Perbandingan Metode Machine Learning dan Deep Learning untuk Klasifikasi Target pada Dataset Teaching Assistant Evaluation

Nama Mahasiswa	:	Elda Serlya Dwi Seviana
NIM	:	233307044
Program Studi	:	D-III Teknologi Informasi
Mata Kuliah	:	Data Science
Dosen Pengampu	:	Gus Nanang Syaifuddiin, S.Kom., M.Kom.
Tahun Akademik	:	2025
Link GitHub Repository	:	https://github.com/eldaserlya/DataSince_Project_UAS.git
Link Video Pembahasan	:	https://youtu.be/NU7q8FXPBPE

1. LEARNING OUTCOMES

1.1 Memahami masalah klasifikasi multikelas pada dataset TAE (memprediksi target kelas 1, 2, atau 3).

1.2 Melakukan eksplorasi data (EDA) untuk memahami distribusi target dan karakteristik fitur.

1.3 Melakukan data preparation yang sesuai untuk data tabular.

1.4 Mengembangkan dan membandingkan tiga model:
Model baseline: Logistic Regression
Model advanced ML: Random Forest
Model deep learning: Neural Network (MLP)

1.5 Mengevaluasi performa model menggunakan metrik klasifikasi yang relevan (minimal Accuracy dan F1-weighted) serta confusion matrix.

1.6 Menentukan "model terbaik" berdasarkan metrik utama.

1.7 Mendokumentasikan eksperimen secara sistematis dalam laporan (problem, data, preprocessing, modeling, evaluasi, kesimpulan).

1.8 Menerapkan reproducibility & software engineering: struktur repo rapi, ada README tambah requirements, kode bisa dijalankan ulang, dan tidak memakai hardcoded path.

2. PROJECT OVERVIEW

2.1 Latar Belakang

Dalam konteks pendidikan, memahami faktor yang memengaruhi kualitas pembelajaran sangat penting agar institusi dapat meningkatkan proses pengajaran. Dataset TAE (Teaching Assistant Evaluation) berisi data evaluasi kelas berdasarkan beberapa atribut seperti kemampuan berbahasa Inggris pengajar, identitas instructor, course, semester, dan ukuran kelas (class_size). Setiap data memiliki label target (kelas 1, 2, atau 3) yang merepresentasikan kategori hasil evaluasi.

Proyek ini dibuat untuk menyelesaikan masalah klasifikasi multikelas: memprediksi target (1/2/3) dari fitur-fitur yang tersedia. Dengan membandingkan beberapa pendekatan di antaranya : model baseline hingga deep learning dan melihat metode mana yang paling efektif untuk data tabular TAE.

3. BUSINESS UNDERSTANDING/PROBLEM UNDERSTANDING

3.1 Problem Statements

Berdasarkan dataset TAE, menyelesaikan masalah terkait klasifikasi multikelas untuk memprediksi target (1, 2, atau 3) dari fitur-fitur yang tersedia (english_speaker, instructor, course, semester, class_size). Tantangan utamanya adalah bagaimana membangun model yang dapat mengklasifikasikan target secara akurat pada data tabular dan membandingkan performa beberapa metode yang berbeda.

3.2 Goals

1. Melakukan EDA untuk memahami karakteristik data dan distribusi kelas target.
2. Melatih dan membandingkan tiga model :
   Logistic Regression sebagai baseline
   Random Forest sebagai model machine learning non-linear,
   Neural Network (MLP) sebagai deep learning untuk data tabular.
3. Mengevaluasi ketiga model menggunakan metrik yang konsisten (misalnya Accuracy dan F1-weighted) serta confusion matrix untuk melihat kesalahan klasifikasi per kelas.
4. Menentukan model terbaik berdasarkan metrik evaluasi utama dan menjelaskan alasan pemilihannya.

3.3 Solution Approach

1. Data Loading (relative path): memuat tae.data menggunakan path relatif agar kode tetap berjalan meskipun runtime Colab restart atau folder berubah.
2. EDA : membuat plot distribusi target, sebaran class_size terhadap target, serta distribusi target berdasarkan semester untuk menemukan pola awal.
3. Preprocessing & Split: melakukan pemisahan data train-test dan transformasi yang diperlukan (misalnya encoding/standarisasi) agar kompatibel dengan model ML dan MLP.
4. Modeling: melatih 3 model (Logistic Regression, RandomForest, Neural Network) menggunakan data training.
5. Evaluation: menghitung metrik evaluasi (Accuracy, F1-weighted, classification report) dan menampilkan/menyimpan confusion matrix untuk masing-masing model.
6. Model Selection: memilih model terbaik berdasarkan metrik utama (misalnya Accuracy, didukung F1-weighted) dan menyimpulkan hasilnya dalam bagian perbandingan model.

4. DATA UNDERSTANDING

4.1 Informasi Dataset

Dataset yang digunakan adalah Teaching Assistant Evaluation (TAE), yaitu dataset klasifikasi yang berisi data evaluasi kinerja Teaching Assistant (asisten pengajar) pada suatu kelas/perkuliahan. Setiap baris data merepresentasikan satu observasi kelas yang dievaluasi, dengan beberapa atribut terkait kondisi kelas dan pengajar, serta satu label target sebagai hasil evaluasi.

TAE termasuk dataset tabular untuk tugas multiclass classification, karena label target memiliki 3 kelas (1, 2, 3). Dataset ini umum digunakan sebagai studi kasus untuk membandingkan performa beberapa metode pembelajaran mesin dalam memprediksi kelas evaluasi berdasarkan fitur-fitur yang tersedia.

4.2 Deskripsi Fitur

- english_speaker: indikator apakah asisten pengajar adalah penutur bahasa Inggris (kategori).
- instructor: kode/ID instruktur (kategori).
- course: kode/ID mata kuliah (kategori).
- semester: kode semester (misalnya 1 = Summer, 2 = Regular).
- class_size: ukuran kelas (numerik).
- target: label kelas (1/2/3) sebagai hasil evaluasi yang diprediksi

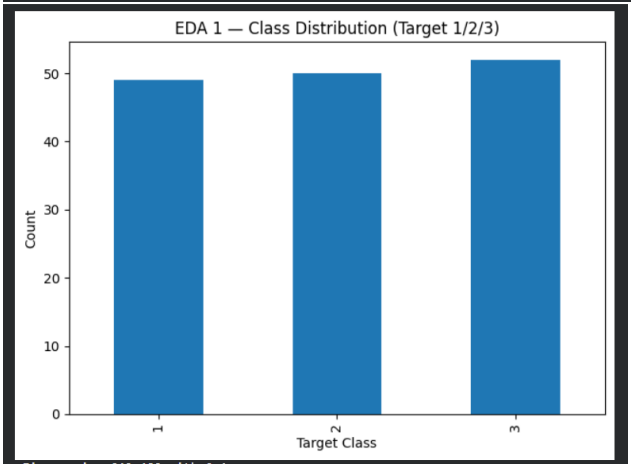
4.3 Kondisi Data

- Data berbentuk tabular dan seluruh kolom memiliki tipe kategori (encode angka) serta satu kolom numerik (class_size).
- Target merupakan multiclass classification (3 kelas) sehingga evaluasi model tidak cukup hanya accuracy, perlu juga melihat metrik seperti F1-weighted dan confusion matrix agar terlihat performa pada tiap kelas.
- Karena beberapa fitur adalah kategori dengan kode angka, preprocessing perlu memastikan representasi fitur sesuai untuk model (misalnya encoding/transformasi yang konsisten antara train dan test).
- Path data dibuat relative agar tidak hardcoded dan tetap aman saat runtime Colab restart.

4.4 Exploratory Data Analysis (EDA)

1. EDA 1 : Distribusi Target

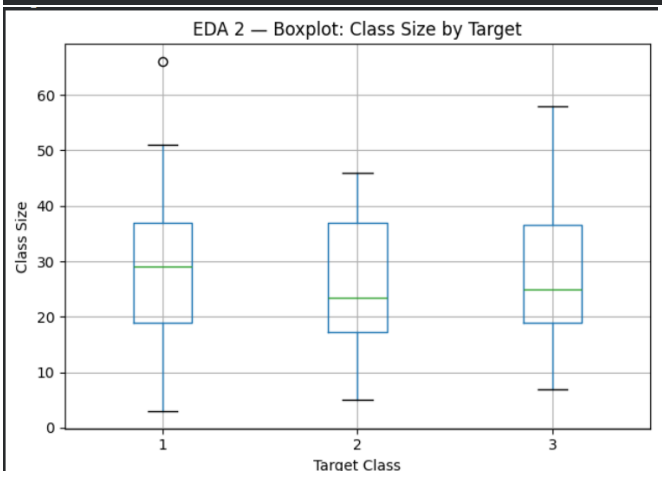
```
plt.figure()
df["target"].value_counts().sort_index().plot(kind="bar")
plt.title("EDA 1 — Class Distribution (Target 1/2/3)")
plt.xlabel("Target Class")
plt.ylabel("Count")
plt.tight_layout()
plt.savefig("images/eda_1_class_distribution_target.png", dpi=200)
plt.show()
```



Visualisasi ini menunjukkan jumlah data untuk masing-masing kelas target (1, 2, 3). Insight yang didapat: distribusi target bisa saja tidak seimbang, sehingga perlu evaluasi tambahan seperti F1-weighted untuk memastikan model tidak hanya bagus di kelas mayoritas.

2. EDA 2 (Sebaran Class_Size per target)

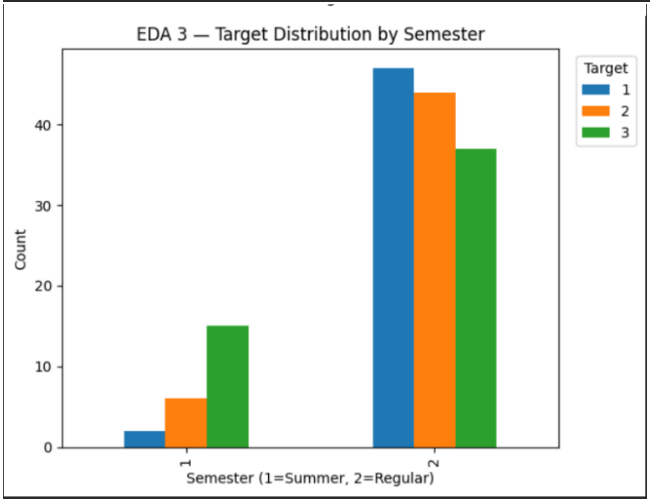
```
plt.figure()
df.boxplot(column="class_size", by="target")
plt.title("EDA 2 — Boxplot: Class Size by Target")
plt.suptitle("")
plt.xlabel("Target Class")
plt.ylabel("Class Size")
plt.tight_layout()
plt.savefig("images/eda_2_boxplot_class_size_by_target.png", dpi=200)
plt.show()
```



Boxplot digunakan untuk melihat perbedaan sebaran ukuran kelas pada tiap target. Insight yang didapat: jika median/penyebaran class_size antar target berbeda, maka class_size berpotensi menjadi fitur yang informatif untuk membedakan kelas target.

3. EDA 3 (Distribusi Target berdasarkan Semester)

```
ct = pd.crosstab(df["semester"], df["target"])
ct.plot(kind="bar")
plt.title("EDA 3 — Target Distribution by Semester")
plt.xlabel("Semester (1=Summer, 2=Regular)")
plt.ylabel("Count")
plt.legend(title="Target", bbox_to_anchor=(1.02, 1), loc="upper left")
plt.tight_layout()
plt.savefig("images/eda_3_target_by_semester.png", dpi=200)
plt.show()
```



Visualisasi ini membandingkan jumlah target untuk masing-masing nilai semester. Insight yang didapat: semester dapat memengaruhi distribusi target, sehingga semester berpotensi relevan dalam prediksi target.

5. DATA PREPARATION

5.1 Data Cleaning

Pada tahap ini dilakukan pengecekan dasar agar data siap diproses, yaitu memastikan kolom target tersedia dan tidak ada nilai yang mengganggu proses modeling. Dataset kemudian dipisahkan menjadi fitur (X) dan label (y) menggunakan:

X = df.drop(columns=["target"])

y = df["target"]

Sehingga proses selanjutnya fokus pada transformasi fitur input tanpa mencampur label target.

5.2 Feature Engineering

Karena dataset memiliki fitur kategorikal, dilakukan penentuan kelompok fitur:

- Fitur kategorikal: english_speaker, instructor, course, semester
- Fitur numerik: class_size

Fitur kategorikal tidak bisa langsung dipakai secara "apa adanya" untuk model tertentu (terutama deep learning), sehingga perlu diubah menjadi representasi numerik yang sesuai.

5.3 Data Transformation

Transformasi dilakukan menggunakan ColumnTransformer agar preprocessing rapi dan konsisten untuk kolom yang berbeda:

OneHotEncoder untuk fitur kategorikal

- Digunakan: OneHotEncoder(handle_unknown="ignore")
- Tujuan: mengubah kategori menjadi kolom-kolom biner (0/1). handle_unknown="ignore" memastikan saat ada kategori baru di data test, proses tetap berjalan tanpa error.

StandardScaler untuk fitur numerik class_size

- Digunakan: StandardScaler()
- Tujuan: menstandarkan nilai class_size agar memiliki skala yang lebih seimbang, membantu model (terutama neural network) belajar lebih stabil.

```
preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
        ("num", StandardScaler(), num_cols),
    ]
)
```

5.4 Data Splitting

Dataset dibagi menjadi data latih dan data uji menggunakan train_test_split dengan ketentuan:

test_size = 0.2 → 80% train, 20% test

random_state = 42 → agar hasil pembagian konsisten saat dijalankan ulang

stratify = y → agar proporsi kelas target (1/2/3) di train dan test tetap seimbang

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Train:", X_train.shape, "Test:", X_test.shape)
```

5.5 Data Balancing

Pada proyek ini tidak digunakan teknik balancing tambahan (seperti oversampling/undersampling). Alasannya karena pembagian data sudah menggunakan stratify, sehingga distribusi kelas di train dan test tetap terjaga. Selain itu, evaluasi model juga akan dilihat menggunakan metrik yang relevan untuk multiclass seperti Accuracy dan F1-weighted.

5.6 Ringkasan Data Preparation

- Memisahkan fitur dan label target (X dan y).
- Mengelompokkan kolom menjadi fitur kategorikal dan numerik.
- Membuat preprocessing dengan OneHotEncoder untuk kategori dan StandardScaler untuk numerik menggunakan ColumnTransformer.
- Membagi data menjadi train-test (80:20) menggunakan stratify untuk menjaga proporsi kelas.
- Tidak menerapkan balancing tambahan karena stratify sudah membantu menjaga keseimbangan distribusi target.

6. MODELING

6.1 Model 1 — Baseline Model

6.1.1 Deskripsi Model

Model baseline yang digunakan adalah **Logistic Regression** untuk tugas klasifikasi multikelas. Logistic Regression dipilih sebagai baseline karena model ini sederhana, cepat dilatih, dan sering digunakan sebagai pembanding awal untuk mengetahui performa minimal yang bisa dicapai sebelum memakai model yang lebih kompleks. Pada proyek ini, Logistic Regression dijalankan dalam bentuk Pipeline agar preprocessing (encoding + scaling) selalu konsisten antara data train dan test.

6.1.2 Hyperparameter

max_iter = 3000

Nilai ini digunakan untuk memastikan proses optimisasi konvergen (tidak berhenti terlalu cepat), terutama karena fitur hasil OneHotEncoder dapat menghasilkan dimensi yang lebih besar.

6.1.3 Implementasi

Implementasi dibuat dengan Pipeline yang berisi:

- prep: preprocessing menggunakan ColumnTransformer (OneHotEncoder untuk fitur kategorikal dan StandardScaler untuk class_size)clf:
- classifier Logistic Regression

```
model_lr = Pipeline([
    ("prep", preprocess),
    ("clf", LogisticRegression(max_iter=3000))
])
```

6.1.4 Hasil Awal

Evaluasi dilakukan menggunakan Accuracy, F1-macro, classification report, dan confusion matrix.

```
=== Logistic Regression ===
Accuracy: 0.5161290322580645
F1-macro: 0.5112022480443533
              precision    recall  f1-score   support

           1       0.44      0.40      0.42        10
           2       0.45      0.50      0.48        10
           3       0.64      0.64      0.64        11

    accuracy                           0.52        31
   macro avg       0.51      0.51      0.51        31
weighted avg       0.52      0.52      0.52        31

Confusion Matrix:
 [[4 4 2]
 [3 5 2]
 [2 2 7]]
```

6.2 Model 2 — ML / Advanced Model

6.2.1 Deskripsi Model

Model advanced yang digunakan adalah **Random Forest Classifier**, yaitu model ensemble berbasis banyak decision tree. Random Forest dipilih karena mampu menangkap pola non-linear dan interaksi antar fitur yang biasanya tidak dapat ditangkap model linear (seperti Logistic Regression). Selain itu, Random Forest relatif robust terhadap noise dan dapat memberikan performa yang baik pada data tabular.

6.2.2 Hyperparameter

- n_estimators = 400 : jumlah pohon (trees) yang dibangun dalam ensemble. Semakin banyak pohon biasanya membuat model lebih stabil (namun waktu training bertambah).
- random_state = 42 : memastikan hasil training konsisten saat dijalankan ulang.

6.2.3 Implementasi

Implementasi model dibuat menggunakan Pipeline yang berisi:
prep: preprocessing menggunakan ColumnTransformer (OneHotEncoder untuk kolom kategori dan StandardScaler untuk class_size)
clf: classifier RandomForestClassifier

```python
from sklearn.ensemble import RandomForestClassifier

model_rf = Pipeline([
    ("prep", preprocess),
    ("clf", RandomForestClassifier(n_estimators=400, random_state=42))
])

model_rf.fit(X_train, y_train)
pred_rf = model_rf.predict(X_test)
```

6.2.4 Hasil Model

Evaluasi dilakukan menggunakan Accuracy, F1-macro, classification report, dan confusion matrix.
Accuracy: (isi dari output)
F1-macro: (isi dari output)

```
=== Random Forest ===
Accuracy: 0.6774193548387096
F1-macro: 0.6761904761904761
              precision    recall  f1-score   support

           1       0.64      0.70      0.67        10
           2       0.60      0.60      0.60        10
           3       0.80      0.73      0.76        11

    accuracy                           0.68        31
   macro avg       0.68      0.68      0.68        31
weighted avg       0.68      0.68      0.68        31

Confusion Matrix:
 [[7 3 0]
 [2 6 2]
 [2 1 8]]
```

6.3 Model 3 — Deep Learning Model

6.3.1 Deskripsi Model

Model deep learning yang digunakan adalah **Neural Network tipe MLP (Multilayer Perceptron)** untuk klasifikasi multikelas. MLP dipilih karena mampu memodelkan hubungan non-linear pada data tabular melalui beberapa lapisan dense (fully-connected). Pada proyek ini, MLP digunakan sebagai pembanding terhadap model machine learning klasik (Logistic Regression dan Random Forest).

6.3.2 Arsitektur Model

Input layer dengan ukuran sesuai jumlah fitur hasil preprocessing (n_features)
Dense(64, ReLU) dengan menangkap pola non-linear awalDropout(0.2) untuk mengurangi risiko overfitting
Dense(32, ReLU) dengan mempelajari representasi lebih ringkas
Dense(3, Softmax) dengan output probabilitas untuk 3 kelas target

```
model_nn = keras.Sequential([
    layers.Input(shape=(n_features,)),
    layers.Dense(64, activation="relu"),
    layers.Dropout(0.2),
    layers.Dense(32, activation
    layers.Dense(n_classes, activation="softmax"),
])
```

Loading...

6.3.3 Input & Preprocessing Khusus

Data fitur ditransform menggunakan preprocess (OneHotEncoder + StandardScaler) agar semua fitur menjadi numerik:

- X_train_t = preprocess.fit_transform(X_train)
- X_test_t = preprocess.transform(X_test)

Hasil transform bisa berupa sparse matrix, sehingga dikonversi menjadi dense array menggunakan .toarray() jika diperlukan.

Label target awalnya 1/2/3, kemudian diubah menjadi 0/1/2 karena loss sparse_categorical_crossentropy mengharuskan label berupa indeks kelas mulai dari 0:

- y_train_nn = (y_train.values - 1)

- y_test_nn = (y_test.values - 1)

6.3.4 Hyperparameter

Optimizer: Adam
Loss: sparse_categorical_crossentropy
Epochs: 30
Batch size: 16
Validation split: 0.2
Dropout rate: 0.2
Hidden units: 64 (layer 1), 32 (layer 2)
Output classes: 3 (softmax)

6.3.5 Implementasi

Implementasi dimulai dengan melakukan transformasi fitur dan penyiapan label, lalu membangun dan compile model:

```
# transform fitur  (karena Keras butuh array numerik)
X_train_t = preprocess.fit_transform(X_train)
X_test_t  = preprocess.transform(X_test)

#  dense array
X_train_t = X_train_t.toarray() if hasattr(X_train_t, "toarray") else X_train_t
X_test_t  = X_test_t.toarray()  if hasattr(X_test_t, "toarray")  else X_test_t

# label target 1/2/3 -> 0/1/2
y_train_nn = (y_train.values - 1)
y_test_nn  = (y_test.values - 1)

n_features = X_train_t.shape[1]
n_classes = 3

model_nn = keras.Sequential([
    layers.Input(shape=(n_features,)),
    layers.Dense(64, activation="relu"),
    layers.Dropout(0.2),
    layers.Dense(32, activation="relu"),
    layers.Dense(n_classes, activation="softmax"),
])

model_nn.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

6.3.6 Training Process

Proses training dilakukan dengan parameter:

- epochs=30, batch_size=16
- validation_split=0.2 untuk memonitor performa pada data validasi

Selama training, dibuat plot loss dan accuracy untuk train vs validation yang disimpan ke folder images/:

- images/nn_loss.png
- images/nn_accuracy.png

digunakan untuk melihat apakah model mengalami overfitting.

```
history = model_nn.fit(
    X_train_t, y_train_nn,
    validation_split=0.2,
    epochs=30,
    batch_size=16,
    verbose=1
)
```

6.3.7 Model Summary

Model summary digunakan untuk menunjukkan jumlah layer, output shape, dan jumlah parameter, model memiliki dua hidden layer (64 dan 32 neuron) dengan dropout 0.2 dan output softmax 3 kelas.

```
=== Model Summary (Neural Network / MLP) ===
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 64)	3,456
dropout_1 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 32)	2,080
dense_8 (Dense)	(None, 3)	99

Model Neural Network/MLP menerima 53 fitur input (hasil transformasi preprocessing), lalu diproses melalui Dense 64 (ReLU) untuk mempelajari representasi awal pola data. Setelah itu terdapat Dropout (tanpa parameter) yang berfungsi mengurangi overfitting dengan menonaktifkan sebagian neuron saat training. Layer berikutnya Dense 32 (ReLU) melakukan penyaringan/penyederhanaan representasi sebelum masuk ke layer output. Terakhir, Dense 3 (Softmax) menghasilkan probabilitas untuk 3 kelas target (1, 2, 3). Total parameter yang dipelajari model adalah 5.635 (3.456 + 2.080 + 99), yang menunjukkan kompleksitas model relatif ringan namun tetap mampu menangkap pola dari fitur hasil encoding dan scaling.

7. EVALUATION

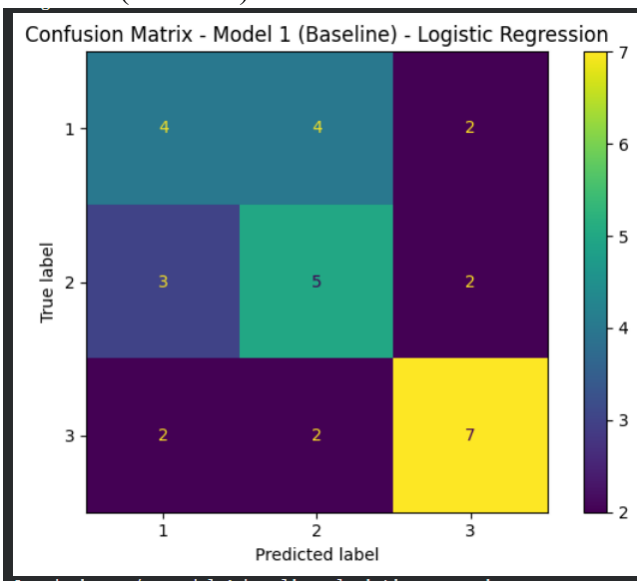
7.1 Metrik Evaluasi

```
                  Model  Accuracy  F1-macro
1         Random Forest  0.645161  0.644444
0   Logistic Regression  0.516129  0.511202
2  Neural Network (MLP)  0.483871  0.485507
Saved: models/classification_reports.txt
```

Berdasarkan evaluasi pada data uji menggunakan metrik Accuracy dan F1-macro, diperoleh bahwa Random Forest memberikan performa terbaik dengan Accuracy 0,6452 dan F1-macro 0,6444. Logistic Regression berada pada posisi kedua dengan Accuracy 0,5161 dan F1-macro 0,5112, sedangkan Neural Network (MLP) menunjukkan performa terendah dengan Accuracy 0,4839 dan F1-macro 0,4855. Dengan demikian, model terbaik untuk klasifikasi target pada dataset TAE adalah Random Forest karena menghasilkan nilai tertinggi pada kedua metrik evaluasi tersebut.

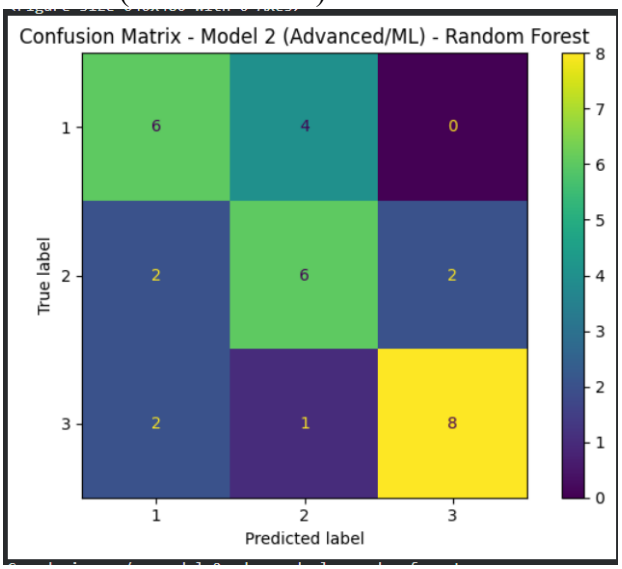
7.2 Hasil Evaluasi Model

7.2.1 Model 1 (Baseline)



onfusion matrix Logistic Regression (Model 1/Baseline) menunjukkan prediksi yang benar ada di bagian diagonal: kelas 1 benar 4 data, kelas 2 benar 5 data, dan kelas 3 benar 7 data. Kesalahan paling sering terjadi karena kelas 1 dan kelas 2 masih sering tertukar, jadi model sulit membedakan dua kelas ini. Masih ada juga beberapa data kelas 1 dan kelas 2 yang diprediksi sebagai kelas 3, dan sebagian data kelas 3 diprediksi sebagai kelas 1 atau kelas 2. Secara umum, model paling baik mengenali kelas 3, tetapi pemisahan antara kelas 1 dan kelas 2 masih lemah sehingga akurasi keseluruhan belum tinggi.

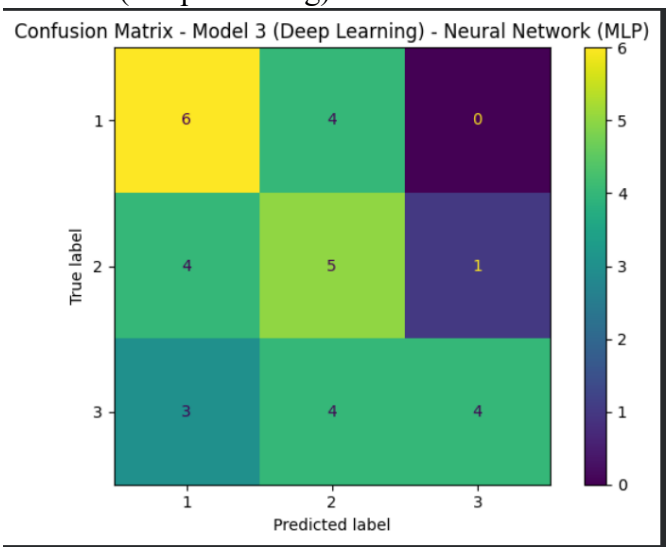
7.2.2 Model 2 (Advanced/ML)



Confusion matrix Random Forest (Model 2/Advanced) menunjukkan prediksi benar paling banyak ada di diagonal: kelas 1 benar 6 data, kelas 2 benar 6 data, dan kelas 3 benar 8 data. Untuk kelas 1, tidak ada data yang salah diprediksi menjadi kelas 3, tetapi masih cukup sering tertukar ke kelas

2 (4 data). Untuk kelas 2, kesalahan terbagi ke kelas 1 (2 data) dan kelas 3 (2 data), jadi kelas 2 masih menjadi kelas yang paling "campur" dengan kelas lain. Untuk kelas 3, sebagian kecil masih salah ke kelas 1 (2 data) dan kelas 2 (1 data), tetapi mayoritas tetap terklasifikasi benar. Secara umum, Random Forest lebih stabil dibanding baseline karena jumlah prediksi benar lebih tinggi dan kesalahan antar kelas lebih sedikit, terutama dalam mengenali kelas 3.

7.2.3 Model 3 (Deep Learning)



Confusion matrix Neural Network (MLP) pada Model 3 menunjukkan prediksi benar masih ada di diagonal, tetapi kesalahan antar kelas cukup besar. Untuk kelas 1, model sudah mengenali dengan baik (6 data benar), namun masih sering tertukar menjadi kelas 2 (4 data) dan tidak ada yang jatuh ke kelas 3. Untuk kelas 2, prediksi benar hanya 5 data, sedangkan 4 data salah menjadi kelas 1 dan 1 data salah menjadi kelas 3, sehingga kelas 2 masih belum stabil. Kinerja terlemah ada pada kelas 3: hanya 4 data yang benar terprediksi sebagai kelas 3, sementara 3 data salah menjadi kelas 1 dan 4 data salah menjadi kelas 2. Ini menunjukkan model deep learning belum mampu membedakan kelas 3 dengan kelas lain secara konsisten, sehingga secara keseluruhan performanya masih kalah dibanding Random Forest pada dataset ini.

7.3 Perbandingan Ketiga Model

Perbandingan tiga model dilakukan menggunakan metrik Accuracy dan F1-macro pada data uji. Hasil menunjukkan Random Forest memberikan performa terbaik dengan Accuracy 0,645 dan F1-macro 0,644, sehingga paling stabil dalam memprediksi ketiga kelas target. Logistic Regression berada di urutan kedua dengan Accuracy 0,516 dan F1-macro 0,511, namun masih cukup sering tertukar antar kelas. Neural Network (MLP) menghasilkan performa terendah dengan Accuracy 0,484 dan F1-macro 0,486, terutama karena kesalahan prediksi yang cukup besar pada kelas 3 dan kelas 2. Berdasarkan metrik utama tersebut, model terbaik adalah Random Forest.

7.4 Analisis Hasil

Berdasarkan hasil evaluasi, Random Forest menunjukkan kinerja terbaik dengan akurasi 0.645 dan F1-macro 0.644, sehingga dipilih sebagai model terbaik. Hasil confusion matrix juga memperlihatkan bahwa Random Forest lebih konsisten dalam mengklasifikasikan data, terutama pada kelas 3 yang memiliki jumlah prediksi benar paling tinggi dibanding model lainnya. Logistic Regression menghasilkan performa menengah (akurasi 0.516; F1-macro 0.511). Kesalahan klasifikasi masih cukup sering terjadi, terutama pada pemisahan antara kelas 1 dan kelas 2, sehingga kualitas prediksi belum sebaik Random Forest. Sementara itu, Neural Network (MLP) memperoleh performa terendah (akurasi 0.484; F1-macro 0.486). Confusion matrix menunjukkan model ini lebih sering melakukan kesalahan prediksi, khususnya pada kelas 3 yang banyak diprediksi menjadi kelas lain.

8. CONCLUSION

8.1 Kesimpulan Utama

Berdasarkan evaluasi pada data uji, Random Forest menjadi model terbaik karena menghasilkan Accuracy 0,645 dan F1-macro 0,644, lebih tinggi dibanding Logistic Regression (Accuracy 0,516; F1-macro 0,511) dan Neural Network/MLP (Accuracy 0,484; F1-macro 0,486).

Secara keseluruhan, tujuan proyek untuk membangun 3 model, melakukan evaluasi, dan menentukan model terbaik telah tercapai sesuai learning outcomes.

8.2 Key Insights

1. Model terbaik untuk dataset TAE pada eksperimen ini adalah Random Forest, karena performanya paling stabil dan nilai metriknya paling tinggi.
2. Kesalahan prediksi paling sering terjadi pada kelas yang polanya mirip sehingga pemisahan antar kelas masih menjadi tantangan utama.
3. Perbandingan menunjukkan bahwa model yang lebih kompleks (MLP) tidak otomatis lebih baik pada data tabular ini, karena hasilnya masih berada di bawah Random Forest.

8.3 Kontribusi Proyek

Proyek ini memberikan eksperimen perbandingan untuk klasifikasi multikelas pada dataset TAE, mulai dari EDA, preprocessing tabular (encoding dan scaling), pelatihan tiga pendekatan model (baseline/Logistic Regression, advanced ML/Random Forest, deep learning/NN MLP), hingga evaluasi dan pemilihan model terbaik. Selain itu, proyek menekankan aspek reproducibility melalui struktur pengerjaan yang terdokumentasi dan pemakaian path relatif agar kode dapat dijalankan ulang secara konsisten

9. FUTURE WORK

Untuk pengembangan selanjutnya, beberapa perbaikan yang dapat dilakukan adalah:

1. Tuning hyperparameter pada Random Forest (misalnya max_depth, min_samples_split, min_samples_leaf) untuk melihat apakah performa masih bisa ditingkatkan.

2. Mencoba model tambahan untuk data tabular seperti Gradient Boosting / XGBoost / LightGBM sebagai pembanding yang sering lebih kuat dari Random Forest.
3. Melakukan tuning Neural Network (MLP) (jumlah layer/neuron, learning rate, batch size, dropout) dan menambahkan early stopping agar training lebih stabil dan tidak overfitting.
4. Menggunakan cross-validation agar evaluasi tidak bergantung pada satu pembagian train-test saja.
5. Menganalisis lebih detail kesalahan pada confusion matrix (terutama kelas yang sering tertukar) dan mencoba feature engineering sederhana untuk membantu pemisahan kelas.

10. REPRODUCIBILITY

10.1 GitHub Repository

Seluruh kode, hasil eksperimen, dan dokumentasi proyek disimpan dalam repository GitHub yang bersifat public. Repository dibuat terstruktur agar mudah dijalankan ulang, dengan pemisahan folder untuk data, notebook, source code, model/output, serta gambar visualisasi (EDA dan evaluasi). Selain itu, proyek menggunakan relative path (bukan hardcoded path), sehingga kode tetap dapat berjalan meskipun lokasi folder berubah atau runtime di-refresh.
https://github.com/eldaserlya/DataSince_Project_UAS.git

10.2 Environment & Dependencies

Dependensi dicantumkan pada file ***requirements.txt*** agar dapat dijalankan ulang pada environment lain dengan memasang dependensi yang sama, lalu menjalankan notebook/script sesuai instruksi pada README.