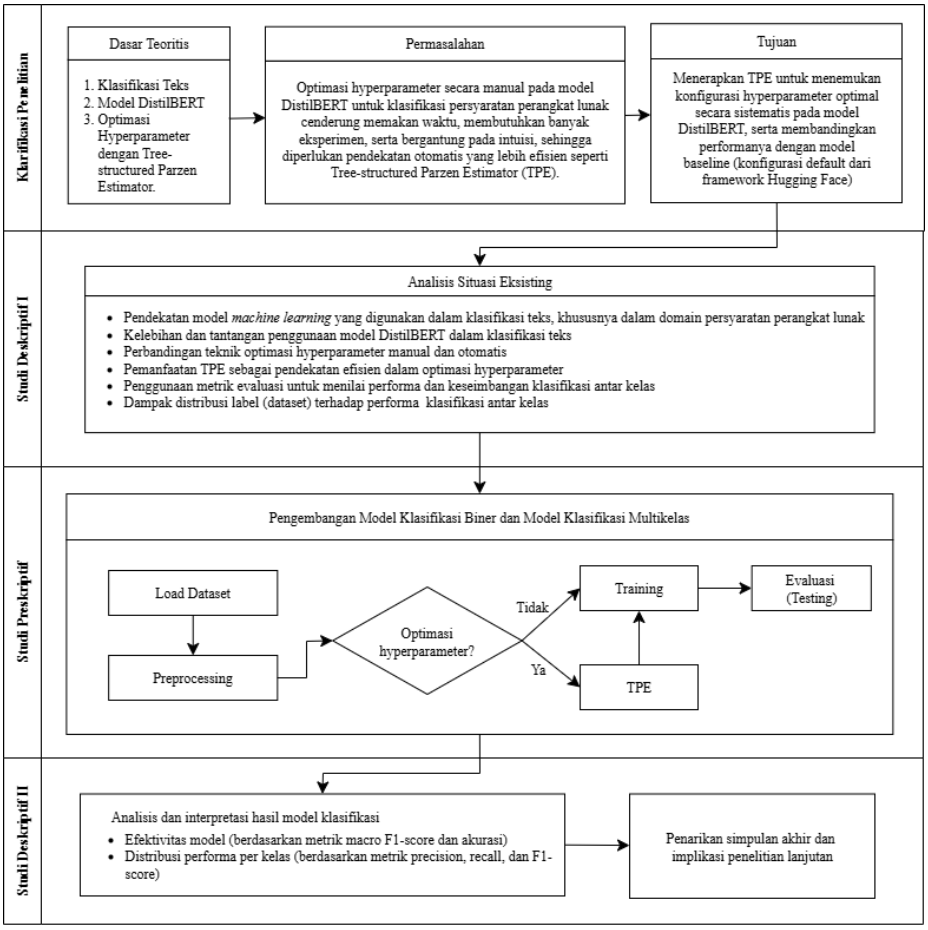
# BAB III
# METODE PENELITIAN

## 3.1 Desain Penelitian

Desain penelitian yang terlihat pada Gambar 3.1 merujuk pada *Design Research Methodology* (DRM). Escudero-Mancebo dkk. (2023) menjelaskan bahwa DRM dapat dimanfaatkan sebagai kerangka kerja untuk mengategorikan dan memahami berbagai aspek penelitian desain teknik, mencakup paradigma penelitian, tujuan penelitian, sifat studi, hingga metode verifikasi. Oleh karena itu, penerapan teknik optimasi hyperparameter dalam penelitian ini akan didasarkan pada data empiris dari tinjauan literatur dan hasil eksperimen. Dengan cara ini, solusi yang diperoleh tidak hanya bersifat teoritis, tetapi juga relevan secara praktis.




Gambar 3.1 Desain Penelitian

### 3.1.1 Klarifikasi Penelitian

Bagian ini bertujuan untuk mengidentifikasi masalah penelitian dan penentuan tujuan yang ingin dicapai (Ali dkk., 2022). Dasar teoritis mencakup kajian mengenai klasifikasi teks, model DistilBERT, serta optimasi hyperparameter menggunakan Tree-structured Parzen Estimator (TPE). Permasalahan yang diangkat berkaitan dengan tantangan optimasi manual yang memakan waktu, bergantung pada intuisi, serta kurang efisien dalam skenario praktis. Oleh karena itu, tujuan utama dari penelitian ini adalah menerapkan TPE sebagai metode optimasi otomatis yang sistematis dan membandingkan performanya dengan model *baseline* yang menggunakan konfigurasi hyperparameter default dari Hugging Face.

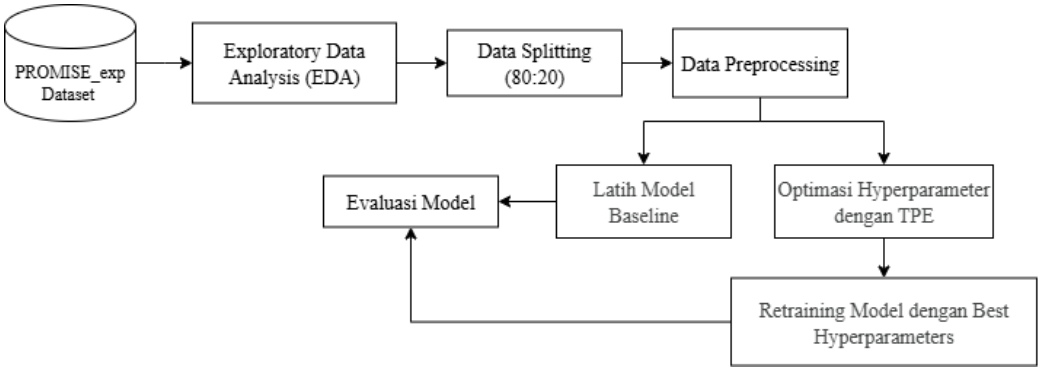
### 3.1.2 Studi Deskriptif I

Bagian ini bertujuan untuk memahami kondisi saat ini melalui pengumpulan dan analisis data yang relevan (Ali dkk., 2022), mencakup peninjauan pendekatan *machine learning* untuk klasifikasi teks, evaluasi kelebihan dan kekurangan model DistilBERT, serta perbandingan teknik optimasi hyperparameter manual dan otomatis dengan fokus pada efisiensi Tree-structured Parzen Estimator. Analisis juga menyoroti pentingnya pemilihan metrik evaluasi yang tepat dan pengaruh distribusi label terhadap performa klasifikasi. Hasil Studi Deskriptif I ini menjadi landasan teoritis dan empiris bagi perumusan masalah dan pengembangan solusi optimasi hyperparameter DistilBERT pada tahap Studi Preskriptif.

### 3.1.3 Studi Preskriptif

Bagian ini merupakan fase perancangan dan implementasi solusi yang memanfaatkan temuan sebelumnya untuk mengatasi masalah yang telah diidentifikasi (Escudero-Mancebo dkk., 2023). Tujuan utamanya adalah mengembangkan dan mengoptimalkan performa model DistilBERT dalam dua skenario klasifikasi, yaitu biner dan multikelas, untuk menghasilkan klasifikasi persyaratan perangkat lunak. Optimasi dilakukan melalui pencarian hyperparameter berbasis Tree-structured Parzen Estimator. Proses ini dirancang

secara sistematis dan terstruktur sebagaimana diilustrasikan pada Gambar 3.2, yang berlaku untuk kedua skenario klasifikasi (biner dan multikelas). Rancangan ini tidak hanya menjadi pedoman dalam pelaksanaan eksperimen, tetapi juga memastikan setiap langkah dapat direplikasi dan divalidasi. Dengan mengikuti proses terstruktur ini, penelitian dapat secara eksplisit menguji dan membandingkan performa model sebelum dan sesudah optimasi hyperparameter, sekaligus memberikan jawaban komprehensif terhadap rumusan masalah penelitian yang telah ditetapkan.




Gambar 3.2 Alur Prosedur Penelitian

### 3.1.3.1 Eksplorasi dan Persiapan Data

Tahap ini diawali dengan eksplorasi dan pemetaan ulang label sesuai kebutuhan dua skenario klasifikasi, yakni biner dan multikelas yang keduanya menggunakan sumber dataset yang sama. Dataset awal terdiri atas 12 label, dengan 1 label *Functional* (F) dan 11 label lainnya merupakan subkategori dari *Non-Functional Requirements* (NFR) (rincian dataset dapat dilihat pada poin 3.2.3). Untuk skenario klasifikasi biner, seluruh label subkategori dari NFR digabung menjadi satu label yakni menjadi NFR, dan label F tidak ada yang berubah sehingga pada skenario klasifikasi biner merepresentasikan klasifikasi persyaratan perangkat lunak secara umum. Sedangkan, skenario multikelas difokuskan pada pengelompokan kategori turunan di dalam kelas NFR, yang mana label F dihapus sehingga hanya menyisakan 11 label dari subkategori NFR. Dataset pada kedua skenario dibagi menjadi *training set* (80%) dan *test set* (20%) untuk memisahkan

data pelatihan dan pengujian secara jelas. Pemilihan rasio 80:20 didasarkan pada temuan empiris Vrigazova (2021) yang menunjukkan bahwa proporsi ini memberikan performa yang stabil tanpa menyebabkan penurunan akurasi yang signifikan, sambil mengoptimalkan efisiensi waktu komputasi dibandingkan dengan proporsi 70:30. Sebagai pembanding, percobaan dengan proporsi 70:30 juga telah dilakukan (lihat Lampiran 2). Hasilnya menunjukkan bahwa meskipun perbedaan akurasi tidak signifikan, rasio 80:20 masih menghasilkan performa yang lebih unggul dan stabil di setiap kelas, dibandingkan dengan 70:30. Temuan ini sejalan dengan hasil Vrigazova (2021) yang menegaskan bahwa 80:20 merupakan pilihan lebih efisien untuk menjaga keseimbangan performa model, terutama ketika dilakukan evaluasi terhadap efektivitas optimasi TPE pada DistilBERT.

Setelah proses pemetaan label, dilakukan pengecekan kualitas dan pembersihan data, serta analisis distribusi dan karakteristik data menggunakan visualisasi sederhana. Selanjutnya, dilakukan encoding label agar dapat diproses oleh algoritma *machine learning*. Pada skenario klasifikasi multikelas, diterapkan pembobotan kelas secara manual pada data latih untuk mengatasi ketidakseimbangan distribusi label, dengan perhitungan bobot yang mengacu pada pendekatan yang digunakan oleh Chaves-Villota dkk. (2024). Pembobotan ini hanya memengaruhi proses pembelajaran model pada data latih tanpa mengubah data uji. Sementara itu, pada klasifikasi biner pembobotan tidak diterapkan.

#### 3.1.3.2 Pengembangan Model *Baseline*

Model *baseline* dikembangkan sebagai tolok ukur awal untuk menilai efektivitas solusi optimasi yang akan diusulkan. Pengembangan model dilakukan dengan membagi data ke set pelatihan dan uji, serta menerapkan model DistilBERT menggunakan hyperparameter bawaan *(default)* dari TrainingArguments Hugging Face. Daftar hyperparameter awal yang digunakan pada model *baseline* ditunjukkan pada Tabel 3.1. Evaluasi awal dilakukan dengan mengukur metrik kualitas model, seperti macro F1-score dan akurasi. Hasil dari *baseline* ini selanjutnya menjadi referensi dalam membandingkan peningkatan kinerja model setelah dilakukan optimasi hyperparameter.

Tabel 3.1 Hyperparameter Awal Model Baseline

| Hyperparameter | Nilai Awal |
|---|---|
| Epochs | 3 |
| Batch Size | 8 |
| Learning Rate | 5e-5 |
| Weight Decay | 0 |

#### 3.1.3.3 Proses Optimasi Hyperparameter dengan TPE

Tahap ini berfokus pada optimasi hyperparameter menggunakan metode Tree-structured Parzen Estimator melalui pustaka Optuna dengan 10 *trial*. Data yang sebelumnya telah dibagi menjadi *training set* (80%) dan *test set* (20%) pada tahap awal, digunakan kembali, di mana *training set* tersebut dipecah lebih lanjut melalui skema validasi silang *Stratified K-Fold* sebanyak 10 lipatan untuk keperluan proses optimasi. Macro F1-score menjadi tujuan utama yang dimaksimalkan selama proses pencarian. Selama proses optimasi, performa model dicatat untuk setiap kombinasi hyperparameter yang diuji, sehingga dapat diperoleh konfigurasi paling optimal sebelum dilakukan pelatihan ulang dan pengujian akhir.

Rentang pencarian hyperparameter yang digunakan dalam tahap ini disajikan pada Tabel 3.2. *Epochs* merupakan jumlah siklus lengkap di mana seluruh dataset training diproses sekali oleh model sebelum satu kali pembaruan parameter selesai (Smith, 2018). Rentang epochs dipilih dalam rentang 3–10 untuk memberi ruang eksplorasi lebih luas dalam menyeimbangkan risiko underfitting dan overfitting (Liu & Wang, 2021). *Batch size* adalah jumlah sampel yang diproses bersamaan sebelum menghitung dan menerapkan gradien untuk pembaruan bobot (Smith, 2018). Nilainya bersifat kategorikal dengan empat opsi yakni 8,16,32, dan 48 yang merepresentasikan variasi dari ukuran kecil hingga menengah untuk mengoptimalkan estimasi gradien dan efisiensi komputasi (Liu & Wang, 2021). *Learning rate* merupakan hyperparameter yang mengontrol seberapa cepat model belajar selama proses pelatihan (Smith, 2018). Rentang 1e-5 hingga 5e-5 diadopsi dari praktik terbaik fine-tuning transformer dengan scheduler warm-up linier dan decay, sehingga update parameter berlangsung halus tanpa lompatan drastis yang merusak bobot pralatih (Y. Wu dkk., 2019). *Learning rate* di bawah 1e-5

memperlambat konvergensi secara signifikan (Milsom dkk., 2025), sedangkan di atas 5e-5 berisiko divergensi, terutama pada lapisan awal model besar (Milsom dkk., 2025). *Weight decay* merupakan bentuk regularisasi L2 yang menambahkan penalti terhadap besarnya bobot ke fungsi *loss* untuk mencegah bobot tumbuh terlalu besar dan mengurangi risiko overfitting (Smith, 2018). Rentangnya menggunakan skala logaritmik dari 1e-4 hingga 1e-1, dengan *baseline* 0.0 untuk memungkinkan evaluasi pengaruh regulasi mulai dari tanpa penalti hingga tingkat regularisasi yang relatif tinggi (Kobayashi dkk., 2024).

Tabel 3.2 Rentang Pencarian Hyperparameter

| Hyperparameter | Rentang Pencarian | Tipe |
|---|---|---|
| Epochs | [3, 10] | Integer (Diskrit) |
| Batch Size | [8, 16, 32, 48] | Kategori |
| Learning Rate | [1e-5, 5e-5] | Float (Skala Log) |
| Weight Decay | [1e-4, 1e-1] | Float (Skala Log) |

#### 3.1.3.4 Pelatihan Ulang Model dan Evaluasi Akhir

Setelah mendapatkan kombinasi hyperparameter terbaik dari hasil Tree-structured Parzen Estimator, semua data pelatihan dan validasi digabungkan, lalu model dilatih ulang dengan pengaturan terbaik tersebut. Setelah training selesai, model dievaluasi menggunakan test set (20% data yang tidak pernah disentuh selama pelatihan). Hasil evaluasi dari pengujian dilaporkan sebagai performa akhir model optimasi, yang akan dibandingkan dengan model *baseline* (konfigurasi hyperparameter *default*).

#### 3.1.3.5 Perbandingan *Baseline* dan TPE

Bagian ini dirancang untuk memberikan evaluasi komparatif antara model baseline dan model hasil optimasi hyperparameter dengan Tree-structured Parzen Estimator. Analisis yang dilakukan meliputi perbandingan metrik utama performa, seperti akurasi, macro F1-score, serta evaluasi performa pada tingkat kelas. Fokus utama tahap ini adalah mengidentifikasi sejauh mana optimasi hyperparameter berbasis Tree-structured Parzen Estimator mampu meningkatkan efektivitas model

dibandingkan baseline, sekaligus memberikan gambaran obyektif mengenai kontribusi solusi yang diusulkan terhadap peningkatan performa klasifikasi persyaratan perangkat lunak. Penjelasan ini menjadi landasan untuk menarik implikasi dan kesimpulan penelitian pada tahapan berikutnya.

### 3.1.4 Studi Deskriptif II

Bagian ini mengevaluasi efektivitas solusi yang telah diusulkan melalui pengujian dan validasi, sehingga dapat menilai sejauh mana tujuan penelitian telah tercapai (Ali dkk., 2022). Peninjauan ini mencakup temuan dari optimasi hyperparameter berbasis Tree-structured Parzen Estimator pada model DistilBERT untuk klasifikasi persyaratan perangkat lunak. Analisis akan dibagi menjadi dua fokus utama, yaitu efektivitas model secara keseluruhan dan analisis distribusi performa model klasifikasi per kelas. Selanjutnya, akan dibahas dampak keseluruhan optimasi Tree-structured Parzen Estimator terhadap model DistilBERT, interpretasi temuan, serta korelasinya dengan literatur dan implikasi praktis di bidang klasifikasi persyaratan perangkat lunak yang sampai akhirnya dapat ditarik simpulan berdasarkan hasil evaluasi tersebut. Rincian langkah teknis pengolahan dan interpretasi data akan dijelaskan lebih lanjut pada sub bab Analisis Data.

#### 3.1.4.1 Analisis Efektivitas TPE terhadap DistilBERT

Pada tahap ini, penelitian akan melakukan analisis efektivitas model DistilBERT dalam tugas klasifikasi persyaratan perangkat lunak dengan menggunakan metrik utamanya yaitu macro F1-score dan akurasi sebagai indikator pendukung. Pendekatan ini dipilih karena macro F1-score memberikan gambaran menyeluruh terhadap performa model pada data yang memiliki distribusi kelas tidak seimbang. Analisis dilakukan dengan membandingkan macro F1-score dan akurasi antara model sebelum dan sesudah optimasi hyperparameter menggunakan Tree-structured Parzen Estimator. Hasil analisis ini akan menjadi dasar untuk menilai sejauh mana optimasi hyperparameter mampu meningkatkan efektivitas model, serta secara langsung menjawab rumusan masalah pertama.

#### 3.1.4.2 Analisis Performa Model pada Level Kelas

Tahap ini bertujuan untuk menganalisis distribusi performa model DistilBERT tiap kelas, baik sebelum maupun sesudah optimasi hyperparameter. Analisis ini penting dilakukan mengingat distribusi label pada data persyaratan perangkat lunak cenderung tidak seimbang. Analisis dilakukan dengan mengevaluasi metrik precision, recall, dan F1-score pada masing-masing kelas. Melalui analisis ini, akan diidentifikasi sejauh mana optimasi hyperparameter berbasis TPE berkontribusi pada peningkatan keseimbangan performa model di seluruh kelas. Temuan dari tahap ini diharapkan dapat memberikan jawaban yang komprehensif terhadap rumusan masalah kedua.

## 3.2 Alat dan Bahan Penelitian

Penelitian ini memanfaatkan seperangkat alat dan bahan untuk mendukung implementasi model, optimasi hyperparameter, dan evaluasi hasil. Berikut merupakan komponen utamanya.

### 3.2.1 *Hardware*

- Intel® Core™ i3-10110U CPU @ 2.10GHz
- RAM 8GB

### 3.2.2 *Software* & *Library*

- Google Colab (runtime type: Python 3)
- Microsoft Excel
- Hugging Face Transformers (v4.51.3)
- Optuna (v4.3.0)
- Scikit-learn (v1.6.1)
- Pandas (v2.2.2)
- Numpy (v2.0.2)
- Matplotlib (v.3.10.0)
- Seaborn (v.0.13.2)
- Datasets (v.4.0.0)
- Torch (v.2.6.0)

### 3.2.3 Dataset

Objek penelitian ini menggunakan dataset PROMISE_exp sebagai basis data persyaratan perangkat lunak yang telah berlabel. Dataset ini telah digunakan juga dalam beberapa penelitian serupa sebelumnya (Binkhonain & Zhao, 2023; Saleem dkk., 2023). PROMISE_exp merupakan dataset hasil perluasan dari dataset PROMISE yang awalnya diperkenalkan oleh Cleland-Huang dkk. (2007). Perluasan ini dilakukan oleh Lima dkk. (2019) sebagai respons terhadap keterbatasan jumlah data pada dataset PROMISE asli yang dinilai terlalu kecil untuk kebutuhan pengembangan dan evaluasi metode *machine learning* dalam klasifikasi persyaratan perangkat lunak. Proses perluasan dataset dilakukan oleh Lima dkk. (2019) dengan cara mengumpulkan dokumen spesifikasi persyaratan perangkat lunak (SRS) tambahan, khususnya dokumen berbahasa Inggris menggunakan mesin pencari Google, kemudian dianalisis manual, ekstraksi, klasifikasi, dan validasi tipe kebutuhan melalui konsensus pakar.

Tabel 3.3 Karakteristik dataset PROMISE_exp

| **Kategori** | | **Label** | **Jumlah** |
|---|---|---|---|
| **Fungsional** | Functional | F | 444 |
| **Non-fungsional** | Security | SE | 125 |
| | Usability | US | 85 |
| | Operability | O | 77 |
| | Performance | PE | 67 |
| | Look & feel | LF | 49 |
| | Availability | A | 31 |
| | Maintainability | MN | 24 |
| | Scalability | SC | 22 |
| | Fault Tolerance | FT | 18 |
| | Legal | L | 15 |
| | Portability | PO | 12 |
| **Total** | | | 969 |

PROMISE_exp terdiri dari 969 persyaratan perangkat lunak berbahasa Inggris yang telah diberi label, dikumpulkan dari 47 dokumen spesifikasi kebutuhan perangkat lunak (Software Requirements Specification/SRS) yang berbeda. Jumlah ini meningkat cukup signifikan dibandingkan dengan dataset PROMISE asli yang hanya memuat 625 persyaratan dari 15 dokumen. Setiap persyaratan diklasifikasikan ke dalam 12 kelas, yaitu 1 kelas kebutuhan fungsional dan 11 kelas kebutuhan non-fungsional, untuk detail distribusi jumlah data per kelas disajikan pada Tabel 3.3.

## 3.3 Instrumen Penelitian

Instrumen penelitian yang digunakan dalam studi ini terdiri dari serangkaian metrik evaluasi yang secara khusus diimplementasikan untuk mengukur performa model klasifikasi pada dataset persyaratan perangkat lunak. Setiap metrik tidak hanya dipilih berdasarkan relevansi teoritisnya, tetapi juga karena kemampuannya merefleksikan kekuatan dan kelemahan model dalam mengklasifikasikan berbagai kategori persyaratan perangkat lunak yang terdapat dalam data.

### 3.3.1 Presisi

Presisi mengukur proporsi prediksi kelas tertentu yang benar-benar sesuai dengan label aslinya (Grandini dkk., 2020). Dalam konteks penelitian ini, misalnya presisi untuk kelas F (fungsional) menunjukkan seberapa banyak prediksi F yang memang benar-benar F di data uji. Nilai presisi yang tinggi mengindikasikan bahwa model jarang memberikan prediksi kelas positif yang keliru *(false positive)* untuk kelas tersebut.

### 3.3.2 Recall

Recall mengukur proporsi data yang sebenarnya termasuk dalam kelas tertentu yang berhasil dikenali dengan benar oleh model (Grandini dkk., 2020). Dalam konteks penelitian ini, recall untuk kelas NF (non-fungsional) berarti persentase persyaratan NF yang berhasil diprediksi sebagai NF dari seluruh persyaratan NF di data uji. Nilai recall yang tinggi berarti model jarang melakukan kesalahan *false negative* untuk kelas tersebut, yaitu gagal mengenali data yang seharusnya termasuk NF.

### 3.3.3 F1-score

F1-score mengukur keseimbangan antara presisi dan recall untuk kelas tertentu (Grandini dkk., 2020). Dalam konteks penelitian ini, F1-score dihitung untuk setiap kelas guna menilai performa model secara seimbang, baik dari sisi ketepatan prediksi maupun kemampuan mengenali seluruh sampel di dataset uji. Misalnya, meskipun presisi suatu kelas tinggi, jika recall-nya rendah, F1-score akan berada pada nilai sedang yang menunjukkan adanya *trade-off* antara keduanya.

### 3.3.4 Akurasi

Akurasi mengukur proporsi total prediksi yang benar terhadap seluruh sampel di dataset uji (Grandini dkk., 2020). Nilai akurasi dapat memberikan gambaran umum kinerja model, namun metrik ini dapat kurang representatif jika distribusi jumlah data antar kelas tidak seimbang (Grandini dkk., 2020). Oleh karena itu, akurasi digunakan sebagai pelengkap terhadap metrik lain seperti F1-score dan macro F1-score yang lebih memperhatikan performa tiap kelas.

### 3.3.5 Macro F1-score

Macro F1-score mengukur rata-rata F1-score dari seluruh kelas tanpa mempertimbangkan jumlah sampel di setiap kelas *(unweighted mean)* (Grandini dkk., 2020). Dalam konteks penelitian ini, macro F1-score digunakan untuk memberikan penilaian yang adil terhadap performa model terutama pada skenario klasifikasi dengan distribusi kelas yang tidak seimbang (Rainio dkk., 2024). Metrik ini memberikan bobot yang sama untuk kelas mayoritas maupun minoritas, baik dalam skenario klasifikasi biner maupun multikelas.

## 3.4 Analisis Data

Analisis data dalam penelitian ini bertujuan untuk mengevaluasi dan menginterpretasikan hasil eksperimen mengenai pengaruh Tree-structured Parzen Estimator sebagai metode optimasi hyperparameter terhadap kinerja model DistilBERT pada tugas klasifikasi persyaratan perangkat lunak. Analisis ini dilaksanakan sesuai dengan kerangka evaluasi yang telah dijabarkan pada bagian Studi Deskriptif II.

### 3.4.1 Analisis Pengaruh TPE terhadap Model DistilBERT Secara Global

Analisis ini difokuskan untuk menilai sejauh mana optimasi hyperparameter dengan metode Tree-structured Parzen Estimator dapat mengoptimasi kinerja model DistilBERT pada tugas klasifikasi persyaratan perangkat lunak. Evaluasi dilakukan dengan menggunakan metrik macro F1-score dan akurasi untuk membandingkan performa model sebelum dan sesudah dilakukan optimasi hyperparameter. Perbandingan ini bertujuan untuk mengidentifikasi peningkatan atau penurunan kinerja model secara keseluruhan, serta menilai efektivitas TPE sebagai strategi optimasi hyperparameter pada skenario klasifikasi biner maupun multikelas.

### 3.4.2 Analisis Pengaruh TPE terhadap Model DistilBERT pada Tingkat Kelas

Analisis ini bertujuan untuk mengidentifikasi sebaran performa model terhadap masing-masing kelas pada dataset, baik untuk skenario klasifikasi biner maupun multikelas. Evaluasi dilakukan menggunakan metrik presisi, recall, dan F1-score pada tingkat kelas, yang disajikan melalui diagram batang serta visualisasi *confusion matrix*. Melalui analisis ini, dapat diketahui kelas mana yang diprediksi dengan baik oleh model dan kelas mana yang cenderung mengalami kesalahan prediksi. Hasil analisis ini diharapkan memberikan gambaran yang lebih komprehensif mengenai kekuatan dan kelemahan model pada setiap kelas, sehingga dapat menjadi acuan dalam perbaikan performa model di masa mendatang.