Penerapan Algoritma Brute Force dalam Feature Selection untuk Model Klasifikasi Machine Learning

Farrell Jabaar Altafataza - 10122057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: fareltaza35@gmail.com , 10122057@mahasiswa.itb.ac.id

***Abstrak*—** Tingginya dimensionalitas dan adanya fitur yang tidak relevan dalam dataset seringkali menurunkan performa model klasifikasi *machine learning*. Penelitian ini mengatasi masalah tersebut dengan menerapkan algoritma *Brute Force* dalam *feature selection* untuk mengidentifikasi *subset* fitur yang paling berpengaruh. Menggunakan "Titanic Dataset" dari Kaggle dan tiga model klasifikasi—Decision Tree, SVM, dan Naïve Bayes—penelitian ini secara sistematis mengevaluasi semua kemungkinan kombinasi fitur untuk menemukan *subset* optimal berdasarkan metrik F1-Score. Hasil eksperimen menunjukkan bahwa dampak *feature selection* sangat bergantung pada model yang digunakan. Model Decision Tree mengalami peningkatan performa signifikan, dengan F1-Score naik dari 0.721 menjadi 0.766. Sebaliknya, model SVM dan Naïve Bayes menunjukkan sedikit penurunan kinerja. Temuan ini membuktikan bahwa *subset* fitur optimal yang dijamin oleh *Brute Force* tidak selalu meningkatkan generalisasi pada semua jenis model. Relevansi sebuah fitur bersifat *model-dependent*, di mana fitur seperti Sex, log_age, dan log_fare secara konsisten terbukti memiliki daya prediktif yang kuat di berbagai model.

Kata Kunci—Feature Selection, Brute Force, Klasifiaksi, Machine Learning, F1-Score.

I. PENDAHULUAN

A. *Latar Belakang*

Di era melimpahnya data, *machine learning* muncul sebagai solusi utama untuk membuat keputusan berdasarkan data. Salah satu tugas fundamental dalam *machine learning* adalah klasifikasi, yaitu proses mengkategorikan data ke dalam kelas-kelas yang telah ditentukan. Keberhasilan model klasifikasi sangat bergantung pada kualitas data yang digunakan untuk melatihnya.

Namun, data yang tersedia di dunia nyata seringkali tidak sempurna. Banyak dataset yang memiliki dimensionalitas yang tinggi, dan seringkali memuat fitur-fitur yang tidak memberikan kontribusi berarti terhadap hasil prediksi. Fitur yang tidak relevan tersebut disebut "*noise"* dan dapat membingungkan model.

Untuk mengatasi permasalahan ini, perlu dilakukan tahapan *preprocessing* data yang disebut *feature selection*. *Feature selection* merupakan proses memilih subset fitur yang paling optimal dari kumpulan fitur asli. *Feature selection* bertujuan untuk menghilangkan fitur-fitur redundan sehingga dapat meningkatkan performa prediksi model.

Salah satu implementasi dari *feature selection* adalah algoritma Brute Force, atau disebut juga sebagai *Exhaustive Feature Search*. Metode ini mengevaluasi kualitas suatu subset fitur dengan cara melatih dan menguji model secara langsung. Meskipun metode ini memiliki biaya komputasi yang tinggi, namun metode ini menjamin menemukan subset fitur yang paling optimal secara global.

B. *Rumusan Masalah*

Berdasarkan latar belakang yang telah dijelaskan, pertanyaan utama yang akan dijawab dalam penelitian ini dirumuskan sebagai berikut:

> Seberapa signifikan peningkatan performa model klasifikasi setelah menggunakan subset fitur paling optimal hasil dari algoritma Brute Force dibandingkan dengan performa model yang menggunakan semua fitur asli?

C. *Tujuan*

Tujuan dari penulisan makalah ini adalah:

1. Menerapkan algoritma Brute Force untuk mengidentifikasi subset fitur paling optimal dari sebuah dataset yang digunakan untuk membangun model klasifikasi.
2. Menganalisis dan membuktikan adanya peningkatan performa pada model klasifikasi setelah menggunakan fitur hasil seleksi, dengan cara membandingkan kinerjanya terhadap model yang menggunakan keseluruhan fitur asli.

II. DASAR TEORI

A. Machine Learning

Machine learning (pembelajaran mesin) adalah sebuah bidang studi yang memberikan komputer kemampuan untuk belajar tanpa diprogram secara eksplisit **[1]**. Secara lebih modern, machine learning didefinisikan sebagai cabang dari ilmu kecerdasan buatan (Artificial Intelligence) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari dan membuat prediksi atau keputusan berdasarkan data **[2]**. Secara umum, machine learning terbagi menjadi tiga kategori utama, yaitu *Supervised Learning, Unsupervised Learning,* dan *Reinforcement Learning*.

Supervised Learning adalah algoritma yang dilatih menggunakan dataset yang telah diberi label, yang berarti setiap data input telah dipasangkan dengan data output (label atau kelas) yang benar. Tujuannya adalah agar model dapat mempelajari fungsi pemetaan dari input ke output untuk memprediksi label data baru. Tugas utama dalam *Supervised Learning* adalah klasifikasi dan regresi.

Unsupervised Learning adalah algoritma yang bekerja dengan data yang tidak memiliki label. Tujuannya adalah untuk menemukan pola, struktur, atau hubungan tersembunyi dalam data, seperti pengelompokan (*clustering*) atau pengurangan dimensi.

Reinforcement Learning adalah sebuah agen (*agent*) yang belajar dengan cara berinteraksi dengan lingkungannya. Agen menerima umpan balik berupa *rewards* (penghargaan) atau *penalties* (hukuman) atas tindakan yang dilakukannya, dan bertujuan untuk mempelajari kebijakan (*policy*) yang dapat memaksimalkan total *reward* yang diterima.

B. Algoritma Brute Force

Algoritma *Brute Force* adalah pendekatan fundamental dalam ilmu komputer untuk pemecahan masalah yang mengandalkan daya komputasi murni. Prinsip dasarnya adalah secara sistematis memeriksa semua kemungkinan kandidat solusi untuk suatu masalah dan mengevaluasi apakah setiap kandidat tersebut memenuhi kriteria yang ditetapkan. Karena sifatnya yang langsung dan tidak menggunakan strategi heuristik atau jalan pintas, metode ini sering dianggap sebagai pendekatan yang paling dasar namun juga paling teliti dalam ruang lingkup algoritma pencarian. Kesederhanaan dalam konsep dan implementasinya menjadikannya titik awal yang penting untuk memahami paradigma desain algoritma yang lebih kompleks [3].

Dalam konteks seleksi fitur (*feature selection)*,implementasi algoritma *Brute Force* masuk ke dalam kategori *Wrapper*. Metode *Wrapper* merupakan metode yang secara langsung menggunakan performa dari model *machine learning* sebagai kriteria evaluasi untuk menemukan subset fitur terbaik. Pendekatan ini dirancang untuk menemukan kombinasi fitur terbaik dengan cara yang paling teliti, yaitu dengan tidak melewatkan satu pun kemungkinan yang ada untuk mencapai performa model yang maksimal.

C. Exhaustive Feature Selector

Exhaustive Feature Selector merupakan salah satu metode *feature selection* yang menggunakan algoritma *Brute Force*. Metode ini juga sering disebut sebagai *Brute Force Feature Selection*. Metode ini bekerja dengan cara mengevaluasi setiap kemungkinan kombinasi subset fitur yang dapat dibentuk dari Kumpulan semua fitur asli. Untuk sebuah dataset dengan N fitur, metode ini menguji semua $2^N - 1$ kemungkinan subset. Sebuah model akan dilatih dan dievaluasi untuk setiap subset, lalu subset dengan performa terbaik akan dipilih menjadi subset paling optimal.

D. Model Klasifikasi

Dalam penelitian ini, digunakan tiga model klasifikasi yaitu:

1. *Decision Tree*

   Decision Tree merupakan salah satu model klasifikasi dan regresi yang paling intuitif dan paling mudah diinterpretasikan dalam *machine learning.* Model ini bekerja dengan cara mempartisi ruang fitur secara rekursif menjadi himpunan bagian yang lebih kecil dan homogen. Struktur model ini menyerupai diagram alir (*flowchart*), di mana setiap node internal merepresentasikan sebuah "tes" terhadap suatu fitur, setiap cabang (*branch*) merepresentasikan hasil dari tes tersebut, dan setiap *node* daun (*leaf node*) merepresentasikan label kelas atau keputusan akhir [4]. Keunggulan model ini adalah sangat mudah untuk diinterpretasikan dan divisualisasikan, serta tidak perlu menormalisasikan data. Akan tetapi, model ini cenderung mengalami *overfitting*, tidak stabil, dan dapat memiliki bias terhadap fitur yang memiliki banyak level.
2. *Naive Bayes*

   Naive Bayes adalah keluarga algoritma klasifikasi probabilistik yang didasarkan pada Teorema Bayes. Model ini menghitung probabilitas setiap kelas berdasarkan nilai-nilai fitur dari data yang diberikan. Disebut "naive" (lugu) karena model ini membuat asumsi penyederhanaan yang kuat, yaitu bahwa semua fitur bersifat independen satu sama lain secara kondisional, mengingat label kelasnya [4]. Keunggulan model ini terletak pada kecepatan dan efisien secara komputasi. Model ini juga bekerja dengan sangat baik pada masalah klasifikasi teks dan data berdimensi tinggi. Kelemahan model ini terletak pada asumsi independensi fitur adalah batasan utamanya dan sering kali tidak realistis pada data dunia nyata, yang dapat menurunkan kinerjanya jika fitur-fitur memiliki korelasi yang kuat [4].
3. *Support Vector Machine (SVM)*

   Support Vector Machine (SVM) adalah model klasifikasi linear dan non-linear yang sangat kuat. Tujuan utama SVM adalah untuk menemukan sebuah *hyperplane* (bidang pemisah) optimal yang dapat memisahkan data dari kelas-kelas yang berbeda dengan margin atau jarak sebesar mungkin. Margin

didefinisikan sebagai jarak antara *hyperplane* dan titik data terdekat dari masing-masing kelas [4]. Model ini sangat efektif di ruang berdimensi tinggi dan efisien dalam penggunaan memori. Namun, model ini kurang efisien ketika dihadapkan dengan jumlah data yang sangat besar.

E. Metrik Evaluasi

Metrik evaluasi yang digunakan merupakan metrik yang diturunkan dari *Confusion Matrix*. *Confusion Matrix* merupakan tabel yang memvisualisasikan performa model dengan membandingkan kelas prediksi dan kelas aktual. Komponen dari *Confusion Matrix* di antaranya adalah:

- *True Positive* (TP)
- *True Negative* (TN)
- *False Positive* (FP)
- *False Negative* (FN).

Metrik- metrik yang digunakan adalah:

1. Akurasi
   Akurasi adalah proporsi prediksi yang benar dari total prediksi. Rumus dari akurasi adalah:
   $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
   Akurasi merupakan metrik yang paling intuitif dan mudah dipahami karena menunjukkan persentase prediksi yang benar secara keseluruhan. Namun, metrik ini bisa memberikan gambaran yang menyesatkan ketika digunakan pada dataset yang tidak seimbang (imbalanced dataset), dimana salah satu kelas mendominasi jumlah instance.
   Metrik ini intuitif tetapi bisa menyesatkan pada dataset yang tidak seimbang (*imbalanced*).
2. Presisi
   Presisi merupakan sebuah metrik yang meninjau apakah semua *instance* yang diprediksi positif benar-benar positif. Rumus dari presisi adalah:
   $$Precision = \frac{TP}{TP + FN}$$
3. Metrik ini sangat penting dalam kasus dimana False Positive memiliki konsekuensi yang serius, seperti dalam diagnosis medis atau deteksi penipuan. Nilai presisi yang tinggi menunjukkan bahwa model jarang melakukan kesalahan dalam memprediksi positif palsu.
4. Recall
   Recall merupakan sebuah metrik yang menjawab pertanyaan "Dari semua instance yang sebenarnya positif, berapa persen yang berhasil diidentifikasi oleh model?". Rumus dari recall adalah:
   $$Recall = \frac{TP}{TP + FN}$$
   Metrik ini sangat krusial dalam aplikasi dimana False Negative memiliki dampak yang besar, seperti dalam skrining penyakit atau sistem keamanan. Recall yang tinggi menunjukkan bahwa model mampu mendeteksi sebagian besar kasus positif.
5. F1-Score
   F1-Score merupakan sebuah metrik yang dibentuk dari presisi dan recall. Rumus dari F1-Score adalah:
   $$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$
   F1-Score sangat berguna untuk mengevaluasi model pada dataset yang tidak seimbang karena memberikan keseimbangan antara presisi dan recall. Nilai F1-Score yang tinggi menunjukkan bahwa model memiliki performa yang baik dalam kedua aspek tersebut, tanpa mengorbankan salah satunya.

III. RANCANGAN PENELITIAN

A. Deskripsi Dataset

Penelitian ini menggunakan dataset "Titanic Dataset" yang bersumber dari Kaggle. Dataset ini dipilih karena menyajikan masalah biner, yaitu memprediksi keselamatan penumpang kapal Titanic dengan kelas 1 berarti selamat dan kelas 0 berarti tidak selamat. Lalu, dataset ini memiliki 11 fitur predictor yang mana jumlah tersebut cukup kecil sehingga pendekatan secara Brute Force masih layak secara komputasi. Selain itu, dataset ini juga merupakan dataset publik sehingga memungkinkan reproduktifitas dan verifikasi hasil oleh peneliti lain.

Berikut merupakan deskripsi dari setiap fitur yang dimiliki oleh dataset ini:

Tabel I. Deskripsi Fitur Titanic Dataset

Nama Fitur	Deskripsi	Tipe Data	Nilai
PassengerId	Kode unik setiap penumpang kapal	Numerik	1-891
Pclass	Kelas tiket penumpang kapal	Kategorikal	1 (1st class), 2 (2nd class), 3 (3rd class)
Name	Nama penumpang	String	"Braund, Mr. Owen Harris", "Heikkinen, Miss. Laina", dll
Sex	Gender penumpang	Kategorikal	male & female
Age	Usia penumpang	Numerik	0.42-80 tahun
SibSp	Jumlah saudara atau pasangan penumpang yang ikut dalam kapal	Numerik	0-8 orang
Parch	Jumlah orang tua atau anak penumpang	Numerik	0-6 orang

	yang ikut dalam kapal		
Ticket	Nomor tiket penumpang	Numerik	681 unique values
Fare	Tarif penumpang	Numerik	0-512.3
Cabin	Nomor kabin penumpang	String	147 unique values
Embarked	Pelabuhan keberangkatan penumpang	Kategorikal	C (Cherbourg), Q (Queenstown), S (Southampton)
Survived	Keselamatan penumpang	Kategorikal	1 (Selamat), 0 (Tidak selamat)

B. Prosedur Implementasi

1. Lingkungan Eksperimen
   Eksperimen dijalankan pada lingkungan komputasi dengan spesifikasi sebagai berikut:
   - Perangkat Keras: CPU Intel(R) Core(TM) Ultra 9 185H, RAM 32 GB
   - Perangkat Lunak: Python 3.11.9, dengan *library* utama: *scikit-learn* untuk pemodelan dan evaluasi, *mlxtend* untuk implementasi *Exhaustive Feature Selector*, *pandas* untuk manipulasi data, dan *matplotlib* untuk visualisasi.
2. *Data Preprocessing*
   Sebelum melakukan pelatihan model dan *feature selection*, dilakukan tahap *preprocessing* yaitu:
   1. Penghapusan Fitur yang Tidak Relevan
      Langkah pertama dalam preprocessing adalah menghapus fitur-fitur yang tidak memberikan nilai signifikan dalam prediksi model atau memiliki masalah data yang parah.
      Fitur seperti PassengerId, Name, dan Ticket dihilangkan karena hanya berfungsi sebagai pengidentifikasi unik penumpang dan tidak memiliki korelasi dengan target prediksi.
      Sementara itu, fitur Cabin juga dihapus karena mengandung 77% missing values, sehingga pengisian nilai kosong tidak akan memberikan hasil yang akurat.
      Dengan menghapus fitur-fitur ini, dataset menjadi lebih ringkas dan fokus pada variabel yang benar-benar berpengaruh.
   2. *Imputation*
      Beberapa fitur dalam dataset masih memiliki nilai yang hilang, tetapi dalam jumlah yang masih dapat ditangani.
      Fitur Age memiliki sekitar 19,8% missing values, dan karena distribusi usia tidak normal, nilai kosong diisi menggunakan median untuk meminimalkan pengaruh outlier. Sementara itu, fitur Embarked hanya memiliki 0,2% missing values, sehingga nilai yang hilang diisi dengan modus (nilai yang paling sering muncul) karena ini merupakan data kategorikal.
   3. *Feature Engineering*
      Proses feature engineering dilakukan untuk menciptakan fitur baru yang lebih informatif atau mengubah fitur yang sudah ada agar lebih sesuai untuk pemodelan. Salah satu teknik yang digunakan adalah transformasi logaritmik pada fitur Age dan Fare karena distribusinya yang skewed. Hasil transformasi ini adalah fitur baru, yaitu log_age dan log_fare, yang memiliki distribusi lebih mendekati normal. Selain itu, fitur SibSp dan Parch digabungkan menjadi FamilySize untuk merepresentasikan ukuran keluarga penumpang, yang mungkin berpengaruh terhadap tingkat kelangsungan hidup.
   4. *Encoding* Fitur Kategorikal
      Karena banyak algoritma machine learning hanya dapat memproses data numerik, fitur kategorikal perlu dikonversi ke dalam bentuk numerik. Fitur Sex diubah menjadi nilai biner (0 untuk male dan 1 untuk female). Sementara itu, fitur Embarked yang memiliki tiga kategori (C, Q, S) diencode menggunakan One Hot Encoding, menghasilkan tiga kolom baru (Embarked_C, Embarked_Q, Embarked_S) dengan nilai 0 atau 1. Fitur asli Embarked kemudian dihapus untuk menghindari masalah multikolinearitas dalam model.
   5. *Feature Scaling*
      Langkah terakhir dalam preprocessing adalah melakukan scaling pada fitur numerik. Beberapa model machine learning, seperti SVM dan regresi linier, sangat sensitif terhadap perbedaan skala antarfitur. Oleh karena itu, digunakan StandardScaler untuk menstandarisasi data sehingga memiliki mean = 0 dan standar deviasi = 1.

   Setelah melakukan *preprocessing*, jumlah fitur predikto pada dataset menjadi 9.
   3. Proses Seleksi Fitur Brute Force
      Proses inti eksperimen ini adalah dengan menggunakan *library mlxtend* dengan kelas *ExhaustiveFeatureSelector*. Proses dimulai dengan menghasilkan 511 kemungkinan subset fitur dai 9 fitur yang tersedia. Untuk setiap subset fitur, algoritma akan melatih model klasifikasi *machine learning* pada data *training* dengan hanya menggunakan subset fitur tersebut.
      Lalu, algoritma mengevaluasi performa model menggunakan *cross-validation* berjumlah 5 lipat agar memastikan performa yang stabil dan tidak

terjadi *overfitting*. Metrik evaluasi utama yang digunakan adalah F1-Score karena dataset mempunyai ketidakseimbangan kelas dalam fitur prediksi.

Terakhir, setelah semua 511 kombinasi subset dievaluasi, algoritma mengidentifikasi subset fitur yang menghasilkan F1-Score tertinggi. Subset ini disebut juga sebagai "subset optimal" untuk model tersebut. Proses ini dilakukan secara terpisah untuk ketiga model klasifikasi.

IV. Hasil dan Analisis

Proses *feature selection* dengan algoritma Brute Force berhasil mengidentifikasi subset fitur yang memberikan performa F1-Score tertinggi untuk masing-masing model klasifikasi. Hasil performa model klasifikasi dirangkum dalam Tabel II. Sebagai perbandingan, tabel ini juga menyertakan performa setiap model ketika dilatih menggunakan **s**emua 9 fitur sebagai *baseline*.

Model Klasifikasi	Jumlah Fitur	Hasil
Decision Tree (*baseline)*	Semua fitur	Decision Tree (baseline) Accuracy: 0.7486 Decision Tree (baseline) Precision: 0.6667 Decision Tree (baseline) Recall: 0.7838 Decision Tree (baseline) F1 Score: 0.7205
Decision Tree	5: ['Pclass', 'Sex', 'Parch', 'log_fare', 'Embarked_Q']	Decision Tree Accuracy: 0.8156 Decision Tree Precision: 0.8060 Decision Tree Recall: 0.7297 Decision Tree F1 Score: 0.7660
SVM (*baseline)*	Semua fitur	SVM (baseline) Accuracy: 0.8156 SVM (baseline) Precision: 0.8060 SVM (baseline) Recall: 0.7297 SVM (baseline) F1 Score: 0.7660
SVM	4: ['Pclass', 'Sex', 'log_age', 'FamilySize']	SVM Accuracy: 0.8101 SVM Precision: 0.8030 SVM Recall: 0.7162 SVM F1 Score: 0.7571
Naïve Bayes (*baseline)*	Semua fitur	Naive Bayes (baseline) Accuracy: 0.7765 Naive Bayes (baseline) Precision: 0.6932 Naive Bayes (baseline) Recall: 0.8243 Naive Bayes (baseline) F1 Score: 0.7531
Naïve Bayes	5: ['Sex', 'log_age', 'log_fare', 'Embarked_S', 'FamilySize']	Naive Bayes Accuracy: 0.7765 Naive Bayes Precision: 0.7429 Naive Bayes Recall: 0.7027 Naive Bayes F1 Score: 0.7222

Hasil seleksi fitur menunjukkan dampak yang berbeda-beda tergantung pada model yang digunakan. Untuk model Decision Tree, seleksi fitur berhasil meningkatkan kinerja secara signifikan, dengan F1-Score naik dari 0.721 menjadi 0.766 dan akurasi meningkat dari 0.749 menjadi 0.816. Ini menunjukkan bahwa untuk Decision Tree, penghapusan fitur yang kurang relevan berhasil meningkatkan kemampuan generalisasi.

Sebaliknya, untuk model SVM dan Naive Bayes, seleksi fitur justru menghasilkan F1-Score yang sedikit lebih rendah. F1-Score SVM turun dari 0.766 menjadi 0.757, dan Naive Bayes turun dari 0.753 menjadi 0.722. Hal ini mengindikasikan bahwa untuk kedua model ini, beberapa fitur yang dihilangkan kemungkinan masih mengandung informasi prediktif yang berguna, dan manfaat dari pengurangan dimensionalitas tidak cukup untuk mengimbangi hilangnya informasi tersebut.

Setiap model klasifikasi menemukan subset fitur optimal yang berbeda. SVM mencapai performa puncaknya dengan hanya 4 fitur, sementara Decision Tree membutuhkan 5 fitur. Hal ini mengindikasikan bahwa relevansi sebuah fitur bersifat *model-dependent*. Cara setiap algoritma belajar dari data (misalnya, SVM mencari margin, Decision Tree mencari pemisahan berdasarkan informasi) memengaruhi fitur mana yang paling berguna baginya.

Beberapa fitur seperti log_age, log_fare dan Sex seringkali muncul di subset optimal, menunjukkan bahwa fitur-fitur ini kemungkinan besar memiliki daya prediktif yang kuat secara universal untuk dataset ini.

V. Kesimpulan dan Saran

A. Kesimpulan

Setelah melakukan pengujian, didapatkan beberapa temuan utama dari eksperimen yang dilakukan kepada dataset "Titanic Dataset", yaitu:

1. Signifikansi perubahan performa setelah seleksi fitur sangat bervariasi dan bergantung pada model klasifikasi yang digunakan.

2. Model Decision Tree mengalami peningkatan performa yang signifikan, menunjukkan bahwa seleksi fitur berhasil menghilangkan fitur yang mengganggu dan meningkatkan generalisasi.

3. Model SVM dan Naive Bayes justru mengalami sedikit penurunan performa, yang mengindikasikan bahwa subset fitur yang lebih kecil tidak selalu lebih baik dan beberapa informasi penting mungkin hilang dalam proses seleksi.

Dapat disimpulkan bahwa penggunaan subset fitur paling optimal yang ditemukan oleh algoritma *brute force* tidak secara otomatis menjamin peningkatan performa yang signifikan untuk semua model. Meskipun *brute force* secara matematis menemukan kombinasi fitur terbaik berdasarkan kriteria evaluasi pada data latih, dampaknya pada data uji (generalisasi) bisa positif, negatif, atau netral, tergantung pada bagaimana algoritma klasifikasi berinteraksi dengan fitur-fitur tersebut.

B. Saran

Berdasarkan temuan penelitian ini, terdapat beberapa saran untuk penelitian selanjutnya.

Pertama adalah melakukan studi komparatif kuantitatif secara langsung antara hasil dari *brute force* dengan metode seleksi fitur heuristik pada dataset yang sama. Analisis tersebut akan dapat mengukur seberapa dekat solusi yang ditemukan oleh metode heuristik dengan solusi optimal yang sebenarnya yang dijamin oleh *brute force*.

Kedua adalah Menerapkan pendekatan ini pada beberapa dataset berdimensi rendah lainnya untuk menguji apakah pola temuan (misalnya, sifat *model-dependent* dari subset optimal) tetap konsisten di berbagai domain masalah.

REFERENCES

[1] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, hlm. 210-229, 1959.

[2] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. The MIT Press, 2022.

[4] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.

[5] F. J. Altafataza. (2025). Implementasi-Brute-Force-pada-Feature-Selection. GitHub repository. [Online]. Available: https://github.com/farrellja1/Implementasi-Brute-Force-pada-Feature-Selection

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Juni 2025

Farrell Jabaar Altafataza
10122057