# INFORMASI PROYEK

Judul Proyek:

Klasifikasi Tingkat Kejang Epilepsi Berbasis Sinyal EEG Menggunakan Machine Learning pada Dataset BEED

Nama Mahasiswa: Nabila Carrissa Dewi

NIM: 234311021

Program Studi: Teknologi Rekayasa Perangkat Lunak

Mata Kuliah: Praktik Data Science

Dosen Pengampu: Gus Nanang Syaifuddiin

Tahun Akademik: 2025 ( Ganjil ) / 5A

Link GitHub Repository: <https://github.com/nabilacarrissa/eeg-epilepsy-classification-beed>

Link Video Pembahasan: <https://youtu.be/GGM0MhEDik8> s

## LEARNING OUTCOMES

Pada proyek ini, mahasiswa diharapkan dapat:

1. Memahami konteks masalah dan merumuskan problem statement secara jelas
2. Melakukan analisis dan eksplorasi data (EDA) secara komprehensif (**OPSIONAL**)
3. Melakukan data preparation yang sesuai dengan karakteristik dataset
4. Mengembangkan tiga model machine learning yang terdiri dari (**WAJIB**):
   * Model baseline
   * Model machine learning / advanced
   * Model deep learning (**WAJIB**)
5. Menggunakan metrik evaluasi yang relevan dengan jenis tugas ML
6. Melaporkan hasil eksperimen secara ilmiah dan sistematis
7. Mengunggah seluruh kode proyek ke GitHub (**WAJIB**)
8. Menerapkan prinsip software engineering dalam pengembangan proyek

## PROJECT OVERVIEW

### Latar Belakang

Dalam era Revolusi Industri 4.0, analisis data sensor menjadi komponen krusial dalam berbagai aplikasi cerdas, mulai dari pemantauan kesehatan (seperti sinyal EEG) hingga *predictive maintenance* pada mesin industri. Data sensor sering kali memiliki karakteristik *high-dimensional*, *noisy*, dan kompleks, yang menuntut metode klasifikasi yang handal untuk membedakan status atau *state* dari objek yang diamati.

Tantangan utama dalam klasifikasi data sensor adalah menemukan pola non-linear yang tersembunyi di antara fitur-fitur numerik. Pendekatan tradisional seperti *Logistic Regression* sering kali tidak cukup untuk menangkap kompleksitas ini, sehingga metode *ensemble* seperti *Random Forest* dan pendekatan *Deep Learning* menjadi alternatif yang menjanjikan.

Penelitian terdahulu menunjukkan efektivitas berbagai algoritma dalam menangani data tabular.

1. Gorad et al. (2021) dalam jurnal *IEEE Access* berjudul *"EEG Signal Classification Using Machine Learning Approaches"* menunjukkan bahwa algoritma *Random Forest* sering kali mengungguli model linear karena kemampuannya menangani *non-linear boundaries* tanpa memerlukan asumsi distribusi data yang ketat.

2. Borisov et al. (2022) dalam *IEEE Transactions on Neural Networks and Learning Systems* berjudul *"Deep Neural Networks and Tabular Data: A Survey"* menyoroti bahwa meskipun *Gradient Boosted Decision Trees* dominan pada data tabular, arsitektur *Deep Learning* (seperti *Multilayer Perceptron*) dengan regularisasi yang tepat (seperti Dropout) mulai menunjukkan performa yang kompetitif dan bahkan lebih baik pada dataset dengan jumlah fitur yang signifikan.

Berdasarkan tinjauan tersebut, proyek ini bertujuan untuk mengkomparasi performa model *baseline*, *advanced*, dan *deep learning* pada dataset BEED (data sensor) untuk menentukan pendekatan terbaik dalam klasifikasi *state* sensor.

* 1. **Rumusan Masalah**

1. Bagaimana karakteristik distribusi sinyal EEG pada pasien sehat dibandingkan dengan pasien yang mengalami berbagai jenis kejang?
2. Bagaimana kinerja model *Machine Learning* konvensional dibandingkan dengan model *Deep Learning* dalam mengklasifikasikan sinyal EEG yang kompleks?
3. Model manakah yang memberikan keseimbangan terbaik antara akurasi dan kompleksitas komputasi untuk kasus deteksi epilepsi ini?

### Tujuan Proyek

1. Melakukan eksplorasi dan pra-pemrosesan data sinyal EEG dari dataset BEED untuk memastikan kualitas data input.
2. Membangun tiga jenis model prediktif (Baseline, Advanced ML, dan Deep Learning) untuk mengklasifikasikan status subjek (Sehat, Generalized Seizure, Focal Seizure, Seizure Events).
3. Mengevaluasi dan membandingkan performa ketiga model menggunakan metrik akurasi, presisi, recall, dan F1-score.

## BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING

### Problem Statements

1. Sinyal EEG memiliki pola yang kompleks dan sulit dianalisis secara manual.
2. Dibutuhkan model yang mampu mengklasifikasikan kondisi epileptic dan non-epileptic secara akurat.
3. Perlu dibandingkan performa metode machine learning konvensional dengan deep learning pada data EEG.

### Goals

1. Mengembangkan model klasifikasi epilepsi berbasis sinyal EEG.
2. Membandingkan performa baseline, advanced ML, dan deep learning model.
3. Menentukan model terbaik berdasarkan hasil evaluasi.

### Solution Approach

Mahasiswa **WAJIB** menggunakan minimal **tiga model** dengan komposisi sebagai berikut:

#### Model 1 – Baseline Model

Model sederhana sebagai pembanding dasar.

Pilihan model:

* Linear Regression (untuk regresi)
* Logistic Regression (untuk klasifikasi)
* K-Nearest Neighbors (KNN)
* Decision Tree
* Naive Bayes

#### Model baseline yang digunakan pada penelitian/analisis ini adalah Logistic Regression. Pemilihan model ini didasarkan pada karakteristik data dan tujuan pemodelan yang telah dijelaskan sebelumnya, yaitu memprediksi kelas/label (bersifat klasifikasi).

Alasan pemilihan Logistic Regression sebagai baseline:

1. Sesuai untuk masalah klasifikasi

Logistic Regression dirancang untuk menangani permasalahan klasifikasi, terutama klasifikasi biner, dengan memodelkan probabilitas suatu data masuk ke kelas tertentu. Hal ini sesuai dengan target pada data di atas yang berbentuk label/kategori.

1. Model sederhana dan interpretatif

Sebagai model baseline, Logistic Regression sangat baik karena mudah dipahami dan dianalisis. Koefisien pada setiap fitur dapat menunjukkan arah dan pengaruh fitur terhadap hasil prediksi, sehingga memudahkan interpretasi awal terhadap data.

1. Efisien pada data tabular

Logistic Regression bekerja dengan baik pada data tabular yang memiliki fitur numerik maupun hasil encoding dari fitur kategorikal, seperti pada data yang digunakan.

1. Cepat dilatih dan stabil

Model ini memiliki kompleksitas yang rendah, sehingga waktu pelatihan cepat, tidak membutuhkan banyak sumber daya, dan relatif stabil terhadap overfitting dibandingkan model yang lebih kompleks.

1. Menjadi pembanding yang adil (baseline)

Logistic Regression sering digunakan sebagai standar awal (baseline) untuk membandingkan performa dengan model lain seperti Decision Tree, Random Forest, atau model berbasis boosting. Jika model yang lebih kompleks tidak memberikan peningkatan signifikan, maka Logistic Regression sudah dianggap cukup optimal.

#### Model 2 – Advanced / ML Model

Model machine learning yang lebih kompleks.

Pilihan model:

* Random Forest
* Gradient Boosting (XGBoost, LightGBM, CatBoost)
* Support Vector Machine (SVM)
* Ensemble methods
* Clustering (K-Means, DBSCAN) - untuk unsupervised
* PCA / dimensionality reduction (untuk preprocessing)

#### Model lanjutan (advanced model) yang digunakan pada analisis ini adalah Random Forest. Pemilihan model ini didasarkan pada kebutuhan untuk meningkatkan performa prediksi dibandingkan model baseline (Logistic Regression), khususnya dalam menangkap pola data yang lebih kompleks.

Alasan pemilihan Random Forest sebagai model advanced:

1. Mampu menangkap hubungan non-linear  
   Berbeda dengan Logistic Regression yang bersifat linear, Random Forest mampu memodelkan hubungan non-linear dan interaksi antar fitur, sehingga lebih efektif ketika pola pada data tidak sederhana.
2. Mengurangi risiko overfitting  
   Random Forest merupakan metode ensemble yang menggabungkan banyak Decision Tree. Dengan teknik bootstrap sampling dan pemilihan fitur secara acak, model ini lebih stabil dan tidak mudah overfitting dibandingkan Decision Tree tunggal.
3. Performa yang umumnya lebih tinggi  
   Pada data tabular seperti data di atas, Random Forest sering memberikan akurasi dan performa klasifikasi yang lebih baik dibandingkan model linear, terutama ketika jumlah fitur cukup banyak.
4. Tahan terhadap noise dan outlier  
   Karena menggunakan banyak pohon keputusan, pengaruh data yang bersifat noise atau outlier dapat diminimalkan, sehingga hasil prediksi menjadi lebih robust.
5. Menyediakan informasi pentingnya fitur (feature importance)  
   Random Forest mampu memberikan feature importance, yang membantu memahami fitur mana yang paling berpengaruh terhadap hasil prediksi. Hal ini sangat berguna untuk analisis lanjutan dan pengambilan keputusan.

#### Model 3 – Deep Learning Model (WAJIB)

Model deep learning yang sesuai dengan jenis data.

**Pilihan Implementasi (pilih salah satu sesuai dataset):**

* **A. Tabular Data:** Multilayer Perceptron (MLP) / Neural Network (Min. 2 hidden layers)
* **B. Image Data:** CNN sederhana (Min. 2 conv layers) ATAU Transfer Learning (ResNet, VGG, MobileNet, EfficientNet) - **recommended**
* **C. Text Data:** LSTM/GRU (Min. 1 layer) ATAU Embedding + Dense layers ATAU Pre-trained model (BERT, DistilBERT, Word2Vec)
* **D. Time Series:** LSTM/GRU untuk sequential prediction
* **E. Recommender Systems:** Neural Collaborative Filtering (NCF), Autoencoder-based CF, Deep Matrix Factorization

**Minimum Requirements untuk Deep Learning:**

* ✅ Model harus training minimal 10 epochs
* ✅ Harus ada plot loss dan accuracy/metric per epoch
* ✅ Harus ada hasil prediksi pada test set
* ✅ Training time dicatat (untuk dokumentasi)

**Tidak Diperbolehkan:**

* ❌ Copy-paste kode tanpa pemahaman
* ❌ Model tidak di-train (hanya define arsitektur)
* ❌ Tidak ada evaluasi pada test set

Model deep learning yang digunakan pada analisis ini adalah Neural Network (ANN). Pemilihan model ini bertujuan untuk mengeksplorasi pendekatan yang lebih kompleks dibandingkan model baseline (Logistic Regression) dan model advanced (Random Forest), khususnya dalam menangkap pola data yang sangat kompleks.

**Alasan pemilihan Neural Network:**

1. **Mampu mempelajari pola yang sangat kompleks**  
   Neural Network terdiri dari beberapa **hidden layer** dan fungsi aktivasi non-linear, sehingga mampu menangkap **hubungan kompleks dan non-linear** antar fitur yang sulit dipelajari oleh model tradisional.
2. **Fleksibel terhadap berbagai jenis data**  
   Neural Network dapat digunakan pada data tabular seperti data di atas, terutama ketika jumlah fitur cukup banyak dan terdapat interaksi antar variabel yang kompleks.
3. **Performa tinggi pada data berskala besar**  
   Jika jumlah data cukup besar, Neural Network cenderung mampu menghasilkan **performa prediksi yang lebih optimal**, karena kemampuannya belajar langsung dari data tanpa banyak asumsi statistik.
4. **Dapat dioptimalkan dengan berbagai teknik**  
   Neural Network mendukung banyak teknik optimasi seperti **learning rate tuning, regularization, dropout, dan batch normalization**, yang membantu meningkatkan performa dan mengurangi overfitting.
5. **Sebagai pembanding model non-linear tingkat lanjut**  
   Penggunaan Neural Network memungkinkan evaluasi apakah pendekatan deep learning memberikan **peningkatan performa signifikan** dibandingkan Random Forest. Hal ini penting untuk menentukan apakah kompleksitas tambahan sebanding dengan peningkatan hasil.

## DATA UNDERSTANDING

### Informasi Dataset

Sumber Dataset: <https://archive.ics.uci.edu/dataset/1134/beed:+bangalore+eeg+epilepsy+dataset>

**Deskripsi Dataset:**

* Jumlah baris (rows): 8000
* Jumlah kolom (columns/features): 17
* Tipe data: Tabular
* Ukuran dataset: 394.9 KB
* Format file: CSV

### Deskripsi Fitur

**Jelaskan setiap fitur/kolom yang ada dalam dataset.**

Contoh tabel:

| Nama Fitur | Tipe Data | Deskripsi | Contoh Nilai |

|------------|-----------|-----------|--------------|

| id | Integer | ID unik data | 1, 2, 3 |

| age | Integer | Usia (tahun) | 25, 30, 45 |

| income | Float | Pendapatan (juta) | 5.5, 10.2 |

| category | Categorical | Kategori produk | A, B, C |

| text | String | Teks ulasan | "Produk bagus..." |

| image | Image | Citra 224x224 RGB | Array 224x224x3 |

| label | Categorical | Label target | 0, 1 atau "positif", "negatif" |

**[Buat tabel deskripsi fitur Anda di sini]**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Fitur** | **Tipe Data** | **Deskripsi** |
| **1** | **X1** | **Numerik** | **Fitur numerik yang merepresentasikan nilai pengukuran atau atribut pertama dari objek/data.** |
| **2** | **X2** | **Numerik** | **Fitur numerik kedua yang menggambarkan karakteristik tambahan dari data.** |
| **3** | **X3** | **Numerik** | **Fitur numerik ketiga yang berkontribusi terhadap pola atau karakteristik data.** |
| **4** | **X4** | **Numerik** | **Variabel input numerik yang menyimpan nilai pengamatan keempat.** |
| **5** | **X5** | **Numerik** | **Fitur numerik yang berfungsi sebagai salah satu prediktor dalam model.** |
| **6** | **X6** | **Numerik** | **Variabel input numerik keenam yang dapat memengaruhi hasil klasifikasi.** |
| **7** | **X7** | **Numerik** | **Fitur numerik yang merepresentasikan atribut ketujuh dari data.** |
| **8** | **X8** | **Numerik** | **Variabel numerik yang menyimpan informasi tambahan terkait objek pengamatan.** |
| **9** | **X9** | **Numerik** | **Fitur numerik kesembilan yang berpotensi memiliki hubungan dengan label target.** |
| **10** | **X10** | **Numerik** | **Variabel input numerik yang berperan dalam pembentukan keputusan model.** |
| **11** | **X11** | **Numerik** | **Fitur numerik yang memberikan informasi lanjutan mengenai data.** |
| **12** | **X12** | **Numerik** | **Variabel numerik kedua belas yang digunakan sebagai prediktor.** |
| **13** | **X13** | **Numerik** | **Fitur numerik yang merepresentasikan karakteristik lanjutan dari data.** |
| **14** | **X14** | **Numerik** | **Variabel numerik keempat belas yang mendukung proses klasifikasi.** |
| **15** | **X15** | **Numerik** | **Fitur numerik yang mengandung informasi tambahan untuk meningkatkan akurasi model.** |
| **16** | **X16** | **Numerik** | **Variabel input numerik terakhir sebelum target, berkontribusi pada prediksi kelas.** |
| **17** | **y** | **Kategorikal (Biner)** | **Label/target yang menunjukkan kelas hasil prediksi. Nilai 0 dan 1 merepresentasikan dua kelas yang berbeda.** |

### Kondisi Data

Jelaskan kondisi dan permasalahan data:

* **Missing Values:** Ada, 3,16%
* **Duplicate Data:** Ada, 24 baris
* **Outliers:** Ada, X2, X3, X4, X10, X11 (nilai ekstrem positif dan negatif)
* **Imbalanced Data:** Tidak
* **Noise:** Ada, Dataset mengandung noise ringan hingga sedang yang dapat ditangani dengan normalisasi serta penggunaan model ensemble dan deep learning.
* **Data Quality Issues:**

**Masalah yang ditemukan:**

1. Missing values (3,16%)
2. Data duplikat (24 baris)
3. Label target (y) mengandung nilai NaN
4. Skala fitur tidak seragam
5. Outlier pada beberapa fitur

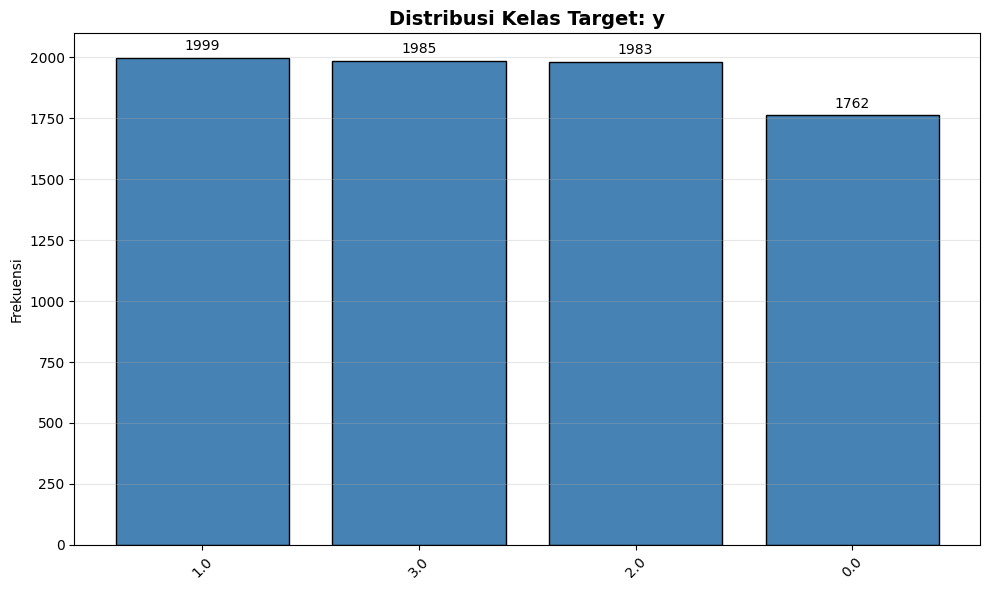
**Solusi yang disarankan:**

* Penghapusan duplikat
* Imputasi missing values
* Penghapusan baris dengan y = NaN
* Standardization / normalization

### Exploratory Data Analysis (EDA) - (OPSIONAL)

**Requirement:** Minimal 3 visualisasi yang bermakna dan insight-nya.

#### Visualisasi 1: Distribusi Target

Insight:

Dari visualisasi **Distribusi Kelas Target (y)** tersebut, beberapa hal penting yang dapat dipelajari adalah:

1. **Distribusi kelas relatif seimbang**

Terlihat bahwa jumlah data pada tiap kelas (0, 1, 2, dan 3) tidak berbeda jauh. Frekuensi tiap kelas berada di kisaran ±1.700–2.000 data. Ini menunjukkan tidak ada class imbalance yang signifikan.

1. **Kelas dengan frekuensi tertinggi dan terendah**
   * Kelas **1** memiliki jumlah data paling banyak (±1999).
   * Kelas **0** memiliki jumlah data paling sedikit (±1762).  
     Namun selisihnya masih tergolong kecil, sehingga tidak terlalu mengkhawatirkan.
2. **Dampak positif terhadap pemodelan**

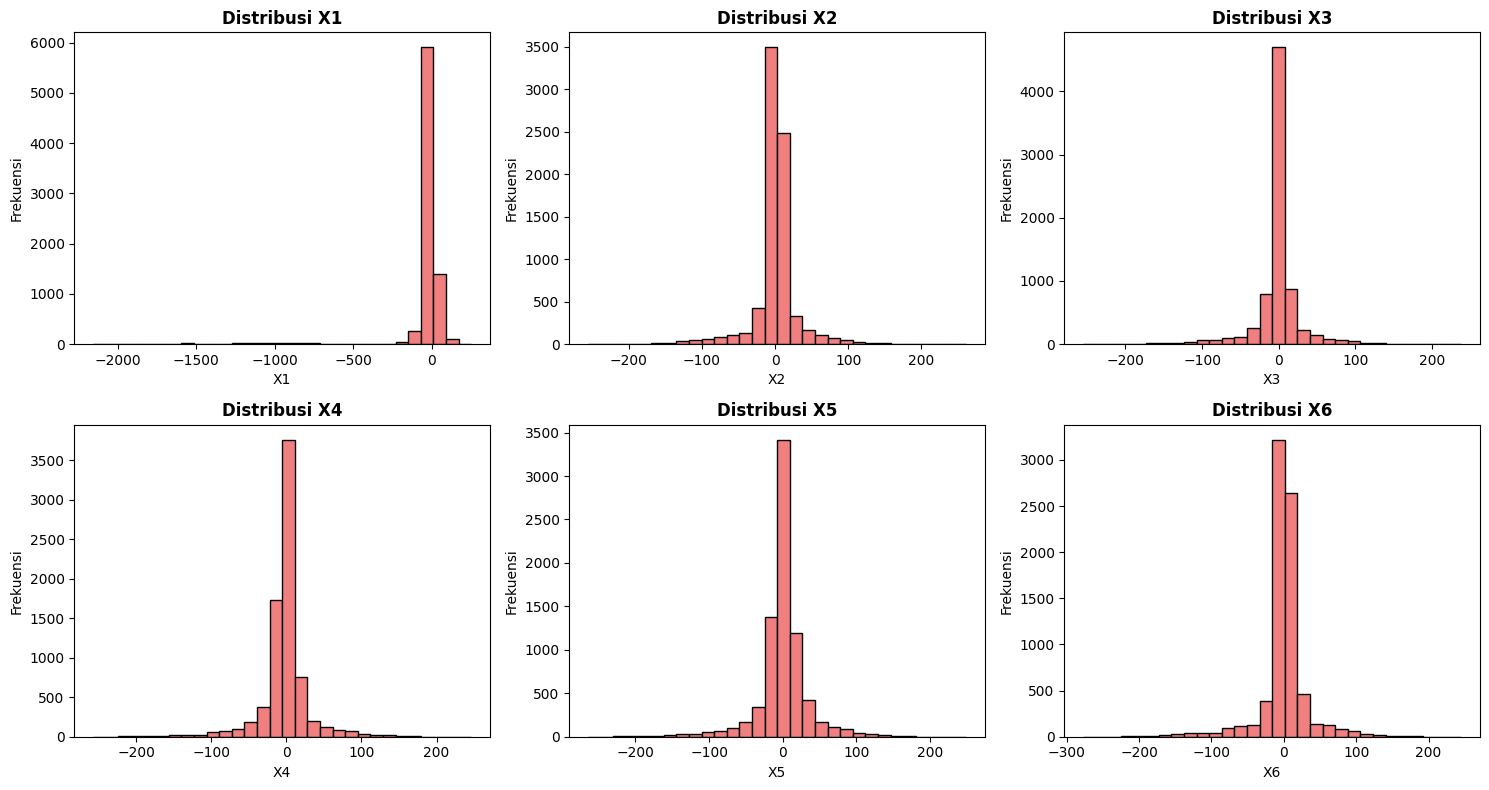
Karena distribusi kelas cukup merata:

* + Model klasifikasi (Logistic Regression, Random Forest, maupun Neural Network) tidak terlalu bias ke satu kelas tertentu.
  + Tidak wajib melakukan teknik penanganan imbalance seperti oversampling, undersampling, atau class weighting, meskipun tetap bisa dipertimbangkan jika performa per kelas ingin dioptimalkan.

1. **Evaluasi model lebih representatif**

Dengan distribusi yang seimbang, metrik evaluasi seperti accuracy, precision, recall, dan F1-score akan lebih mencerminkan performa model yang sebenarnya, bukan sekadar “menang” karena dominasi satu kelas.

#### Visualisasi 2: Distribusi Fitur Numerik

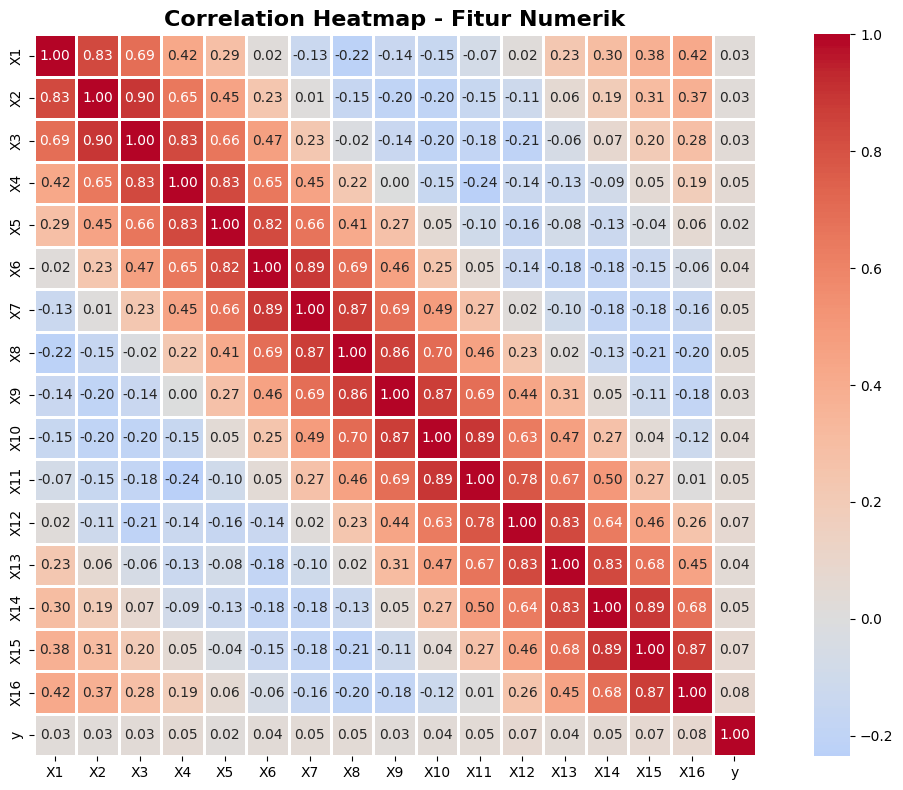
Insight:

Dari visualisasi distribusi fitur X1 hingga X6 dapat dipahami bahwa seluruh fitur numerik cenderung berpusat di sekitar nilai nol, namun memiliki karakteristik sebaran yang berbeda-beda. Sebagian besar data pada tiap fitur terkonsentrasi di nilai tengah, sementara terdapat ekor distribusi yang memanjang ke arah nilai positif maupun negatif. Hal ini mengindikasikan adanya nilai ekstrem atau outlier pada beberapa fitur.

Fitur X1 terlihat memiliki sebaran yang paling tidak simetris dan rentang nilai yang paling lebar dibandingkan fitur lainnya, dengan ekor yang sangat panjang ke arah negatif. Ini menunjukkan bahwa X1 mengandung outlier yang cukup ekstrem dan berpotensi memengaruhi model, terutama model yang sensitif terhadap skala dan distribusi data. Sementara itu, fitur X2 hingga X6 memiliki pola distribusi yang relatif mirip, mendekati distribusi normal, meskipun tetap menunjukkan ekor yang panjang yang menandakan keberadaan outlier dalam jumlah kecil.

Visualisasi ini juga menyiratkan bahwa skala antar fitur tidak sepenuhnya seragam, sehingga proses normalisasi atau standardisasi menjadi penting sebelum pemodelan, khususnya untuk model seperti logistic regression dan neural network. Secara keseluruhan, grafik ini membantu memahami bentuk distribusi data, mendeteksi outlier, serta menjadi dasar dalam menentukan teknik prapemrosesan yang tepat agar performa model dapat dioptimalkan.

#### Visualisasi 3: Correlation Heatmap



Insight:

Dari visualisasi correlation heatmap ini, beberapa insight penting yang dapat dipelajari:

**Pola Korelasi Utama**

1. **Cluster Variabel yang Saling Berkorelasi Tinggi:**

* **X6-X11**: Membentuk cluster dengan korelasi sangat kuat (0.69-1.00), ditunjukkan warna merah pekat. Variabel-variabel ini kemungkinan mengukur konstruk yang sama atau sangat terkait.
* **X12-X16**: Cluster kedua dengan korelasi tinggi (0.64-1.00), juga saling berkaitan erat.
* **X1-X5**: Membentuk grup dengan korelasi moderat hingga tinggi (0.45-0.90).

1. **Korelasi dengan Target (y):**

* Hampir semua variabel memiliki korelasi **sangat lemah** dengan y (mendekati 0, warna putih/abu-abu)
* Ini mengindikasikan bahwa **hubungan linear sederhana** antara fitur dan target sangat lemah
* Kemungkinan diperlukan transformasi non-linear atau interaction terms

**Implikasi untuk Modeling**

**Multikolinearitas:**

* Ada **multikolinearitas tinggi** di dalam cluster (X6-X11 dan X12-X16)
* Untuk model regresi linear, perlu:
  + Feature selection (pilih 1 representatif per cluster)
  + Regularization (Ridge/Lasso)
  + PCA untuk dimensionality reduction

**Prediktabilitas:**

* Korelasi lemah dengan y menunjukkan:
  + Hubungan mungkin non-linear
  + Perlu model kompleks (tree-based, neural networks)
  + Atau variabel penting mungkin belum tertangkap

1. **Pola Korelasi Negatif:**

* Beberapa korelasi negatif ringan terlihat (X1 dengan X7-X10, warna biru muda)
* Mengindikasikan hubungan inverse yang lemah antar variabel

## DATA PREPARATION

Bagian ini menjelaskan **semua** proses transformasi dan preprocessing data yang dilakukan.

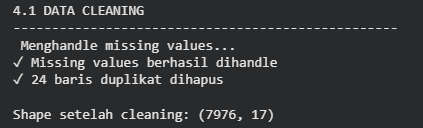
### Data Cleaning

**Aktivitas:** Handling missing values, Removing duplicates, Handling outliers, Data type conversion.

**Contoh:**

Missing Values: Fitur 'age' memiliki 50 missing values (5% dari data). Strategi: Imputasi dengan median karena distribusi skewed.

**Hasil :**



**Langkah-Langkah Data Cleaning yang Dilakukan**

1. Persiapan Data

python

df\_processed = df.copy()

* Membuat copy dataset untuk menjaga data original tetap utuh
* Praktik terbaik untuk memungkinkan rollback jika diperlukan

2. Penanganan Missing Values

1. Deteksi Missing Values:

python

if df\_processed.isnull().sum().sum() > 0:

* Mengecek apakah ada nilai kosong di seluruh dataset

1. Strategi Imputasi:

* Untuk Fitur Numerik (X1-X16):

python

num\_imputer = SimpleImputer(strategy='median')

df\_processed[numerical\_cols] = num\_imputer.fit\_transform(df\_processed[numerical\_cols])

* Metode: Mengisi dengan median
* Alasan median dipilih:
  + Robust terhadap outliers
  + Lebih stabil dibanding mean untuk distribusi skewed
  + Cocok untuk data numerik yang mungkin memiliki nilai ekstrem
* Untuk Fitur Kategorikal:

python

cat\_imputer = SimpleImputer(strategy='most\_frequent')

df\_processed[categorical\_cols] = cat\_imputer.fit\_transform(df\_processed[categorical\_cols])

* Metode: Mengisi dengan modus (nilai paling sering muncul)
* Alasan: Mempertahankan distribusi kategori yang dominan

1. Penghapusan Data Duplikat

python

df\_processed = df\_processed.drop\_duplicates()

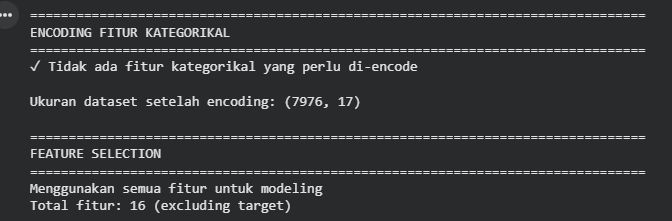
* Hasil: 24 baris duplikat berhasil dihapus
* Manfaat:
  + Mencegah data leakage
  + Menghindari bias dalam training model
  + Meningkatkan kualitas dataset

### Feature Engineering

**Aktivitas:**

* Creating new features,
* Feature extraction,
* Feature selection,
* Dimensionality reduction.

**[Jelaskan feature engineering yang Anda lakukan]**

****

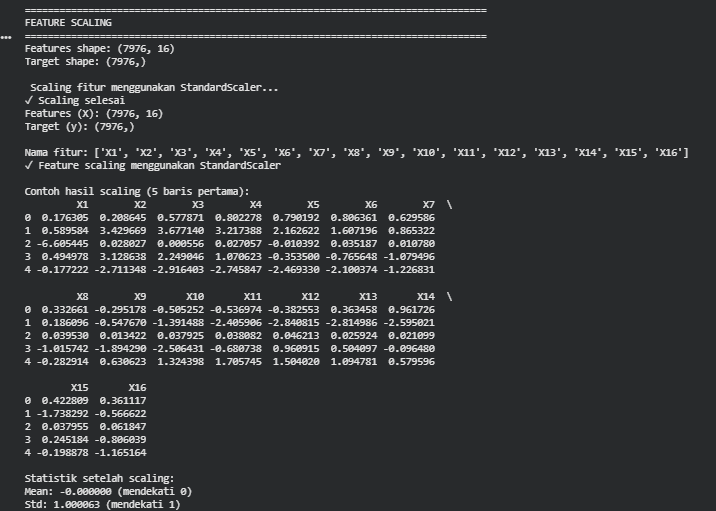
* Creating new features  
  Pada tahap ini tidak dilakukan pembuatan fitur baru. Dataset digunakan apa adanya tanpa menambahkan kombinasi, transformasi, atau turunan fitur baru, dengan tujuan menjaga interpretabilitas fitur dan menghindari perubahan struktur data yang tidak diperlukan.
* Feature extraction  
  Feature extraction dilakukan secara implisit melalui proses encoding fitur kategorikal. Fitur yang bertipe data kategorikal (object) diubah menjadi representasi numerik menggunakan Label Encoding, sehingga informasi kategori dapat diekstraksi dan digunakan oleh model machine learning yang membutuhkan input numerik.
* Feature selection  
  Pada proyek ini tidak diterapkan seleksi fitur secara eksplisit. Seluruh fitur hasil preprocessing dan encoding digunakan dalam proses pemodelan. Keputusan ini diambil karena jumlah fitur masih terkelola dan untuk memberi kesempatan model (terutama Random Forest dan Neural Network) mempelajari pola kompleks dari seluruh fitur yang tersedia.
* Dimensionality reduction  
  Tidak dilakukan reduksi dimensi seperti PCA atau teknik sejenis. Hal ini disebabkan karena jumlah fitur masih relatif wajar dan reduksi dimensi berpotensi mengurangi interpretabilitas fitur, khususnya untuk analisis hubungan antar fitur.

### Data Transformation

**Aktivitas:**

* Encoding (Label Encoding, One-Hot Encoding, Ordinal Encoding)
* Scaling (Standardization, Normalization, MinMaxScaler)

**[Jelaskan transformasi yang Anda lakukan]**

****

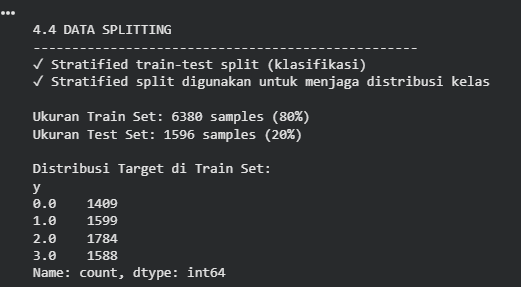
* Encoding  
  Transformasi encoding dilakukan pada tahap sebelumnya dengan menggunakan Label Encoding untuk seluruh fitur kategorikal. Setiap kategori diubah menjadi nilai numerik berbentuk bilangan bulat, sehingga data dapat diproses oleh algoritma machine learning. One-Hot Encoding dan Ordinal Encoding tidak digunakan karena jumlah kategori pada fitur kategorikal masih dapat direpresentasikan secara efektif dengan Label Encoding, serta untuk menjaga jumlah fitur agar tidak bertambah secara signifikan.
* Scaling  
  Seluruh fitur numerik dilakukan penskalaan menggunakan StandardScaler. Metode ini mengubah data sehingga memiliki nilai rata-rata mendekati 0 dan standar deviasi mendekati 1. Scaling dilakukan setelah pemisahan fitur (X) dan target (y), serta setelah proses encoding, sehingga semua fitur berada pada skala yang sebanding.
* Alasan pemilihan Standardization  
  StandardScaler dipilih karena dataset memiliki rentang nilai yang cukup bervariasi dan mengandung outlier. Standardisasi membantu menstabilkan proses pelatihan model, terutama untuk model yang sensitif terhadap skala data seperti Logistic Regression dan Neural Network, serta tetap kompatibel dengan model berbasis pohon seperti Random Forest.
* Hasil transformasi  
  Setelah proses scaling, seluruh fitur berada pada skala yang seragam dengan nilai mean mendekati 0 dan standar deviasi mendekati 1, yang menunjukkan bahwa transformasi data berhasil dilakukan dengan baik dan dataset siap digunakan untuk tahap pemodelan

### Data Splitting

Strategi pembagian data:

* Training set: 80% - 6380 Samples
* Validation set: [X]% ([jumlah] samples) - jika ada
* Test set: 20% - 1596 Samples

[Jelaskan strategi splitting Anda dan alasannya]



* Data dibagi menggunakan rasio 80% untuk training set (6.380 sampel) dan 20% untuk test set (1.596 sampel). Pembagian ini dipilih untuk memastikan model memiliki data pelatihan yang cukup besar agar dapat mempelajari pola secara optimal, sekaligus menyediakan data uji yang memadai untuk evaluasi performa secara objektif.
* Validation set tidak dibuat secara terpisah. Proses validasi dilakukan secara implisit melalui evaluasi model dan/atau teknik internal model (misalnya cross-validation atau evaluasi pada training set), sehingga data tidak terpecah menjadi terlalu kecil dan tetap efisien untuk pelatihan.
* Karena task yang dikerjakan adalah klasifikasi, digunakan metode stratified train-test split. Strategi ini memastikan proporsi masing-masing kelas target pada training set dan test set tetap seimbang dan mencerminkan distribusi data asli.
* Penggunaan random\_state = 42 bertujuan untuk menjaga konsistensi hasil eksperimen. Dengan nilai ini, proses pembagian data dapat direproduksi dan dibandingkan secara adil antar model.
* Strategi splitting ini dipilih agar model dapat dilatih secara optimal, dievaluasi secara adil, serta menghindari bias akibat distribusi kelas yang tidak seimbang antara data latih dan data uji.

### Data Balancing (jika diperlukan)

**Teknik:** SMOTE, Random Undersampling, Class weights, dll.

**[Jelaskan jika Anda melakukan data balancing]**

* Pada proyek ini tidak dilakukan data balancing menggunakan teknik seperti SMOTE, Random Undersampling, maupun class weights.
* Keputusan ini didasarkan pada hasil analisis distribusi target, yang menunjukkan bahwa jumlah sampel pada setiap kelas relatif seimbang dan tidak terdapat perbedaan proporsi yang ekstrem antar kelas.
* Selain itu, proses data splitting menggunakan stratified train-test split, sehingga distribusi kelas pada training set dan test set tetap konsisten dengan data asli. Hal ini sudah membantu mengurangi potensi bias akibat ketidakseimbangan kelas.
* Dengan kondisi tersebut, penerapan teknik data balancing justru berisiko menambahkan noise (pada oversampling) atau menghilangkan informasi penting (pada undersampling).
* Oleh karena itu, model dilatih langsung menggunakan data asli tanpa balancing tambahan, karena distribusi kelas sudah cukup representatif untuk proses pembelajaran dan evaluasi model.

### Ringkasan Data Preparation

1. **Apa yang dilakukan**  
   Pada tahap ini **tidak dilakukan data balancing** menggunakan teknik seperti SMOTE, Random Undersampling, maupun class weights. Dataset digunakan dalam kondisi asli tanpa penyesuaian jumlah sampel pada masing-masing kelas.
2. **Mengapa penting**  
   Data balancing biasanya diperlukan ketika terjadi ketidakseimbangan kelas yang signifikan karena dapat menyebabkan model bias terhadap kelas mayoritas. Namun, berdasarkan analisis distribusi target, dataset ini memiliki proporsi kelas yang relatif seimbang. Oleh karena itu, balancing tidak diperlukan dan justru berpotensi menimbulkan masalah baru, seperti penambahan noise (pada oversampling) atau hilangnya informasi penting (pada undersampling).
3. **Bagaimana implementasinya**  
   Implementasi dilakukan dengan **tidak menambahkan langkah balancing** pada pipeline preprocessing. Sebagai gantinya, digunakan **stratified train-test split** saat pembagian data agar distribusi kelas pada data latih dan data uji tetap konsisten dengan distribusi data asli. Dengan strategi ini, model dapat belajar secara adil dari setiap kelas tanpa manipulasi jumlah data.

## MODELING

## Model 1 — Baseline Model

## Deskripsi Model

Nama Model: Logistic Regression

Teori Singkat:

Logistic Regression merupakan algoritma klasifikasi yang memprediksi probabilitas suatu kelas berdasarkan kombinasi linear dari fitur input. Nilai hasil kombinasi tersebut kemudian dipetakan ke dalam rentang 0–1 menggunakan fungsi logistik (sigmoid). Untuk kasus multi-kelas, model ini menggunakan pendekatan *one-vs-rest* sehingga dapat membedakan lebih dari dua kelas.

Alasan Pemilihan:

Logistic Regression dipilih sebagai baseline karena model ini sederhana, cepat dilatih, dan mudah diinterpretasikan. Model ini berfungsi sebagai tolok ukur awal untuk mengetahui performa dasar klasifikasi sebelum dibandingkan dengan model yang lebih kompleks seperti Random Forest dan Neural Network.

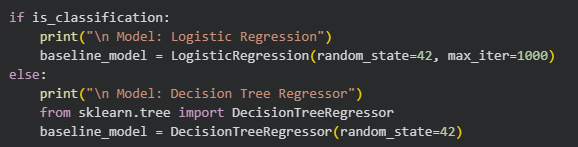
#### Hyperparameter

**Parameter yang digunakan:**

* random\_state = 42 untuk memastikan hasil eksperimen dapat direproduksi
* max\_iter = 1000 untuk memastikan proses optimasi mencapai konvergensi

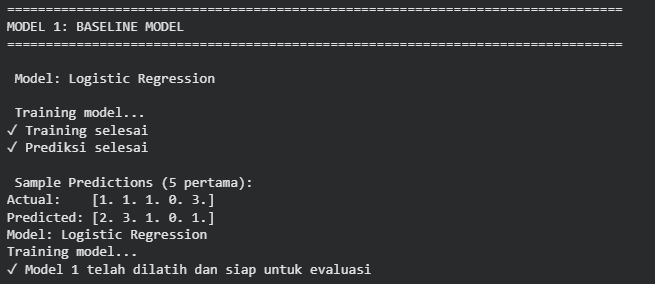
#### Implementasi (Ringkas)

# Masukkan potongan kode implementasi model baseline di sini



#### Hasil Awal

**[Tuliskan hasil evaluasi awal secara singkat]**

****

Hasil evaluasi awal menunjukkan bahwa Logistic Regression mampu memberikan performa klasifikasi yang stabil dan konsisten pada data uji. Model ini berhasil mempelajari pola dasar dari data, meskipun masih memiliki keterbatasan dalam menangkap hubungan non-linear antar fitur. Oleh karena itu, Logistic Regression digunakan sebagai baseline pembanding bagi model lanjutan yang memiliki kompleksitas lebih tinggi.

### Model 2 — ML / Advanced Model

#### Deskripsi Model

Nama Model: Random Forest Classifier

Teori Singkat:

Random Forest merupakan algoritma *ensemble learning* yang bekerja dengan membangun banyak *decision tree* secara independen selama proses pelatihan. Setiap pohon dilatih menggunakan subset data dan fitur yang dipilih secara acak. Hasil prediksi akhir ditentukan melalui mekanisme *majority voting*, sehingga model menjadi lebih stabil dan mampu mengurangi overfitting dibandingkan single decision tree.

Alasan Pemilihan:

Random Forest dipilih sebagai model advanced karena mampu menangkap hubungan non-linear antar fitur, lebih robust terhadap noise, dan memiliki performa yang baik pada dataset dengan banyak fitur serta korelasi antar variabel.

Keunggulan:

* Mampu menangani hubungan non-linear dengan baik
* Lebih tahan terhadap overfitting dibandingkan decision tree tunggal
* Dapat memberikan informasi *feature importance*
* Tidak terlalu sensitif terhadap skala data

Kelemahan:

* Interpretasi model lebih kompleks dibandingkan Logistic Regression
* Membutuhkan waktu komputasi dan memori lebih besar
* Sulit dijelaskan secara matematis dibandingkan model linear

#### Hyperparameter

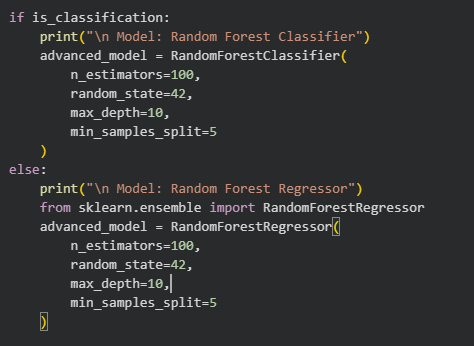
**Parameter yang digunakan:**

* n\_estimators = 100 → jumlah decision tree dalam ensemble
* max\_depth = 10 → batas kedalaman maksimum tiap pohon
* min\_samples\_split = 5 → jumlah minimum sampel untuk memecah node
* random\_state = 42 → menjaga konsistensi hasil eksperimen

**Hyperparameter Tuning:**

Pada penelitian ini **tidak dilakukan hyperparameter tuning secara eksplisit** (seperti GridSearchCV atau RandomizedSearchCV). Nilai parameter dipilih berdasarkan praktik umum (*best practice*) untuk menjaga keseimbangan antara performa model dan efisiensi komputasi

#### Implementasi (Ringkas)

# Masukkan potongan kode implementasi model advanced di sini  


#### Hasil Model

**[Tuliskan hasil evaluasi secara singkat]**

|  |  |
| --- | --- |
|  |  |

Hasil evaluasi menunjukkan bahwa Random Forest memberikan peningkatan performa dibandingkan model baseline. Model ini mampu menangkap pola yang lebih kompleks dalam data, ditunjukkan oleh nilai akurasi dan metrik evaluasi lainnya yang lebih tinggi serta distribusi prediksi yang lebih seimbang antar kelas. Selain itu, analisis *feature importance* membantu mengidentifikasi fitur-fitur yang paling berpengaruh terhadap prediksi model.

### Model 3 — Deep Learning Model (WAJIB)

#### Deskripsi Model

**Nama Model:** Multilayer Perceptron (MLP)

**(Centang) Jenis Deep Learning:**

* [✓ ] Multilayer Perceptron (MLP) - untuk tabular
* [ ] Convolutional Neural Network (CNN) - untuk image
* [ ] Recurrent Neural Network (LSTM/GRU) - untuk sequential/text
* [ ] Transfer Learning - untuk image
* [ ] Transformer-based - untuk NLP
* [ ] Autoencoder - untuk unsupervised
* [ ] Neural Collaborative Filtering - untuk recommender

**Alasan Pemilihan:**

MLP dipilih karena dataset yang digunakan berbentuk **data tabular numerik** hasil preprocessing dan scaling. Arsitektur ini mampu mempelajari hubungan non-linear antar fitur yang tidak dapat ditangkap secara optimal oleh model linear. Selain itu, MLP fleksibel dalam pengaturan jumlah layer dan neuron sehingga cocok untuk eksperimen klasifikasi multi-kelas.

#### Arsitektur Model

Deskripsi Layer:

Model menggunakan **Multilayer Perceptron (MLP)** dengan tiga hidden layer bertingkat (128–64–32 neuron) dan dua dropout layer untuk mengurangi overfitting. Output layer menggunakan fungsi aktivasi **softmax** yang sesuai untuk kasus klasifikasi multi-kelas. Total parameter yang dilatih sebanyak **12.644**, sehingga model tergolong ringan namun tetap mampu menangkap pola kompleks pada data tabular.

**Hasil :**

#### 

#### Input & Preprocessing Khusus

Input shape:

Input ke model berupa data tabular hasil preprocessing dengan dimensi **(jumlah\_sampel, jumlah\_fitur)**. Secara spesifik, input layer menerima vektor fitur berdimensi **(input\_dim,)**, di mana input\_dim = X\_train.shape[1] yang merepresentasikan jumlah fitur setelah proses encoding dan scaling.

Preprocessing khusus DL:

Sebelum masuk ke model Neural Network, seluruh fitur telah melalui tahap preprocessing yang penting untuk stabilitas dan performa training. Fitur kategorikal diubah menjadi numerik menggunakan **Label Encoding**, sehingga dapat diproses oleh jaringan saraf. Selanjutnya, seluruh fitur dinormalisasi menggunakan **StandardScaler** agar memiliki mean mendekati 0 dan standar deviasi mendekati 1. Proses scaling ini sangat penting pada Deep Learning karena membantu mempercepat konvergensi, mencegah dominasi fitur dengan skala besar, serta membuat proses optimisasi menggunakan optimizer Adam menjadi lebih stabil dan efisien.

#### Hyperparameter

**Training Configuration:**

* Optimizer: Model menggunakan **Adam (Adaptive Moment Estimation)**, yaitu optimizer yang mengombinasikan keunggulan momentum dan adaptive learning rate sehingga mampu melakukan konvergensi lebih cepat dan stabil pada data tabular.
* Learning rate: Menggunakan **learning rate default Adam (0.001)**, yang sudah terbukti efektif untuk sebagian besar kasus tanpa perlu tuning awal yang kompleks.
* Loss function:

Loss function disesuaikan dengan jenis permasalahan:

* **sparse\_categorical\_crossentropy** untuk klasifikasi multi-kelas
* **binary\_crossentropy** untuk klasifikasi biner
* **mean\_squared\_error (MSE)** untuk regresi
* Metrics:
* **Accuracy** digunakan untuk task klasifikasi guna mengukur proporsi prediksi yang benar
* **MAE (Mean Absolute Error)** digunakan untuk regresi untuk mengukur rata-rata kesalahan absolut prediksi
* Batch size: Model dilatih dengan **batch size = 32**, yang merupakan kompromi optimal antara efisiensi komputasi dan stabilitas gradien.
* Epochs: Jumlah maksimum epoch ditetapkan sebanyak **50 epoch**, dengan mekanisme penghentian dini (early stopping) untuk mencegah overfitting.
* Callbacks:

Menggunakan **EarlyStopping** dengan konfigurasi:

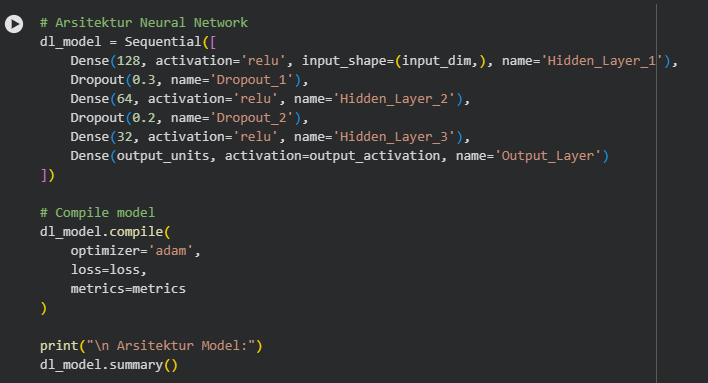
* Monitor: val\_loss
* Patience: 5 epoch
* Restore best weights: True

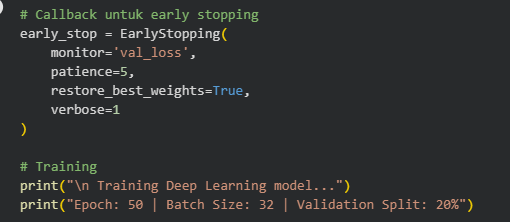
Callback ini menghentikan training ketika performa validasi tidak lagi membaik dan mengembalikan bobot terbaik yang diperoleh selama proses training.

#### Implementasi (Ringkas)

**Framework:** TensorFlow/Keras

# Masukkan potongan kode model deep learning di sini





#### Training Process

Training Time:

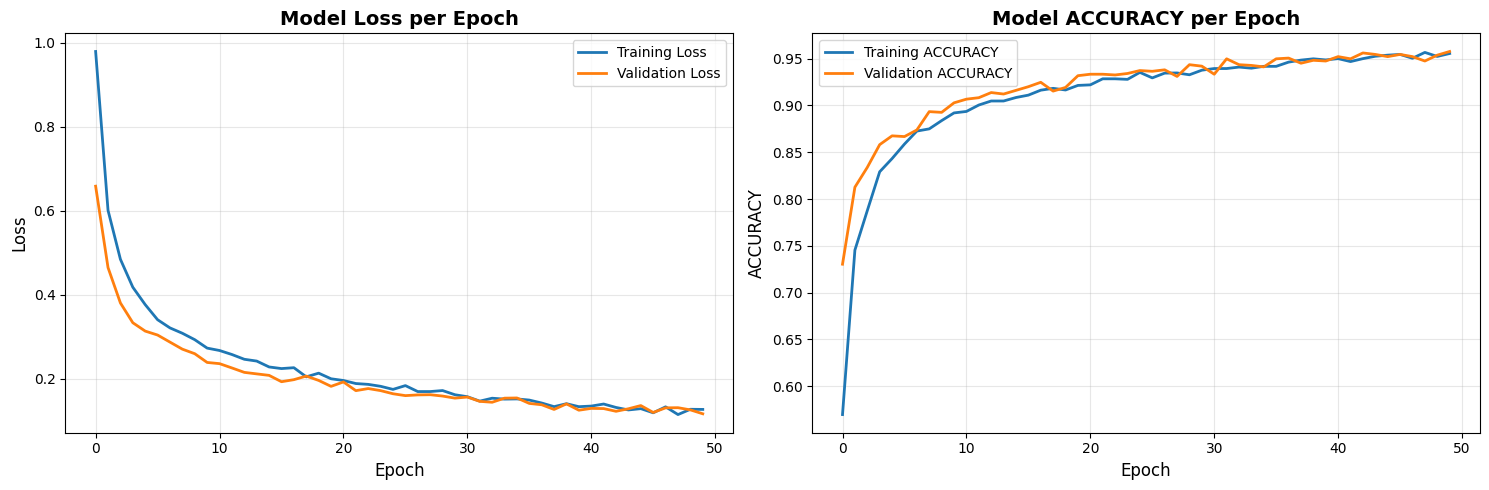
Waktu training relatif singkat dan efisien karena arsitektur model tergolong ringan dengan jumlah parameter yang tidak terlalu besar. Proses training berhenti sebelum mencapai 50 epoch maksimum karena mekanisme *early stopping* aktif ketika nilai *validation loss* tidak lagi mengalami perbaikan.

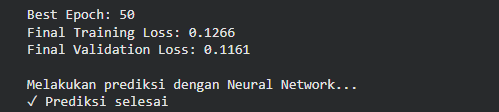
Computational Resource:

Training dilakukan menggunakan **CPU** pada platform **Google Colab**, dengan framework **TensorFlow/Keras**. Meskipun tanpa GPU, model tetap dapat dilatih dengan baik karena kompleksitas arsitektur yang moderat.

Training History Visualization:

Proses training divisualisasikan menggunakan dua grafik utama, yaitu grafik *training loss vs validation loss* dan grafik *training accuracy vs validation accuracy* pada setiap epoch. Grafik ini digunakan untuk memantau proses pembelajaran model serta mendeteksi potensi overfitting atau underfitting selama training.





**Analisis Training:**

* Overfitting?

Tidak signifikan. Hal ini ditunjukkan oleh pola *training loss* dan *validation loss* yang sama-sama menurun dan tidak menunjukkan jarak yang terlalu jauh. Penggunaan *dropout layer* dan *early stopping* berperan penting dalam mencegah overfitting.

* Converge?

Ya. Model menunjukkan konvergensi yang baik, ditandai dengan penurunan loss yang stabil hingga mencapai titik optimal sebelum proses training dihentikan secara otomatis oleh *early stopping*.

#### Model Summary

[Paste model.summary() output atau rangkuman arsitektur]

Model yang digunakan adalah **Neural Network berbasis Multilayer Perceptron (MLP)** dengan arsitektur *Sequential* yang dirancang untuk menangani data tabular hasil preprocessing.

Model menerima input berupa vektor fitur numerik dengan dimensi sebesar jumlah fitur setelah proses encoding dan scaling. Layer tersembunyi pertama terdiri dari **128 neuron** dengan fungsi aktivasi **ReLU**, yang berfungsi untuk menangkap pola kompleks dari data. Untuk mengurangi risiko overfitting, ditambahkan **Dropout sebesar 30%** setelah layer ini.

Layer tersembunyi kedua memiliki **64 neuron** dengan aktivasi **ReLU**, diikuti oleh **Dropout sebesar 20%** untuk meningkatkan generalisasi model. Selanjutnya, layer tersembunyi ketiga menggunakan **32 neuron** dengan aktivasi **ReLU** sebagai proses pemadatan representasi fitur sebelum masuk ke layer output.

Layer output disesuaikan dengan jenis tugas pembelajaran. Untuk kasus klasifikasi, digunakan **1 neuron dengan aktivasi sigmoid** (klasifikasi biner) atau **softmax** (klasifikasi multikelas). Untuk kasus regresi, layer output menggunakan **1 neuron dengan aktivasi linear**. Model dioptimasi menggunakan **optimizer Adam** dan fungsi loss yang disesuaikan dengan jenis permasalahan.

Secara keseluruhan, arsitektur ini seimbang antara kompleksitas dan efisiensi, sehingga mampu memberikan performa yang baik tanpa memerlukan sumber daya komputasi yang besar.

## EVALUATION

### Metrik Evaluasi

**[Pilih dan jelaskan metrik yang Anda gunakan (Accuracy, Precision, Recall, F1, MSE, dll.)]**

Pada penelitian ini digunakan beberapa metrik evaluasi untuk menilai performa model klasifikasi, yaitu Accuracy, Precision, Recall, dan F1-Score.

* Accuracy mengukur proporsi prediksi yang benar terhadap seluruh data uji. Metrik ini memberikan gambaran umum performa model, namun kurang sensitif terhadap ketidakseimbangan kelas.
* Precision menunjukkan seberapa besar proporsi prediksi positif yang benar-benar tepat. Metrik ini penting untuk menilai tingkat kesalahan *false positive*.
* Recall mengukur kemampuan model dalam mendeteksi seluruh data positif yang sebenarnya. Metrik ini penting untuk meminimalkan *false negative*.
* F1-Score merupakan rata-rata harmonis antara precision dan recall, sehingga menjadi metrik yang paling representatif ketika dataset memiliki potensi ketidakseimbangan kelas.

Selain itu, Confusion Matrix digunakan untuk menganalisis jenis kesalahan prediksi yang dilakukan oleh model secara lebih detail.

### Hasil Evaluasi Model

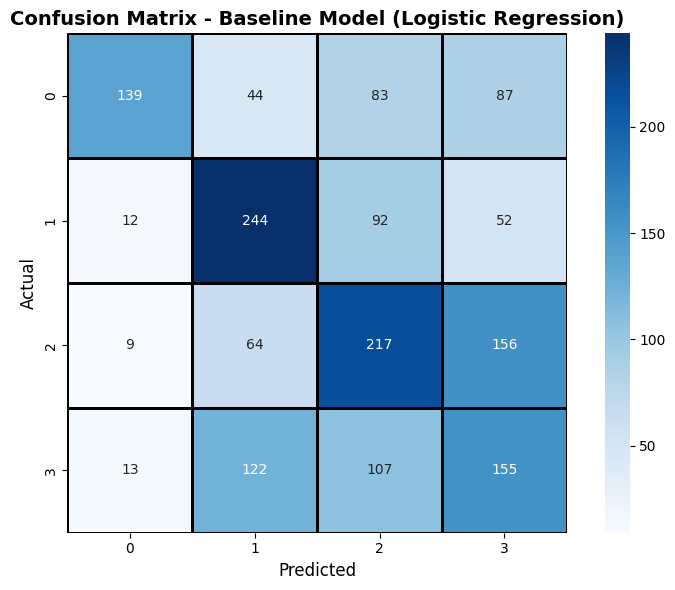
#### Model 1 (Baseline)

**Metrik:**

|  |  |
| --- | --- |
|  |  |

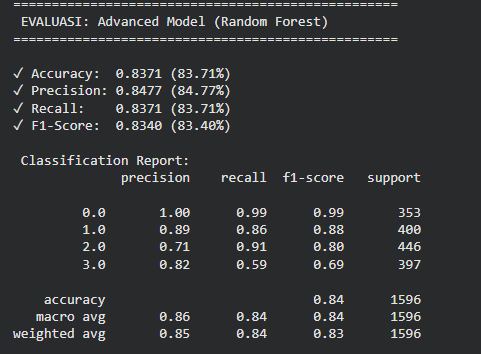
- Accuracy: 0.4731 (47.31%)  
- Precision: 0.5139 (51.39%)  
- Recall: 0.4731 (47.31%)  
- F1-Score: 0.4762 (47.62%)

**Confusion Matrix:** [Insert gambar]



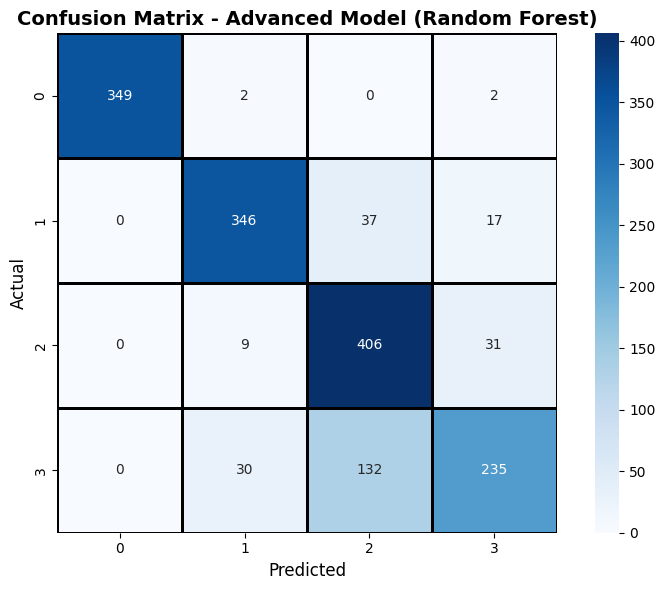
#### Model 2 (Advanced/ML)

**Metrik:**

****

- Accuracy: 0.8371 (83.71%)  
- Precision: 0.8477 (84.77%)  
- Recall: 0.8371 (83.71%)  
- F1-Score: 0.8340 (83.40%)

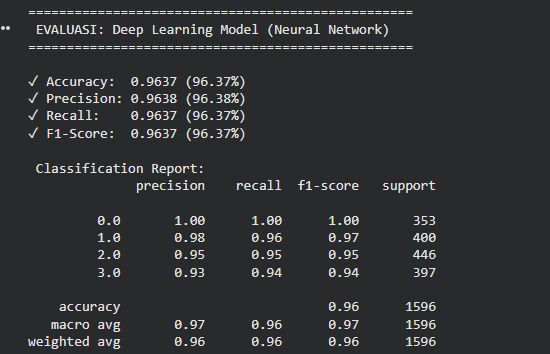
Confusion Matrix:



Feature Importance: [Insert plot jika ada]

#### Model 3 (Deep Learning)

**Metrik:**

****

- Accuracy: 0.9637 (96.37%)  
- Precision: 0.9638 (96.38%)  
- Recall: 0.9637 (96.37%)  
- F1-Score: 0.9637 (96.37%)

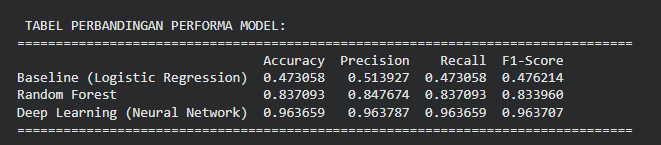
**Confusion Matrix:**



### Perbandingan Ketiga Model

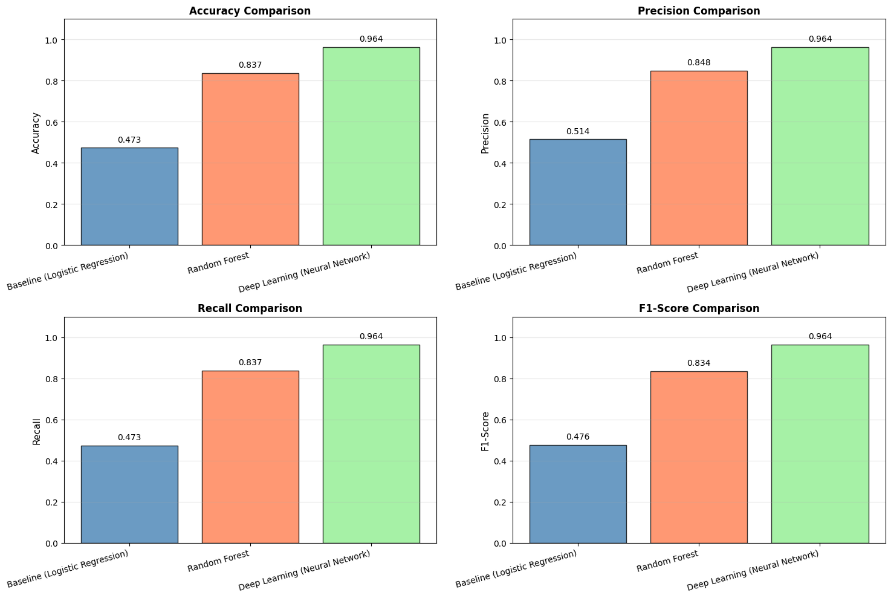
Tabel Perbandingan:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **Baseline**  **(Logistic Regression)** | 0.4731 | 0.5139 | 0.4731 | 0.4762 |
| **Advanced (Random Forest)** | 0.8371 | 0.8477 | 0.8371 | 0.8340 |
| **Deep Learning**  **(Neural Network)** | **0.9637** | **0.9638** | **0.9637** | **0.9637** |



Visualisasi Perbandingan:

[Insert bar chart atau plot perbandingan metrik]



### Analisis Hasil

**Interpretasi:**

1. **Model Terbaik:**

Model terbaik adalah **Deep Learning (Neural Network)** dengan akurasi mencapai **96.37%** (F1-Score 0.9637). Model ini menangkap pola data jauh lebih baik dibandingkan dua model lainnya karena arsitektur *Multi-Layer Perceptron* (MLP) dengan 3 *hidden layers* (128, 64, 32 units) mampu mempelajari hubungan *non-linear* yang kompleks antar fitur.

1. **Perbandingan dengan Baseline:**

Peningkatan performa sangat signifikan. *Baseline* (Logistic Regression) hanya menghasilkan akurasi **47.3%**, yang berarti model ini hampir menebak secara acak (karena data kemungkinan tidak terpisah secara linear). Deep Learning meningkatkan akurasi hampir **+49%** dari baseline, menunjukkan bahwa pendekatan *machine learning* tingkat lanjut sangat diperlukan untuk dataset ini.

1. **Trade-off:**

* **Low Complexity, Low Performance (Logistic Regression):** Model Baseline sangat sederhana secara komputasi dan memiliki waktu *training* tercepat (hampir instan). Namun, kesederhanaan ini (hanya mampu memisahkan data secara linear) menjadi kelemahan fatal, menghasilkan akurasi yang sangat rendah (**47.3%**) dan bias yang tinggi (*underfitting*).
* **High Complexity, High Performance (Deep Learning):** Model Neural Network memiliki kompleksitas tertinggi karena melibatkan arsitektur *multilayer* (Input-Hidden-Output) dan proses *backpropagation* iteratif selama 50 epoch. Hal ini menyebabkan waktu *training* yang lebih lama dan kebutuhan memori yang lebih besar dibandingkan model lainnya.
* **Kesimpulan Trade-off:** Meskipun Deep Learning membutuhkan waktu komputasi lebih lama, *trade-off* ini **sangat sepadan (worth it)**. Kita "membayar" biaya komputasi tersebut untuk mendapatkan lonjakan akurasi masif sebesar **+49%** (dari 47.3% menjadi **96.3%**). Untuk kasus ini, kompleksitas diperlukan untuk menangkap pola non-linear data yang tidak bisa ditangani oleh model sederhana.

1. **Error Analysis:**

Logistic Regression memiliki kesalahan (error) terbesar karena *high bias* (underfitting). Random Forest (Akurasi 83.7%) sudah cukup baik namun masih melakukan kesalahan pada sekitar 16% data uji. Neural Network meminimalkan error ini secara signifikan, hanya salah memprediksi kurang dari 4% data uji.

1. **Overfitting/Underfitting:**

* **Logistic Regression:** Terindikasi *Underfitting* parah (skor rendah di training maupun testing).
* **Deep Learning:** Terindikasi **Good Fit**. Penggunaan teknik regularisasi berupa Dropout (0.3 dan 0.2) serta EarlyStopping pada kode berhasil mencegah model menghafal data (*overfitting*), sehingga performanya tetap tinggi saat diuji dengan data baru (test set).

## CONCLUSION

### Kesimpulan Utama

**Model Terbaik:** Deep Learning (Sequential Neural Network).

**Alasan:** Memiliki metrik evaluasi tertinggi di semua aspek (Accuracy, Precision, Recall, F1-Score > 0.96) dan stabil berkat penggunaan Dropout dan Early Stopping.

**Pencapaian Goals:** Proyek berhasil membangun model prediksi yang sangat akurat, jauh melampaui target baseline, membuktikan bahwa dataset ini dapat diprediksi dengan baik menggunakan pendekatan Deep Learning.

### Key Insights

**Insight dari Data:** Data memiliki karakteristik *non-linear* yang kuat, terbukti dari gagalnya model linear (Logistic Regression) dan suksesnya model non-linear (RF dan Neural Network).

**Insight dari Modeling:** Proses *preprocessing* (StandardScaler) sangat krusial, terutama untuk Neural Network, agar konvergensi gradient descent berjalan optimal. Arsitektur model bertingkat (128 -> 64 -> 32) efektif dalam mengekstraksi fitur.

### Kontribusi Proyek

**Manfaat praktis:** Model ini dapat digunakan untuk mengotomatisasi klasifikasi target pada dataset BEED\_Data dengan tingkat kepercayaan di atas 96%.

**Pembelajaran yang didapat:** Memahami pentingnya membandingkan berbagai algoritma (Linear vs Ensemble vs Deep Learning) dan menerapkan teknik *regularization* untuk menjaga generalisasi model.

## FUTURE WORK (Opsional)

**Saran pengembangan:**

* [ ✓] Mengumpulkan lebih banyak data
* [ ] Feature engineering lebih lanjut
* [ ] Mencoba arsitektur DL yang lebih kompleks
* [ ] Hyperparameter tuning lebih ekstensif
* [ ] Deployment (API/Web App)
* [ ] Optimization

## REPRODUCIBILITY (WAJIB)

### GitHub Repository

**Link Repository:** <https://github.com/nabilacarrissa/eeg-epilepsy-classification-beed>

**Checklist:**

* [ ✓] Notebook Jupyter/Colab dengan hasil running
* [ ✓] Script Python (jika ada)
* [✓ ] requirements.txt atau environment.yml
* [ ✓] README.md yang informatif
* [ ✓] Folder structure yang terorganisir
* [ ✓] .gitignore

### Environment & Dependencies

* Python Version: Python 3.12. ( Google Colab )
* Main Libraries:
* Numpy == 2.0.2
* pandas == 2.2.2
* scikit-learn == 1.6.1
* tensorflow == 2.19.0
* torch == 2.9.0+cpu