# BAB III

# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Pada program magang Kementerian Keuangan *batch* 1, mahasiswa magang ditempatkan ke dalam tim reguler pengelola informasi keuangan daerah. Mahasiswa diberikan kesempatan untuk membantu beberapa pekerjaan serta proyek yang berkaitan dengan pengelolaan dan pengolahan data keuangan daerah. Mahasiswa diberikan tugas untuk melakukan analisis trend terhadap data APBD pada jangka waktu tertentu. Selama kerja magang berlangsung, mahasiswa dibimbing dan diberikan arahan secara langsung oleh mentor dan diawasi oleh ketua tim reguler Pengelolaan Informasi Keuangan Daerah.


Gambar 3.1 Alur Koordinasi Kerja di Direktorat SIPT

Gambar 3.1, merupakan bagan yang menggambarkan posisi Tim Reguler Pengelolaan Informasi Keuangan Daerah dalam struktur instansi Kemenkeu, khususnya DJPK. Tim Reguler Pengelola Informasi Keuangan Daerah bertanggung jawab dalam mengelola seluruh informasi dan data terkait dengan APBD untuk dilakukan analisis lebih lanjut. Posisi mahasiswa magang sebagai *Data Analyst* mendorong untuk selalu aktif dan memberikan ide-ide baru untuk mendukung pekerjaan dalam tim reguler. Selama masa magang, mahasiswa mendapatkan tanggung jawab untuk merancang dan membangun sebuah *dashboard* interaktif

yang berfungsi sebagai alat bantu dalam menganalisis trend data keuangan daerah berdasarkan rentang tahun tertentu. Proyek ini bertujuan untuk menyajikan data secara visual dan informatif, sehingga dapat mempermudah para pemangku kepentingan dalam memahami perkembangan anggaran, realisasi belanja, serta perbandingan antar tahun. *Dashboard* ini dilengkapi dengan fitur filter tahun, kategori belanja, serta grafik interaktif yang memungkinkan pengguna untuk menelusuri data secara dinamis. Dalam proses pengembangannya, mahasiswa menggunakan *tools* seperti Microsoft Power BI dan Python, serta berkolaborasi dengan tim data untuk memastikan keakuratan dan relevansi informasi yang ditampilkan. Selain aspek teknis, mahasiswa juga turut serta dalam beberapa rapat koordinasi dan diskusi internal, yang menjadi pengalaman baru dan bermanfaat dalam memahami alur kerja, kebutuhan pengguna akhir, serta pentingnya komunikasi lintas divisi dalam menyelesaikan suatu proyek.

### 3.2 Tugas dan Uraian Kerja Magang

Mahasiswa telah menyelesaikan kegiatan kerja magang dengan total 640 jam kerja, sesuai dengan ketentuan kurikulum yang berlaku di Program Studi Sistem Informasi. Selama kerja magang berlangsung, mahasiswa diberikan tanggung jawab dalam proses *preprocessing data*, desain serta pengembangan *dashboard*. Seluruh pekerjaan yang dilakukan berfokus pada pengembangan *dashboard* interaktif agar memudahkan dalam penyajian data sesuai dengan kebutuhan pengguna dan para pemangku kepentingan. Berbagai tugas dilakukan secara bertahap dan berkesinambungan sepanjang periode magang, mulai dari melakukan *cleansing* data, desain *mockup dashboard*, menerapkan desain ke dalam *tools* visualisasi, hingga evaluasi dan perbaikan berdasarkan *feedback* pengguna. Rincian lengkap mengenai aktivitas yang dilakukan selama proses pengembangan aplikasi dan pelaksanaan magang dijelaskan bagian berikut.

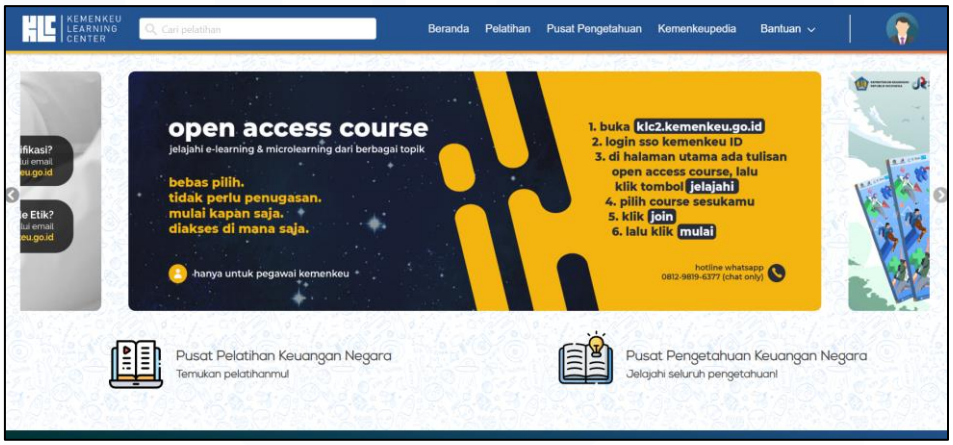
Tabel 3.1 Tabel Uraian Kerja Magang

| No. | Kegiatan | Minggu Ke- | Waktu Mulai | Waktu Selesai |
|---|---|---|---|---|
| 1 | Pembekalan materi TKD | 1-2 | 18 Februari 2025 | 28 Februari 2025 |
| 2 | Pembekalan materi HKPD | 1-2 | 18 Februari 2025 | 28 Februari 2025 |
| 3 | Pembekalan materi Pengelolaan Keuangan Daerah | 1-2 | 18 Februari 2025 | 28 Februari 2025 |
| 4 | Pembukaan program magang DJPK | 3 | 3 Maret 2025 | 3 Maret 2025 |
| Analisis Trend APBD | | | | |
| 5 | *Cleansing* data Realisasi APBD | 3-6 | 7 Maret 2025 | 28 Maret 2025 |
| 6 | Merancang visualisasi untuk data Realisasi APBD | 7-10 | 8 April 2025 | 30 April |
| 7 | Presentasi progres | 10 | 30 April 2025 | 30 April 2025 |
| 8 | *Troubleshoot shapefile map* Indonesia | 10-12 | 2 Mei 2025 | 23 Mei 2025 |
| 9 | *Update* Data APBD 2025 | 13-14 | 23 Mei 2025 | 28 Mei 2025 |
| 10 | Membangun *dashboard* Anggaran Realisasi APBD 2021-2025 | 15-17 | 2 Juni 2025 | 20 Juni 2025 |
| Deteksi Hubungan Semantik Berbasis NLP | | | | |
| 11 | Mengembangkan model klasifikasi kontekstual | 14-17 | 26 Mei 2025 | 16 Juni 2025 |
| 12 | *Deployment* | 16-17 | 28 Mei 2025 | 20 Juni 2025 |

### 3.2.1 Pembekalan melalui Kemenkeu *Learning Center* (KLC)

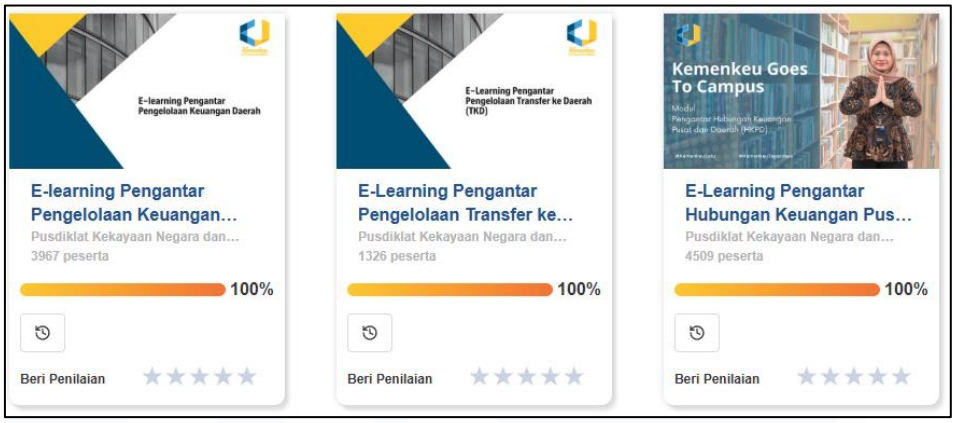
Kementerian Keuangan (Kemenkeu) memiliki komitmen yang kuat dalam mendukung pengembangan kompetensi mahasiswa magang melalui penyediaan program pelatihan berbasis digital yang dapat diakses secara fleksibel melalui browser, baik pada perangkat desktop maupun *mobile*.

Program ini bertujuan untuk membekali mahasiswa dengan pengetahuan yang selaras dengan tuntutan pekerjaan di lingkungan Direktorat Jenderal Perimbangan Keuangan (DJPK). Melalui pembekalan ini, mahasiswa magang tidak hanya memperoleh pemahaman mendalam mengenai berbagai aspek keuangan daerah, tetapi juga diarahkan untuk mengembangkan kemampuan analitis serta keterampilan pendukung lainnya dalam menghadapi dunia kerja. Program ini menjadi langkah strategis kementerian Keuangan dalam mempersiapkan mahasiswa agar mampu beradaptasi secara cepat dan berkontribusi secara optimal dalam konteks lingkungan kerja yang sesungguhnya.




Gambar 3.2 *Website* Kemenkeu *Learning Center*

Dalam pelaksanaannya, Kementerian Keuangan memanfaatkan Kemenkeu *Learning Center* (KLC) sebagai platform pembelajaran daring yang diperuntukkan bagi seluruh pegawai maupun mahasiswa magang. Melalui platform ini, peserta dapat mengakses berbagai materi pembelajaran dan program pelatihan yang dirancang secara khusus untuk mendukung peningkatan kompetensi di bidang keuangan negara. Struktur konten dalam KLC disusun dalam bentuk modul-modul interaktif, yang dilengkapi dengan tes formatif untuk mengukur tingkat pemahaman peserta. Setiap tes disertai dengan ambang batas kelulusan yang berfungsi sebagai indikator keberhasilan dan penguasaan materi.




Gambar 3.3 Materi Pembekalan

Secara khusus bagi mahasiswa magang, terdapat tiga modul utama yang diwajibkan untuk diselesaikan sebagai bagian dari persyaratan kelulusan program magang, yaitu Hubungan Keuangan Pusat dan Daerah (HKPD), Transfer ke Daerah (TKD), dan Pengelolaan Keuangan Daerah. Peserta magang diwajibkan untuk secara aktif mempelajari dan menyelesaikan seluruh materi yang tercakup dalam ketiga modul tersebut, dalam rangka membangun pemahaman yang komprehensif terkait konsep, kebijakan, serta implementasi pengelolaan keuangan daerah yang menjadi bagian dari ruang lingkup kerja DJPK. KLC memberikan fleksibilitas bagi pengguna untuk dapat mengakses materi kapan saja dan di mana saja, yang memungkinkan peserta magang untuk mengoptimalkan proses belajar, sekaligus memperkuat kesiapan dalam menghadapi tantangan praktis. Dengan demikian, program ini tidak hanya berperan sebagai sarana penguatan kapabilitas akademis, tetapi juga sebagai jembatan yang mengintegrasikan teori akademik dengan praktik kerja nyata dalam sektor keuangan publik.

### 3.2.2 Pembukaan Program Magang DJPK

Kegiatan orientasi lingkungan kerja merupakan langkah awal yang dilaksanakan oleh Kementerian Keuangan sebagai sarana pengenalan bagi para peserta magang dari program Magang Kementerian Keuangan. Kegiatan ini berfungsi sebagai pembukaan program Magang Kementerian

Keuangan *batch* 1, khususnya di lingkungan DJPK yang diselenggarakan pada tanggal 3 Maret 2025. Sekitar 30 mahasiswa mengikuti acara ini secara langsung (*offline*) maupun *online*. Tujuan dari kegiatan ini adalah untuk memberikan pengenalan menyeluruh mengenai instansi, meliputi struktur organisasi, tugas dan fungsi tiap unit, serta budaya kerja yang diterapkan di lingkungan instansi. Kegiatan ini dipandu secara langsung oleh Biro Sumber Daya Manusia (SDM) selaku penanggung jawab program magang mahasiswa. Setelah rangkaian kegiatan pembukaan program magang selesai, para mahasiswa magang akan diarahkan ke direktorat masing-masing dan diperkenalkan kepada seluruh tim di dalamnya. Selanjutnya, mahasiswa akan menempati tim reguler sesuai dengan penempatan yang telah ditentukan, dan mulai melakukan diskusi serta mendapatkan pemahaman lebih lanjut mengenai peran dan tanggung jawab posisi magang melalui bimbingan mentor.

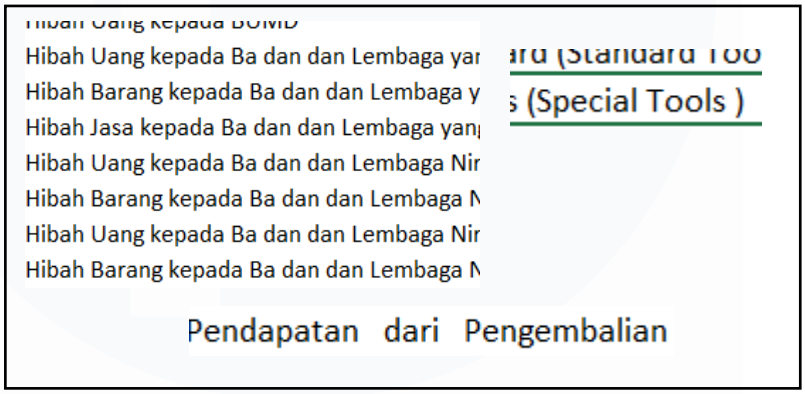
### 3.2.3 Analisis Trend APBD

dalam proyek ini, mahasiswa diminta untuk melakukan analisis trend data anggaran realisasi APBD. Salah satu *output* utama dari proyek ini adalah perancangan *dashboard* interaktif yang menyajikan data secara visual. *Dashboard* ini dirancang untuk memberikan gambaran yang komprehensif mengenai berbagai indikator keuangan yang dikelola oleh Kementerian Keuangan (Kemenkeu), seperti trend belanja dan pendapatan secara historis. Dalam proses perancangannya, mahasiswa terlibat dalam beberapa tahapan peting, mulai dari pengumpulan kebutuhan pengguna hingga visualisasi data yang akan ditampilkan pada *dashboard*. Salah satu aspek penting dalam pengembangan *dashboard* ini adalah bagaimana data yang telah diolah dapat disajikan secara informatif, intuitif, dan mudah dipahami oleh para pemangku kepentingan. Oleh karena itu, mahasiswa juga mempelajari prinsip-prinsip desain antarmuka pengguna yang relevan dalam pengembangan *dashboard* berbasis data. Selain itu, mahasiswa akan menggunakan berbagai *tools* analisis data dan visualisasi, seperti Power BI

dan Python untuk mengolah data dan menyajikan dalam format yang dinamis dan interaktif.

#### 3.2.3.1 *Cleansing* data Realisasi APBD

Setelah melakukan revisi nomenklatur akun objek, mahasiswa langsung melanjutkan ke proses *preprocessing* data khususnya data realisasi APBD 2021-2024. Mahasiswa melakukan eksplorasi data terlebih dahulu untuk mengetahui masalah pada data, sebelum menentukan apa saja yang harus dilakukan terhadap data tersebut.




Gambar 3.4 Masalah pada data

Gambar 3.4 merupakan beberapa contoh masalah yang terjadi pada data. Data dalam beberapa kolom menunjukkan banyaknya inkonsistensi dan kesalahan format penulisan. Salah satu permasalahan utama adalah adanya spasi yang tidak konsisten, seperti spasi ganda antar kata, spasi sebelum tanda baca (seperti sebelum tanda koma atau tanda kurung). Hal ini mengakibatkan satu jenis data dapat muncul dalam beberapa bentuk penulisan yang berbeda, seperti "Belanja Barang dan Jasa" bisa tercatat sebagai "Belanja Barang, dan Jasa". Inkonsistensi seperti ini menyulitkan proses agregasi data karena sistem akan menganggapnya sebagai kategori yang berbeda, padahal secara semantik memiliki makna yang sama. Masalah lainnya muncul dalam bentuk pemisahan frasa atau istilah teknis yang seharusnya ditulis utuh, seperti "Badan Usaha Milik Swasta" atau "Organisasi Kemasyarakatan yang

Berbadan Hukum Indonesia". Dalam beberapa data, frasa seperti ini terpotong secara tidak wajar, seperti kalimat "Ba dan dan Usaha Milik Swasta" sehingga tidak lagi mewakili entitas yang dimaksud dalam laporan keuangan.

Masalah signifikan lainnya adalah perbedaan format untuk data yang sebenarnya mengacu pada kodefikasi yang sama. Contohnya, data dengan kode awal yang sama (misalnya "826152") dapat memiliki deskripsi kategori yang sedikit berbeda karena variasi tanda baca atau kalimat. Hal sebaliknya juga terjadi, ketika data dengan deskripsi yang sama namun memiliki sedikit kesalahan pada penulisan kodefikasi. Hal tersebut menyebabkan sistem gagal mengelompokkan atau menggabungkan data yang seharusnya identik, sehingga frekuensi untuk satu jenis transaksi terpecah menjadi beberapa kategori yang berbeda. Seluruh masalah inkonsistensi ini berdampak secara langsung pada analisis trend anggaran dan realisasi. Ketika data kategori tidak ditulis secara konsisten, maka pola frekuensi kemunculan dari suatu jenis pendapatan atau belanja akan tampak tersebar. Akibatnya, trend dari suatu jenis transaksi tertentu tidak dapat dikenali secara utuh karena tersembunyi di balik beberapa versi penulisan yang berbeda. hal ini sangat mengganggu proses analisis, visualisasi, maupun pengambilan keputusan berbasis data, karena jumlah aktual dari suatu jenis aktivitas fiskal menjadi bias dan tidak representatif. Oleh karena itu, proses standarisasi format teks menjadi sangat penting untuk memastikan data dapat digunakan secara optimal.

Untuk mengatasi berbagai permasalahan tersebut, diperlukan proses pembersihan dan standarisasi data yang sistematis. Langkah-langkah ini bertujuan untuk menyelaraskan format penulisan, menyatukan data semantik, serta menghilangkan variasi yang tidak relevan agar data menjadi konsisten dan dapat diolah lebih lanjut secara akurat.

```
import pandas as pd


df = pd.read_excel("E:\\Kumpulan Tugas\\Magang\\Kemenkeu\\DJPK\\Raw Data\\Anggaran-Realisasi 2021-2024 (RAW).xlsx", sheet_na
df.head()
```

| | kodepemda | namapemda | standarutamamap | standarkelompokmap | standarjenismap | standarobjekmap | standarrincimap | standarsubrincimap | sum | tal |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | Provinsi Aceh | 4. PENDAPATAN DAERAH | 41. PENDAPATAN ASLI DAERAH (PAD) | 4101. Pajak Daerah | 410101. Pajak Kendaraan Bermotor (PKB) | 41010101. PKB-Mobil Penumpang-Sedan | 410101010001. PKB-Mobil Penumpang-Sedan-Pribadi | 1.010669e+10 | 2 |
| 1 | 1.0 | Provinsi Aceh | 4. PENDAPATAN DAERAH | 41. PENDAPATAN ASLI DAERAH (PAD) | 4101. Pajak Daerah | 410101. Pajak Kendaraan Bermotor (PKB) | 41010102. PKB-Mobil Penumpang-Jeep | 410101020001. PKB-Mobil Penumpang-Jeep-Pribadi | 5.480120e+10 | 2 |
| 2 | 1.0 | Provinsi Aceh | 4. PENDAPATAN DAERAH | 41. PENDAPATAN ASLI DAERAH (PAD) | 4101. Pajak Daerah | 410101. Pajak Kendaraan Bermotor (PKB) | 41010103. PKB-Mobil Penumpang-Minibus | 410101030002. PKB-Mobil Penumpang-Minibus-Umum | 3.105026e+11 | 2 |
| 3 | 1.0 | Provinsi Aceh | 4. PENDAPATAN DAERAH | 41. PENDAPATAN ASLI DAERAH (PAD) | 4101. Pajak Daerah | 410101. Pajak Kendaraan Bermotor (PKB) | 41010104. PKB-Mobil Bus-Microbus | 410101040002. PKB-Mobil Bus-Microbus-Umum | 2.889223e+09 | 2 |
| 4 | 1.0 | Provinsi Aceh | 4. PENDAPATAN DAERAH | 41. PENDAPATAN ASLI DAERAH (PAD) | 4101. Pajak Daerah | 410101. Pajak Kendaraan Bermotor (PKB) | 41010105. PKB-Mobil Bus-Bus | 410101050002. PKB-Mobil Bus-Bus-Umum | 7.158826e+08 | 2 |

Gambar 3.5 *Import dataset*

Proses *preprocessing* data dimulai dengan *import library* Pandas sebagai pembaca *file*, kemudian *import file dataset* yang akan dilakukan *cleansing* dan disimpan pada DataFrame bernama df. Selanjutnya, mahasiswa membuat duplikat DataFrame df dengan nama data. Tujuan menduplikat menjadi DataFrame baru adalah mahasiswa dapat memanggil kembali DataFrame baru sebagai reset tanpa mengimpor *dataset* dari awal kembali. Setelah DataFrame berhasil dibuat, maka mahasiswa lanjut ke proses *cleansing data*.

```
data['kodepemda'] = data['kodepemda'].apply(lambda x: str(f"{float(x):05.2f}"))
```

Gambar 3.6 Potongan kode untuk *reformat* kolom

Langkah awal yang dilakukan mahasiswa dalam *preprocessing* data adalah melakukan format ulang terhadap kolom kode daerah (kodepemda). Saat data tersebut diimpor ke dalam Python, ditemukan bahwa format kode mengalami perubahan otomatis oleh sistem, dari format *string number* seperti 01.00 menjadi format numerik desimal seperti 1.0. Hal ini umum terjadi ketika *file dataset* dibaca menggunakan *library* seperti Pandas, karena nilai 01.00 dikenali sebagai angka bukan sebagai teks. Akibatnya, nil di depan terhapus dan jumlah digit desimal menjadi tidak sesuai dengan standar kode daerah yang ditetapkan. Untuk mengatasi hal ini, mahasiswa menggunakan fungsi ".apply()" pada

kolom “kodepemda” untuk menerapkan transformasi dengan *lambda function*, sebagaimana terlihat pada gambar 3.6. Fungsi ini mengonversi setiap elemen dalam kolom menjadi tipe *float* terlebih dahulu, lalu diformat ulang menggunakan *string formatting* Python. Format ini memastikan bahwa nilai yang dihasilkan memiliki total lima karakter termasuk uda angka di belakang koma, dan menambahkan nol di depan jika diperlukan. Misalnya, kode 1.0 akan diubah menjadi 01.00, sesuai dengan format kode pemerintah daerah yang digunakan dalam *database*.

```
def clean_text(text):
    """Fungsi untuk membersihkan teks dari kesalahan format."""
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\s+([.,)])', r'\1', text)
    text = re.sub(r'([(])\s+', r'\1', text)
    return text.strip()

# Terapkan pembersihan hanya pada kolom "standar"
data["standar"] = data["standar"].apply(clean_text)

```

Gambar 3.7 Fungsi *Clean Text*

Gambar 3.7 merupakan fungsi bernama “clean_text” yang digunakan untuk membersihkan teks dari kesalahan format penulisan pada kolom tertentu. Fungsi ini dirancang untuk menghilangkan inkonsistensi seperti spasi ganda, spasi sebelum tanda baca, dan spasi yang tidak perlu di sekitar tanda kurung. Langkah pertama dalam fungsi ini adalah menghapus semua spasi berlebih dengan menggantinya menjadi satu spasi tunggal menggunakan ekspresi reguler “\s+”. Selanjutnya, fungsi akan menghapus spasi yang muncul sebelum tanda baca seperti koma dan titik, sehingga format penulisan menjadi lebih rapi dan konsisten. Setelah itu, fungsi juga memastikan tidak ada spasi yang salah tempat setelah tanda kurung buka atau sebelum tanda kurung tutup untuk menjaga konsistensi struktur kalimat. Di akhir fungsi, terdapat metode “strip()” yang digunakan untuk menghapus spasi di awal dan akhir teks. Fungsi ini kemudian diterapkan secara langsung pada

kolom “standar” dalam DataFrame bernama data. Penerapan proses pembersihan ini sangat penting untuk mencegah terjadinya pengelompokan data yang tidak akurat akibat variasi penulisan yang seharusnya merujuk pada kategori yang sama.

```
10 def correct_cut_words(text):
11     pattern = r"\b(\w{2,5})[\s,]+dan\b"
12 
13     matches = re.findall(pattern, text)
14     if matches:
15         for match in matches:
16             if match.lower().startswith("ba"):
17                 replacement = "Badan"
18             elif match.lower().startswith("berba"):
19                 replacement = "Berbadan"
20             else:
21                 replacement = match
22 
23             text = re.sub(rf"\b{match}[\s,]+dan\b", f"{replacement} dan", text)
24 
25     return text
```

Gambar 3.8 Fungsi *correct_cut_words*

Pada gambar 3.8 terdapat fungsi dengan nama “correct_cut_words(text)” merupakan fungsi yang digunakan untuk mengatasi masalah fragmentasi kata yang sering terjadi dalam pemrosesan teks bahasa Indonesia, terutama pada frasa normal seperti “Badan Usaha Milik Negara” yang terpotong menjadi "Berba dan Hukum Indonesia”. Masalah ini umumnya muncul akibat kesalahan dalam proses tokenisasi atau pemrosesan dokumen digital yang sempurna. Fungsi ini dimulai dengan mendefinisikan *pattern regex* yang secara sistematis mencari pola tertentu dalam teks. *Pattern* ini terdiri dari beberapa komponen penting di mana “\b” berfungsi sebagai *word boundary* untuk memastikan pencocokan dimulai dari batas kata yang tepat. Kemudian “(\w{2,5})” merupakan *capture group* yang menangkap dua hingga lima karakter alfanumerik yang merepresentasikan bagian kata yang tidak terpotong, “[\s,]+” mencocokkan satu atau lebih spasi atau koma yang memisahkan bagian kata dengan kata “dan”, serta “dan\b” mencocokkan kata “dan” sebagai kata terpisah yang diakhiri dengan *word boundary*.

Setelah *pattern* didefinisikan, fungsi menggunakan "re.findall(pattern, text)" untuk mencari semua kemunculan pola tersebut dalam teks *input*. Hasil pencarian berupa *list* yang berisi bagian-bagian kata yang tertangkap oleh *capture group*, bukan seluruh *match pattern*. Misalnya, jika *input* adalah "Ba dan Usaha Milik Negara", maka yang dikembalikan adalah ["Ba"], sedangkan untuk *input* "Berba dan Hukum Indonesia" akan mengembalikan ["Berba"]. Pendekatan ini memungkinkan fungsi untuk fokus pada bagian kata yang perlu diperbaiki tanpa harus memproses seluruh *pattern*. Proses selanjutnya melibatkan iterasi melalui setiap *match* yang ditemukan dengan menggunakan *loop for* yang hanya dijalankan jika ada *matches* yang ditemukan, mencegah *error* pada teks yang tidak memiliki *pattern* yang dicari. Untuk setiap *match*, fungsi menerapkan *match* dalam huruf kecil dimulai dengan "ba" menggunakan "match.lower().startswith("ba"), dan jika kondisi ini terpenuhi, maka kata tersebut diasumsikan sebagai "Badan" yang terpotong. Kondisi kedua memeriksa apakah *match* dimulai dengan "berba", dan jika benar maka diasumsikan sebagai "Berbadan" yang terpotong. Jika tidak ada kondisi yang terpenuhi, fungsi akan menggunakan kata asli sebagai *fallback mechanism* untuk mencegah perubahan yang tidak diinginkan. Terakhir, merupakan proses penggantian dalam teks yang mengganti seluruh *pattern* yang cocok dengan format yang benar. Penggunaan *raw f-string* membantu menghindari masalah dengan *escape character*, sementara *pattern* dinamis secara spesifik mencocokkan *match* yang ditemukan beserta spasi atau koma dan kata "dan" yang mengikutinya. *String* pengganti "f"{replacement} dan" memastikan format yang konsisten dengan spasi normal antara kata yang diperbaiki dengan kata "dan".

```python
# Fungsi utama untuk memformat teks
def format_text(text):
    original_text = text
    text = re.sub(r'\s+', ' ', text).strip()

    # Menghapus spasi setelah tanda "-"
    text = re.sub(r'(?<=\S)-\s+', '-', text)
    text = re.sub(r'\s+-(?=\S)', '-', text)

    # Memastikan koma sebelum "dan" hanya jika ada lebih dari dua kategori
    match = re.search(r'^(.*?),?\s*dan\s+)(.*?)$', text)
    if match:
        before_dan, connector, after_dan = match.groups()

        if ',' in before_dan and not any(p in after_dan for p in protected_phrases):
            text = f"{before_dan}, dan {after_dan}"
        else:
            text = f"{before_dan} dan {after_dan}"

    # Menghapus koma setelah kata-kata dalam exception_words
    for word in exception_words:
        pattern = rf'(?i)\b({word}),\b'
        text = re.sub(pattern, r'\1', text)

    # Koreksi kata-kata yang terpotong secara otomatis
    text = correct_cut_words(text)

    # Menghapus spasi di sekitar "/"
    text = re.sub(r'\s*/\s*', '/', text)

```

Gambar 3.9 Fungsi *format text*

Gambar 3.9 merupakan fungsi untuk melakukan format dan membersihkan teks yang dirancang secara komprehensif dengan tujuan menangani berbagai masalah *formatting* yang umum terjadi dalam pemrosesan dokumen. Fungsi ini terdiri dari beberapa tahapan pembersihan teks yang sistematis, mulai dari normalisasi spasi hingga koreksi kata-kata yang terpotong, sehingga menghasilkan teks yang lebih bersih dan konsisten untuk pemrosesan lebih lanjut. Fungsi ini dimulai dengan menyimpan teks asli sebagai referensi dengan “original_text = text” dan melakukan normalisasi spasi. Operasi ini mengganti semua jenis *whitespace* berupa spasi ganda, tab, *newline*, atau kombinasinya dengan satu spasi tunggal, sementara “.strip()” menghilangkan spasi di awal dan akhir teks. Selanjutnya adalah menghapus spasi setelah tanda baca tertentu menggunakan dua operasi *regex* berturut-turut. Operasi pertama menghilangkan spasi setelah tanda hubung (-) yang didahului oleh karakter non-spasi, dengan tujuan untuk memperbaiki kata-kata berimbuhan atau kata majemuk yang terpisah. Operasi selanjutnya

berfungsi untuk menghilangkan spasi sebelum tanda hubung yang diikuti karakter non-spasi dan menangani kasus seperti “kata- kata” menjadi “kata-kata”. Kedua operasi ini menggunakan *lookbehind* dan *lookahead assertion* untuk memastikan bahwa tanda hubung benar-benar berada antara kata yang bermakna. Bagian kompleks dari fungsi ini adalah penanganan koma sebelum kata “dan” dengan mempertimbangkan konteks kalimat. Fungsi ini bertujuan untuk mencari pola di mana koma muncul sebelum kata “dan”, kemudian menganalisis apakah koma tersebut diperlukan atau tidak. Jika *pattern* ditemukan, fungsi akan mengekstrak tiga bagian yaitu teks sebelum koma, *connector* berupa koma dan spasi, serta teks setelah kata “dan”. Sistem kemudian melakukan pengecekan apakah ada koma dalam bagian “before_dan” dan tidak ada karakter dalam daftar “protected_phrases” di bagian “after_dan”. Jika kondisi ini terpenuhi, maka koma sebelum “dan” akan dihapus karena dianggap tidak perlu dalam konteks enumerasi sederhana.

Tahap selanjutnya adalah menghapus koma setelah kata-kata dalam daftar “exception_words” menggunakan *loop* yang mengiterasi setiap kata dalam daftar tersebut. Untuk setiap kata, fungsi membuat *pattern* yang menggunakan *flag case_sensitive* dan *word boundaries* untuk memastikan pencocokan yang tepat. *Pattern* ini kemudian digunakan untuk menghilangkan kota yang tidak diperlukan setelah kata-kata tertentu yang mungkin merupakan bagian dari frasa yang lebih besar atau memiliki fungsi khusus dalam kalimat. Salah satu fitur penting dalam fungsi ini adalah pemanggilan fungsi “correct_cut_words(text)” yang secara khusus menangani masalah kata-kata yang terpotong, terutama dalam konteks istilah formal seperti “Badan” dan “Berbadan” yang sering muncul dalam dokumen resmi. Fungsi ini bekerja dengan mengidentifikasi pola fragmentasi kata yang umum terjadi dalam pemrosesan dokumen dan memperbaikinya secara otomatis untuk

memastikan bahwa istilah-istilah penting tetap utuh dan dapat diinterpretasi dengan benar. Selanjutnya adalah normalisasi spasi di sekitar tanda garis miring dengan mengganti pola "spasi-slash-spasi" dengan menghilangkan spasinya. Operasi ini penting dilakukan untuk konsistensi dalam penulisan kata yang menggunakan tanda garis miring, serta memastikan bahwa tidak ada spasi yang tidak perlu di sekitar tanda tersebut. Setelah seluruh fungsi dijalankan, maka langsung diterapkan untuk kolom-kolom sesuai dengan kebutuhan/permasalahan yang ada.

```
kamus_wilayah = {
    "Kab. Aceh Barat": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Barat Daya": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Besar": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Jaya": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Selatan": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Singkil": ("Provinsi Aceh", "Sumatera"),
    "Kab. Aceh Tamiang": ("Provinsi Aceh", "Sumatera"),
```

Gambar 3.10 Potongan kode untuk pengelompokan daerah

Setelah melalui tahap *data cleansing* dan *data formatting* terhadap *dataset* anggaran realisasi APBD, mahasiswa menghadapi struktur data awal di mana informasi administratif seperti nama kabupaten/kota dan provinsi masih tercampur dalam satu kolom. Hal ini menghambat proses analisis lanjutan, terutama saat diperlukan visualisasi geografis yang terstruktur dan pengelompokan wilayah berdasarkan pulau. Untuk memenuhi permintaan dari mentor, yakni membuat visualisasi berbasis peta yang mampu mengelompokkan daerah berdasarkan pulau, mahasiswa melakukan proses pemecahan data wilayah menjadi tiga komponen, yaitu nama kabupaten/kota, nama provinsi, dan nama pulau. Proses ini dimulai dengan menyusun sebuah struktur kamus Python dengan nama "kamus_wilayah", seperti pada gambar 3.10. kamus ini berisi pasangan *key-value*, di mana *key* berupa nama kabupaten/kota, sedangkan *value* merupakan *tuple* yang berisi dua elemen, yaitu

nama provinsi dan nama pulau. Contohnya adalah "Kab. Aceh Barat": ("Provinsi Aceh", "Sumatera") menunjukkan bahwa Aceh Barat termasuk dalam Provinsi Aceh yang berada di Pulau Sumatera. Dengan struktur ini, setiap daerah dapat dengan mudah ditelusuri dan dikategorikan ke dalam wilayah geografis yang lebih besar berdasarkan letak pulaunya.

Pendekatan ini penting dilakukan, karena Power BI tidak secara otomatis mengenali atribut geografis seperti pulau atau provinsi jika informasi tersebut tidak disediakan dalam kolom terpisah. oleh karena itu, pemisahan dan klasifikasi wilayah ini menjadi kunci untuk membangun *map visual* berbasis *Shape Map*, yang mengandalkan atribut yang jelas untuk membuat segmentasi warna, filter, atau hierarki visual. Selain itu, struktur ini juga mendukung proses integrasi data dari berbagai sumber, memperkuat validitas analisis spasial, dan memfasilitasi pengguna fungsi pemetaan dan pengelompokan dalam proses visualisasi.

#### 3.2.3.2 Merancang visualisasi data Realisasi APBD

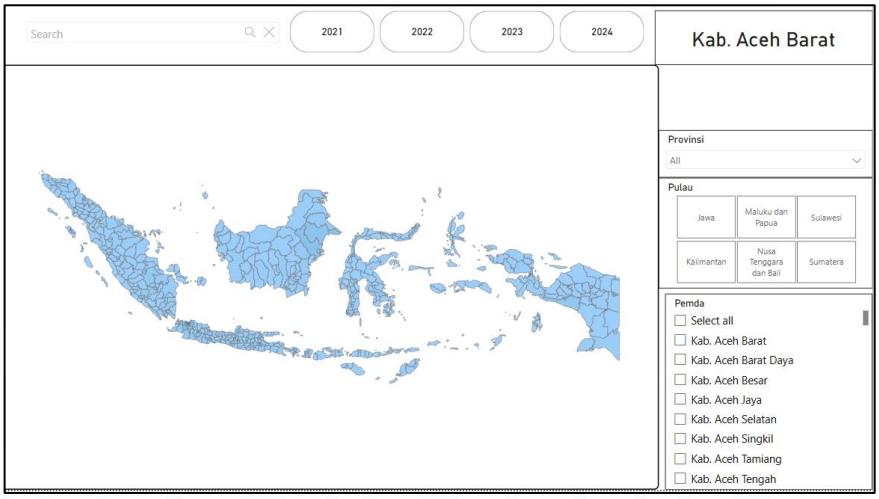
setelah menyelesaikan proses pembersihan dan standarisasi data, mahasiswa melanjutkan ke tahap berikutnya yaitu pembuatan visualisasi sebagai bentuk eksplorasi awal terhadap data realisasi APBD yang telah disiapkan. Visualisasi ini bertujuan untuk memahami pola, distribusi, serta potensi informasi yang dapat ditarik dari data, baik dalam konteks trend waktu, komposisi anggaran, maupun perbandingan antar kategori belanja dan pendapatan. Untuk mendukung proses ini, mahasiswa menggunakan Microsoft Power BI yang merupakan sebuah *business intelligence tools* yang memungkinkan pengguna untuk membuat *dashboard* interaktif dan representasi visual data yang informatif. Melalui Microsoft Power BI, mahasiswa dapat mengolah data ke dalam bentuk grafik seperti *bar chart*, *pie chart*, hingga tabel interaktif

yang membantu dalam mengidentifikasi anomali data, kecenderungan belanja, dan proporsi realisasi anggaran dari berbagai sektor. Proses eksplorasi visual ini menjadi langkah penting sebelum masuk ke tahap analisis yang lebih dalam, karena membantu dalam mengenali area-area strategis yang memerlukan perhatian atau yang memiliki kontribusi signifikan terhadap struktur APBD secara keseluruhan. Dalam proses pembuatan visualisasi, mahasiswa menyiapkan data yang sudah bersih dan terstruktur dalam format Excel untuk diimpor ke power BI melalui menu "Get Data".

```
= Table.ReplaceValue(#"Replaced Value3","Gunung Kidul","Gunungkidul",Replacer.ReplaceText,{"namapemda"})
```

Gambar 3.11 Transformasi Teks pada Kolom *namapemda*

Setelah data berhasil dimuat, mahasiswa melanjutkan ke tahap penyesuaian beberapa nama daerah untuk memastikan konsistensi nama daerah menggunakan Power Query Editor yang terintegrasi dalam Power BI.




Gambar 3.12 Visualisasi peta Indonesia

Gambar 3.12 merupakan visualisasi peta yang merepresentasikan data Anggaran Realisasi APBD 2021-2024 di seluruh wilayah di Indonesia, yang dibangun menggunakan fitur

*Shape Map* pada aplikasi Microsotft Power BI dengan memanfaatkan *file* TopoJSON yang diperoleh dari repositori GitHub sebagai *foundation* geografis. Peta ini mencakup seluruh 529 kabupaten dan kota yang tersebar di 38 provinsi di Indonesia, yang divisualisasikan dalam bentuk area poligon berwarna biru muda yang menggambarkan batas administratif masing-masing daerah dengan akurasi tinggi. Penggunaan TopoJSON sebagai sumber geometri geografis memberikan keunggulan signifikan dalam hal ukuran *file* yang lebih *compact* dibandingkan format GeoJSON konvensional, serta kemampuan *encoding* topologi yang memungkinkan representasi batas-batas administratif yang presisi dengan efisiensi *storage* yang optimal.

Dalam proses pembangunan visualisasi map tersebut, mahasiswa mengintegrasikan tiga kolom utama dari *dataset* ke dalam struktur peta secara terstruktur serta konfigurasi integrasi antara *file* TopoJson dan data tabular melalui proses *geographic matching* yang kompleks. Tahap awal implementasi melibatkan impor *file* TopoJSON Indonesia dari GitHub *repository*, kemudian dilakukan proses *parsing* dan *validation* untuk memastikan integritas struktur geometri dan konsistensi *attribute naming*. Power BI secara otomatis melakukan *topology decoding* dari format TopoJSON menjadi representasi geografis yang dapat ditampilkan, dengan mempertahankan relasi topologi antar *boundaries* untuk optimasi *rendering performance*. Untuk mengatasi inkonsistensi format penamaan antara data APBD dan *attribute fields* dalam TopoJSON, mahasiswa mengimplementasikan formula *Data Analysis Expressions* (DAX) untuk *data cleansing* dan *standarization*.

```
LokasiBersih =
IF(
    [Namapemda] = "Kab. Gorontalo",
    [Namapemda],
    SUBSTITUTE(
        SUBSTITUTE(
            [Namapemda],
            "Kab. ", ""
        ),
        "Provinsi ", ""
    )
)
```

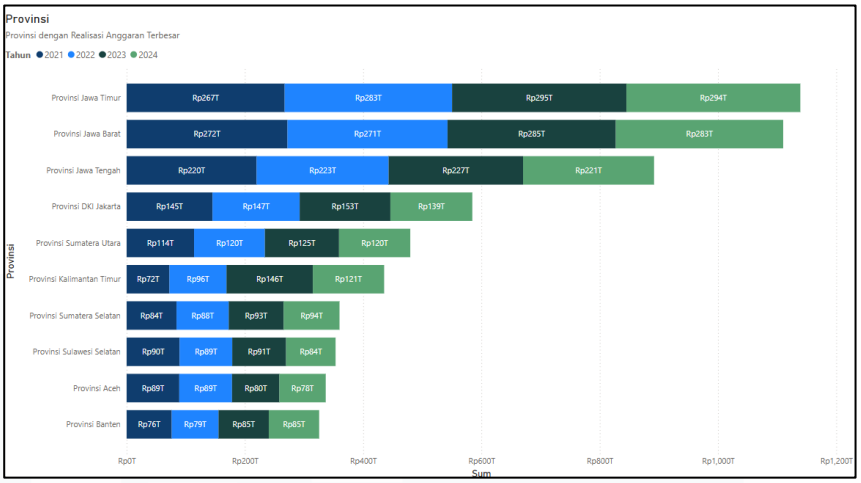
Gambar 3.13 DAX untuk normalisasi nama pemda

Gambar 3.13 merupakan formula DAX yang digunakan untuk menghilangkan imbuhan seperti “Kab.” Dan “Provinsi” dari nama daerah pada data, dengan pengecualian untuk “Kab. Gorontalo” yang dipertahankan formatnya untuk menghindari konflik dengan “Kota Gorontalo”. Implementasi DAX ini menggunakan *nested SUBSTITUTE functions* untuk melakukan *multiple string replacement* secara berurutan. Sementara *conditional logic IF statement* memastikan penanganan yang cepat untuk *edge cases* yang memerlukan perlakukan khusus, sehingga menghasilkan *clean geographic identifiers* untuk mencocokkan data secara akurat dengan TopoJSON *properties*.

Terdapat tiga kolom utama yang digunakan dalam pembangunan visualisasi map, yaitu nama daerah, nilai realisasi, dan tahun anggaran. Kolom pertama adalah nama daerah/nama pemda, yang berfungsi sebagai *geographic identifier* atau kunci penghubung utama antara data dengan entitas geografis dalam peta serta *properties object* dalam *file* TopoJSON. Kolom ini harus mengikuti standarisasi nomenklatur administratif sesuai dengan peraturan Menteri Dalam Negeri tentang kode dan Data Wilayah Administrasi Pemerintahan. Proses standarisasi ini mencakup penanganan *exception cases* seperti daerah dengan perubahan nama, pemekaran wilayah, atau status ekonomi khusus, dengan tingkat akurasi

*geocoding* yang tinggi untuk memastikan setiap entitas geografis ter petakan dengan benar terhadap *polygon geometry* yang sesuai dalam struktur TopoJSON. Kolom kedua adalah nilai realisasi, yang berisi informasi kuantitatif berupa angka yang merepresentasikan jumlah dana APBD yang telah terealisasi, baik secara total maupun per kategori belanja seperti belanja operasi, belanja modal, dan lainnya. Nilai ini dinyatakan dalam satuan rupiah dengan jangkauan yang bervariasi, tergantung pada skala dan kapasitas fiskal masing-masing daerah. Data realisasi ini divisualisasikan melalui gradasi warna pada peta atau yang bisa disebut *sequential color scale*, di mana intensitas warna biru yang semakin pekat menunjukkan tingginya nilai realisasi anggaran di suatu wilayah, sementara warna yang lebih terang mengindikasikan realisasi yang relatif rendah. Dengan penggunaan sistem pewarnaan ini tidak hanya memudahkan interpretasi visual, tetapi juga memungkinkan identifikasi cepat terhadap pola spasial seperti *clustering* daerah dengan performa serupa atau perbedaan regional dalam pengelolaan keuangan daerah. Kolom terakhir adalah tahun anggaran yang berisi periode waktu tertentu. Kolom ini mencakup rentang tahun anggaran dari 2021 hingga 2024 yang disesuaikan dengan ketersediaan data dan mengikuti siklus tahun anggaran pemerintah Indonesia. Penggunaan kolom ini memungkinkan pengguna untuk melakukan analisis *time series* yang spesifik, seperti perbandingan antar tahun untuk melihat pertumbuhan atau penurunan realisasi anggaran serta identifikasi pola musiman dalam realisasi anggaran. Mahasiswa memanfaatkan fitur *slicer* di Power BI untuk memudahkan navigasi data dari waktu ke waktu.

Selain visualisasi peta, mahasiswa juga membuat berbagai bentuk visual lainnya untuk mendukung analisis data secara menyeluruh. Beberapa visual yang digunakan meliputi *bar chart*, *line chart*, dan *treemap*.

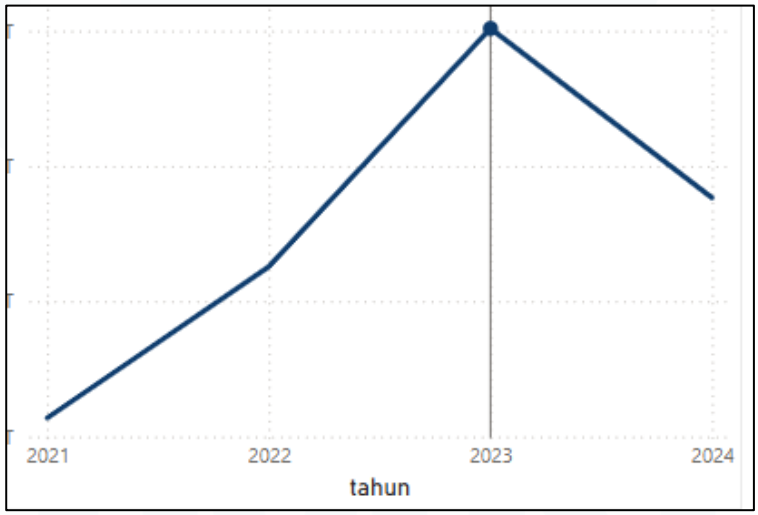



Gambar 3.14 visualisasi *stacked bar chart*

Gambar 3.14 merupakan visualisasi *stacked bar chart* horizontal yang menggambarkan seluruh provinsi dengan realisasi anggaran APBD terbesar di Indonesia selama periode 2021 hingga 2024. Visualisasi ini merupakan representasi grafis yang strategis dalam konteks analisis keuangan publik, karena APBD merupakan instrumen utama pemerintah daerah dalam melaksanakan fungsi pelayanan publik, pembangunan infrastruktur, dan stimulasi ekonomi regional. Visualisasi ini dibuat menggunakan fitur *stacked bar chart* yang terdapat di Power BI, yang memungkinkan penggabungan nilai dari beberapa kategori (tahun) dalam satu sumbu utama, sehingga pengguna dapat membandingkan total kumulatif sekaligus distribusi nilai per tahun dalam satu representasi visual. Data diorganisir dengan struktur setiap baris mewakili kombinasi provinsi dan tahun, dengan kolom tersendiri untuk nilai realisasi anggaran. Dalam Power BI, visualisasi dibuat dengan meletakkan *field* "Provinsi" ke area *Axis* (Sumbu Y), *field* "Tahun" ke area *Legend* untuk membedakan warna setiap segmen, dan *field* "Nilai" ke area *Values* (Sumbu X). Power BI secara otomatis akan menghitung agregasi data menggunakan fungsi SUM untuk menjumlahkan total realisasi per provinsi dan per tahun. Untuk

memastikan urutan provinsi berdasarkan realisasi terbesar, dilakukan *sorting* secara *descending*.

Skema warna yang digunakan untuk membedakan masing-masing tahun adalah biru tua (2021), biru muda (2022), hijau tua (2023), dan hijau muda (2024). Pemilihan warna ini dilakukan melalui menu Format Visual di panel *Properties*, di mana setiap tahun dapat diberikan warna khusus dengan memilih "Data Colors" dan memilih warna untuk setiap kategori tahun. Gradiasi warna dari biru ke hijau digunakan karena memberikan kontras yang jelas namun tetap harmonis secara visual, serta mudah dibedakan oleh pengguna dengan gangguan penglihatan warna.



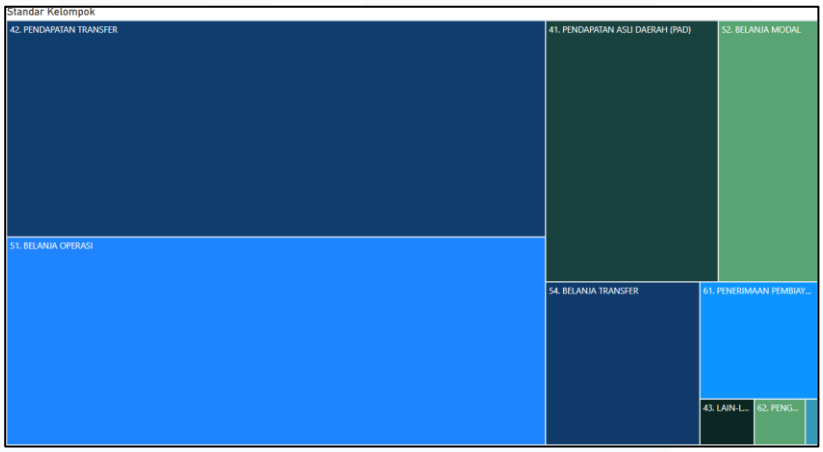

Gambar 3.15 Visualisasi *line chart*

Gambar 3.15 menampilkan visualisasi berupa *line chart* yang menggambarkan trend total realisasi APBD secara nasional selama periode tahun 2021 hingga 2024. Grafik ini dibangun dengan tujuan untuk memberikan representasi visual yang jelas dan ringkas terhadap dinamika perubahan jumlah relasi anggaran di seluruh kabupaten dan kota di Indonesia dari tahun ke tahun. Sumbu X menunjukkan tahun anggaran mulai dari 2021 hingga 2024, sedangkan sumbu Y menampilkan total nilai realisasi anggaran yang

telah dijumlahkan dari seluruh daerah, dengan satuan yang digunakan adalah triliun rupiah. Nilai-nilai ini ditampilkan dalam format ter standarisasi agar mudah dibaca dan dibandingkan antar waktu. Dari visualisasi tersebut, terlihat adanya pola pertumbuhan yang cukup konsisten selama tiga tahun pertama, dimulai dari tahun 2021 hingga mencapai puncaknya pada tahun 2023. Selama periode tersebut, nilai realisasi APBD secara nasional terus menunjukkan peningkatan, mencerminkan adanya dorongan belanja yang kuat. Trend kenaikan ini dapat dikaitkan dengan berbagai inisiatif pemerintah daerah dalam mempercepat pemulihan ekonomi pasca pandemi, termasuk peningkatan investasi di sektor infrastruktur dan pelayanan publik sebagai bagian dari pelaksanaan program-program prioritas daerah. Namun, memasuki tahun 2024, grafik menunjukkan adanya penurunan dalam total realisasi anggaran. Walaupun nilainya masih berada pada tingkat yang relatif tinggi dibandingkan dua tahun awal, penurunan ini menandai potensi perubahan kebijakan atau dinamika baru dalam pengelolaan fiskal daerah.

Visualisasi ini dibangun dari hasil agregasi nilai realisasi berdasarkan kolom tahun pada tabel data yang telah diproses sebelumnya melalui Power Query Editor. Proses ini melibatkan penggabungan data dari berbagai entitas daerah, penyamaan format, serta konversi angka ke dalam bentuk numerik berskala triliun untuk memastikan konsistensi dan keakuratan dalam Power BI. Komponen *line chart* dipilih karena sifatnya yang sangat efektif dalam menggambarkan perubahan trend waktu, serta kemampuannya untuk menampilkan arah pertumbuhan dan titik balik (*turning point*) secara jelas. Desain visual yang sederhana mampu menyampaikan pesan kompleks mengenai pergerakan anggaran nasional secara makro. Visualisasi ini memiliki peran penting dalam melihat pola kenaikan dan penurunan dari tahun ke tahun dan melakukan

investigasi penyebab kenaikan dan evaluasi terhadap penurunan yang terjadi. Selain itu, visualisasi ini juga berguna bagi pihak-pihak yang membutuhkan gambaran umum atas anggaran dan realisasi APBD, termasuk auditor, hingga masyarakat luas dengan mudah.




Gambar 3.16 Visualisasi *Treemap*

Gambar 3.16 merupakan visualisasi *Treemap* yang merepresentasikan struktur anggaran daerah yang diorganisasikan berdasarkan klasifikasi standar kelompok, di mana setiap kotak dalam visualisasi menggambarkan satu kelompok anggaran dengan ukuran yang proporsional terhadap kontribusi nilai kelompok tersebut dalam keseluruhan struktur anggaran realisasi. Visualisasi ini memungkinkan pengguna dengan cepat mengidentifikasi komponen-komponen dominan dalam anggaran melalui perbandingan ukuran relatif antar kotak, sementara perbedaan warna digunakan sebagai pembeda visual antar kelompok tanpa mengandung makna tambahan. Keunggulan dari penggunaan visualisasi *TreeMap* terletak pada kemampuannya untuk menyajikan informasi hierarkis dan proporsional dalam satu tampilan yang komprehensif. Dibandingkan dengan tabel numerik tradisional yang memerlukan waktu lebih lama untuk dipahami dan dianalisis, *TreeMap* memungkinkan pemahaman instan tentang struktur dan proporsi relatif setiap komponen anggaran melalui persepsi visual

yang intuitif. Penggunaan area sebagai representasi nilai memberikan keunggulan psikologis dalam pemahaman, karena secara natural *user* akan lebih mudah memproses informasi spasial dibandingkan dengan deretan angka yang abstrak.

Efisiensi komunikasi yang dihasilkan oleh visualisasi *TreeMap* sangat signifikan dalam konteks pelaporan keuangan daerah yang melibatkan *stakeholder* dengan latar belakang yang beragam. Seluruh kalangan dapat dengan mudah memahami pesan utama dari struktur anggaran tanpa harus memiliki keahlian teknis yang mendalam dalam analisis keuangan. Visualisasi ini memungkinkan identifikasi cepat terhadap area-area yang memerlukan perhatian khusus, seperti ketergantungan berlebihan pada sumber pendapatan tertentu atau ketidakseimbangan dalam alokasi belanja, yang mungkin tidak terlihat jelas dalam format tabel konvensional. Kemampuan *TreeMap* untuk menampilkan *multiple level of detail* dalam satu tampilan merupakan keunggulan strategis yang sangat berharga dalam analisis anggaran yang kompleks. *User* dapat sekaligus melihat gambaran besar dari struktur anggaran melalui perbandingan kota-kotak besar, sambil tetap dapat mengidentifikasi komponen-komponen kecil yang mungkin memiliki signifikansi strategis tertentu fleksibilitas visual ini memungkinkan analisis yang bersifat *top-down* maupun *bottom-down* dalam satu *interface* yang terintegrasi. Aspek interaktif yang dapat dikembangkan dari visualisasi *TreeMap* memberikan dimensi tambahan dalam eksplorasi data anggaran realisasi APBD. Kemampuan untuk melakukan *drill-down* ke level yang lebih detail, *filtering* berdasarkan kriteria tertentu, dan perbandingan antar periode waktu dapat diimplementasikan dengan lebih intuitif dibandingkan dengan tabel konvensional. Interaktivitas ini memungkinkan pengguna untuk melakukan analisis *ad-hoc* dan

eksplorasi data yang lebih mendalam sesuai dengan kebutuhan spesifik.

Dari perspektif transparansi dan akuntabilitas publik, visualisasi *TreeMap* memberikan kontribusi yang sangat berharga dan mewujudkan *good governance* dalam pengelolaan keuangan daerah. Kompleksitas informasi anggaran yang biasanya sulit dipahami oleh masyarakat umum dapat disederhanakan tanpa mengurangi akurasi dan kelengkapan informasi. Hal ini mendukung prinsip transparansi fiskal yang menjadi salah satu pilar utama dalam pengelolaan keuangan yang akuntabel. Efisiensi dalam proses pengambilan keputusan juga merupakan keunggulan yang tidak dapat diabaikan. Visualisasi *TreeMap* memungkinkan pengambil keputusan untuk dengan cepat mengidentifikasi area-area yang memerlukan intervensi kebijakan, melakukan perbandingan skenario alternatif, dan mengkomunikasikan keputusan kepada *stakeholder* dengan lebih efektif. Kecepatan dalam pemahaman informasi ini sangat krusial dalam konteks anggaran yang sering kali memiliki *deadline* yang ketat dan melibatkan negosiasi yang kompleks antar berbagai pihak.

Pendapatan Transfer menempati posisi sebagai kotak terbesar dalam visualisasi, mengindikasikan bahwa kelompok ini merupakan komponen utama dalam struktur pendapatan daerah. Dominasi yang signifikan dari Pendapatan Transfer ini mencerminkan realitas sistem desentralisasi fiskal di Indonesia, di mana pemerintah daerah masih bergantung pada mekanisme transfer dana dari pemerintah pusat. Ketergantungan fiskal yang tinggi menunjukkan bahwa kapasitas fiskal mandiri daerah belum mencapai tingkat yang memadai untuk membiayai kebutuhan pengeluaran daerah secara independen. Fenomena ini umumnya terjadi karena adanya kesenjangan vertikal antara kewenangan

pengeluaran yang telah didesentralisasikan kepada daerah dengan kapasitas pendapatan yang dapat dihimpun oleh daerah dari sumber-sumber lokal. Belanja Operasi menempati posisi kotak terbesar kedua, menandakan bahwa sebagian besar alokasi anggaran daerah diperuntukkan bagi pembiayaan kegiatan operasional rutin pemerintahan. Komponen ini mencakup berbagai jenis belanja seperti belanja pegawai yang meliputi gaji, tunjangan, dan honorarium seluruh aparatur sipil negara di daerah, serta belanja barang dan jasa yang mencakup pengadaan kebutuhan operasional kantor, pemeliharaan rutin, dan berbagai kegiatan pelayanan publik sehari-hari. Dominasi belanja operasi ini mencerminkan karakteristik umum pengelolaan keuangan daerah yang masih sangat berorientasi pada pembiayaan aktivitas rutin dibandingkan investasi jangka panjang. Kondisi ini dapat diinterpretasikan sebagai indikasi bahwa daerah masih dalam tahap konsolidasi kelembagaan dan operasional, namun di sisi lain juga menunjukkan perlunya evaluasi efisiensi dalam pengelolaan belanja rutin agar dapat mengalokasikan lebih banyak sumber daya untuk kegiatan pembangunan.

Pendapatan Asli Daerah (PAD) menunjukkan ukuran kotak yang berada pada kategori menengah, mengindikasikan bahwa upaya daerah untuk menggalang pendapatan dari sumber-sumber lokal telah menunjukkan hasil yang cukup berarti, namun belum mencapai tingkat dominan dalam struktur pendapatan keseluruhan. Komponen Pendapatan Asli Daerah ini mencakup berbagai sumber seperti pajak daerah yang meliputi pajak kendaraan bermotor, pajak air permukaan, dan berbagai jenis pajak daerah lainnya sesuai dengan kewenangan yang dimiliki masing-masing tingkatan pemerintah daerah. Selain itu, retribusi daerah yang dikenakan atas pelayanan publik tertentu, hasil pengelolaan kekayaan daerah yang dipisahkan, dan lain-lain pendapatan asli daerah yang sah juga berkontribusi dalam komponen ini. Proporsi PAD yang masih lebih

kecil dibandingkan pendapatan transfer mengindikasikan bahwa potensi sumber-sumber pendapatan lokal masih dapat dioptimalkan lebih lanjut melalui berbagai strategi seperti intensifikasi dan ekstensifikasi pajak dan retribusi, pengembangan aset produktif daerah, serta peningkatan iklim investasi untuk memperluas basis pajak. Selanjutnya, Belanja Modal memperlihatkan ukuran kotak yang juga berada pada kategori menengah, menunjukkan bahwa alokasi untuk investasi pembangunan infrastruktur dan pengadaan aset-aset produktif mendapat perhatian yang cukup signifikan dalam struktur anggaran daerah. Belanja Modal ini mencakup berbagai jenis investasi seperti pembangunan dan rehabilitasi infrastruktur jalan, jembatan, dan jaringan irigasi, pembangunan gedung-gedung pelayanan publik seperti sekolah, puskesmas, dan kantor pemerintahan, serta pengadaan peralatan dan mesin yang diperlukan untuk mendukung kelancaran operasional pemerintahan dan pelayanan publik. Meskipun proporsinya cukup signifikan, namun fakta bahwa belanja modal tidak sebesar Belanja Operasi yang menunjukkan adanya ketidakseimbangan dalam alokasi anggaran yang lebih condong pada pembiayaan aktivitas rutin dibandingkan investasi jangka panjang. Kondisi ini perlu mendapat perhatian khusus mengingat investasi modal merupakan faktor kunci dalam mendorong pertumbuhan ekonomi daerah, meningkatkan daya saing, dan menciptakan *multiplier effect* yang positif bagi perekonomian lokal.

Belanja Transfer menampilkan ukuran kotak yang berada pada kategori kecil hingga menengah, mencerminkan adanya mekanisme retribusi keuangan dari pemerintah daerah kepada tingkatan pemerintahan yang lebih rendah atau kepada pihak-pihak tertentu yang memerlukan bantuan keuangan. Komponen ini meliputi transfer dana ke pemerintah desa dalam bentuk Alokasi Dana Desa, bantuan keuangan kepada partai politik sesuai dengan

ketentuan peraturan perundang-undangan, bantuan keuangan untuk kegiatan kemasyarakatan, serta berbagai bentuk transfer lainnya yang sah menurut regulasi yang berlaku. Proporsi yang relatif lebih kecil dibandingkan belanja operasi dan modal menunjukkan bahwa fungsi redistribusi keuangan ini penting, namun tidak menjadi komponen dominan dalam struktur pengeluaran daerah. Kemudian, Penerimaan Pembiayaan memperlihatkan ukuran kotak yang berada pada kategori kecil hingga menengah, mengindikasikan bahwa sumber-sumber pembiayaan di luar pendapatan rutin memiliki kontribusi yang terbatas namun tetap relevan dalam struktur keuangan daerah. komponen ini dapat mencakup berbagai sumber, seperti Sisa Lebih Perhitungan Anggaran tahun sebelumnya yang mencerminkan kemampuan daerah dalam mengelola anggaran secara efisien sehingga menghasilkan surplus, penerimaan dari hasil penjualan aset daerah yang sudah tidak produktif atau tidak diperlukan lagi, penerimaan pinjaman daerah untuk pembiayaan investasi infrastruktur tertentu yang telah mendapat persetujuan dari otoritas yang berwenang, serta sumber-sumber pembiayaan lain yang sah. Proporsi yang tidak dominan dari komponen ini menunjukkan bahwa daerah tidak terlalu bergantung pada sumber-sumber pembiayaan yang bersifat temporer atau yang berpotensi menimbulkan kewajiban di masa mendatang.

Lain-lain Pendapatan Daerah yang Sah menunjukkan ukuran kotak yang relatif kecil, mengindikasikan bahwa sumber-sumber pendapatan di luar PAD dan pendapatan transfer memberikan kontribusi yang terbatas dalam struktur pendapatan daerah secara keseluruhan. Kategori ini dapat mencakup berbagai jenis pendapatan seperti hibah dari pihak ketiga yang sah, pendapatan dari dengan dan sanksi administrasi, hasil dari kegiatan-kegiatan insidental yang dilakukan pemerintah daerah, serta berbagai sumber pendapatan lain yang diakui secara hukum namun tidak dapat dikategorikan ke

dalam PAD atau pendapatan transfer. Kontribusi yang kecil dari komponen ini menunjukkan bahwa sumber-sumber pendapatan alternatif ini bukan merupakan andalan utama dalam struktur pembiayaan daerah, namun tetap memiliki peran sebagai pelengkap dalam diversifikasi sumber pendapatan. Terakhir, Pengeluaran Pembiayaan memperlihatkan ukuran kotak yang paling kecil dalam keseluruhan visualisasi, mengindikasikan bahwa komponen ini memiliki proporsi yang sangat terbatas dan tidak signifikan dalam struktur anggaran daerah. Pengeluaran Pembiayaan ini dapat mencakup berbagai jenis seperti pembayaran pokok pinjaman daerah yang jatuh tempo, penyertaan modal pemerintah daerah pada Badan Usaha Milik Daerah atau entitas bisnis lainnya yang strategis bagi daerah, pembentukan cadangan untuk keperluan tertentu, serta pengeluaran-pengeluaran lain yang berkaitan dengan aspek pembiayaan daerah. Proporsi yang sangat kecil dari komponen ini menunjukkan bahwa daerah tidak memiliki beban kewajiban pembiayaan yang berat dan tidak melakukan investasi modal yang besar pada entitas-entitas di luar struktur pemerintahan langsung.

#### 3.2.3.3 Presentasi Progres

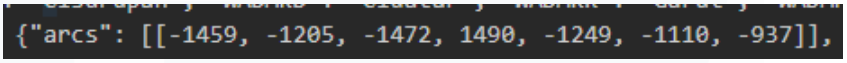
Mahasiswa melaksanakan sesi presentasi progres pada hari yang telah dijadwalkan sebagai bagian dari rangkaian proyek yang sedang dikerjakan. Kegiatan ini diselenggarakan secara *offline* di ruang rapat dan *online* melalui Microsoft Teams untuk menciptakan suasana kondusif dan terjadinya interaksi langsung agar lebih efisien. Dengan presentasi yang dilakukan, mahasiswa secara sistematis menguraikan berbagai aktivitas dan kegiatan yang telah berhasil dilakukan selama periode magang berlangsung. Setiap tahapan proses dijelaskan secara kronologis, dimulai dari fase perencanaan awal hingga implementasi solusi yang telah dilakukan. Dalam pemaparannya, mahasiswa tidak hanya menyajikan daftar pekerjaan yang telah diselesaikan, namun juga menjelaskan

mengenai metode yang digunakan. Dalam presentasi, mahasiswa juga mengungkapkan mengenai berbagai hambatan dan tantangan yang dihadapi selama proses pengerjaan proyek. Setiap tantangan dijelaskan secara detail, termasuk dampak yang mungkin terjadi terhadap kualitas *output* yang dihasilkan. Mahasiswa juga memaparkan beberapa upaya yang dilakukan untuk mengatasi setiap permasalahan, dengan menangani masalah menggunakan alternatif solusi yang ada. Selama presentasi berlangsung, tim mulai memberikan berbagai pertanyaan analitis dan mendalam. Tidak hanya untuk mengklarifikasi informasi yang telah disampaikan, namun lebih menggali wawasan tentang proses dan solusi yang dilakukan mahasiswa dalam menghadapi setiap situasi. Mentor dan *Supervisor* kemudian memberikan evaluasi yang mencakup apresiasi sekaligus menyampaikan masukan-masukan untuk perbaikan dan pengembangan lebih lanjut. Berbagai saran yang diberikan tidak hanya bersifat teknis, namun juga mencakup perspektif lainnya yang dapat memperkuat proyek yang sedang dikerjakan.

#### 3.2.3.4 *Troubleshoot shapefile map* Indonesia

setelah melakukan sesi presentasi progres, mahasiswa secara tidak sengaja menemukan adanya kesalahan dalam pemetaan geografis Indonesia yang digunakan dalam visualisasi. Kesalahan ini muncul pada *file* TopoJSON, yaitu format *file* yang dipakai untuk mendukung pemetaan data geografis dalam aplikasi. Masalah yang teridentifikasi adalah hilangnya beberapa wilayah administratif, antara lain Kota Magelang, Kota Tebing Tinggi, serta beberapa daerah lain di Indonesia yang seharusnya tampil dalam visualisasi peta. masalah ini tidak hanya mengganggu visualisasi, namun juga berpotensi menyebabkan bias dalam analisis data spasial karena informasi geografis yang tidak lengkap. Setelah dilakukan penelusuran lebih dalam, mahasiswa menemukan bahwa penyebab

utama masalah ini berasal dari kekurangan data pada *shapefile* sumber yang kemudian dikonversi menjadi TopoJSON. Mahasiswa menemukan bahwa beberapa daerah yang hilang tidak memiliki entri "arcs" di dalam *file* TopoJSON tersebut. Dalam konteks TopoJSON, "arcs" adalah representasi vektor fundamental dari jalur atau garis yang membentuk batas-batas wilayah geografis. Setiap "arcs" merupakan urutan koordinat yang dikodekan menggunakan teknik *delta encoding* untuk mengoptimalkan ukuran *file*, di mana setiap titik koordinat dinyatakan sebagai perubahan relatif dari titik sebelumnya.




Gambar 3.17 Contoh struktur "arcs" TopoJSON

Gambar 3.17 merupakan contoh struktur "arcs" yang ditemukan dalam *file* TopoJSON, di mana setiap angka merepresentasikan perubahan koordinat x dan y secara bergantian. "arcs" digunakan untuk membangun poligon atau garis kompleks yang menggambarkan bentuk dan batas wilayah administratif, seperti provinsi, kota, kabupaten, dan unit administratif lainnya. Proses pembentukan poligon dilakukan dengan menggabungkan satu atau lebih "arcs" sesuai dengan referensi yang terdapat dalam objek geometri. Tanpa adanya "arcs", visualisasi tidak dapat menentukan posisi spasial, bentuk geometris, atau batas wilayah dari objek tersebut. Akibatnya, wilayah tersebut akan hilang atau tidak ditampilkan dalam *rendering* peta akhir. Untuk mengatasi masalah tersebut, mahasiswa melakukan serangkaian langkah, dimulai dengan validasi *shapefile* menggunakan *software* QGIS atau ArcGIS. Proses validasi ini meliputi pemeriksaan integritas geometri, deteksi kesalahan topologi, dan verifikasi kelengkapan atribut untuk memastikan bahwa seluruh wilayah yang diinginkan memang memiliki geometri yang valid dan lengkap. Langkah

selanjutnya adalah mencari dan menggunakan *shapefile* alternatif terbaru, khususnya dari sumber resmi seperti Badan Pusat Statistik (BPS) yang menyediakan peta digital wilayah administrasi Indonesia, atau Bada Informasi Geospasial (BIG) yang memuat data geografi administrasi secara lengkap dan *update*. Proses perbaikan dilanjutkan dengan melakukan konversi ulang dari *shapefile* ke TopoJSON menggunakan *tools* seperti *mapshaper* untuk memastikan tidak ada data yang hilang selama proses konversi. Tahap akhir dari proses *troubleshooting* adalah validasi menyeluruh terhadap *file* TopoJSON hasil konversi, yang meliputi pemeriksaan struktur "arcs", verifikasi referensi geometri, dan *testing rendering* untuk memastikan semua wilayah memiliki entri "arcs" yang sesuai dan dapat ditampilkan dengan benar dalam visualisasi.

#### 3.2.3.5 *Update* data APBD 2025

Setelah presentasi berlangsung, mentor memberikan arahan bahwa akan ada *update* pada data yang digunakan dalam proyek. Mentor menyampaikan bahwa data terbaru yang mencakup periode tahun 2025 juga akan dimasukkan ke dalam analisis. Oleh karena itu, mahasiswa diminta untuk menyesuaikan proses yang digunakan agar dapat mengakomodasi keberadaan data terbaru tersebut. Proses pembaruan data ini dilakukan oleh mentor yang bertanggung jawab untuk melakukan penarikan data terbaru langsung dari *database*.

| Sum of sum | Column Labels | | | | |
|---|---|---|---|---|---|
| Row Labels | 2021 | 2022 | 2023 | 2024 | 2025 |
| 4. Pendapatan Daerah | 2.325.646.080.523.670 | 2.325.095.279.109.420 | 2.491.712.798.018.910 | 2.618.327.265.171.450 | 1.640.446.611.621.770 |
| 5. Belanja Daerah | 2.492.595.177.307.600 | 2.385.247.015.532.060 | 2.557.656.495.929.360 | 2.695.669.349.083.100 | 1.635.830.566.465.900 |
| 6. Pembiayaan Daerah | 213.579.886.933.543 | 256.819.051.506.653 | 260.879.740.802.562 | 194.601.174.231.714 | 96.628.356.860.746 |
| (blank) | | | | - | 39.545.031.896 |
| **Grand Total** | **5.031.821.144.764.820** | **4.967.161.346.148.130** | **5.310.249.034.750.830** | **5.508.597.788.486.270** | **3.372.865.989.916.520** |

Gambar 3.18 Data APBD terbaru

Gambar 3.18 merupakan contoh tampilan data terbaru yang dirangkum menggunakan *pivot table* sebagai ilustrasi awal. Setelah proses penarikan selesai, data yang telah diperbarui tersebut akan diberikan kepada mahasiswa. Selanjutnya, mahasiswa dapat

memproses data tersebut, mulai dari *data understanding, preprocessing*, hingga visualisasi sesuai dengan proyek.

#### 3.2.3.6 Membangun *Dashboard* Anggaran Realisasi APBD

Data terbaru yang mencakup tahun 2025 perlu dilakukan *data cleansing* kembali untuk memastikan konsistensi format serta kelengkapan nilai anggaran dan realisasi. Pembersihan data ini mencakup penyesuaian nama kolom, penyatuan istilah antar tahun, serta penghapusan nilai duplikat dan anomali. Dengan data yang telah siap, tahap selanjutnya adalah membangun *dashboard* visual sebagai media utama penyajian informasi. Pembangunan *dashboard* ini bertujuan untuk menyajikan data realisasi anggaran APBD dari tahun 2021 hingga 2025 secara interaktif, sehingga pengguna dapat mengeksplorasi informasi keuangan daerah dengan mudah dan cepat.

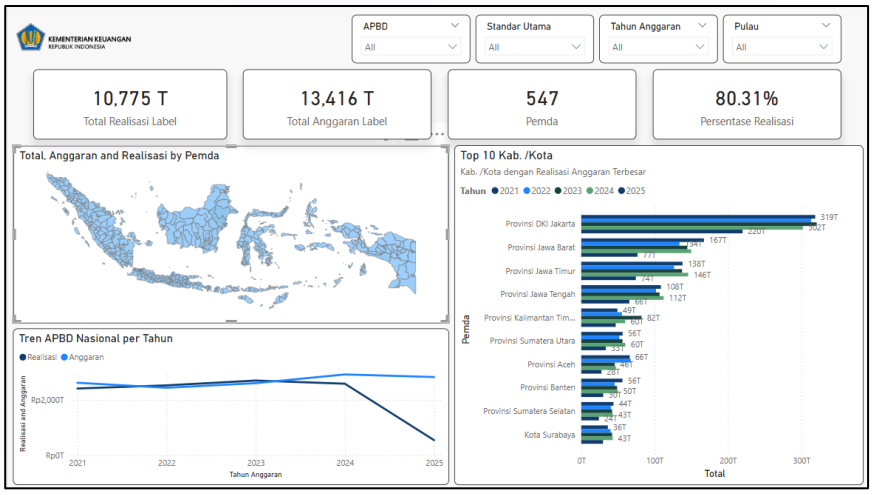
Untuk mempermudah proses visualisasi dan analisis nilai total dari anggaran dan realisasi APBD, mahasiswa melakukan pemisahan terhadap kedua jenis data tersebut yang sebelumnya berada dalam satu kolom yang sama, yaitu kolom “APBD”. Pada kolom ini informasi mengenai jenis data dituliskan dalam bentuk tabel, sementara nilai nominal disimpan pada kolom “sum”. Untuk mengelola hal ini secara lebih efektif dalam visualisasi Power BI, mahasiswa menggunakan rumus DAX untuk membuat pengukuran terpisah antara nilai anggaran dan realisasi.

```
1 Realisasi =
2     CALCULATE(
3         SUM('Sheet1'[sum]),
4         FILTER('Sheet1', 'Sheet1'[APBD] = "Realisasi")
5 )
```

Gambar 3.19 Rumus DAX untuk memperoleh nilai

Gambar 3.19 merupakan contoh untuk memperoleh total nilai realisasi. Formula tersebut bekerja dengan menjumlahkan

seluruh nilai pada kolom “sum”, tetapi hanya untuk baris data yang memiliki label “Realisasi” pada kolom “APBD”. Fungsi CALCULATE digunakan untuk menerapkan konteks filter secara eksplisit, sedangkan fungsi Filter menyaring data agar hanya baris-baris dengan kategori “Realisasi” yang dihitung. Dengan cara ini, mahasiswa dapat menampilkan nilai realisasi secara mandiri di dalam *dashboard*, terpisah dari nilai anggaran. Dengan metode ini dapat memudahkan proses visualisasi dan perbandingan data keuangan. Mahasiswa dapat menampilkan metrik seperti total anggaran dan realisasi dalam bentuk visualisasi *card*, *bar chart,* dan berbagai visual lainnya. Selain itu, pemisahan ini memungkinkan perhitungan lanjutan, seperti menghitung persentase capaian realisasi terhadap anggaran secara lebih akurat. Pendekatan ini tidak hanya meningkatkan kemudahan dalam memahami data, tetapi juga mendukung analisis yang lebih mendalam terhadap kinerja pengelolaan keuangan daerah dari tahun ke tahun.




Gambar 3.20 *Dashboard* Anggaran Realisasi APBD

Setelah pemisahan nilai anggaran dan realisasi berhasil dilakukan menggunakan rumus DAX, tahap berikutnya yang dilakukan mahasiswa adalah membangun *dashboard* interaktif sebagai sarana penyajian informasi keuangan daerah secara

komprehensif. *Dashboard* ini dirancang menggunakan Power BI dengan pendekatan visual yang informatif dan mudah dipahami, serta dilengkapi dengan filter dinamis untuk memudahkan eksplorasi data oleh pengguna. fokus utama dari *dashboard* ini adalah menyajikan data anggaran dan realisasi APBD nasional secara ringkas, akurat, dan berbasis waktu yang mencakup periode tahun 2021 hingga 2025.

Pada bagian atas *dashbaord* ditampilkan empat indikator utama dalam bentuk *card visual*, yaitu total nilai realisasi, total nilai anggaran, jumlah pemerintah daerah, serta persentase realisasi terhadap anggaran. Keempat indikator ini memberikan gambaran umum mengenai skala dan capaian pengelolaan keuangan daerah dalam lima tahun terakhir. Di atasnya terdapat elemen *slicer* atau filter interaktif yang memungkinkan pengguna menyaring tampilan data berdasarkan jenis APBD, standar belanja, tahun anggaran, dan wilayah pulau untuk menyesuaikan fokus analisis yang diinginkan. Visualisasi selanjutnya menampilkan peta Indonesia dengan pencatatan nilai anggaran dan realisasi yang terdistribusi berdasarkan lokasi pemerintah daerah. Peta ini membantu pengguna dalam memahami sebaran geografis dana APBD di seluruh Indonesia secara visual. Di samping itu, terdapat grafik batang horizontal yang menampilkan Top 10 Provinsi dan Kota/Kabupaten dengan nilai realisasi anggaran terbesar dalam lima tahun terakhir. Visualisasi ini memudahkan identifikasi daerah dengan kontribusi belanja tinggi, seperti DKI Jakarta, Jawa Barat, Jawa timur, yang secara konsisten menempati urutan teratas dalam jumlah realisasi anggaran. Sementara itu, grafik garis di bagian bawah *dashboard* menampilkan trend APBD nasional per tahun, dengan perbandingan antara nilai anggaran dan realisasi dari tahun 2021 hingga 2025. Grafik ini memberikan pemahaman terhadap dinamika belanja pemerintah daerah secara agregat, termasuk fluktuasi yang mungkin

terjadi akibat faktor kebijakan fiskal, kondisi ekonomi, atau respons terhadap peristiwa tertentu seperti pandemi.

Berdasarkan rangkaian proses yang telah dilakukan, mulai dari *input data*, *cleansing*, pemisahan nilai anggaran dan realisasi menggunakan rumus DAX, hingga pembangunan *dashboard* interaktif, dapat disimpulkan bahwa visualisasi data memainkan peran penting dalam mendukung transparansi dan pemahaman terhadap pengelolaan keuangan daerah. *Dashboard* yang dibangun berhasil menyajikan informasi anggaran dan realisasi APBD nasional periode 2021-2025 secara menyeluruh dan mudah diakses, baik dari sisi agregat maupun per wilayah. Melalui penggunaan elemen visual seperti grafik, peta interaktif, serta indikator kinerja, pengguna dapat dengan cepat mengidentifikasi pola belanja, trend realisasi, dan capaian anggaran dari ratusan pemerintah daerah di Indonesia. Selain itu, fitur filter yang tersedia memungkinkan analisis yang lebih mendalam dan terfokus sesuai kebutuhan pengguna. dengan demikian, *dashboard* ini tidak hanya berfungsi sebagai alat pelaporan, tetapi juga sebagai media analisis strategi yang dapat mendukung pengambilan keputusan yang lebih berbasis data di sektor publik.

### 3.2.4 *Minor Project*: Deteksi Hubungan Semantik Berbasis NLP

Pada akhir bulan Mei, mahasiswa berkesempatan untuk berpartisipasi dalam suatu proyek yang berfokus pada pengembangan sistem untuk klasifikasi hubungan kontekstual antara dua kalimat dalam suatu *dataset*. Inisiasi proyek diawali dengan sesi *brainstorming* yang dipandu oleh mentor, dengan tujuan merumuskan pendekatan dan metode yang tepat untuk mengatasi permasalahan yang diidentifikasi. Dalam sesi *brainstorming*, mentor memaparkan latar belakang serta konteks permasalahan yang dihadapi oleh tim kerja. permasalahan utama yang diangkat berkaitan dengan kebutuhan mendesak untuk memverifikasi apakah dua kolom teks dalam *dataset* memiliki hubungan kontekstual yang

bermakna atau tidak. Kemampuan untuk memahami keterkaitan informasi antar dua kolom teks ini sangat penting dalam memastikan konsistensi data yang digunakan dalam pengambilan keputusan. Sebagai ilustrasi, salah satu contoh kasus yang dihadapi adalah seperti pada gambar 3.21.

| Kegiatan DBH CHT | KEPMENDAGRI |
|---|---|
| Pelatihan Peningkatan Kualitas tembakau | Peningkatan kapasitas kelembagaan penyuluhan pertanian di kecamatan dan desa |

Gambar 3.21 Contoh data yang akan digunakan

Dalam kasus tersebut, sistem diharapkan mampu menentukan apakah kedua kegiatan yang tercantum memiliki hubungan kontekstual, misalnya apakah keduanya merupakan bagian dari program pengembangan sektor pertanian tembakau atau justru merupakan kegiatan yang tidak saling terkait. Kemampuan otomatis untuk melakukan penilaian semacam ini akan sangat membantu dalam proses validasi data kegiatan lintas kementerian/lembaga.

Setelah memperoleh pemahaman yang jelas mengenai permasalahan yang dihadapi, mahasiswa berkolaborasi untuk mengusulkan berbagai solusi inovatif, dengan mempertimbangkan keterbatasan sumber daya dan kebutuhan organisasi. Salah satu pendekatan yang disepakati melalui diskusi adalah pengembangan model klasifikasi kontekstual berbasis *Natural Language Processing* (NLP). Model ini diharapkan mampu menganalisis hubungan semantik kompleks antara dua kalimat dengan tingkat akurasi yang tinggi, serta mengklasifikasikan hubungan tersebut ke dalam kategori yang telah ditentukan, seperti hubungan erat atau tidak memiliki hubungan. Pada tahap perancangan, pendekatan yang dipertimbangkan meliputi penggunaan teknik *sentence embedding* untuk mentransformasi kalimat menjadi representasi vektor multidimensi yang dapat dibandingkan secara matematis. Kemudian dilakukan perhitungan kesamaan semantik menggunakan metrik seperti *cosine similarity*, pemanfaatan model *pre-trained* seperti *Sentence-BERT*, serta penerapan algoritma berbasis *machine learning* maupun *deep learning* untuk

memetakan hubungan antar kalimat. Dengan pendekatan ini, proyek tidak hanya bertujuan menghasilkan model yang akurat dan optimal, tetapi juga mampu memberikan *insight* yang mendalam bagi tim dalam memahami pola hubungan tersembunyi dalam data secara lebih strategis.

#### 3.2.4.1 Membangun Model Klasifikasi Kontekstual

Program ini merupakan implementasi analisis kemiripan semantik antara dua kolom teks dalam *dataset* menggunakan teknik *Natural Language Processing* (NLP). Tujuan utama program adalah untuk mengatur seberapa mirip makna dari pasangan teks menggunakan teknologi *sentence embeddings*. Program ini menggunakan model *transformer pre-trained* yang telah dilatih pada jutaan pasang kalimat untuk memahami konteks dan makna teks, bukan hanya berdasarkan kesamaan kata kunci.

```
1 import pandas as pd
2 import torch
3 from sklearn.metrics.pairwise import cosine_similarity
4 from sentence_transformers import SentenceTransformer
5 import numpy as np
```

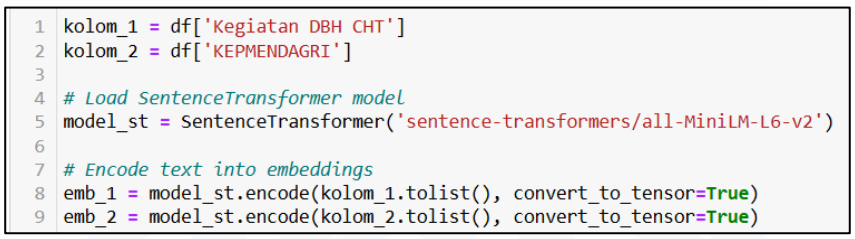
Gambar 3.22 *Import Library* dan Modul

Analisis *similarity* dimulai dengan mengimpor berbagai *library* dan modul yang diperlukan untuk menjalankan analisis *similarity*. Pandas digunakan sebagai *library* fundamental untuk manipulasi dan analisis data terstruktur dalam bentuk Dataframe. *PyTorch* (*Torch*) berfungsi sebagai *framework deep learning* yang menjadi *backend* untuk model *transformers* yang menyediakan infrastruktur komputasi tensor yang efisien. *Scikit-learn* berkontribusi melalui fungsi *cosine_similarity* yang mengimplementasikan perhitungan kemiripan antar vektor dengan efisien dan akurat. *SentenceTransformer* merupakan *library* khusus yang telah dioptimalkan untuk menghasilkan *sentence embeddings* berkualitas serta memberikan kemudahan dalam proses *encoding*

teks menjadi representasi vektor. Terakhir, *NumPy* memberi dukungan untuk operasi *array* dan komputasi numerik yang menjadi fondasi perhitungan matematis dalam program.


Gambar 3.23 *Import dataset*

Proses selanjutnya adalah *load dataset* dengan format *Excel* ke dalam struktur data Pandas *DataFrame* menggunakan fungsi "read_excel". Pada tahap ini, program mengasumsikan bahwa *file* Excel berisi minimal dua kolom teks yang akan dibandingkan kemiripannya. Penggunaan parameter *engine* yang sesuai seperti "openpyxl" memiliki kompabilitas yang lebih baik, serta melakukan eksplorasi data awal menggunakan fungsi seperti "head()" dan "info()" untuk memahami struktur dan karakteristik *dataset*. Langkah ini juga penting untuk memvalidasi bahwa kolom-kolom yang dibutuhkan tersedia dalam *dataset* dan tidak ada masalah dengan format atau *encoding file*.




Gambar 3.24 Ekstraksi kolom yang akan digunakan

Tahap *preprocessing* melibatkan ekstraksi dua kolom spesifik dari DataFrame, yaitu "Kegiatan DBH CHT" dan "KEPMENDAGRI". Program ini menggunakan model *SentenceTransformer* dengan arsitektur "sentence-transformers/all-MiniLM-L6-v2" yang merupakan pilihan optimal untuk *task similarity matching*. Model ini dibangun berdasarkan arsitektur "MiniLM" yang merupakan versi ringkas dari "BERT" namun tetap

mempertahankan kemampuan pemahaman semantik yang kuat. Dengan sekitar 22.7 juta parameter, model ini menawarkan keseimbangan ideal antara performa dan efisiensi komputasi, menghasilkan *embeddings* dengan dimensi 384 yang cukup handal untuk menangkap makna teks. Keunggulan utama model ini terletak pada dukungan multilingual untuk lebih dari 100 bahasa termasuk Bahasa Indonesia. Kemampuannya memahami konteks dan semantik teks bukan hanya kesamaan dari kata kunci, tetapi juga optimasi khusus untuk *task sentence similarity* yang telah dilatih pada jutaan pasang kalimat. Model ini juga memiliki kecepatan *encoding* yang tinggi sehingga cocok untuk memproses *dataset* berukuran besar.

Selanjutnya, terdapat proses *encoding* yang menjadi inti dari analisis *similarity*, di mana setiap kalimat dari kedua kolom dikonversi menjadi representasi vektor numerik menggunakan metode *encode* dari *SentenceTransformer*. Proses ini melibatkan beberapa tahap internal yang kompleks namun dapat dieksekusi dengan baik oleh *library*. Pertama, teks *input* dipecah menjadi token-token individual melalui proses *tokenization* yang mempertimbangkan struktur linguistik dan *subword information*. Kemudian, setiap token dikonversi menjadi *embedding vector* melalui *embedding layer* yang telah dilatih. Selanjutnya *embeddings* dari semua token dalam satu kalimat diagregasi melalui *pooling mechanism* untuk menghasilkan satu vektor representatif per kalimat. Parameter "convert_to_tensor=True" memastikan *output* dalam format PyTorch tensor yang lebih efisien untuk operasi komputasi selanjutnya, sedangkan "tolist()" mengonversi *Pandas series* menjadi *Python list* yang kompatibel dengan *input function*. Hasil akhir berupa tensor dengan *shape* "[n_sentence, 384]" di mana setiap baris merepresentasikan satu kalimat dalam ruang vektor 384-dimensional. Representasi vektor ini menangkap makna semantik

kalimat, konteks antar kata, nuansa bahasa seperti sinonim dan polisemi, serta struktur yang memungkinkan model memahami kemiripan makna meski menggunakan kata-kata yang berbeda.

```
# Hitung cosine similarity
similarity_scores = cosine_similarity(emb_1, emb_2).diagonal()
df['similarity_score'] = similarity_scores
df['is_similar'] = df['similarity_score'] >= 0.5  # threshold 50%

# Simpan hasil
df.to_excel("hasil_similarity.xlsx", index=False)

```

Gambar 3.25 Kode untuk menghitung *cosine similarity*

Selanjutnya, dilakukan perhitungan *similarity* menggunakan *cosine similarity* yang merupakan metrik standar untuk mengukur kemiripan antar vektor dalam ruang multidimensional. *Cosine similarity* bekerja dengan menghitung *cosinus* sudut antara dua vektor, dengan rumus matematika:

$$Cosine\ similarity = \frac{\vec{A} \cdot \vec{B}}{\overrightarrow{\|A\|} \times \overrightarrow{\|B\|}}$$

Di mana, A dan B adalah vektor embeddings dari kedua teks yang dibandingkan. Nilai hasil perhitungan berkisar antara -1 hingga 1, di mana 1 menunjukkan kemiripan sempurna (vektor searah), 0 menunjukkan tidak ada kemiripan (vektor tegak lurus), dan -1 menunjukkan oposisi sempurna (vektor berlawanan arah). Dalam konteks *text similarity*, nilai negatif jarang terjadi karena teks umumnya memiliki beberapa elemen kesamaan. Penggunaan diagonal () pada hasil "cosine_similarity" sangat penting karena akan menghasilkan matriks "nxn" yang berisi *similarity score* antara setiap kombinasi vektor, sedangkan proyek ini hanya membutuhkan perbandingan antara teks pada baris yang sama. Diagonal () mengekstrak elemen-elemen diagonal dari matriks ini, menghasilkan *array* 1D yang berisi *similarity score* untuk setiap pasangan teks yang sejajar. Interpretasi praktis dari nilai *similarity*

yaitu 80% hingga 100% menunjukkan kemiripan sangat tinggi, 60% hingga 80% menunjukkan kemiripan tinggi, 40% hingga 60% menunjukkan kemiripan sedang, 20% hingga 40% menunjukkan kemiripan rendah, dan di bawah 20% menunjukkan teks yang berbeda secara signifikan.

Hasil perhitungan *similarity* kemudian diintegrasikan kembali ke dalam DataFrame asli dengan menambahkan dua kolom baru yang memberikan perspektif berbeda terhadap tingkat kemiripan. Kolom "*similarity_score*" berisi nilai hasil perhitungan *cosine similarity* yang memberikan informasi tentang tingkat kemiripan dengan presisi tinggi, sehingga memungkinkan analisis yang lebih detail berdasarkan tingkat *similarity*. Kolom "is_similar" merupakan transformasi *binary* dari *similarity score* dengan menggunakan *threshold* 50% sebagai *cut-off point*, yang memberikan klasifikasi sederhana berupa *True/False* untuk memudahkan analisis kategorikal. Pemilihan *threshold* 50% merupakan pendekatan konservatif yang memastikan hanya pasangan teks dengan kemiripan yang cukup signifikan yang diklasifikasikan sebagai *similar* dan juga menyeimbangkan antara *precision* dan *recall* dalam deteksi *similarity*.

| Kegiatan DBH CHT | KEPMENDAGRI | similarity_score | is_similar |
|---|---|---|---|
| Pelatihan Peningkatan Kualitas te | Peningkatan kapasitas kelem | 0,626792908 | TRUE |
| Pelatihan Peningkatan Kualitas te | Pengembangan kapasitas kel | 0,554527998 | TRUE |
| Pelatihan Peningkatan Kualitas te | Pembentukan dan Penyeleng | 0,661085606 | TRUE |
| Pelatihan Peningkatan Kualitas te | Diseminasi Informasi Teknis, | 0,493218601 | FALSE |
| Pelatihan Peningkatan Kualitas te | Pengembangan kapasitas kel | 0,554527998 | TRUE |
| Pelatihan Peningkatan Kualitas te | Peningkatan kapasitas kelem | 0,626792908 | TRUE |
| Penanganan panen dan pasca pa | Peningkatan pascapanen dar | 0,362905115 | FALSE |
| Pelatihan Peningkatan Kualitas te | Diseminasi Informasi Teknis, | 0,493218601 | FALSE |

Gambar 3.26 Hasil *similarity score check*

Tahap akhir *text similarity* adalah menyimpan analisis ke dalam *file* Excel baru yang berisi semua informasi asli ditambah dengan hasil perhitungan *similarity*. Fungsi "to_excel" dengan

parameter “index=False” memastikan bahwa *row index* pandas tidak disertakan dalam *output file*, sehingga menghasilkan *file* yang lebih bersih dan mudah dipahami. *File output* berisi kolom-kolom asli dari *dataset* awal beserta dua kolom tambahan hasil analisis, yaitu “similarity_score” dan “is_similar”, yang memberikan *insight* tentang tingkat kemiripan antar teks. *Output file* ini akan langsung digunakan sebagai *input* untuk tahap *modeling* selanjutnya, khususnya untuk menganalisis dan memahami pola relasi kontekstual antar teks dalam *dataset*.

Pembangunan model klasifikasi kontekstual ini merupakan tahap lanjutan dari analisis *similarity* yang telah dilakukan sebelumnya. Model ini dirancang untuk mengklasifikasikan teks berdasarkan pemahaman kontekstual yang mendalam dengan memanfaatkan kombinasi teknologi *deep learning* dengan BERT dan pendekatan *machine learning* tradisional. *Pipeline* ini mengintegrasikan *preprocessing* data hasil *similarity analysis*, *feature extraction* menggunakan *transformer models*, dan klasifikasi menggunakan algoritma *machine learning* untuk menghasilkan model yang kuat dan akurat dalam memahami kontekstual teks.

```
import pandas as pd
import torch
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from transformers import BertTokenizer, BertModel
from torch.optim import AdamW
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
from tqdm import tqdm
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import re
```

Gambar 3.27 *Import library* dan modul untuk pemodelan selanjutnya

Pembangunan model klasifikasi kontekstual dimulai dengan mengimpor berbagai *library* penting yang mendukung pemodelan secara menyeluruh. *Library* fundamental seperti Pandas menjadi dasar utama dalam manipulasi dan analisis data terstruktur. Pandas

digunakan untuk memuat dataset hasil analisis *similarity*, melakukan *preprocessing* terhadap label target, membagi data menjadi set pelatihan dan validasi, serta mengelola metadata yang diperlukan dalam proses evaluasi model. Di samping itu, Pandas juga sangat membantu dalam proses evaluasi model. Di samping itu, Pandas juga sangat membantu dalam proses *Exploratory Data Analysis* (EDA) seperti analisis distribusi kelas, penanganan nilai hilang, dan transformasi variabel kategorikal dalam format numerik yang sesuai untuk kebutuhan pemodelan.

Selanjutnya, NumPy digunakan untuk mendukung kmoputasi numerik secara efisien melali array multidimensi. Dalam konteks pembangunan model, NumPy berperan penting dalam operasi matematis terhadap *embedding*, perhitungan statistik fitur, serta transformasi data seperti normalisasi, reduksi dimensi, dan penskalaan fitur. Kemampuan ini mendukung kestabilan numerik dan performa model secara keseluruhan. Untuk kebutuhan *machine learning*, Scikit-learn menjadi *tools* yang menyediakan berbagai fungsi penting. Fungsi "train_test_split" digunakan untuk membagi dataset secara proporsional agar distribusi kelas tetap seimbang antara data latih dan uji. Untuk evaluasi performa model, "classification_report" dan "confusion_matrix" menyajikan metrik evaluasi baik per kelas maupun secara keseluruhan. *Confusion matrix* juga memberikan gambaran visual terhadap hasil prediksi, membantu dalam mengidentifikasi kelas yang sering salah diprediksi, serta mengukur potensi bias dalam model.

Dalam hal pemrosesan bahasa (NLP), *library* Hugging Face Transformers menyediakan akses terhadap model-model terkini berbasis arsitektur transformer. Tokenisasi menggunakan BertTokenizer memungkinkan transformasi teks mentah menjadi token yang dapat dipahami oleh model BERT, termasuk penanganan

token spresial, *subword tokenization* untuk kata-kata asing, serta pembentukan attention mask agar model dapat menangani *input* dengan panjang yang berbeda-beda. Sementara itu, BertModel digunakan untuk menghasilkan representasi kontekstual yang kaya, sehingga model dapat menangkap makna semantik dan pola sintaks dalam setiap kalimat. Untuk proses pelatihan dan optimasi, digunakan *framework* PyTorch yang mendukung komputasi berbasis tensor dengan akselerasi GPU serta mekanisme *autodifferentation* yang esensial dalam proses *backpropagation*. PyTorch memunngkinkan fleksibilitas dalam membangun arsitektur model yang sesuai, mendefinisikan *loss function* secara kustom, dan menerapkan strategi optimasi lanjutan. Salah satu *optimizer* yang digunakan adalah AdamW, yang merupakan sebuah varian dari "torch.optim" yang secara eksplisit mengintegrasikan regulasi *weight decay*.

Selain *library* utama di atas, erdapat juga sejumlah *library* yang mendukung alur kerja *machine learning*. Salah satunya adalah TQDM, yang digunakan untuk menampilkan *progress bar* selama proses pelatihan model berlangsung. Visualisasi ini sangat membantu dalam memantau durasi dari tiap *epoch* dan proses *batch*, serta mendeteksi jika terjadi *bottleneck.* Untuk kebutuhan khusus Bahasa Indonesia, digunakan *library* Sastrawi yang menyediakan algoritma *stemming*. Proses *stemming*ini berfungsi menghilangkan imbuhan kata sehingga model lebih fokus pada akar kata dan makna dasar. Meski demikian, penggunaan *stemming* perlu dilakukan secara hati-hati karena berisiko menghilangkan konteks penting yang justru dibutuhkan dalam klasifikasi relasi antar kalimat. Terakhir, modul Regular Expressions (re) digunakan untuk *preprocessing* teks tingkat lanjut. Dengan kemampuan *pattern matching*, modul ini sangat berguna untuk membersihkan data dari karakter yang tidak relevan, menstandarkan format teks, engekstrak

pola tertentu, hingga menormalisasi teks yang inkonsisten. Kemampuan ini penting dalam mendukung pembersihan data berbasis aturan sebelum proses tokenisasi dan pemodelan lebih lanjut dilakukan.

```
#Load Data
df = pd.read_excel("C:\\Users\\frhnz\\DJPK_SIPT\\NLP\\hasil_similarity.xlsx", sheet_name='Sheet1')
df.head()
```

| | Kegiatan DBH CHT | KEPMENDAGRI | similarity_score | label |
|---|---|---|---|---|
| 0 | Pelatihan Peningkatan Kualitas tembakau | Peningkatan kapasitas kelembagaan penyuluhan p... | 0.626793 | 1 |
| 1 | Pelatihan Peningkatan Kualitas tembakau | Pengembangan kapasitas kelembagaan petani di k... | 0.554528 | 1 |
| 2 | Pelatihan Peningkatan Kualitas tembakau | Pembentukan dan Penyelenggaraan Sekolah Lapang... | 0.661086 | 1 |
| 3 | Pelatihan Peningkatan Kualitas tembakau | Peningkatan kapasitas kelembagaan penyuluhan p... | 0.626793 | 1 |
| 4 | Pelatihan Peningkatan Kualitas tembakau | Pengembangan kapasitas kelembagaan petani di k... | 0.554528 | 1 |

*Gambar 3.28 Import dataset* (dengan *similarity score*)

Setelah mengimpor seluruh *library* yang diperlukan, proses selanjutnya adalah memuat *dataset* menggunakan fungsi "read_excel()" dari *library* Pandas. Setelah data berhasil dimuat ke dalam sebuah DataFrame, fungsi "df.head()" digunakan untuk menampilkan lima baris pertama dari data guna memastikan struktur dan isi *dataset* telah terbaca dengan benar. Dua kolom pertama berisi deskripsi teks dari kegiatan yang berasal dari dua sumber berbeda yang akan dianalisis hubungan semantisnya. Kolom "similarity_score" merepresentasikan nilai kemiripan antara kedua teks dengan rentang 0 hingga 1. Sementara itu, kolom label berfungsi sebagai target klasifikasi yang menunjukkan apakah kedua kegiatan memiliki kemiripan atau tidak, berdasarkan *thresholding* terhadap skor *similarity*. Proses ini menjadi tahap krusial dalam *pipeline deep learning* karena memastikan bahwa data yang digunakan dalam pelatihan dan evaluasi model telah terstruktur dan sesuai dengan kebutuhan analisis lanjutan.

```python
# Text Preprocessing
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def preprocess_text(text):
    # Hilangkan karakter khusus (tanda baca dipertahankan)
    text = re.sub(r"[^a-zA-Z0-9\s.,?!:()]", "", str(text))
    # Stemming
    text = stemmer.stem(text)
    return text.strip()

df['Kegiatan DBH CHT'] = df['Kegiatan DBH CHT'].apply(preprocess_text)
df['KEPMENDAGRI'] = df['KEPMENDAGRI'].apply(preprocess_text)
```

*Gambar 3.29* Kode untuk *preprocessing text*

Selanjutnya dilakukan *preprocessing* dengan tujuan untuk membersihkan dan menormalisasi data teks mentah agar dapat digunakan secara optimal dalam tahap klasifikasi. *Preprocessing data* dilakukan melalui dua langkah utama, yaitu penghapusan karakter non-alfabetik serta proses *stemming* terhadap kata-kata dalam teks menggunakan Bahasa Indonesia. Implementasi proses ini dilakukan dengan menggunakan *library* Sastrawi, yang menyediakan *StemmerFactory* untuk menghasilkan objek *stemmer*. Objek tersebut digunakan untuk mengubah setiap kata menjadi bentuk dasarnya, sehingga variasi pada kata dapat disederhanakan menjadi satu bentuk umum, seperti “meningkatkan” menjadi “tingkat”. Kemudian, fungsi “preprocess_text” digunakan untuk mengotomatisasi proses pembersihan dan *stemming*. Fungsi ini memanfaatkan ekspresi reguler untuk menghapus karakter-karakter khusus yang tidak termasuk dalam huruf alfabet, angka, spasi, dan tanda baca dasar seperti titik, koma, tanda tanya, dan tanda seru. Fungsi ini penting untuk menghindari gangguan semantik akibat adanya simbol-simbol asing yang tidak relevan dalam konteks kalimat. Setelah tahap pembersihan selesai, teks kemudian diproses melalui *stemmer* yang telah dibuat sebelumnya, dan hasil akhirnya dikembalikan dalam bentuk *string* bersih yang telah disederhanakan. Fungsi *preprocessing* ini diterapkan secara langsung pada dua kolom utama pada *dataset*. Kedua kolom tersebut berisi uraian kegiatan dari

dua sumber berbeda, yang menjadi dasar dalam penilaian apakah terdapat hubungan kontekstual di antara keduanya.

```python
# Split Data (80% Train, 20% Test)
train_df, test_df = train_test_split(df, test_size=0.2, stratify=df['label'], random_state=42)

# Tokenizer (IndoBERT)
tokenizer = BertTokenizer.from_pretrained("indobenchmark/indobert-base-p1")

# Dataset Class
class RelasiDataset(torch.utils.data.Dataset):
    def __init__(self, dataframe, tokenizer):
        self.dataframe = dataframe
        self.tokenizer = tokenizer

    def __len__(self):
        return len(self.dataframe)

    def __getitem__(self, idx):
        item = self.dataframe.iloc[idx]

        encoding = self.tokenizer(item['Kegiatan DBH CHT'], truncation=True, padding='max_length', max_length=128, return_te

        return {
            'input_ids': encoding['input_ids'].squeeze(0),
            'attention_mask': encoding['attention_mask'].squeeze(0),
            'label': torch.tensor(item['label'], dtype=torch.long),
            'similarity_scores': torch.tensor(item['similarity_score'], dtype=torch.float)
        }
```

Gambar 3.30 Pembagian Data dan Definisi *Dataset Class* untuk Model Kontekstual

Tahap krusial dalam pembentukan *dataset* dimulai dengan pemisahan data menggunakan fungsi “train_test_split” dengan proporsi 80% untuk data pelatihan dan 20% untuk data pengujian. Pemisahan ini dilakukan berdasarkan label klasifikasi guna mempertahankan distribusi kelas yang seimbang di kedua *subset*. Strategi ini penting untuk mengurangi bias akibat ketidakseimbangan kelas serta memastikan validitas statistik dalam evaluasi model. Penggunaan parameter “random_state” menjamin dapat mereproduksi hasil, sementara rasio 80:20 dipilih berdasarkan *best practice* dalam pembelajaran mesin, karena memberikan data yang cukup bagi model untuk mempelajari pola kompleks, sekaligus menyisakan porsi yang memadai untuk evaluasi performa secara objektif. integrasi *tokenizer* IndoBERT melibatkan sejumlah konfigurasi untuk normalisasi huruf kapital, standarisasi karakter, dan menjaga keseimbangan antara efisiensi komputasi dan kelengkapan informasi. *Tokenizer* ini menggunakan skema *subword tokenization* WordPiece, yang sangat efektif dalam menangani variasi Bahasa Indonesia serta kata-kata yang tidak dikenal.

Implementasi kelas *dataset* "RelasiDataset" dirancang secara modular dan efisien yang memungkinkan *lazy loading*. *Lazy loading* adalah pemrosesan data yang dilakukan saat diperlukan, bukan saat inisialisasi sehingga dapat menghemat memori. Konstruktor kelas menyimpan DataFrame hasil *preprocessing* dan objek *tokenizer*, serta melakukan reset indeks untuk memastikan konsistensi dalam akses data. Metode "__len__" memenuhi standar *interface PyTorch* dan memungkinkan pembagian *batch* yang optimal, sementara "__getitem__" menangani tokenisasi secara langsung dan fleksibel, termasuk konversi indeks tensor ke format *list* bila diperlukan. Fitur lainnya yang penting dari implementasi ini adalah integrasi *similarity scores* yang mencerminkan tingkat kemiripan semantik antar kalimat. Integrasi informasi tambahan ini memungkinkan model untuk tidak hanya mempelajari representasi dari teks mentah, tetapi juga memperhitungkan sinyal eksternal yang relevan. Hal ini berpotensi meningkatkan akurasi dan kemampuan menafsirkan model dalam tugas klasifikasi hubungan kalimat yang bersifat kompleks.

```python
from transformers import AutoModel

# 7. DataLoader
train_dataset = RelasiDataset(train_df, tokenizer)
test_dataset = RelasiDataset(test_df, tokenizer)

train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=16)

# 8. Model Architecture
class HybridModel(nn.Module):
    def __init__(self, hidden_size=769, num_labels=2):  # Sesuaikan
        super(HybridModel, self).__init__()
        self.bert = AutoModel.from_pretrained("indobenchmark/indobert-base-p1")
        self.linear = nn.Linear(hidden_size, num_labels)

    def forward(self, input_ids, attention_mask, similarity_scores=None):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        pooled_output = outputs.last_hidden_state[:, 0, :]

        if similarity_scores is not None:
            pooled_output = torch.cat([pooled_output, similarity_scores.unsqueeze(1)], dim=1)

        return self.linear(pooled_output)

# 9. Inisialisasi Model
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = HybridModel().to(device)
```

Gambar 3.31 Implementasi *Hybrid Model* Berbasis IndoBERT

Tahap lanjutan dari pengembangan sistem klasifikasi hubungan kontekstual antar kalimat adalah pembentukan arsitektur model. Proses ini diawali dengan pembuatan objek "DataLoader" untuk data pelatihan dan pengujian. *Dataset* ini kemudian diarahkan ke dalam "DataLoader" PyTorch yang bertugas untuk menyusun data dalam bentuk *batch* untuk efisiensi pelatihan. "batch_size" digunakan sebesar 16 untuk menyeimbangkan antara akurasi pelatihan dan efisiensi penggunaan memori GPU, sementara "shuffle" pada "train_loader" memastikan bahwa urutan data diacak agar model tidak *overfit* terhadap urutan data tertentu. Selanjutnya, rancangan arsitektur model dilakukan melalui deklarasi kelas "HybridModel" yang merupakan struktur dasar model dalam PyTorch. Model memuat "AutoModel" dengan menggunakan model *pretrained* IndoBERT dari repositori "indobenchmark/indobert-base-p1". Model ini bertugas mengekstraksi representasi kontekstual dari *input* kalimat. Selain itu, layer linier digunakan sebagai lapisan klasifikasi akhir, yang akan mengubah keluaran dari IndoBERT menjadi prediksi label. Parameter "hidden_size" dan "num_labels" menunjukkan bahwa *output* dari BERT memiliki dimensi yang bisa disesuaikan dengan *output* aktual dari model.

Metode "forward" mendefinisikan alur data selama proses inferensi dan pelatihan. *Input* berupa "input_ids" dan "attention_mask" dikirimkan ke model BERT untuk menghasilkan representasi kalimat. BERT menghasilkan *output* yang diekstraksi dari token "CLS", yang biasanya mewakili keseluruhan makna kalimat. Selanjutnya, terdapat mekanisme fleksibel untuk mengintegrasikan fitur tambahan berupa "similarity_scores". Jika skor ini tersedia, maka akan digabungkan dengan representasi "CLS", sehingga model memperoleh informasi tambahan semantik dalam proses klasifikasi. Pendekatan ini menerapkan arsitektur *hybrid*, di mana model tidak hanya bergantung pada representasi

kalimat dari BERT, tetapi juga pada sinyal eksternal dari perhitungan kesamaan semantik, seperti *cosine similarity*. Metode ini kemudian ditutup dengan layer linier untuk menghasilkan prediksi akhir berdasarkan gabungan fitur yang tersedia. proses inisialisasi model dilakukan dengan menentukan perangkat komputasi berdasarkan ketersediaan GPU. Metode ini penting untuk memastikan pelatihan model berlangsung secara optimal, khususnya saat menggunakan model skala besar seperti IndoBERT yang memiliki hubungan komputasi tinggi.

Dalam banyak kasus klasifikasi khususnya pada domain publik, data yang digunakan cenderung mengalami ketidakseimbangan distribusi label. Ketidakseimbangan ini berisiko menyebabkan model bias terhadap kelas mayoritas, sehingga performa pada kelas minoritas menurun drastis. Oleh karena itu, pendekatan yang umum digunakan untuk mengatasi permasalahan ini adalah dengan menerapkan strategi penyesuaian bobot kelas, seperti yang ada pada gambar 3.31.

```python
# 10. Hitung Class Weight (Untuk Imbalance Data)
from sklearn.utils.class_weight import compute_class_weight

# Calculate class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_df['label']),
    y=train_df['label']
)
class_weights = torch.tensor(class_weights, dtype=torch.float).to(device)

# Loss function with class weights
loss_fn = nn.CrossEntropyLoss(weight=class_weights)

# 11. Loss Function & Optimizer
loss_fn = nn.CrossEntropyLoss(weight=class_weights)
optimizer = AdamW(model.parameters(), lr=2e-5)
```

Gambar 3.32 Penanganan Ketidakseimbangan Kelas dan *Inisialisasi Optimizer*

Modul "sklearn.utils.class_weight" memiliki fungsi "compute_class_weight" yang digunakan untuk menghitung bobot masing-masing kelas secara otomatis berdasarkan frekuensi kemunculannya. Parameter "class_weight" memastikan bahwa kelas

yang lebih jarang akan diberi bobot yang lebih tinggi, sehingga model akan belajar lebih keras pada kelas-kelas tersebut. nilai bobot yang dihasilkan kemudian dikonversi ke tensor PyTorch menggunakan "torch.tensor" dengan tipe data *float* dan dipindahkan ke perangkat komputasi agar dapat digunakan dalam proses pelatihan di GPU. Selanjutnya terdapat *loss function* yang merupakan fungsi standar untuk klasifikasi *multi* kelas. Dengan menyertakan parameter "weight", model akan menghitung *loss* berdasarkan bobot yang disesuaikan sehingga kontribusi *error* dari kelas minoritas akan diperhitungkan lebih besar daripada kelas mayoritas. Strategi ini sanat efektif dalam meningkatkan sensitivitas model terhadap kelas yang kurang terwakili dan menjaga keseimbangan performa antar kelas. Langkah terakhir adalah konfigurasi *optimizer*, yaitu mekanisme yang digunakan untuk memperbarui bobot model selama proses *training* berdasarkan hasil perhitungan *loss*. Dalam kode pada gambar 3.32 digunakan *optimizer* AdamW, yaitu versi yang telah disempurnakan dari Adam dan direkomendasikan untuk model berbasis *transformer* seperti BERT. *Optimizer* ini mendukung regulasi *weight decay* secara eksplisit, yang dapat membantu mencegah *overfitting*. Nilai *learning rate* yang digunakan adalah 2e-5, yang merupakan nilai umum untuk *fine-tuning* model *pre-training* skala besar, karena nilai *learning rate* yang terlalu tinggi dapat merusak bobot awal hasil *pre-training*.

Tahap pelatihan model (*training loop*) merupakan proses selanjutnya sekaligus menjadi inti dari proses pembelajaran dalam arsitektur *deep learning*. Fungsi "train_epoch" pada gambar 3.33 berfungsi untuk menjalankan satu siklus penuh (*epoch*) pelatihan model dengan menggunakan seluruh data dalam satu *DataLoader*. Fungsi ini menerima beberapa parameter penting, yaitu model, *data loader*, *loss function*, *optimizer,* dan perangkat komputasi.

```python
# 12. Training Loop
def train_epoch(model, data_loader, loss_fn, optimizer, device):
    model.train()
    total_loss = 0
    all_preds = []
    all_labels = []
    all_probs = []

    for batch in tqdm(data_loader, desc="Training"):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['label'].to(device)
        similarity_scores = batch['similarity_scores'].to(device)  # Get similar

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask, similarity_scores)  # Pass sim
        loss = loss_fn(outputs, labels)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        _, preds = torch.max(outputs, dim=1)
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

    avg_loss = total_loss / len(data_loader)
    accuracy = np.mean(np.array(all_preds) == np.array(all_labels))
    return avg_loss, accuracy
```

Gambar 3.33 Implementasi *Training Loop* dengan *Similarity Score*

Proses pelatihan dimulai dengan mengaktifkan mode pelatihan model, yang memastikan bahwa lapisan-lapisan seperti *dropout* dan *batch normalization* berperilaku sesuai dengan kondisi *training*. Kemudian, beberapa variabel ditetapkan, seperti ”total_loss” untuk mengakumulasi nilai kerugian total per *epoch*, serta “all_preds” dan “all_labels” berfungsi menyimpan prediksi dan label aktual dari seluruh *batch* untuk kebutuhan evaluasi metrik akurasi. Iterasi utama dilakukan melalui *looping* yang membaca *batch* data secara progressif dari data *loader* dengan tampilan *progress bar* menggunakan *library* “tqdm”. Sebelum proses *forward pass*, dilakukan pengaturan ulang gradien terlebih dahulu dengan “optimizer” agar nilai gradien dari *batch* sebelumnya tidak terbawa ke iterasi saat ini. Model kemudian menerima *input* dari token, *attention mask*, dan skor kemiripan semantik sebagai *input* dan menghasilkan prediksi. Fungsi kerugian dihitung dengan membandingkan *ouputs* dan *labels* kemudian dilakukan *backpropagation* untuk menghitung gradien dan diikuti dengan

“optimizer.step()” untuk memperbarui bobot model berdasarkan gradien tersebut.

Setelah proses optimasi pada *batch* selesai, nilai *loss* ditambahkan ke “total_loss” dan prediksi label ditentukan dengan “torch.max” yang memilih kelas dengan probabilitas tertinggi. Hasil prediksi dan label aktual kemudian dikonversi ke NumPy dan disimpan untuk evaluasi. Fungsi ini kemudian mengembalikan nilai rata-rata *loss* dan akurasi untuk satu *epoch*, yang dapat digunakan untuk *monitoring* performa model selama proses pelatihan.

Selanjutnya terdapat tahap *evaluation loop* yang merupakan bagian dengan tujuan untuk mengukur performa model terhadap data yang tidak terlihat selama pelatihan. Fungsi “eval_model” pada kode di atas dirancang untuk melakukan evaluasi model pada data uji atau data validasi dengan tetap mempertahankan efisiensi komputasi dan objektivitas evaluasi.

```
# 13. Evaluation Loop
def eval_model(model, data_loader, loss_fn, device):
    model.eval()
    total_loss = 0
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for batch in tqdm(data_loader, desc="Evaluating"):
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['label'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)  # Get similarity_scores from batch

            outputs = model(input_ids, attention_mask, similarity_scores)  # Pass similarity_scores
            loss = loss_fn(outputs, labels)

            total_loss += loss.item()
            _, preds = torch.max(outputs, dim=1)
            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    avg_loss = total_loss / len(data_loader)
    accuracy = np.mean(np.array(all_preds) == np.array(all_labels))

    # Classification Report
    print("\nClassification Report:")
    print(classification_report(all_labels, all_preds, target_names=['Tidak ada hubungan', 'Ada hubungan']))

    # Confusion Matrix
    print("\nConfusion Matrix:")
    print(confusion_matrix(all_labels, all_preds))

    return avg_loss, accuracy
```

Gambar 3.34 Fungsi Evaluasi dan Analisis Klasifikasi Model

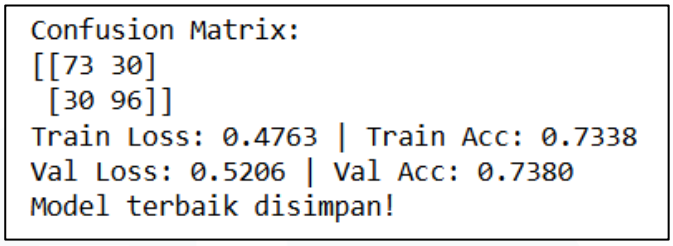
Langkah pertama dalam fungsi ini adalah mengaktifkan mode evaluasi pada model dengan “model_eval()” yang menonaktifkan lapisan tertentu seperti *dropout* dan *batch*

*normalization* yang berperilaku berbeda antara pelatihan dan validasi. Selanjutnya, tiga variabel utama diinisiasi, yaitu "total_loss" untuk akumulasi nilai kerugian, serta "all_preds" dan "all_labels" untuk menyimpan prediksi model dan label sebenarnya secara bertahap. Blok "with torch.no_grad():" digunakan untuk memastikan bahwa proses evaluasi tidak mengaktifkan perhitungan gradien. Hal ini penting karena gradien tidak dibutuhkan pada saat inferensi dan dapat mengurangi konsumsi memori GPU serta mempercepat proses evaluasi. Bagian akhir fungsi ini menyediakan laporan metrik klasifikasi yang lengkap menggunakan fungsi "classification_report" dari "sklearn", yang mencakup *precision*, *recall*, dan *F1-score* untuk setiap kelas. Selain itu *confusion matrix* juga digunakan untuk memberikan gambaran visual tentang distribusi kesalahan model dalam mengklasifikasikan masing-masing kelas. Kedua metrik ini penting untuk memberikan wawasan mendalam tentang performa model, terutama ketika berhadapan dengan data yang tidak seimbang atau klasifikasi yang kompleks. Fungsi evaluasi kemudian mengembalikan nilai rata-rata *loss* dan akurasi evaluasi sebagai indikator kinerja model pada data yang tidak digunakan dalam pelatihan. Tahapan ini memungkinkan untuk mengukur apakah model mengalami *overfitting*, *underfitting*, atau telah belajar dengan baik.

```python
# 14. Train Model dengan Early Stopping
EPOCHS = 10
best_accuracy = 0
patience = 3  # jumlah epoch tanpa peningkatan sebelum menghentikan training
counter = 0   # menghitung jumlah epoch tanpa peningkatan

for epoch in range(EPOCHS):
    print(f"\nEpoch {epoch + 1}/{EPOCHS}")
    train_loss, train_acc = train_epoch(model, train_loader, loss_fn, optimizer, device)
    val_loss, val_acc = eval_model(model, test_loader, loss_fn, device)

    print(f"Train Loss: {train_loss:.4f} | Train Acc: {train_acc:.4f}")
    print(f"Val Loss: {val_loss:.4f} | Val Acc: {val_acc:.4f}")

    # Cek apakah akurasi validasi membaik
    if val_acc > best_accuracy:
        torch.save(model.state_dict(), 'best_model_3.pth')
        best_accuracy = val_acc
        counter = 0  # reset counter jika ada peningkatan
        print("Model terbaik disimpan!")
    else:
        counter += 1
        print(f"Tidak ada peningkatan akurasi. Early stopping counter: {counter}/{patience}")

        if counter >= patience:
            print("Early stopping dilakukan.")
            break
```

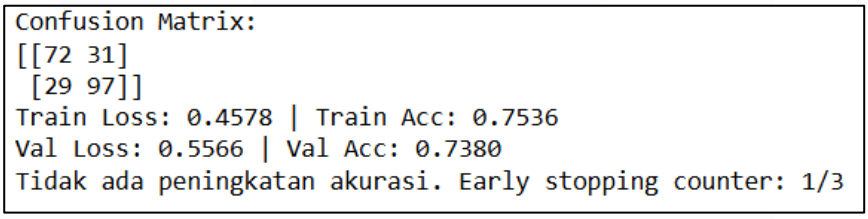
Gambar 3.35 *Training model* dengan *Early Stopping*

Proses selanjutnya adalah pelatihan model yang ditampilkan pada gambar 3.35 yang dilengkapi dengan mekanisme *early stopping*, yang bertujuan untuk menghentikan proses pelatihan secara otomatis jika tidak terjadi peningkatan kinerja pada data validasi selama sejumlah iterasi tertentu. Dalam konteks ini, parameter *EPOCHS* ditetapkan sebanyak 10, menunjukkan batas maksimum iterasi pelatihan, sementara "patience" merepresentasikan jumlah maksimum *epoch* berturut-turut tanpa peningkatan akurasi validasi yang diizinkan sebelum pelatihan dihentikan lebih awal. Terdapat juga variabel "counter" yang digunakan untuk melacak jumlah *epoch* tanpa peningkatan tersebut. Pada setiap iterasi *epoch*, model dilatih menggunakan fungsi "train_epoch" dan performanya dievaluasi melalui fungsi "eval_model". Hasil dari kedua fungsi ini adalah metrik *loss* dan *accuracy* baik untuk data latih maupun data validasi. Informasi ini kemudian dicetak sebagai *logging* untuk memudahkan *monitoring* selama proses pelatihan.



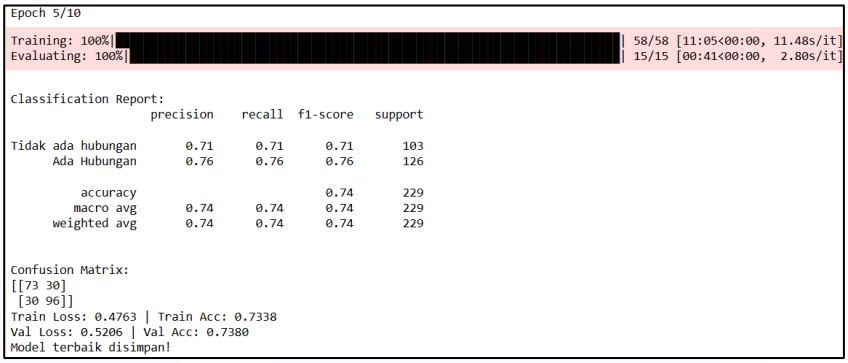

Gambar 3.36 Hasil Evaluasi Model Terbaik

Blok "if" berikutnya memeriksa apakah akurasi validasi yang baru saja dihitung melebihi nilai akurasi terbaik sebelumnya (*best_accuracy*). Jika terjadi peningkatan, maka model disimpan ke dalam *file* menggunakan "torch.save()", nilai "best_accuracy" diperbarui, dan "counter" dimulai kembali dari nol untuk menunjukkan bahwa model mengalami perbaikan performa. Ini merupakan strategi penting untuk mempertahankan model terbaik yang ditemukan selama pelatihan.




Gambar 3.37 Evaluasi *Epoch* dengan *Early Stopping* Aktif

Sebaliknya, jika akurasi validasi tidak menunjukkan peningkatan, nilai "counter" akan bertambah satu. Jika "counter" mencapai nilai "pattience", maka sistem akan mengeluarkan pesan "*Early stopping* dilakukan" dan menghentikan pelatihan menggunakan perintah *break*. Mekanisme ini secara efektif mencegah model dari overfitting akibat pelatihan yang terlalu panjang dan tidak produktif, terutama ketika model telah mencapai titik jenuh dalam proses belajar.




Gambar 3.38 Evaluasi Kinerja Model (*Epoch* ke-5)

Evaluasi performa model pada *epoch* ke-5 menunjukkan hasil yang cukup baik dalam melakukan klasifikasi hubungan kontekstual antar kalimat. Berdasarkan metrik *classification report*, model memperoleh akurasi keseluruhan sebesar 0.74, yang menunjukkan bahwa 74% prediksi yang dihasilkan model pada data uji sesuai dengan label *ground truth*. Metrik ini diperkuat oleh *macro average* dan *weighted average* yang juga berada pada angka 0.74 untuk ketiga metrik utama. Konsistensi antara *macro* dan *weighted average* mengindikasikan bahwa distribusi kelas dalam *dataset* cukup seimbang dan model tidak menunjukkan bias signifikan terhadap kelas tertentu. Jika dianalisis lebih lanjut berdasarkan performa per kelas, label "Tidak ada hubungan" mencatat nilai *precision, recall,* dan *f1-score* sebesar 0.71, sementara label "Ada hubungan" menunjukkan performa yang lebih tinggi dengan skor 0.76 pada ketiga metrik tersebut. Hal ini mengindikasikan bahwa model sedikit lebih efektif dalam mengidentifikasi adanya hubungan antar kalimat dibandingkan dengan mengenali ketiadaan hubungan. Dengan jumlah *support* yang relatif seimbang, perbedaan performa ini dapat menjadi bahan evaluasi lanjutan untuk memperbaiki sensitivitas model terhadap kelas "Tidak ada hubungan".

Informasi lain dari *confusion matrix* menyatakan bahwa dengan 73 data yang diklasifikasikan benar sebagai "Tidak ada

hubungan", 96 diklasifikasikan benar sebagai "Ada hubungan". Sementara masing-masing terdapat 30 kesalahan klasifikasi silang antara dua kelas tersebut. ini berarti *false positive* dan *false negative* terjadi dengan frekuensi yang serupa, namun jumlah *true positive* lebih tinggi pada kelas "Ada hubungan", yang sesuai dengan nilai *recall* dan *f1-score* yang lebih baik untuk kelas tersebut. Dari sisi *loss*, nilai *train loss* sebesar 0.47 dan *validation loss* sebesar 0.52 menunjukkan bahwa model masih mengalami generalisasi yang cukup baik, dengan selisih yang tidak terlalu besar antara performa pada data pelatihan dan data uji. Selain itu, *train accuracy* sebesar 73.38% dan *validation accuracy* sebesar 73.8% mengindikasikan bahwa model tidak mengalami *overfitting* secara signifikan hingga *epoch* ke-5. Model pada titik ini dinyatakan sebagai model terbaik karena berhasil meningkatkan akurasi validasi dibandingkan dengan beberapa *epoch* sebelumnya. Hal ini ditunjukkan oleh pernyataan dalam log sistem yang menunjukkan bahwa parameter model pada *epoch* ke-5 disimpan sebagai representasi terbaik berdasarkan kriteria evaluasi validasi. Maka dari itu, hasil dari *epoch* ini dapat digunakan dalam kesimpulan akhir maupun tahap *deployment*.

Berdasarkan hasil pelatihan yang telah mencapai titik optimal, tahap evaluasi dilakukan untuk memastikan kualitas prediksi model melalui berbagai metrik performa dan visualisasi hasil. Pada tahap evaluasi model, dilakukan proses analisis menyeluruh terhadap performa klasifikasi untuk memastikan bahwa model tidak hanya belajar dari data latih, tetapi juga mampu menyimpulkan pada data uji.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    accuracy_score,
    f1_score,
    precision_score,
    recall_score,
    cohen_kappa_score,
    roc_auc_score,
    RocCurveDisplay
)
```

Gambar 3.39 Inisialisasi *library* evaluasi dan visualisasi model

Kode pada gambar 3.39 merupakan inisialisasi *library* yang digunakan dalam visualisasi dan pengukuran performa model, yang merupakan tahap lanjutan setelah proses pelatihan dan validasi selesai. *Library Matplotlib* dan *Seaborn* diimpor untuk keperluan visualisasi hasil evaluasi, khususnya dalam menggambarkan *confusion matrix* dan *ROC curve* yang memberikan pandangan intuitif terhadap prediksi model terhadap berbagai kelas. Selanjutnya, terdapat fungsi "classification_report", "accuracy_score", "precision_score", "recall_score", dan "f1_score" yang diimpor dari model *sklearn.metrics* digunakan untuk menghitung metrik kunci performa klasifikasi. Kombinasi metrik ini memungkinkan penilaian yang komprehensif terhadap ketepatan prediksi, kemampuan menangkap seluruh hal yang relevan, serta keseimbangan antara keduanya melalui *F1-score*. Untuk kasus klasifikasi biner atau multi-kelas yang cenderung *imbalanced*, penggunaan *precision* dan *recall* menjadi sangat penting untuk menilai performa model secara adil terhadap semua kelas.

Selain itu, fungsi "confusion_matrix" akan memberikan representasi tabular terhadap jumlah prediksi benar dan salah untuk

setiap kelas. Hal ini memungkinkan identifikasi kesalahan spesifik, misalnya jika model sering salah dalam membedakan antara dua kelas tertentu. Metrik "cohen_kappa_score" digunakan untuk mengukur tingkat kesepakatan antara prediksi model dan label aktual dengan mempertimbangkan kemungkinan kesepakatan yang terjadi secara acak, sehingga menjadikannya metrik evaluasi yang kuat dalam konteks klasifikasi yang kompleks. Evaluasi berbasis probabilitas juga dilakukan untuk memberikan informasi penting mengenai pertukaran antara *true positive rate* dan *false positive rate*, yang sangat berguna terutama dalam pengambilan keputusan yang berbasis risiko atau klasifikasi dengan distribusi kelas yang tidak seimbang. *ROC-AUC* menjadi indikator seberapa baik model dapat memisahkan kelas secara umum, dan sering digunakan sebagai metrik tambahan di samping akurasi.

```python
def plot_confusion_matrix(model, data_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            preds = torch.argmax(torch.softmax(outputs, dim=1), dim=1)

            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    cm = confusion_matrix(all_labels, all_preds)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Pred: Tidak Ada', 'Pred: Ada'],
                yticklabels=['Actual: Tidak Ada', 'Actual: Ada'])
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.title('Confusion Matrix')
    plt.show()

    return cm
```

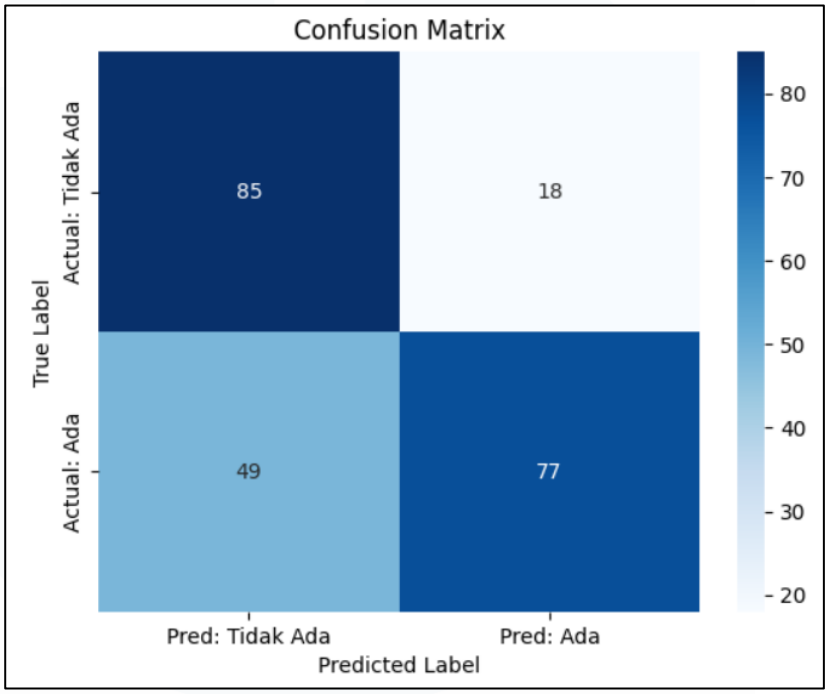
*Gambar 3.40* Fungsi Visualisasi *Confusion Matrix*

Gambar 3.40 menunjukkan fungsi "plot_confusion_matrix", yang digunakan untuk mengevaluasi performa model klasifikasi melalui *confusion matrix*. Fungsi ini berperan penting dalam

memberikan pemahaman mendalam tentang bagaimana model melakukan prediksi terhadap masing-masing kelas, terutama dalam tugas klasifikasi biner. Fungsi ini dimulai dengan mengatur model ke mode evaluasi agar semua lapisan seperti *dropout* dan *batch normalization* tidak aktif selama inference. Kemudian, dua list kosong disiapkan untuk menyimpan hasil prediksi dan label sebenarnya dari seluruh *batch* data. Proses evaluasi dilakukan tanpa menghitung gradien untuk menghemat memori dan mempercepat perhitungan karena tidak diperlukan *backpropagation* pada tahap ini. Dalam *loop batch*, data *input* seperti "inpud_ids", "attention_mask", dan "similarity_scores" diambil dari setiap "data_loader" dan dipindahkan ke perangkat yang sesuai. Selanjutnya, model digunakan untuk menghasilkan *output* prediksi yang kemudian diubah menjadi kelas diskrit menggunakan "torch_argmax" setelah menerapkan *softmax* pada *output model*. Prediksi dan label aktual kemudian diekstrak ke CPU dan dikonversi ke NumPy untuk kemudahan proses berikutnya.

Setelah seluruh prediksi dan label dikumpulkan, fungsi "confusion_matrix" digunakan untuk memperlihatkan jumlah prediksi benar dan salah untuk masing-masing kelas, dan memberikan gambaran langsung tentang jenis kesalahan yang sering terjadi. Contohnya adalah apakah model sering mengklasifikasikan kalimat yang sebenarnya memiliki hubungan sebagai "tidak ada hubungan", atau sebaliknya. Visualisasi matriks dilakukan menggunakan "seaborn.heatmap", dengan skema warna biru dan anotasi nilai numerik. Label sumbu disesuaikan untuk merepresentasikan prediksi dan label aktual dalam bahasa Indonesia, sehingga lebih mudah dipahami oleh audiens atau pemangku kepentingan. Label sumbu X menunjukkan label hasil prediksi model, sedangkan label sumbu Y menunjukkan label sebenarnya dari data. Fungsi ini diakhiri dengan "plt.show()" untuk

menampilkan plot dan *return cm* yang mengembalikan *confusion matrix* sebagai *array* yang memungkinkan pengguna untuk menyimpan atau melakukan analisis lanjutan.




Gambar 3. 41 Visualisasi *Confusion Matrix*

Visualisasi *confusion matrix* pada gambar 3.41 memberikan gambaran kinerja model klasifikasi dalam memprediksi hubungan kontekstual antar kalimat melalui fungsi “plot_confusion_matrix”. Dari hasil evaluasi, terlihat bahwa sebanyak 85 *instance* dari kelas “Tidak ada hubungan” berhasil diprediksi dengan benar oleh model, sementara 18 *instance* dari kelas yang sama salah diklasifikasikan sebagai “Ada hubungan”. Di sisi lain, terdapat 77 *instance* dari kelas “Ada hubungan” yang diprediksi dengan benar, namun 49 *instance* dari kelas ini salah diklasifikasikan sebagai “Tidak ada hubungan”. Distribusi ini menunjukkan bahwa model memiliki kemampuan yang cukup baik dalam mengidentifikasi kalimat yang benar-benar tidak memiliki hubungan. Namun demikian, model masih menghadapi tantangan dalam mengenali kalimat yang memiliki

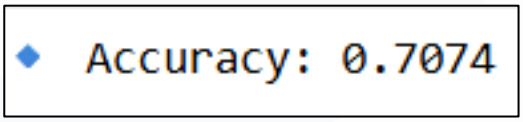
hubungan yang terlihat dari tingginya jumlah *false negative*. Hal ini bisa disebabkan oleh beberapa faktor, seperti kurangnya representasi data pada kelas positif, ambiguitas semantik, atau keterbatasan pemahaman model terhadap konteks tertentu. Secara umum, model cenderung lebih sering memilik prediksi "Tidak ada hubungan" yang bisa dilihat dari total prediksi kelas negatif dibandingkan kelas positif. Ini menunjukkan adanya potensi bias terhadap kelas mayoritas atau kecenderungan model untuk menghindari prediksi positif. Untuk mengatasi hal ini, terdapat beberapa strategi lanjutan yang dapat dipertimbangkan, seperti *class balancing*, augmentasi data untuk kelas minoritas, atau *fine-tuning* dengan *loss function* berbobot untuk meningkatkan sensitivitas model terhadap kelas "Ada hubungan".

Selain melalui *confusion matrix*, evaluasi model juga memerlukan untuk menghitung akurasi model klasifikasi berdasarkan hasil prediksi terhadap data uji menggunakan fungsi "evaluate_accuracy".

```python
from sklearn.metrics import accuracy_score

def evaluate_accuracy(model, data_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            preds = torch.argmax(torch.softmax(outputs, dim=1), dim=1)

            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    accuracy = accuracy_score(all_labels, all_preds)
    print(f"🔹 Accuracy: {accuracy:.4f}")
    return accuracy
```

Gambar 3.42 Fungsi Evaluasi Akurasi Model

Akurasi merupakan metrik evaluasi dasar yang mengukur proporsi prediksi yang benar terhadap seluruh jumlah data. Proses ini sangat penting untuk memberikan gambaran umum mengenai seberapa baik model mengenali pola pada data yang belum pernah dilihat sebelumnya. Fungsi dimulai dengan menetapkan mode evaluasi dan menonaktifkan fitur-fitur khusus agar prediksi stabil. Prediksi yang diperoleh serta label yang sebenarnya dikumpulkan dari tiap *batch* yang dikonversi ke *array NumPy* agar dapat digunakan oleh fungsi evaluasi dari *Scikit-learn*. Setelah seluruh *data batch* diproses, nilai akurasi dihitung menggunakan "accuracy_score" untuk menghitung jumlah prediksi benar dibagi dengan total jumlah data. Nilai akurasi ditampilkan dengan format empat angka desimal dan dikembalikan sebagai *output* fungsi.




Gambar 3.43 Hasil Evaluasi Akurasi Model

Hasil evaluasi model menunjukkan model mencapai akurasi sebesar 0.7074, atau sekitar 70,74% pada data pengujian. Ini berarti bahwa dari seluruh sampel dalam *dataset* uji, sekitar 70% berhasil diklasifikasikan dengan benar oleh model. Nilai akurasi ini memberikan indikasi bahwa model memiliki performa yang cukup baik dalam mengenali pola hubungan antar teks berdasarkan *input* yang diberikan. Meskipun belum mencapai akurasi yang sangat tinggi, skor ini tetap layak dalam konteks *natural language processing* berbahasa Indonesia yang cenderung lebih kompleks akibat morfologi yang kaya dan variasi semantik yang tinggi. Namun, akurasi saja belum cukup untuk memberikan gambaran utuh tentang kinerja model, terutama jika terdapat ketidakseimbangan kelas. Maka dari itu, diperlukan beberapa evaluasi lanjutan

menggunakan metrik tambahan yang lebih sensitif terhadap performa model klasifikasi.

```python
from sklearn.metrics import precision_recall_fscore_support

def evaluate_prf(model, data_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            preds = torch.argmax(torch.softmax(outputs, dim=1), dim=1)

            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    precision, recall, f1, _ = precision_recall_fscore_support(
        all_labels, all_preds, average=None, labels=[0, 1]
    )

    print("🔹 Precision, Recall, F1 per class:")
    print(f"  Kelas 0 (Tidak Ada Hubungan) : Precision={precision[0]:.4f}, Recall={recall[0]:.4f}, F1={f1[0]:.4f}")
    print(f"  Kelas 1 (Ada Hubungan)       : Precision={precision[1]:.4f}, Recall={recall[1]:.4f}, F1={f1[1]:.4f}")

    return precision, recall, f1
```

Gambar 3.44 Fungsi Evaluasi *Precision, Recall, dan F1-score* per Kelas

Pada tahap evaluasi berikutnya, proses difokuskan untuk menghitung dan menampilkan metrik evaluasi yang lebih mendalam, yaitu *precision*, *recall*, dan *f1-score* untuk masing-masing kelas. Ketiga metrik ini sangat penting terutama ketika data yang digunakan memiliki ketidakseimbangan antar kelas, sehingga akurasi saja tidak cukup untuk menggambarkan performa sebenarnya dari model. Fungsi "evaluate_prf" menggunakan "precision_recall_fscore_support" yang memungkinkan evaluasi terpisah untuk setiap label. Fungsi ini pertama-tama mengatur mode evaluasi dan menonaktifkan proses *backpropagation* untuk menghemat memori serta mempercepat inferensi. Model kemudian melakukan prediksi dengan mengeluarkan *logit*, yang kemudian dikonversi menjadi prediksi kelas melalui proses *softmax* dan *argmax*. Setelah seluruh prediksi dan label aktual dikumpulkan, *precision*, *recall*, dan *f1-score* dihitung untuk masing-masing kelas. Hasil evaluasi ini dicetak ke konsol dengan format yang informatif yang memberikan nilai metrik untuk masing-masing label. Fungsi ini berguna dalam memahami seberapa baik model mampu

mendeteksi hubungan antar teks dalam dua skenario, seperti apakah teks memang saling terkait (kelas 1), atau tidak memiliki hubungan (kelas 0).

```
Precision, Recall, F1 per class:
Kelas 0 (Tidak Ada Hubungan) : Precision=0.6343, Recall=0.8252, F1=0.7173
Kelas 1 (Ada Hubungan)       : Precision=0.8105, Recall=0.6111, F1=0.6968
```

*Gambar 3.45* Fungsi Evaluasi *Precision, Recall, dan F1-score* per Kelas

Hasil evaluasi model melalui fungsi "evaluate_prf" menunjukkan performa yang cukup seimbang, namun dengan karakteristik yang berbeda pada masing-masing kelas. Untuk kelas 0 (tidak ada hubungan), model menunjukkan *recall* yang tinggi sebesar 0.8525, yang berarti sebagian besar *instance* kelas 0 berhasil dikenali dengan benar oleh model. Namun demikian, nilai *precision* hanya sebesar 0.6343, yang mengindikasikan bahwa dari seluruh prediksi yang dianggap sebagai kelas 0 oleh model, hanya sekitar 63% yang benar-benar merupakan kelas 0. nilai *f1-score* untuk kelas ini adalah 0.7173, yang menandakan keseimbangan antara *precision* dan *recall*. Sebaliknya, pada kelas 1 (ada hubungan), model menunjukkan *precision* yang tinggi sebesar 0.8105, yang berarti sebagian besar prediksi kelas 1 memang benar adanya. Namun, *recall*-nya cenderung lebih rendah yaitu 0.6111, yang menunjukkan bahwa model gagal mengenali cukup banyak *instance* kelas 1 yang sebenarnya. *F1-score* untuk kelas 1 adalah 0.6968, sedikit lebih rendah dari kelas 0 yang menandakan kesenjangan antara kemampuan model dalam mengidentifikasi dan memastikan keakuratan prediksi kelas ini.

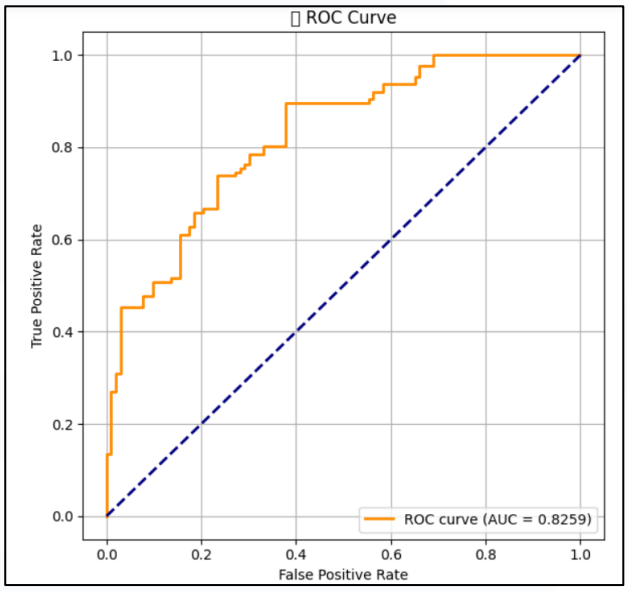
Proses evaluasi berikutnya melibatkan visualisasi dan perhitungan *ROC Curve* (*Receiver Operating Characteristic Curve*) beserta nilai AUC (*Area Under the Curve)* untuk mengevaluasi performa model dalam hal klasifikasi biner secara lebih menyeluruh.

```python
def plot_roc_auc(model, data_loader, device):
    model.eval()
    all_probs = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            probs = torch.softmax(outputs, dim=1)[:, 1]  # Probabilitas kelas 1

            all_probs.extend(probs.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    # Hitung ROC curve
    fpr, tpr, _ = roc_curve(all_labels, all_probs)
    roc_auc = auc(fpr, tpr)

    # Plot
    plt.figure(figsize=(6, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.4f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.grid()
    plt.tight_layout()
    plt.show()
```

Gambar 3.46 Fungsi Visualisasi *ROC Curve* dan Perhitungan *AUC*

Fungsi ini dimulai dengan menetapkan model dalam mode evaluasi dan menonaktifkan proses perhitungan gradien agar evaluasi berjalan lebih efisien tanpa mempengaruhi parameter model. Selanjutnya, model memproses setiap *batch* dari data untuk menghasilkan probabilitas prediksi kelas 1 melalui fungsi *softmax* dan menyimpannya ke dalam *list* ”all_probs”. Label sebenarnya juga dikumpulkan untuk dibandingkan dengan probabilitas prediksi tersebut. setelah seluruh *data batch* diproses, metrik *ROC* dihitung menggunakan “roc_curve” yang akan menghasilkan tiga komponen, yaitu *false positive rate* (FPR), *true positive rate* (TPR), dan *threshold*. Kemudian, nilai AUC dihitung menggunakan fungsi “auc()” sebagai area di bawah kurva ROC tersebut. nilai AUC ini menggambarkan kemampuan model dalam membedakan antar kelas. Nilai AUC mendekati 1 menunjukkan performa klasifikasi yang sangat baik, sementara nilai mendekati 0.5 mengindikasikan performa mendekati acak. Langkah terakhir adalah memvisualisasikan ROC Curve menggunakan “matplotlib”. Kurva

ROC ditempatkan dengan fpr sebagai sumbu X dan tpr sebagai sumbu Y.




Gambar 3.47 Visualisasi *ROC Curve*

Hasil evaluasi model berdasarkan *ROC Curve* menunjukkan bahwa model memiliki performa klasifikasi yang cukup baik dalam membedakan antara dua kelas, yaitu "Tidak ada hubungan" dan "Ada hubungan". Kurva oranye yang membentang dari titik (0,0) hingga (1,1) menggambarkan hubungan antara *true positive rate* (TPR) dan *false positive rate* (FPR) untuk berbagai nilai ambang probabilitas. Model yang ideal akan memiliki kurva yang melengkung tajam ke arah kiri atas grafik, menunjukkan TPR dan FPR rendah yang berarti model mampu mengenali banyak positif sebenarnya sekaligus menghindari prediksi positif palsu. Grafik pada gambar 3.47 menunjukkan nilai AUC sebesar 0.8259, yang berarti bahwa model memiliki peluang besar sekitar 82.6% untuk membedakan secara benar antara sampel positif dan negatif secara acak. Nilai ini berada jauh di atas *baseline* 0.5 yang digambarkan oleh garis diagonal biru sebagai representasi model acak, yang mengindikasikan bahwa model ini memiliki performa diskriminatif

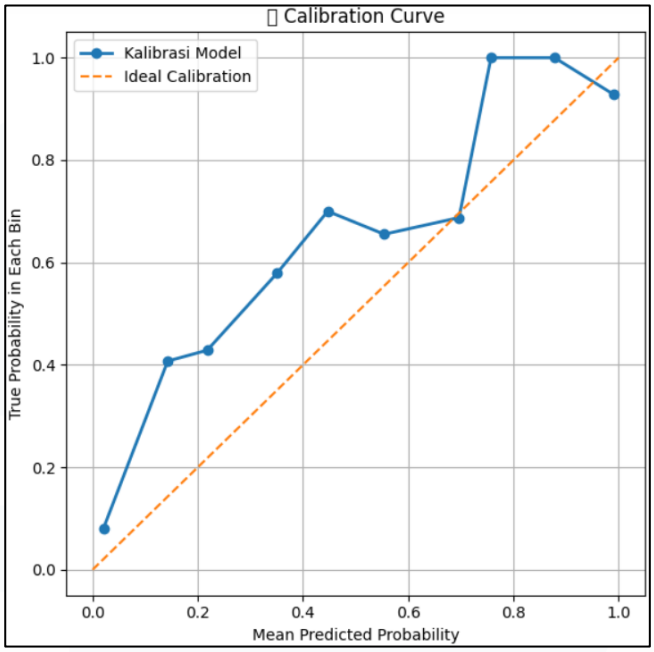
yang baik. secara umum, AUC di atas 0.8 dianggap sangat baik dalam konteks banyak aplikasi klasifikasi. Dengan demikian, berdasarkan *ROC Curve* dan nilai AUC dapat disimpulkan bahwa model memiliki kapabilitas yang cukup kuat dalam mengklasifikasikan data ke dalam dua kelas target dengan tingkat kesalahan yang relatif rendah. Ini menegaskan bahwa model tidak hanya mengandalkan akurasi keseluruhan, tetapi juga memperhatikan keseimbangan sensitivitas yang sangat penting terutama dalam situasi data tidak seimbang.

Proses evaluasi berikutnya adalah visualisasi *Calibration Curve* dan perhitungan *Brier Score*, yang keduanya bertujuan untuk mengukur seberapa baik probabilitas prediksi model mencerminkan kenyataan. Evaluasi ini berfokus pada aspek kalibrasi model, untuk melihat kecocokan antara probabilitas prediksi dengan frekuensi aktual kemunculan kelas positif. Dalam konteks klasifikasi biner, kalibrasi yang baik berarti bahwa jika model memberi probabilitas 0.7 untuk satu sampel, maka sekitar 70% dari sampel serupa memang benar termasuk ke dalam kelas positif.

```python
def plot_calibration_curve(model, data_loader, device):
    model.eval()
    all_probs = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            probs = torch.softmax(outputs, dim=1)[:, 1]  # Probabilitas kelas 1

            all_probs.extend(probs.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    # Brier Score
    brier = brier_score_loss(all_labels, all_probs)
    print(f"Brier Score: {brier:.4f} (semakin rendah, semakin baik)")

    # Calibration Curve
    prob_true, prob_pred = calibration_curve(all_labels, all_probs, n_bins=10)

    plt.figure(figsize=(6, 6))
    plt.plot(prob_pred, prob_true, marker='o', linewidth=2, label='Kalibrasi Model')
    plt.plot([0, 1], [0, 1], linestyle='--', label='Ideal Calibration')
    plt.title("Calibration Curve")
    plt.xlabel("Mean Predicted Probability")
    plt.ylabel("True Probability in Each Bin")
    plt.legend()
    plt.grid()
    plt.tight_layout()
```

Gambar 3.48 Fungsi Visualisasi *Calibration Curve*

Pada bagian pertama evaluasi ini, *Brier Score* dihitung menggunakan fungsi "brier_score_loss". Skor ini mengukur seberapa jauh probabilitas yang diprediksi menyimpang dari label sebenarnya dalam skala 0-1. Nilai *Brier Score* yang semakin rendah menandakan bahwa model memberikan prediksi probabilitas yang lebih tepat atau terkalibrasi dengan baik. Selanjutnya, *Calibration Curve* ditetapkan untuk memvisualisasikan hubungan antara *mean predicted probability* (sumbu X) dan *true probability in each bin* (sumbu Y). Garis putus-putus berwarna merah menggambarkan kalibrasi ideal, di mana nilai probabilitas prediksi sepenuhnya sesuai dengan kenyataan. Sementara itu, garis dengan tanda bulat menunjukkan kalibrasi aktual dari model. Jika garis tersebut dekat dengan garis ideal, maka model memiliki kalibrasi probabilitas yang baik. Sebaliknya, jika terlalu melengkung ke atas atau bawah menunjukkan model *overconfident* atau *underconfident*.




*Gambar 3.49* Visualisasi *Calibration Curve*

Gambar 3.49 merupakan hasil evaluasi model klasifikasi dalam bentuk *Calibration Curve* yang digunakan untuk mengukur seberapa baik probabilitas prediksi model. Kurva pada gambar menunjukkan bahwa pada tentang probabilitas rendah (0.0-0.3), model cenderung *underconfident* sehingga memberikan probabilitas yang lebih rendah dari kenyataannya. Ini berarti bahwa dalam konteks klasifikasi kontekstual, model cenderung terlalu berhati-hati saat menghadapi contoh yang diprediksi rendah walaupun pada kenyataannya kelas positif lebih sering muncul dari yang diperkirakan. di sisi lain, pada kisaran probabilitas menengah (0.3-0.7), kurva model cukup dekat dengan garis ideal. Hal tersebut menandakan kalibrasi yang baik dan memberikan tingkat kepercayaan yang sesuai dengan kenyataan. Namun, pada probabilitas tinggi (di atas 0.75), kurva sedikit melengkung ke atas garis ideal dan menyentuh nilai maksimum sehingga mengindikasikan adanya *overconfidence* yang menyebabkan model terlalu yakin pada prediksi kelas positif yang dibuat.

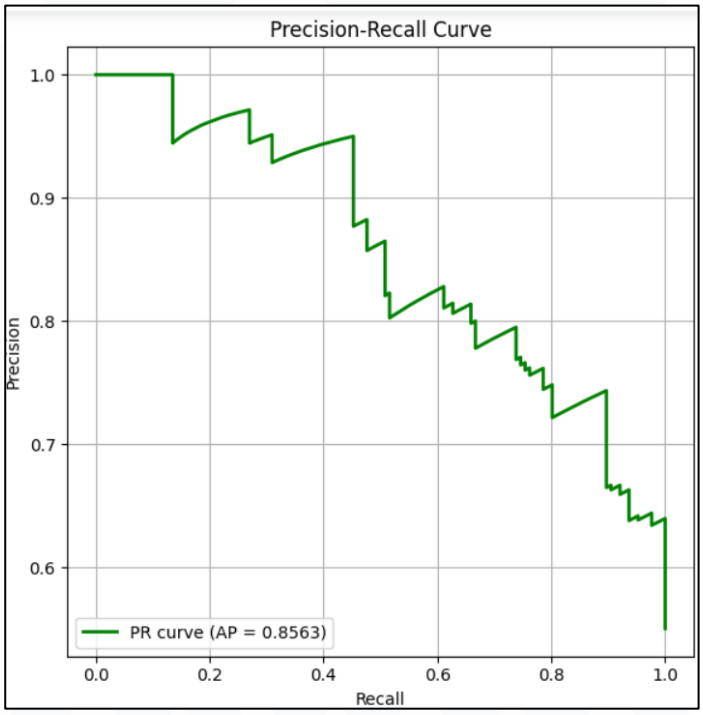
Proses evaluasi selanjutnya adalah visualisasi *precision-Recall Curve* serta perhitungan *Average Precision* yang bertujuan untuk mengevaluasi performa model klasifikasi, khususnya pada kasus data yang tidak seimbang.

```python
def plot_precision_recall_curve(model, data_loader, device):
    model.eval()
    all_probs = []
    all_labels = []

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            similarity_scores = batch['similarity_scores'].to(device)
            labels = batch['label'].to(device)

            outputs = model(input_ids, attention_mask, similarity_scores)
            probs = torch.softmax(outputs, dim=1)[:, 1]

            all_probs.extend(probs.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    # Hitung PR Curve
    precision, recall, _ = precision_recall_curve(all_labels, all_probs)
    avg_prec = average_precision_score(all_labels, all_probs)

    # Plot
    plt.figure(figsize=(6, 6))
    plt.plot(recall, precision, color='green', linewidth=2, label=f'PR curve (AP = {avg_prec:.4f})')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision-Recall Curve')
    plt.legend(loc='lower left')
    plt.grid()
    plt.tight_layout()
    plt.show()
```

Gambar 3.50 Fungsi visualisasi *Precision Recall Curve*

Evaluasi ini dilakukan dengan mendefinisikan fungsi "plot_precision_recall_curve" yang menerima model, *data loader*, dan *device* sebagai parameter. Fungsi ini dimulai dengan menempatkan model dalam mode evaluasi dan menyiapkan *list* kosong untuk menyimpan probabilitas prediksi serta label sebenarnya. Dengan menggunakan "torch.no_grad()", model tidak melakukan perhitungan gradien, sehingga proses evaluasi menjadi lebih efisien. Untuk setiap *batch data*, *input* berupa "input_ds", "attention_mask", dan "similarity_scores" dikirim ke *device* dan menghitung hasil prediksi dari model. *Output* model kemudian diarahkan ke fungsi *softmax* untuk mendapatkan probabilitas, dan hanya probabilitas dari kelas positif yang diambil. Seluruh nilai probabilitas dan label sebenarnya dikumpulkan dan disimpan dalam *list* untuk dihitung metrik evaluasinya. Setelah semua data diproses,

dilakukan perhitungan kurva menggunakan fungsi "precision_recall_curve" dari *Scikit-learn*, yang menghasilkan nilai-nilai *precision* dan *recall* pada berbagai ambang batas (*threshold*). Selain itu, nilai *Average Precision* dihitung menggunakan "average_precision_score" yang memberikan ringkasan kinerja model dalam satu angka, di mana nilai yang lebih tinggi menunjukkan performa yang lebih baik dalam mempertahankan presisi tinggi sambil meningkatkan *recall.*




Gambar 3.51 visualisasi *Precision Recall Curve*

Gambar 3.51 menunjukkan hasil evaluasi model klasifikasi dalam bentuk *Precision-Recall Curve* (PR) dengan sumbu X merepresentasikan nilai *recall* (sensitivitas) dan sumbu Y menunjukkan nilai *precision*. Nilai *recall* menunjukkan seberapa banyak kasus positif yang berhasil terdeteksi oleh model, sementara nilai *precision* menunjukkan seberapa akurat prediksi positif yang dilakukan oleh model. Secara umum, model yang baik akan menghasilkan kurva PR yang tinggi dan cenderung melengkung ke

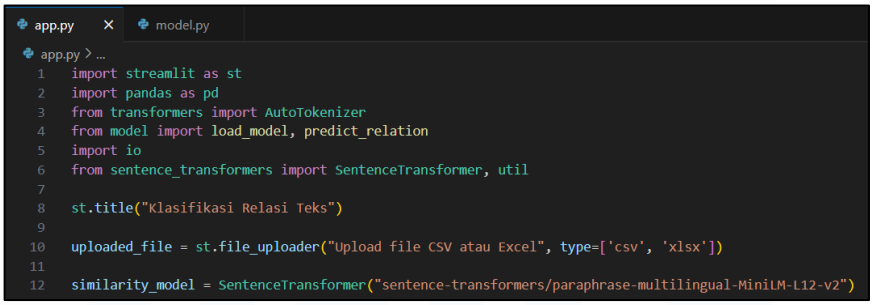
kanan atas yang berarti kombinasi *precision* dan *recall* yang sama-sama tinggi. Berdasarkan visualisasi dari gambar 3.51, kurva menunjukkan performa yang sangat baik dengan nilai *Average Precision* (AP) sebesar 0.8563. nilai AP ini merupakan ringkasan dari area di bawah kurva dan mencerminkan kemampuan model dalam mempertahankan presisi tinggi seiring dengan meningkatnya *recall*. Dengan makna lain, model ini mampu mengidentifikasi banyak kasus positif sambil tetap menjaga kesalahan prediksi positif dalam jumlah yang relatif rendah. Hal tersebut terlihat dari bagian awal kurva yang dimulai dengan *precision* mendekati 1.0 dan tetap stabil cukup lama meskipun *recall* meningkat sebelum akhirnya *precision* mulai menurun secara bertahap.

Berdasarkan rangkaian evaluasi yang telah dilakukan, dapat disimpulkan bahwa model klasifikasi yang dikembangkan menunjukkan performa yang cukup baik dalam mengidentifikasi hubungan kontekstual antar kalimat, meskipun masih terdapat beberapa aspek yang perlu ditingkatkan. Dari hasil *confusion matrix*, model terbukti lebih andal dalam mengenali kalimat tanpa hubungan, namun masih kesulitan mendeteksi kalimat yang memiliki hubungan yang terlihat dari tingginya jumlah *false negative*. Hal ini juga tercermin dalam evaluasi metrik *precision, recall,* dan *f1_score*, di mana *recall* untuk kelas positif relatif lebih rendah dibanding *precision*-nya. Dengan demikian, ada indikasi bahwa model masih melewatkan cukup banyak *instance* yang relevan. Evaluasi melalui *ROC Curve* memperkuat temuan ini dengan nilai AUC sebesar 0.8259, yang menandakan bahwa model memiliki kemampuan diskriminatif yang baik dalam membedakan antara dua kelas. Sementara itu, evaluasi *Calibration Curve* menunjukkan bahwa model memiliki kalibrasi probabilitas yang cukup baik di rentang menengah, namun masih menunjukkan tanda *underconfidence* pada prediksi rendah dan sedikit *overconfident*

pada prediksi tinggi. Terakhir, dari hasil *Precision-Recall Curve,* model menunjukkan kinerja yang sangat baik dengan *Average Precision* sebesar 0.8563, yang mengindikasikan kemampuan yang stabil dalam menjaga keseimbangan antara presisi dan sensitivitas meskipun menghadapi data yang tidak seimbang. Namun, model ini sangat disarankan untuk dilakukan peningkatan pada sensitivitas terhadap kelas positif melalui strategi seperti penyeimbangan data, augmentasi, atau penyesuaian *loss function* untuk menghasilkan model yang lebih seimbang dan optimal dalam berbagai kondisi kontekstual.

#### *3.2.4.2 Deployment*

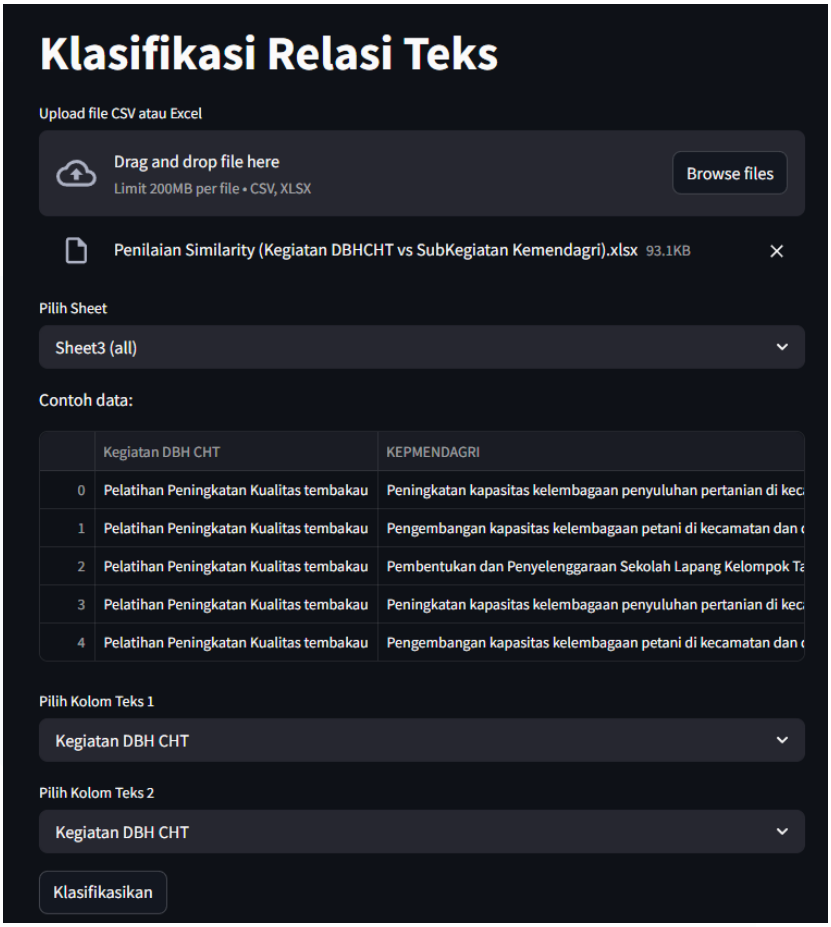
Setelah melalui tahapan eksplorasi data, pemodelan, validasi, serta evaluasi terhadap akurasi dan reliabilitas model, langkah selanjutnya adalah proses *deployment*. Proses ini bertujuan untuk mentransformasikan model yang telah dikembangkan menjadi aplikasi yang dapat digunakan oleh pengguna secara langsung melalui *interface* web.




Gambar 3.52 Potongan kode *app.py*

Gambar 3.52 merupakan *file* "app.py" yang berfungsi sebagai *entry point* utama aplikasi yang mengatur seluruh alur kerja, mulai dari pengunggahan data hingga proses klasifikasi dan pengunduhan hasil. Struktur aplikasi disusun secara modular dengan memisahkan tanggung jawab komponen, seperti manajemen *input file*, *preprocessing data*, inferensi model, dan penyajian hasil.

Arsitektur ini mendukung kemudahan pemeliharaan dan mempercepat pengembangan fitur tambahan di masa depan. *Interface* juga dirancang dengan mengedepankan pengalaman pengguna, memberikan umpan balik visual dan validasi *input* pada setiap tahapan proses untuk mencegah terjadinya kesalahan. Inisialisasi aplikasi dimulai dengan mengimpor berbagai *library* penting, termasuk Streamlit untuk *web interface*, Pandas untuk manupulasi data tabular, *transformers* untuk model IndoBERT, dan "sentence-transformers" untuk perhitungan *similarity embedding*. Masing-masing *library* memiliki peran spesifik dalam *pipeline* aplikasi. Pandas berfungsi menangani struktur data, *transformers* digunakan untuk memuat dan menjalankan model IndoBERT, sedangkan "sentence-transformers" digunakan untuk menghasilkan representasi vektor dari teks untuk menghitung kemiripan semantik antar pasangan teks. Model IndoBERT yang merupakan varian BERT terlatih khusus untuk bahasa Indonesia diimpor melalui *library transformers* untuk mendukung pemahaman bahasa lokal yang lebih baik dibandingkan model multilingual. Sementara itu "paraphrase-multilingual-MiniLM-L12-v2" dari "sentence-transformers" digunakan karena efisiensi dan kemampuannya menghasilkan *embedding* berkualitas tinggi untuk berbagai bahasa, termasuk bahasa Indonesia. Integrasi antar *library* ini dioptimalkan untuk menyeimbangkan antara performa klasifikasi dan efisiensi komputasi.




Gambar 3.53 Halaman *website* menggunakan *streamlit*

Antarmuka pengguna dirancang dengan prinsip kesederhanaan dan kemudahan. *Header* aplikasi ditampilkan untuk memberi konteks langsung kepada pengguna menggunakan "st.title". Alur kerja aplikasi dibuat mengikuti tahapan logis yang saling berurutan, sehingga pengguna hanya dapat melanjutkan ke tahap berikutnya setelah menyelesaikan tahap sebelumnya dengan tujuan untuk menjaga antarmuka tetap bersih dan informatif. Komponen *file uploader* mendukung format CSV dan Excel untuk fleksibilitas pengguna. validasi format dilakukan secara otomatis, termasuk pemilihan *sheet* pada *file* Excel melalui "st.selectbox" yang menampilkan daftar *sheet* secara dinamis. Hal ini memungkinkan pengguna memilih *sheet* yang relevan tanpa perlu *preprocessing* manual, serta menghindari kesalahan pemilihan

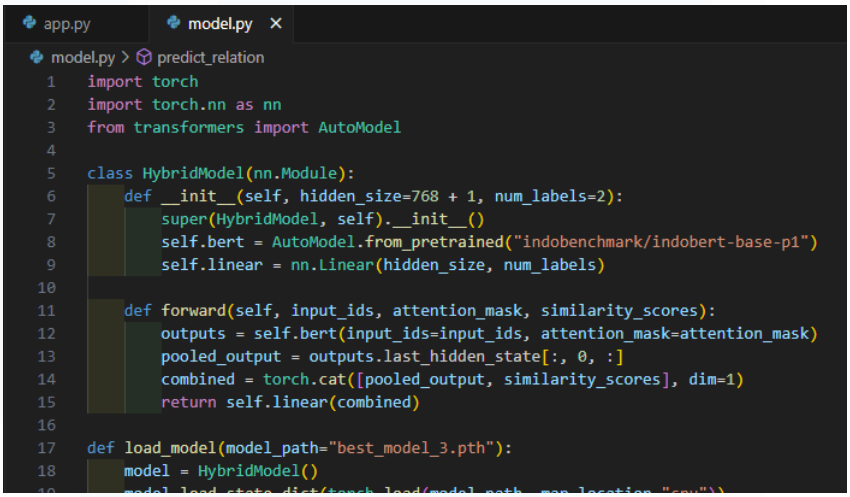
*sheet*. Setelah *file* berhasil dimuat, lima baris pertama ditampilkan menggunakan "st.dataframe()" sebagai *preview* untuk membantu pengguna melakukan verifikasi awal terhadap struktur data, nama kolom, dan konten. *Preview* ini menjadi bagian dari mekanisme pemeriksaan kualitas data sebelum klasifikasi dilakukan. Pengguna kemudian memilih dua kolom teks melalui *dropdown* yang secara otomatis mendeteksi kolom yang tersedia. validasi diterapkan untuk memastikan kolom yang dipilih berbeda dan mengandung teks yang valid. Jika terdapat kesalahan, aplikasi akan memberikan *feedback* yang jelas dan solutif, sehingga meminimalkan potensi *error* selama proses klasifikasi.

```
30    if st.button("Klasifikasikan"):
31        with st.spinner("Memuat model..."):
32            tokenizer = AutoTokenizer.from_pretrained("indobenchmark/indobert-base-p1")
33            model = load_model()
34
35        pred_labels = []
36        confidences = []
37
38        for _, row in df.iterrows():
39            text1 = str(row[col1])
40            text2 = str(row[col2])
41
42            # Hitung similarity
43            emb1 = similarity_model.encode(text1, convert_to_tensor=True)
44            emb2 = similarity_model.encode(text2, convert_to_tensor=True)
45            similarity_score = util.cos_sim(emb1, emb2).item()  # float scalar
46
47            # Prediksi dengan similarity
48            label, conf = predict_relation(model, tokenizer, text1, text2, similarity_score)
49            pred_labels.append(label)
50            confidences.append(f"{conf * 100:.2f}%")
51
52        df['Label Prediksi'] = pred_labels
53        df['Confidence Level'] = confidences
54
55        st.success("Klasifikasi selesai!")
56        st.dataframe(df)
```

Gambar 3.54 Pola *lazy loading* di app.py

Pemanggilan model dilakukan menggunakan pola *lazy loading*, di mana_model IndoBERT dan *tokenizer* hanya dimuat ketika tombol "Klasifikasikan" ditekan. Strategi ini dirancang untuk menghemat memori dan mempercepat waktu *startup* aplikasi. Fungsi "load_model()" memanfaatkan *caching* untuk mencegah pemuatan ulang model yang sama dalam satu sesi yang sangat penting dalam konteks aplikasi interaktif. Model "sentence-

transformers" digunakan sebagai fitur tambahan dengan menghitung skor kemiripan semantik antar teks. Skor ini kemudian digunakan sebagai *input* tambahan untuk model IndoBERT yang melakukan klasifikasi akhir. Pendekatan ini menghasilkan kombinasi kekuatan antara *sentence-level similarity* dan *token-level attention* untuk menciptakan sistem klasifikasi yang lebih akurat.




Gambar 3.55 Potongan kode *model.py*

Dalam proses *deployment* aplikasi klasifikasi relasi teks, "model.py" juga memiliki peran penting sebagai penghubung antara *user interface* dengan model yang telah dilatih sebelumnya. *File* ini mengimplementasikan struktur model klasifikasi yang disebut *HybridModel*, yaitu sebuah arsitektur *hybrid* yang menggabungkan kemampuan representasi semantik dari IndoBERT dengan informasi *similarity score* yang telah dihitung. Pendekatan ini memungkinkan model untuk tidak hanya mempertimbangkan konteks dari pasangan teks, tetapi juga memperkuat keputusan klasifikasi berdasarkan skor kemiripan semantik antara teks yang diuji.

Pada bagian inisialisasi model, *HybridModel* memanfaatkan *AutoModel* dari Hugging Face untuk memuat *pre-trained model* yang telah dilakukan *fine-tune* khusus untuk bahasa Indonesia. *Output* dari model BERT berupa "last_hidden_state" diambil khusus

pada token pertama yang mengandung representasi keseluruhan *input* teks. Representasi ini kemudian dikombinasikan dengan nilai *similarity score* untuk membentuk fitur gabungan yang lebih kaya secara semantik dan numerik. Fitur gabungan ini memiliki dimensi 768+1, dan diproses oleh layer linear terakhir untuk menghasilkan dua nilai logit sebagai *output* hasil akhir klasifikasi biner berupa label 0 dan 1. Fungsi lainnya yang bernama load_model berfungsi untuk memuat parameter model terlatih dari *file* eksternal. Fungsi ini sangat penting dalam konteks *deployment* karena memungkinkan aplikasi untuk langsung menggunakan model yang telah dilatih tanpa perlu melakukan *retraining*. Model dimuat ke dalam mode evaluasi yang memastikan bahwa *dropout* dan layer lainnya dimatikan selama inferensi untuk menjaga konsistensi hasil prediksi.

Kemudian terdapat fungsi lainnya yang menjadi kunci dalam *pipeline* inferensi, dengan nama "predict_relation". Fungsi ini menerima dua teks dan nilai *similarity score* sebagai *input* untuk dilakukan tokenisasi terhadap teks menggunakan *tokenizer* IndoBERT. Tokenisasi dilakukan secara paralel terhadap dua teks dengan hasil berbentuk tensor agar sesuai dengan *input model*. *Similarity score* juga dirancang dalam bentuk tensor dua dimensi dan digabungkan dengan hasil BERT di dalam fungsi *forward* pada model. Setelah model mengeluarkan *output* logis, dilakukan proses *softmax* untuk mengubah logit menjadi probabilitas dan *argmax* untuk memilih label dengan probabiltas tertinggi. Fungsi ini juga mengembalikan *confidence score* yang merupakan probabilitas dari label prediksi terpilih untuk ditampilkan pada *interface* untuk membantu interpretasi hasil.

Setiap pasangan teks dalam *dataset* dikodekan menggunakan *SentenceTransformer* untuk dikonversi menjadi representasi vektor berdimensi tinggi. Representasi ini mempertimbangkan konteks kata dalam kalimat melalui mekanisme *attention* yang memungkinkan

pemahaman semantik yang lebih dalam. Perhitungan skor *similarity* dilakukan dengan *cosine similarity* yang mengukur kesamaan arah antara dua vektor dalam ruang vektor. Nilai *similarity* ini digunakan untuk memperkuat keputusan klasifikasi oleh model dan memberikan informasi tambahan terkait kedekatan semantik dua teks yang dibandingkan. Terdapat juga fungsi "predict_relation" yang bertanggungjawab terhadap proses klasifikasi akhir. Fungsi ini menerima *input* berupa pasangan teks dan skor *similarity* dan mengeluarkan label prediksi serta *confidence score*. *Pipeline* ini mengintegrasikan informasi dari dua sumber utama untuk meningkatkan ketepatan prediksi. *Confidence score* dihitung berdasarkan probabilitas keluaran dari model dan memberikan gambaran tentang tingkat keyakinan model terhadap prediksi yang dibuat. Skor ini berguna untuk proses evaluasi kualitas prediksi serta *manual review*, khususnya untuk kasus prediksi dengan nilai *confidence* yang rendah. Kalibrasi model juga dipertimbangkan untuk memastikan skor probabilitas menggambarkan peluang aktual secara akurat.

| | Kegiatan DBH CHT | KEPMENDAGRI | Label Prediksi | Confidence Level |
|---|---|---|---|---|
| 2 | Pelatihan Peningkatan Kualitas | Peningkatan kapasitas kelembag | 1 | 77.99% |
| 3 | Pelatihan Peningkatan Kualitas | Pengembangan kapasitas kelem | 1 | 77.32% |
| 4 | Pelatihan Peningkatan Kualitas | Pembentukan dan Penyelenggar | 1 | 83.44% |
| 5 | Pelatihan Peningkatan Kualitas | Peningkatan kapasitas kelembag | 1 | 77.99% |
| 6 | Pelatihan Peningkatan Kualitas | Pengembangan kapasitas kelem | 1 | 77.32% |
| 7 | Pelatihan Peningkatan Kualitas | Peningkatan kapasitas kelembag | 1 | 77.99% |
| 8 | Pelatihan Peningkatan Kualitas | Pengembangan kapasitas kelem | 1 | 77.32% |
| 9 | Pelatihan Peningkatan Kualitas | Diseminasi Informasi Teknis, So | 1 | 65.20% |
| 10 | Pelatihan Peningkatan Kualitas | Peningkatan kapasitas kelembag | 1 | 77.99% |
| 11 | Pelatihan Peningkatan Kualitas | Pengembangan kapasitas kelem | 1 | 77.32% |
| 12 | Pelatihan Peningkatan Kualitas | Diseminasi Informasi Teknis, So | 1 | 65.20% |

Gambar 3.56 *Output* dari model klasifikasi kontekstual

*Output* dari aplikasi klasifikasi relasi teks disajikan dalam format tabel yang terdiri dari pasangan teks yang dibandingkan, label prediksi hasil klasifikasi, serta tingkat keyakinan (*confidence level*) terhadap hasil tersebut. kolom Label Prediksi menunjukkan klasifikasi model, di mana nilai 1 menandakan bahwa model

memprediksi adanya hubungan antara kedua teks dan 0 menandakan bahwa model memprediksi tidak adanya hubungan antara kedua teks. selain label, aplikasi juga memberikan informasi *confidence level* terhadap prediksi yang diberikan dan ditampilkan dalam bentuk persentase. Nilai ini mencerminkan probabilitas yang dihitung oleh model terhadap kelas prediksi, dan menjadi indikator penting untuk menilai seberapa yakin model dalam memutuskan adanya relasi antar teks. sebagai contoh, *confidence* sebesar 83.44% menunjukkan bahwa model sangat yakin terhadap keputusannya pada baris tersebut. sementara nilai-nilai seperti 65.20% dapat diinterpretasikan sebagai prediksi yang kurang pasti dan mungkin memerlukan tinjauan manual.

### 3.3 Kendala yang Ditemukan

Dalam menjalankan kegiatan praktik kerja magang, mahasiswa menghadapi beberapa kendala, yaitu:

1) Data yang digunakan dalam *dashboard* untuk analisis trend bersifat dinamis dan berubah setiap harinya, disebabkan oleh waktu penyampaian data yang bervariasi dari masing-masing pemerintah daerah. Hal ini sering kali menghambat kelancaran proses kerja analisis yang dilakukan karena harus terus diperbarui sesuai dengan data terbaru.
2) Penggunaan *PyTorch* dalam mengembangkan model klasifikasi kontekstual berbasis *Natural Language Processing* (NLP) merupakan pengalaman baru, sehingga mahasiswa perlu mempelajari dan mengeksplorasi terkait dengan alur dan penggunaannya.
3) Terjadi kendala teknis pada *ShapeFile* yang digunakan dalam pembuatan visualisasi peta Indonesia di Power BI. Beberapa wilayah tidak dapat ditampilkan dengan baik akibat ketidaksesuaian data

spasial, sehingga mempengaruhi kualitas visualisasi dan interpretasi data geografis.

### 3.4 Solusi atas Kendala yang Ditemukan

Dari seluruh kendala yang dialami oleh mahasiswa selama kegiatan praktik kerja magang berlangsung, terdapat solusi yang dilakukan, antara lain:

1) Untuk mengatasi permasalahan data yang terus berubah setiap hari, tim memutuskan untuk menerapkan *cutoff* data secara rutin. Dengan keputusan ini, analisis dilakukan berdasarkan data yang diterima hingga batas waktu tertentu, sehingga proses kerja menjadi lebih terstruktur dan tidak terganggu oleh perubahan data yang bersifat dinamis.
2) Mahasiswa aktif melakukan pendalaman secara mandiri terkait dengan penggunaan *PyTorch* untuk pengembangan model klasifikasi kontekstual berbasis NLP. Proses ini dilakukan melalui pembelajaran melalui dokumentasi resmi serta eksplorasi *library* yang relevan, sehingga pemahaman terhadap alur kerja dan implementasi teknologi tersebut dapat meningkat secara bertahap.
3) Untuk menyelesaikan permasalahan visualisasi peta Indonesia, mahasiswa melakukan upaya perbaikan terhadap *ShapeFile* dengan menggunakan *Visual Studio Code* (VS Code). Proses ini juga melibatkan perbandingan dengan mengacu pada sumber-sumber geografis resmi lainnya, sehingga visualisasi dapat ditampilkan secara akurat dan menyeluruh.