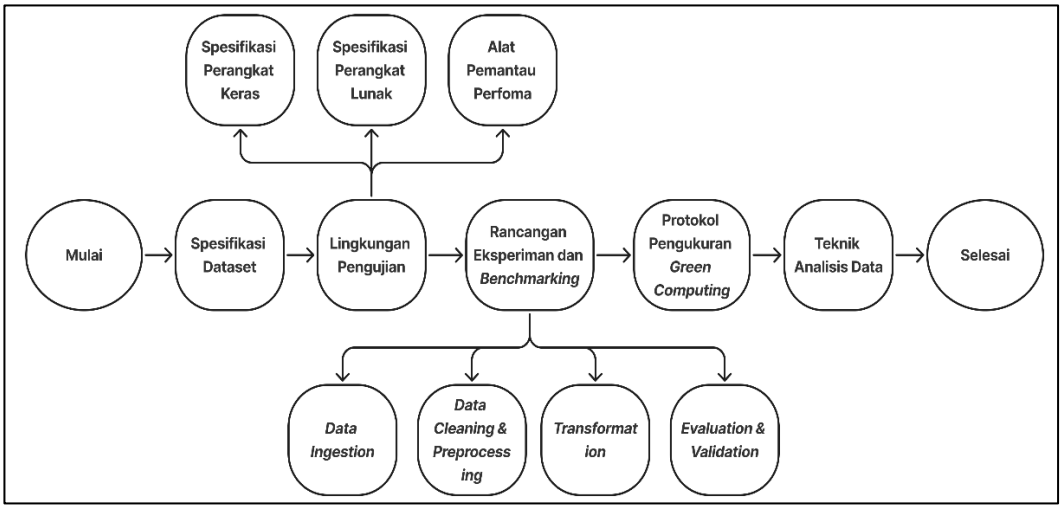
BAB III

METODOLOGI PENELITIAN

3.1 Metodologi

Metode komparatif digunakan untuk membandingkan performa dua platform analitik, *R* (menggunakan *RStudio*) dan *Python*, dalam konteks *Big Data Analytics* berbasis *Green Computing*. Empat metrik utama digunakan untuk membandingkan performa, yaitu waktu eksekusi, utilisasi *CPU*, penggunaan memori, dan konsumsi energi. Metrik-metrik ini berfungsi sebagai indikator performa dan efisiensi penggunaan sumber daya komputasi.

Dalam penelitian ini, istilah *Big Data Analytics* mengacu pada pendekatan *pipeline* analitik, sedangkan analitik data skala menengah diuji pada perangkat kelas menengah sesuai dengan rancangan eksperimen. Seluruh pengujian dilakukan pada *testbed* yang sama untuk menjaga keterkendalian faktor lingkungan. Setiap skenario pengujian pada masing-masing platform dieksekusi sebanyak tiga kali. Hasil pengujian kemudian diringkas menggunakan nilai rata-rata (*mean*) untuk meningkatkan reliabilitas dan mengurangi pengaruh variabilitas sistem.



Gambar 3.1 Alur Penelitian

3.2 Spesifikasi Dataset

Penelitian ini menggunakan *"AI4I 2020 Predictive Maintenance"*, sebuah dataset sintetis yang merepresentasikan skenario *predictive maintenance* pada lingkungan manufaktur dan disediakan sebagai alternatif ketika data dunia nyata sulit diperoleh serta dibagikan secara terbuka (Matzka, 2020). Dataset yang digunakan berisi 10.000 observasi dengan 14 atribut, serta digunakan sebagai input yang sama pada *R* (melalui *RStudio*) dan *Python* untuk menjaga kesetaraan data uji.

Berdasarkan karakteristik dataset, data tidak memiliki *missing value*. Struktur data mencakup atribut identitas, atribut kategorikal, variabel proses, serta label kegagalan mesin beserta lima jenis kegagalan spesifik (*TWF, HDF, PWF, OSF, dan RNF*). Meskipun ukuran data berada pada skala menengah, dataset ini memadai untuk menguji perbedaan efisiensi eksekusi *pipeline* analitik dan pemanfaatan sumber daya pada kedua platform dalam lingkungan komputasi personal. Karakteristik tersebut mendukung pelaksanaan pengukuran parameter *Green Computing* pada kedua platform.

3.3 Lingkungan Pengujian

Lingkungan pengujian dirancang menggunakan satu *testbed* komputasi yang sama untuk seluruh eksperimen agar perbandingan performa antara *R* (melalui *RStudio*) dan *Python* berlangsung secara terkontrol dan konsisten. Tujuan pengendalian ini adalah untuk mengurangi pengaruh elemen luar yang dapat mengurangi kredibilitas hasil pengukuran. Untuk mengukur parameter *Green Computing*, lingkungan pengujian terdiri dari tiga komponen utama spesifikasi perangkat keras, konfigurasi perangkat lunak, dan alat pemantau performa.

3.3.1 Spesifikasi Perangkat Keras

Tabel 3.1 Spesifikasi Perangkat Keras

Komponen	**Spesifikasi**
Prosesor	Intel Core i5-8250U @ 1.60 GHz
RAM	8GB DDR4
Penyimpanan	SSD 256GB
Sistem Operasi	Windows 11 Home

Sebagai hasil dari spesifikasi perangkat keras yang digunakan dalam penelitian ini, lingkungan komputasi personal kelas menengah yang memiliki keterbatasan sumber daya dapat diwakili secara realistis. Konfigurasi ini tidak dimaksudkan untuk merepresentasikan komputasi *Big Data* berskala *enterprise*, melainkan untuk menilai performa *R* (melalui *RStudio*) dan *Python* pada perangkat dengan sumber daya relatif terbatas, khususnya dari sisi memori dan prosesor. Dengan demikian, fokus penelitian adalah untuk mengevaluasi performa dan

efisiensi pemanfaatan sumber daya pada perangkat kelas menengah. Untuk menilai sensitivitas performa terhadap variasi spesifikasi perangkat keras, pengujian dapat dilakukan pada perangkat dengan kapasitas RAM lebih besar, misalnya lebih dari 16 GB.

3.3.2 Spesifikasi Perangkat Lunak

Tabel 3.2 Spesifikasi Perangkat Lunak

Kategori	***R* (melalui *RStudio*)**	***Python***
Versi/Platform	*R* 4.5.1	*Python* 3.12.9
IDE/*Environment*	*RStudio* 2023.09.1	*Jupyter Notebook* 7.4.5
Pustaka	*dplyr, data.table*	*pandas, polars*

Pemilihan versi perangkat lunak dan pustaka (*library*) pada kedua platform berdasarkan ketersediaan rilis stabil pada periode penelitian serta tingkat adopsi yang luas oleh komunitas, sehingga mengurangi risiko inkonsistensi akibat perubahan atau *bug* yang belum matang. Selain itu, pustaka yang digunakan dipilih karena mendukung operasi analisis data secara efisien pada dataset berskala menengah hingga besar. Untuk menjaga keadilan komparasi, pustaka pada lingkungan *R* (melalui *RStudio*) dan *Python* dipetakan secara fungsional agar operasi yang setara dapat diimplementasikan secara optimal pada masing-masing ekosistem.

Penelitian ini mengolah dan mengubah data di ekosistem *R* (melalui *RStudio*) menggunakan pustaka dplyr dan data.table. Pada sisi *Python*, *pandas* adalah pustaka pengolahan data standar, dan *polars* adalah pustaka alternatif yang berfokus

pada meningkatkan efisiensi dan performa *preprocessing dataframe*. Pemilihan perangkat lunak dan pustaka dilakukan dengan mempertimbangkan seberapa performa dan efisien sumber daya digunakan dalam lingkungan komputasi personal kelas menengah, menurut prinsip *Green Computing* (Sneha, dkk., 2021).

3.3.3 Alat Pemantau Performa

Dalam penelitian ini, pendekatan hibrida digunakan untuk mengukur parameter *Green Computing*. Untuk utilisasi *CPU* dan penggunaan memori, pengukuran dilakukan pada level proses (*process-level*) menggunakan instrumen berbasis skrip yang tertanam dalam kode *R* dan *Python*. Pendekatan ini memastikan bahwa hanya sumber daya yang digunakan oleh proses *R* dan *Python* yang diukur, tanpa terpengaruh oleh aktivitas sistem operasi atau aplikasi lain yang berjalan bersamaan.

Berbeda dengan itu, konsumsi energi diukur menggunakan perangkat lunak eksternal *Intel Power Gadget* (IPG) pada level paket *CPU* (*CPU package-level*). IPG memantau konsumsi energi seluruh prosesor, bukan hanya proses *R* atau *Python* secara spesifik. Untuk meminimalkan bias pengukuran energi, lingkungan pengujian dijaga agar hanya menjalankan proses *R* atau *Python* saat pengukuran berlangsung, dengan menutup aplikasi lain yang tidak diperlukan dan memastikan tidak ada beban komputasi signifikan lainnya yang berjalan bersamaan. Namun, perbedaan konsumsi energi yang teramati dapat diasumsikan sebagai cerminan dari karakteristik eksekusi *pipeline* analitik pada masing-masing platform. Selama eksekusi *pipeline*, pencatatan *CPU* dan memori dilakukan dengan interval sampling 0,5 detik.

Tabel 3.3 Alat Pemantau Performa dan Konfigurasi Pencatatan

Parameter	***R* (melalui *RStudio*)**	***Python***	**Level Pengukuran**	**Interval Sampling**	**Alat Eksternal**
Waktu Eksekusi	`Sys.time()` dan `proc.time()`	`time.time()` dan `timeit.default_timer()`	*Run-level*	–	–
Utilisasi *CPU*	`ps::ps_cpu_times()`	`psutil.Process().cpu_percent()`	*Process-level*	0,5 s	–
Penggunaan Memori	`ps::ps_memory_info()$rss`	*`psutil.Process().memory_info().rss`*	*Process-level*	0,5 s	–
Konsumsi Energi	–	–	*CPU package-level*	Mengikuti interval *logging* IPG	*Intel Power Gadget* 3.5

Pengukuran pada level proses (*process-level*) dipilih untuk memastikan bahwa hanya sumber daya yang digunakan oleh proses *R* dan *Python* yang diukur, tanpa terpengaruh oleh aktivitas sistem operasi atau aplikasi lain yang berjalan

bersamaan. Pendekatan ini memberikan perbandingan yang lebih adil dan terkontrol antara kedua platform. Pemilihan level proses dilakukan agar perbandingan *CPU* dan memori tidak terdistorsi oleh beban aplikasi lain di luar proses pengujian.

3.4 Rancangan Eksperimen dan *Benchmarking*

Eksperimen dirancang untuk membandingkan performa *R* dan *Python* pada *pipeline* analitik yang terkontrol. Untuk melakukan *benchmarking*, tahapan kerja yang sama digunakan pada dataset *"AI4I 2020 Predictive Maintenance"*, dengan urutan prosedur yang sama pada kedua platform. Sintaks dan mekanisme internal pustaka tetap berbeda sesuai dengan karakteristik ekosistem. Namun, tujuan operasi, struktur fitur-target, dan keluaran antara diterapkan sehingga beban kerja bersifat sebanding. Seluruh prosedur yang melibatkan randomisasi dikendalikan dengan *seed* 42 untuk menjaga keterulangan hasil pada kedua platform.

3.4.1 *Data Ingestion*

Pada tahap ini, dataset dimasukkan ke dalam struktur *dataframe* pada masing-masing platform dari berkas *CSV*. Setelah data dimasukkan, penamaan kolom diseragamkan agar pemanggilan atribut pada tahap berikutnya dapat dilakukan secara konsisten di kedua lingkungan.

3.4.2 *Data Cleaning & Preprocessing*

Sebelum transformasi dan pemodelan, langkah ini dilakukan untuk memastikan bahwa data berada dalam kondisi yang layak untuk analisis dan setara secara metodologis. Proses terdiri dari:

a. Menghapus atribut identitas dan non-fitur,

b. Menempatkan variabel kegagalan mesin sebagai label klasifikasi,
c. Menghapus *TWF*, *HDF*, *PWF*, *OSF*, dan *RNF* dari matriks fitur (X) agar tidak terjadi label *leakage*, mengingat kolom-kolom tersebut merepresentasikan subkategori dari target *Machine failure*.

Sebelum memasuki tahap transformasi, data pada *R* (melalui *RStudio*) dan *Python* berada pada struktur fitur-target yang sebanding berkat perlakuan yang konsisten tersebut.

3.4.3 *Transformation*

Data disiapkan untuk digunakan dalam proses pemodelan pada tahap transformasi. Tahap ini melibatkan operasi utama seperti:

a. *Encoding* variabel kategorikal jenis menjadi variabel numerik dengan pemetaan yang setara,
b. Menyusun matriks fitur (X) dan target (y) yang konsisten,
c. Memastikan seluruh fitur berada dalam format numerik yang dapat diproses oleh algoritma klasifikasi pada kedua platform.

3.4.4 *Evaluation & Validation*

Dalam proses *benchmarking*, tahap evaluasi dan validasi merupakan tugas komputasi akhir. Untuk memastikan proporsi kelas pada kedua subset data, pembagian stratifikasi berdasarkan kelas target digunakan untuk membagi data menjadi data latih dan data uji dengan rasio 70:30. Teknik *oversampling SVM-SMOTE* pada kedua platform digunakan untuk mengatasi ketidakseimbangan kelas pada data latih, dengan pengendalian *seed* dan proses randomisasi dapat direplikasi.

Penelitian ini menggunakan *Logistic Regression* sebagai *workload* pembanding yang ekuivalen karena memiliki implementasi yang jelas pada kedua platform. Implementasi pada *R* menggunakan `glm(Machine.failure ~ ., family = binomial, data = data_train)`, sedangkan pada *Python* menggunakan `sklearn.linear_model.LogisticRegression(solver='lbfgs', max_iter=1000, random_state=42)`. Dengan standarisasi ini, *pipeline* yang sebanding dan terkontrol digunakan untuk membandingkan sampai tahap pemodelan.

Ringkasan pemetaan operasi pada *R* (melalui *RStudio*) dan *Python* untuk setiap tahapan disajikan pada Tabel 3.6.

Tabel 3.4 Operasi *Benchmarking* pada *R* dan *Python*

Tahap	**Operasi Utama**	***R* (melalui *RStudio*) (*Library*/Fungsi)**	***Python* (*Library*/Fungsi)**	**Parameter/ Standar**
Data Ingestion	Membaca *CSV* ke *dataframe*	`data.table::fread()`	`pandas.read_csv()`	Input dataset sama
Data Ingestion	Penyeragaman nama kolom	`make.names()`	`df.rename()`	Konsisten antar platform
Data Cleaning dan *Preprocessing*	Menghapus kolom	`dplyr::select(-`	`df.drop(['Product`	Kolom yang dihapus disetarakan

	identitas/non-fitur	`Product.ID, -UDI)`	`ID','UDI'], axis=1)`	
Data Cleaning dan *Preprocessing*	Menetapkan target	`y <- data$Machine.failure`	`y = df['Machine failure']`	Target: *Machine failure*
Data Cleaning dan Preprocessing	Menghapus label turunan dari fitur	`dplyr::select(-TWF,-HDF,-PWF,-OSF,-RNF)`	`df.drop(['TWF','HDF','PWF','OSF','RNF'], axis=1)`	Mencegah *label leakage*
Transformation	*Encoding Type* → numerik	`factor(Type, levels=c('L','M','H')) → numerik`	`replace({'L':0,'M':1,'H':2})`	*Mapping*: L=0, M=1, H=2
Evaluation & Validation	*Train–test split* 70:30 (*stratified*)	`set.seed(42); caret::createDataPartition(y, p=0.7, times=1, list=FALSE)`	`train_test_split(test_size=0.3, stratify=y, random_state=42)`	*Seed*: 42
Evaluation & Validation	*Oversampling* data latih (*SVM-SMOTE*)	`set.seed(42); smotefamily:`	`oversample = SVMSMOTE(ran`	*Seed*: 42

		:SVM.SMOTE(X_train, y_train)	dom_state = 42)	
Evaluation & Validation	Model klasifikasi ekuivalen	glm(Machine.failure ~ ., family=binomial, data=train)	LogisticRegression(solver='lbfgs', max_iter=1000, random_state=42)	Model: *Logistic Regression* Seed: 42

Penelitian ini menggunakan pendekatan *functional equivalence*, di mana operasi pada kedua platform disetarakan berdasarkan tujuan dan keluaran akhir, bukan berdasarkan implementasi teknis yang identik baris per baris. Perbedaan sintaks dan mekanisme internal dipertahankan sebagai bagian dari karakteristik ekosistem masing-masing platform, sehingga perbandingan performa mencerminkan kekuatan dan kelemahan alami dari *R* dan *Python* dalam konteks penggunaan nyata.

3.5 Protokol Pengukuran *Green Computing*

Protokol pengukuran disusun berdasarkan prinsip *Green Software* yang menekankan efisiensi energi, efisiensi perangkat keras, dan pemanfaatan sumber daya dalam evaluasi perangkat lunak. Berdasarkan prinsip tersebut, pengukuran dilakukan menggunakan empat parameter yaitu waktu eksekusi, utilisasi *CPU*,

penggunaan memori, dan konsumsi energi untuk merepresentasikan performa komputasi serta penggunaan sumber daya pada *R* (melalui *RStudio*) dan *Python*. Penelitian ini menggunakan metode hibrida berupa pencatatan berbasis skrip untuk pengukuran waktu eksekusi, utilisasi *CPU*, dan penggunaan memori, serta penggunaan alat eksternal *Intel Power Gadget* (IPG) untuk mengukur konsumsi energi pada tingkat paket *CPU*.

Karakteristik masing-masing metrik menentukan tingkat pengukuran yang berbeda. Instrumen yang tertanam dalam kode *R* dan *Python* digunakan untuk mengukur penggunaan *CPU* dan memori pada level proses. Metode ini digunakan untuk memastikan bahwa metrik yang dicatat menunjukkan sumber daya yang digunakan oleh proses eksekusi *pipeline* pada masing-masing platform. Ini mengurangi dampak aktivitas sistem operasi dan aplikasi lain yang berjalan secara bersamaan.

Namun, konsumsi energi diukur menggunakan *Intel Power Gadget* (IPG) pada level paket *CPU* karena IPG memantau energi prosesor secara keseluruhan, bukan hanya konsumsi proses *R* atau *Python* secara terpisah. Untuk mengurangi bias, pengukuran energi dilakukan dalam lingkungan yang terkontrol. Ini dilakukan dengan menutup aplikasi latar belakang yang tidak diperlukan, menjamin bahwa tidak ada proses komputasi intensif lain berjalan, dan menjalankan pengujian satu platform sekaligus. Dengan pengendalian ini, perbedaan konsumsi energi yang dicatat dianggap sebagai proksi yang konsisten. Ini mencerminkan karakteristik eksekusi *pipeline* analitik pada kedua platform.

Eksperimen tersebut dilakukan pada *testbed* yang sama dengan interval sampling *CPU* dan memori sebesar 0,5 detik. Prosedur eksekusi juga digunakan untuk kedua platform dengan cara yang sama.

3.5.1 Pengukuran Waktu Eksekusi

Waktu eksekusi adalah durasi yang dibutuhkan untuk menyelesaikan *pipeline* analitik pada setiap platform dalam satu kali pengujian. Skrip pengujian mencatat durasi dalam satuan milidetik (ms). Pencatatan dimulai tepat sebelum *pipeline* berjalan dan dihentikan setelah seluruh tahapan selesai. Nilai ini merepresentasikan performa ujung ke ujung pada tahapan analitik yang sebanding. Untuk pelaporan, waktu eksekusi diringkas sebagai *mean*, maksimum, dan minimum dalam satuan milidetik (ms).

3.5.2 Pengukuran Utilisasi *CPU*

Utilisasi *CPU* dicatat untuk merepresentasikan beban prosesor selama *pipeline* dieksekusi. Pengukuran dilakukan pada level proses, sehingga nilai yang dicatat merepresentasikan penggunaan *CPU* oleh proses pengujian. Pada *Python*, utilisasi *CPU* proses dicatat menggunakan `psutil.Process().cpu_percent()` pada interval sampling 0,5 detik selama *pipeline* berjalan. Pada *R* (melalui *RStudio*), waktu *CPU* proses diambil menggunakan `ps::ps_cpu_times()` pada setiap sampling, lalu utilisasi dihitung dari perubahan waktu *CPU* proses antar sampling sesuai interval yang sama. Nilai *CPU* diringkas menjadi *mean*, *P95*, dan maksimum dalam satuan persentase (%).

3.5.3 Pengukuran Penggunaan Memori

Penggunaan memori dicatat secara periodik selama *pipeline* berjalan dengan interval sampling 0,5 detik. Pada *R* (melalui *RStudio*), pencatatan dilakukan pada level proses dengan mengambil `RSS` proses menggunakan `ps::ps_memory_info()$rss`. Pada *Python*, pencatatan dilakukan menggunakan `psutil.Process().memory_info().rss` untuk mengambil `RSS` proses *Python*. Nilai memori disajikan dalam satuan *megabyte* (MB) dan diringkas menjadi *mean*, *P95*, dan maksimum.

3.5.4 Pengukuran Konsumsi Energi

Konsumsi energi dapat dihitung dengan menggunakan *Intel Power Gadget* (IPG) 3.5 pada tingkat paket *CPU*. IPG menghasilkan berkas *CSV* yang berisi nilai energi kumulatif dalam satuan *Joule* (J). Untuk menghitung konsumsi energi per *run*, selisih energi kumulatif antara akhir dan awal eksekusi *pipeline* dihitung dengan persamaan (1) berikut:

$$E_{run} = E_{akhir} - E_{awal} \quad (1)$$

Keterangan:

E_{run} = Total konsumsi energi spesifik selama satu skenario pengujian berjalan.

E_{akhir} = Nilai metrik kumulatif energi pada *timestamp* akhir eksekusi.

E_{awal} = Nilai metrik kumulatif energi pada *timestamp* awal eksekusi.

Nilai E_{akhir} dan E_{awal} didapat dari *log CSV* IPG pada *timestamp* yang sesuai dengan waktu mulai dan waktu selesai eksekusi *pipeline* masing-masing, dengan E_{akhir} adalah energi kumulatif *CPU* pada akhir eksekusi dan E_{awal} adalah energi kumulatif *CPU* pada awal eksekusi.

Untuk menghindari pencatatan yang tumpang tindih, pengukuran dilakukan secara bergantian untuk masing-masing platform *R* (melalui *RStudio*) dan *Python*. Nilai energi mencakup konsumsi seluruh prosesor, bukan hanya proses *R* atau *Python* secara terpisah, karena pengukuran dilakukan pada level paket *CPU*. Untuk memastikan validitas perbandingan, pengujian dilakukan dalam lingkungan yang terkontrol. Ini dilakukan dengan menghentikan aplikasi latar belakang yang tidak diperlukan dan memastikan bahwa tidak ada proses komputasi intensif lainnya yang dilakukan selama pengukuran.

Pengukuran ini tidak melihat konsumsi energi komponen lain seperti RAM, GPU, dan perangkat penyimpanan, tetapi cukup untuk membandingkan kedua platform secara relatif karena dilakukan pada perangkat keras, konfigurasi, dan prosedur yang sama. Untuk pelaporan hasil, konsumsi energi pada tiap platform digambarkan sebagai rata-rata (*mean*), nilai tertinggi (maksimum), dan nilai terendah (minimum) dalam satuan *Joule* (J).

3.5.5 Pengulangan Pengujian dan Perhitungan Nilai Akhir

Setiap skenario pengujian pada masing-masing platform dieksekusi sebanyak 3 (tiga) kali sebagai pengulangan eksperimen. Nilai akhir untuk waktu eksekusi dan konsumsi energi dilaporkan menggunakan ringkasan antar-*run*, yaitu *mean*, maksimum, dan minimum.

Untuk metrik yang dicatat secara periodik selama eksekusi, yaitu utilisasi *CPU* dan penggunaan memori, ringkasan hasil dilaporkan dalam bentuk *mean*, *P95*, dan *peak* guna menggambarkan kecenderungan nilai, stabilitas, serta potensi

lonjakan selama eksekusi. Pendekatan ini digunakan untuk meningkatkan keterulangan hasil dan meminimalkan pengaruh variabilitas sistem.

3.6 Teknik Analisis Data

Pada skenario uji yang sama, analisis data dilakukan untuk membandingkan performa penggunaan sumber daya komputasi *R* (melalui *RStudio*) dan *Python*. Setiap skenario pengujian pada masing-masing platform dieksekusi sebanyak tiga kali, sebagai eksperimen pengulangan. Hasil pengukuran dianalisis menggunakan statistik deskriptif sebagai ringkasan utama. Untuk menunjukkan perbedaan antar platform, data disajikan dalam bentuk grafik dan tabel. Prinsip yang berkaitan dengan *Green Computing* adalah bahwa platform yang menggunakan sumber daya komputasi yang lebih sedikit untuk *workload* yang sama dianggap lebih efisien dari segi penggunaan sumber daya.

3.6.1 Statistik Deskriptif

Untuk membandingkan performa antar platform, statistik deskriptif digunakan untuk meringkas data pengujian. Ringkasan untuk metrik tingkat operasi, yaitu waktu eksekusi dan konsumsi energi, dilaporkan dalam bentuk rata-rata $\pm$ SD. Ini menunjukkan nilai pusat dan variabilitas antar-operasi, serta nilai maksimum dan minimum untuk menunjukkan rentang hasil.

Ringkasan untuk metrik seperti utilisasi *CPU* dan memori yang dicatat secara berkala selama eksekusi diberikan dalam bentuk mean sebagai kecenderungan umum *P95* ditunjukkan sebagai representasi nilai tinggi yang paling umum selama eksekusi dan *peak* ditunjukkan sebagai lonjakan maksimum yang menunjukkan fase komputasi yang paling sulit pada operasi tertentu.

3.6.2 Analisis Komparatif dan Penyajian Data

Hasil pengukuran disajikan dalam dua bentuk tabel yang menguraikan nilai ringkasan statistik untuk masing-masing platform dan grafik yang menunjukkan perbandingan visual pada parameter tertentu. Dengan melihat konsistensi pola pada tiga pengulangan, perbedaan dalam performa dievaluasi. Jika satu platform secara konsisten menunjukkan waktu eksekusi dan konsumsi energi yang lebih rendah, serta utilisasi *CPU* dan memori yang lebih rendah, maka platform tersebut dianggap memiliki performa yang lebih baik pada *workload* yang sama dalam kerangka *Green Computing*. Untuk menunjukkan seberapa besar perbedaan yang sebenarnya, selisih *mean*, persentase perbedaan, dan rasio antar platform masing-masing parameter dihitung.

Untuk mengkontekstualisasikan hasil, penelitian dibandingkan dengan penelitian sebelumnya yang relevan tentang perbandingan *R* dan *Python*, serta penelitian *Green Computing* yang melihat performa dan konsumsi energi pada tugas analitik data. Perbandingan ini dilakukan dengan mempertimbangkan perbedaan metodologi, fitur tugas, pustaka yang digunakan, dan lingkungan pengujian. Penelitian ini tidak melakukan uji signifikansi statistik karena jumlah pengulangan masih terbatas (tiga kali per platform). Untuk meningkatkan ketelitian konfirmasi temuan, penelitian lanjutan dengan jumlah replikasi yang lebih besar harus menggunakan uji inferensial seperti interval kepercayaan dan ukuran efek.