanjutan.

Setelah data berhasil dimuat, dilakukan eksplorasi awal (*initial data exploration*) dengan melihat beberapa baris pertama dari *dataset* menggunakan fungsi df.head(). Dari hasil ini, terlihat bahwa sebagian besar

data memiliki atribut yang relevan untuk proses deteksi *clone*, khususnya kolom email dan tanggal pendaftaran, yang dapat digunakan untuk mengidentifikasi pola pendaftaran ganda dengan alamat email mirip atau waktu pendaftaran berdekatan. Tahapan ini menjadi dasar dalam proses selanjutnya, yaitu menentukan fitur potensial untuk analisis *clone member*. Beberapa fitur yang dipertimbangkan antara lain kesamaan nama, domain email, serta jarak waktu antar pendaftaran akun. Dengan memahami struktur dan distribusi awal data, proses analisis dapat diarahkan secara lebih efektif untuk menemukan indikasi *clone member*.

3.3.1.6 Similarity *detection* dengan FuzzyWuzz.

```
from fuzzywuzzy import fuzz
from tqdm import tqdm
import pandas as pd
from concurrent.futures import ThreadPoolExecutor

# Ekstrak username & domain
df_sample = df.sample(n=10000, random_state=42).copy()
df_sample['username'] = df_sample['EMAIL'].str.extract(r'(^[^@]+)')
df_sample['domain'] = df_sample['EMAIL'].str.extract(r'@(.+)$')
df_sample['CREATED_DATE'] = pd.to_datetime(df_sample['CREATED_DATE'])

# Fungsi proses 1 parent row (untuk parallel)
def process_parent(i):
    parent = df_sample.iloc[i]
    clones, clone_names, clone_emails, clone_dates, clone_domains = [], [], [], [], []
    name_sim_scores, email_sim_scores, domain_sim_scores, date_diffs = [], [], [], []

    for j in range(i + 1, len(df_sample)):
        clone = df_sample.iloc[j]

        name_sim = fuzz.token_sort_ratio(str(parent['MEMBER_NAME']).lower(), str(cloxne['MEMBER_NAME']).lower())
        email_sim = fuzz.token_sort_ratio(str(parent['EMAIL']).lower(), str(clone['EMAIL']).lower())
        domain_sim = fuzz.ratio(str(parent['domain']).lower(), str(clone['domain']).lower())
        date_diff = abs((parent['CREATED_DATE'] - clone['CREATED_DATE']).days)

        if name_sim >= 85 and email_sim >= 80 and domain_sim >= 85 and date_diff <= 3:
            clones.append(clone['CREATED_BY'])
            clone_names.append(clone['MEMBER_NAME'])
            clone_emails.append(clone['EMAIL'])
            clone_dates.append(str(clone['CREATED_DATE'].date()))
            clone_domains.append(clone['domain'])
            name_sim_scores.append(name_sim)
            email_sim_scores.append(email_sim)
            domain_sim_scores.append(domain_sim)
            date_diffs.append(date_diff)
```

```
    if clones:
        return {
            'PARENT_ID': parent['CREATED_BY'],
            'PARENT_NAME': parent['MEMBER_NAME'],
            'PARENT_EMAIL': parent['EMAIL'],
            'PARENT_DOMAIN': parent['domain'],
            'PARENT_DATE': parent['CREATED_DATE'].date(),
            'CLONE_ID': clones,
            'CLONE_NAME': clone_names,
            'CLONE_EMAIL': clone_emails,
            'CLONE_DOMAIN': clone_domains,
            'CLONE_DATE': clone_dates,
            'NAME_SIMILARITY': name_sim_scores,
            'EMAIL_SIMILARITY': email_sim_scores,
            'DOMAIN_SIMILARITY': domain_sim_scores,
            'DATE_DIFF_DAYS': date_diffs
        }
    return None  # Kalau tidak ada clone

# Jalankan secara paralel
def detect_similar_parallel(max_workers=8):
    indices = list(range(len(df_sample)))
    results = []
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        for res in tqdm(executor.map(process_parent, indices), total=len(indices), desc="Parallel Deteksi"):
            if res:
                results.append(res)
    return pd.DataFrame(results)

# Eksekusi
result_df = detect_similar_parallel(max_workers=16)
```

Gambar 3.41 Deteksi *clone member* dengan *library* FuzzyWuzzy

Gambar 3.41 menunjukkan tahapan mencoba deteksi *clone member* dengan *library* FuzzyWuzzy. Pada tahap ini dilakukan proses deteksi kemiripan (*similarity detection*) antar akun menggunakan *library* FuzzyWuzzy untuk mengidentifikasi kemungkinan adanya *clone member* atau akun duplikat. Proses ini diawali dengan mengekstraksi atribut penting seperti *member name*, *email*, *domain*, dan *created date* guna mempermudah

proses perbandingan antar akun. Metode FuzzyWuzzy memanfaatkan algoritma *Levenshtein distance* untuk menghitung tingkat kesamaan antar *string*. Dalam tahap ini, tiga aspek utama yang dibandingkan meliputi *name similarity*, *email similarity*, dan *domain similarity*, dengan tambahan perhitungan selisih waktu pembuatan akun (*date difference*). Akun dikategorikan mirip apabila memenuhi ambang batas kemiripan tertentu, yaitu ≥ 85% untuk *name* dan *domain*, serta ≥ 80% untuk *email*, dengan selisih tanggal pembuatan maksimal tiga hari.

Namun, proses ini masih bersifat percobaan (*trial and error*) karena beban komputasinya sangat tinggi dan membutuhkan kapasitas RAM yang besar. Untuk menghindari *crash* dan memastikan proses berjalan lancar, percobaan dilakukan hanya pada 10.000 sampel data dari keseluruhan populasi. Selain itu, pemrosesan dilakukan secara paralel menggunakan *ThreadPoolExecutor* untuk mempercepat eksekusi. Langkah ini bertujuan untuk menguji kelayakan metode FuzzyWuzzy dalam mendeteksi kemiripan antar *member* sebelum diterapkan pada seluruh *dataset*. Hasil sementara menunjukkan potensi metode ini dalam mendeteksi pola duplikasi, namun diperlukan optimalisasi lebih lanjut agar dapat diterapkan pada data berskala besar secara efisien.

3.3.1.7 Similarity *detection* dengan *longitude* dan *latitude*.

Pada tahap ini dilakukan deteksi kemiripan antar *member* dengan mempertimbangkan lokasi geografis berdasarkan data *longitude* dan *latitude*. Pendekatan ini dikombinasikan dengan pengukuran kemiripan teks menggunakan RapidFuzz untuk atribut seperti *member name, email*, dan domain. Selain itu, digunakan juga jarak waktu pembuatan akun (*created date*) serta jarak lokasi yang dihitung menggunakan rumus Haversine untuk mengetahui kedekatan dua titik di permukaan bumi. Proses ini dijalankan secara paralel menggunakan ProcessPoolExecutor agar lebih efisien, mengingat beban pemrosesan yang tinggi. Karena penggunaan memori dan waktu komputasi yang sangat besar, percobaan ini dibatasi hanya pada

500.000 sampel data. Tujuannya adalah untuk mengidentifikasi potensi akun ganda (*clone members*) yang tidak hanya mirip secara teks, tetapi juga dibuat dalam waktu dan lokasi yang berdekatan, sehingga dapat menjadi indikasi kuat adanya duplikasi akun.

Namun, metode ini tidak digunakan pada tahap akhir analisis karena hasilnya kurang efektif dalam mengidentifikasi akun duplikat secara akurat. Hal ini disebabkan oleh kemungkinan bahwa beberapa *member* memang secara alami memiliki lokasi geografis yang berdekatan, misalnya karena tinggal di area yang sama atau melakukan pendaftaran di toko fisik yang sama. Akibatnya, penggunaan jarak *longitude* dan *latitude* justru dapat menghasilkan banyak *false positive*, yaitu akun yang dianggap mirip padahal bukan duplikasi sebenarnya.

3.3.1.8 Similarity *detection* dengan *secure device id*.

```
from rapidfuzz.fuzz import token_sort_ratio, ratio
from tqdm import tqdm
from concurrent.futures import ProcessPoolExecutor

df_sample = df_filtered.copy()
df_sample['username'] = df_sample['EMAIL'].str.extract(r'(^[^@]+)')
df_sample['domain'] = df_sample['EMAIL'].str.extract(r'@(.+)$')
df_sample['name_prefix'] = df_sample['MEMBER_NAME'].str[:3].str.lower()
df_sample = df_sample.reset_index(drop=True)
df_sample=df_sample.sort_values(by=['secure_device_id','create_at'])

def process_parent(i):
    parent = df_sample.iloc[i]
    clones, clone_names, clone_emails, clone_dates, clone_domains = [], [], [], [], []
    name_sim_scores, email_sim_scores, domain_sim_scores, date_diffs = [], [], [], []

    candidates = df_sample[df_sample['name_prefix'] == parent['name_prefix']]

    for j in candidates.index:
        if j <= i:
            continue

        clone = df_sample.loc[j]

        if clone['secure_device_id'] != parent['secure_device_id']:
            continue

        name_sim = token_sort_ratio(str(parent['MEMBER_NAME']), str(clone['MEMBER_NAME']))
        email_sim = token_sort_ratio(str(parent['EMAIL']), str(clone['EMAIL']))
        domain_sim = ratio(str(parent['domain']), str(clone['domain']))
        date_diff = abs((parent['create_at'] - clone['create_at']).days)

        if name_sim >= 85 and email_sim >= 80 and domain_sim >= 85:
            clones.append(clone['CREATED_BY'])
            clone_names.append(clone['MEMBER_NAME'])
            clone_emails.append(clone['EMAIL'])
            clone_dates.append(str(clone['create_at']))
            clone_domains.append(clone['domain'])
            name_sim_scores.append(name_sim)
            email_sim_scores.append(email_sim)
            domain_sim_scores.append(domain_sim)
            date_diffs.append(date_diff)
```

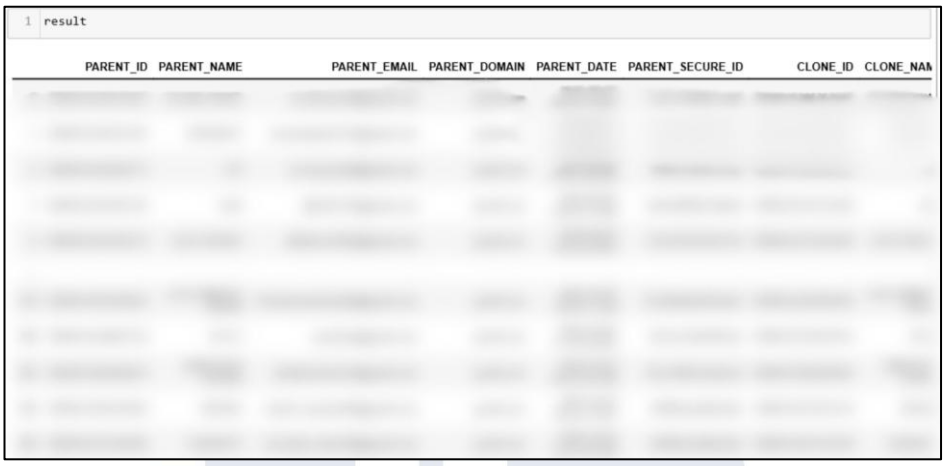
```
    if clones:
        return {
            'PARENT_ID': parent['CREATED_BY'],
            'PARENT_NAME': parent['MEMBER_NAME'],
            'PARENT_EMAIL': parent['EMAIL'],
            'PARENT_DOMAIN': parent['domain'],
            'PARENT_DATE': parent['create_at'],
            'PARENT_SECURE_ID': parent['secure_device_id'],
            'CLONE_ID': clones,
            'CLONE_NAME': clone_names,
            'CLONE_EMAIL': clone_emails,
            'CLONE_DOMAIN': clone_domains,
            'CLONE_DATE': clone_dates,
            'NAME_SIMILARITY': name_sim_scores,
            'EMAIL_SIMILARITY': email_sim_scores,
            'DOMAIN_SIMILARITY': domain_sim_scores,
            'DATE_DIFF_DAYS': date_diffs
        }
    return None

def detect_similar_parallel(max_workers=15):
    indices = list(range(len(df_sample)))
    results = []
    with ProcessPoolExecutor(max_workers=max_workers) as executor:
        for res in tqdm(executor.map(process_parent, indices), total=len(indices), desc="Fast Matching"):
            if res:
                results.append(res)
    return pd.DataFrame(results)

result_df = detect_similar_parallel(max_workers=15)
print(result_df.head())
```

Gambar 3.42 Deteksi clone member dengan secure device id

Gambar 3.40 menunjukan proses deteksi persamaan menggunakan atribut secure_device_id. Atribut secure_device_id terbukti jauh lebih reliabel dan dapat dipercaya untuk mengidentifikasi keterkaitan antar akun. Atribut ini merepresentasikan perangkat fisik yang digunakan oleh pengguna dengan tingkat konsistensi tinggi, karena setiap perangkat memiliki identitas unik yang sulit untuk dimanipulasi atau digandakan. Dengan demikian, penggunaan secure_device_id memungkinkan proses analisis dilakukan dengan tingkat akurasi yang lebih baik dalam mendeteksi potensi akun yang saling berkaitan atau diduga sebagai *clone member*.

Selain itu, secure_device_id juga mampu meminimalkan kemungkinan kesalahan dalam pengelompokan data, karena nilai yang digunakan bersifat tetap dan tidak berubah-ubah meskipun pengguna melakukan pembaruan aplikasi, berpindah lokasi, atau menggunakan akun berbeda pada perangkat yang sama. Hal ini menjadikannya sebagai indikator yang sangat kuat dalam proses validasi dan deteksi hubungan antar akun. Oleh karena itu, meskipun beberapa pendekatan lain sempat dicoba pada tahap eksplorasi data, analisis selanjutnya difokuskan pada pemanfaatan secure_device_id sebagai fitur utama. Pendekatan ini dinilai lebih efektif, efisien, dan relevan untuk mencapai tujuan analisis, yaitu mengidentifikasi pola keterkaitan antar akun secara akurat dan mendukung proses deteksi *clone member* dengan hasil yang lebih dapat diandalkan.

3.3.1.9 Data *cleaning* hasil *similarity detection*



Gambar 3.43 *Output* hasil *similarity detection* menggunakan *secure device id*

Gambar 3.43 menunjukkan hasil akhir data yang telah melalui proses pembersihan dan penataan pasangan akun (*parent-clone*) berdasarkan atribut *secure_device_id*. Hasil pada gambar tersebut merepresentasikan data yang sudah bersih dan siap digunakan untuk analisis lebih lanjut. Proses ini dilakukan untuk memastikan bahwa setiap pasangan akun yang diidentifikasi benar-benar relevan, tidak memiliki duplikasi, dan bebas dari pasangan yang bersifat redundan atau terbalik. Tahapan pembersihan dimulai dengan penyesuaian format kolom agar seluruh kolom seperti CLONE_ID, CLONE_NAME, CLONE_EMAIL, dan CLONE_DOMAIN memiliki format yang konsisten berbentuk *list*. Hal ini dilakukan agar proses *explode* dapat berjalan dengan baik dan menghasilkan struktur data yang rapi. Setelah itu, data di-*explode* untuk memisahkan setiap pasangan akun ke dalam baris tersendiri, sehingga setiap hubungan *parent-clone* dapat dilihat secara individual tanpa tumpang tindih data.

Langkah berikutnya adalah melakukan penyaringan terhadap data yang tidak relevan. Penyaringan ini bertujuan untuk menghapus pasangan akun yang sebenarnya adalah akun yang sama, yaitu ketika semua atribut seperti ID, nama, email, domain, dan tanggal memiliki kesamaan penuh. Dengan demikian, hanya hubungan antar akun yang benar-benar berbeda

yang dipertahankan. Selanjutnya, dilakukan penghapusan pasangan terbalik (*reverse pairs*), yaitu kondisi di mana kombinasi akun muncul dua kali dengan urutan berbeda. Untuk menangani hal ini, dibuat pasangan unik (*PAIR_KEY*) berdasarkan kombinasi terurut antara *parent* dan *clone* agar tidak ada pasangan yang muncul ganda. Proses ini dilakukan secara berulang hingga tidak ditemukan lagi pasangan duplikat di dalam *dataset*. Tahapan akhir adalah memastikan bahwa setiap akun *clone* hanya muncul satu kali di dalam *dataset*. Apabila satu akun *clone* terhubung dengan beberapa akun *parent*, maka hanya relasi pertama yang dipertahankan berdasarkan urutan waktu pembuatan akun (*PARENT_DATE*). Langkah ini dilakukan untuk menjaga integritas dan keunikan relasi antar akun.

3.3.1.10 Modeling dan pengolahan *output* model dan penggabungan *list member* untuk *voucher targeted* September 2025.

```
Training Accuracy: 0.9921568627450981
Testing Accuracy : 0.990665663498387

Classification Report (Test Data):
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    284228
           1       0.97      0.99      0.98     85268

    accuracy                           0.99    369496
```

Gambar 3.44 Evaluasi model *voucher targeted* September 2025

Gambar 3.44 menampilkan hasil evaluasi model Random Forest yang digunakan untuk proses *voucher targeted* pada kategori *baby needs* untuk periode September 2025. Proses pemodelan dilakukan dengan tahapan yang sama seperti yang telah dijelaskan pada poin 3.3.1.3, namun kali ini model difokuskan secara spesifik pada kategori produk kebutuhan bayi agar hasil rekomendasi *voucher* lebih relevan terhadap segmen pengguna yang dituju. Model menghasilkan performa yang sangat baik dengan akurasi pelatihan sebesar 0.992 dan akurasi pengujian sebesar 0.991, menunjukkan tingkat generalisasi yang tinggi tanpa indikasi *overfitting*. Berdasarkan *classification*

report, kelas mayoritas (label 0) memiliki nilai *precision* sebesar 1.00, *recall* 0.99, dan F1-score 0.99, sedangkan kelas minoritas (label 1) yang mewakili pengguna potensial untuk *voucher targeted* menunjukkan *precision* 0.97, *recall* 0.99, dan F1-score 0.98.

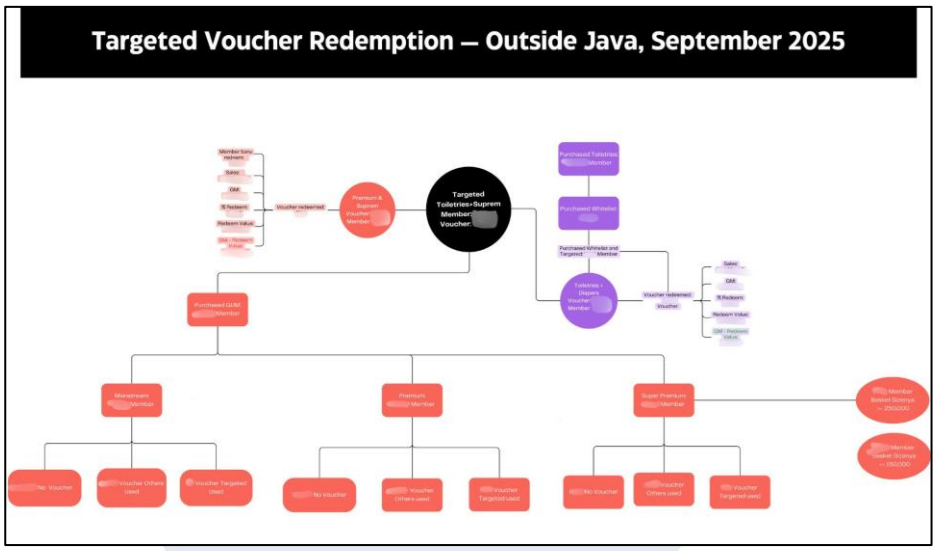
Gambar 3.45 Penambahan fitur proba untuk voucher targeted September 2025

Gambar 3.45 menampilkan proses penambahan fitur probabilitas (proba) pada hasil prediksi model Random Forest untuk program *voucher targeted* bulan September 2025. Nilai probabilitas ini dihasilkan dari fungsi predict_proba(), yang menunjukkan seberapa besar kemungkinan seorang *member* termasuk ke dalam kategori target penerima *voucher* berdasarkan perilaku pembeliannya. Dalam tahapan ini, setiap *member* juga diberikan skor probabilitas antara 0 hingga 1. Nilai tersebut merepresentasikan tingkat keyakinan model bahwa *member* tersebut memiliki kecenderungan tinggi untuk melakukan transaksi atau merespons *voucher* yang diberikan. Selanjutnya, dilakukan proses penyaringan menggunakan batas ambang (*threshold*) 0.5, di mana hanya *member* dengan nilai probabilitas di atas 0.5 yang dipilih sebagai *targeted members*.

Dengan demikian, *output* pada gambar tersebut merupakan kumpulan *member* yang memiliki potensi paling besar untuk menebus *voucher* berdasarkan hasil analisis prediktif model. Data ini kemudian menjadi dasar untuk proses selanjutnya, yaitu pengiriman voucher secara *targeted* kepada

member yang memiliki peluang tinggi dalam meningkatkan *engagement* dan *redeem rate* pada periode September 2025.

3.3.1.11 *Monitoring* hasil program *voucher targeted* September 2025



Gambar 3.46 *Monitoring voucher targeted* September 2025

Gambar 3.46 menampilkan alur pemantauan hasil distribusi *voucher targeted* selama periode September 2025. Visualisasi ini memperlihatkan bagaimana *voucher* dialokasikan kepada *member* yang telah teridentifikasi melalui model sebagai kandidat potensial *redeem*, kemudian dilacak kembali performanya melalui aktivitas *redeem* yang terjadi di transaksi aktual. Alur tersebut memberikan gambaran yang lebih jelas mengenai efektivitas program, dimulai dari jumlah *voucher* yang berhasil dikirim, jumlah *member* yang benar-benar melakukan *redeem*, hingga kontribusi transaksi yang dihasilkan dari aktivitas tersebut.

Melalui tampilan ini, keterlibatan *member* dapat diamati secara langsung, sehingga memberikan *insight* awal apakah strategi *targeting* berhasil mengarahkan penawaran kepada *member-member* yang relevan. Selain itu, visualisasi ini juga membantu mengidentifikasi gap antara jumlah distribusi dan realisasi *redeem*, yang dapat menjadi dasar evaluasi untuk menentukan apakah perlu dilakukan penyesuaian strategi, seperti optimalisasi

threshold probabilitas atau diferensiasi penawaran berdasarkan segmentasi perilaku belanja.

3.3.1.12 Modeling dan pengolahan *output* model dan penggabungan *list member* untuk *voucher targeted* Oktober 2025

Gambar 3.47 *Monitoring voucher targeted* September 2025

Gambar 3.47 menunjukkan tahapan berikutnya yang berfokus pada proses pemodelan dan pengolahan *output* model untuk menentukan daftar *member* yang akan menerima *voucher targeted* pada periode Oktober 2025. Proses dimulai dengan menerapkan model klasifikasi yang telah dilatih sebelumnya untuk menghasilkan probabilitas kecenderungan *redeem voucher* dari setiap *member*. Probabilitas ini kemudian digunakan sebagai dasar seleksi, di mana *member* dengan nilai probabilitas di atas *threshold* yang ditetapkan dianggap memiliki potensi tinggi untuk melakukan *redeem*, sehingga dimasukkan ke dalam daftar distribusi *voucher*.

3.3.2 Kendala yang Ditemukan

Selama proses pelaksanaan magang, terdapat beberapa kendala yang muncul, terutama terkait pemahaman struktur data dan proses teknis dalam pengolahan data *member clone*, yaitu:

1. Keterbatasan pemahaman terhadap struktur data yang dimiliki perusahaan serta konteks industri ritel yang menjadi landasan operasional perusahaan. Banyak istilah teknis dan istilah bisnis yang digunakan dalam dokumen

maupun diskusi bersama tim yang belum familiar, sehingga membutuhkan waktu untuk melakukan pemetaan istilah dan menyesuaikannya dengan konteks analisis yang dikerjakan.

2. Kompleksitas struktur *database* perusahaan. Sistem *database* perusahaan terdiri dari banyak tabel dengan relasi yang tidak selalu terlihat secara langsung. Data yang dibutuhkan untuk analisis *clone member* tersebar di berbagai tabel seperti tabel profil *member*, *device*, dan riwayat transaksi, sehingga proses identifikasi *clone member* memerlukan logika *join* yang tepat dan efisien. Tantangan bertambah ketika eksekusi *query* harus tetap optimal mengingat volume data yang besar.
3. Kesulitan dalam menentukan logika identifikasi *clone member*. Salah satu kendala teknis utama adalah pada penentuan kriteria untuk mengidentifikasi *clone member*. Tidak semua *member* dengan lebih dari satu *device* dapat langsung dikategorikan sebagai *clone*.

3.3.3 Solusi atas Kendala yang Ditemukan

Bagian ini berisi solusi atas kendala yang ditemukan selama proses kerja magang

1. Peningkatan pemahaman terhadap struktur data dan konteks industri. Untuk mengatasi keterbatasan pemahaman terhadap istilah dan struktur data perusahaan, dilakukan pendekatan bertahap melalui eksplorasi dokumentasi internal, membaca materi presentasi tim data, dan mempelajari definisi atribut langsung dari metadata tabel. Selain itu, melakukan diskusi aktif dengan supervisor dan rekan kerja menjadi strategi penting untuk mempercepat pemahaman konteks bisnis. Catatan pribadi juga dibuat untuk mendokumentasikan istilah yang sering muncul agar proses analisis di tahap selanjutnya menjadi lebih efisien.
2. Dalam Pemetaan *database* secara sistematis dan eksplorasi tabel secara bertahap. Untuk menghadapi kompleksitas struktur *database*, dilakukan pemetaan tabel inti yang relevan dengan proses identifikasi *clone member*, seperti tabel profil, tabel *device*, dan riwayat aktivitas *login*. Eksplorasi

dimulai dari tabel dengan jumlah atribut paling fundamental menggunakan *query* sederhana untuk memahami struktur dan relasinya. Setelah itu, dilakukan penyusunan diagram relasi sederhana agar memudahkan proses *join* data. Konsultasi langsung dengan tim data juga membantu memastikan bahwa tabel dan *logic* yang digunakan sudah sesuai dengan standar yang diterapkan perusahaan.

3. Penyusunan logika identifikasi *clone member* secara iteratif dan berbasis validasi. Untuk mengatasi kesulitan dalam menentukan kriteria yang tepat dalam identifikasi *clone member*, dilakukan pendekatan iteratif dengan membangun logika secara bertahap. Proses ini dimulai dari identifikasi berdasarkan satu atribut utama seperti secure_device_id, kemudian dikombinasikan dengan atribut tambahan seperti account_card, waktu login, dan pola aktivitas untuk meningkatkan akurasi. Setiap versi logika diuji terlebih dahulu dengan mengambil sampel data, kemudian divalidasi bersama tim untuk meminimalisir potensi *false positive* dan *false negative*. Pendekatan ini membantu menghasilkan *rule* yang lebih stabil dan dapat diterapkan secara konsisten pada data skala besar.