ANALISIS KLASIFIKASI GENDER BERDASARKAN NAMA MENGGUNAKAN TF-IDF, RANDOM FOREST DAN MULTILAYER PERCEPTRON

Disusun Oleh :

Nama	: Ardhiyonda Wahyu Putra Viardhana
NIM	: 234311032
Program Studi	: Teknologi Rekayasa Perangkat Lunak
Mata Kuliah	: Data Science

Dosen Pengampu:

Gus Nanang Syaifuddiin, S.Kom., M.Kom.

2025 / SEMESTER 5

1. Learning Outcomes

Pada proyek ini, mahasiswa diharapkan mampu:

1. Memahami permasalahan klasifikasi gender berbasis data teks nama.
2. Melakukan proses data preparation dan ekstrasi fitur teks menggunakan metode TF-IDF.
3. Mengembangkan dan membandingkan model baseline, machine learning lanjutan, dan deep learning.
4. Mengevaluasi performa model klasifikasi menggunakan metrik evaluasi yang relevan.
5. Menyusun proyek data science secara sistematis dan terdokumentasi.

2. Project Overview

2.1 Latar Belakang

Klasifikasi gender berdasarkan nama merupakan salah satu permasalahan klasifikasi teks yang banyak digunakan sebagai studi kasus dalam bidang Data Science dan *Natural Language Processing* (NLP). Dalam berbagai sistem informasi, data gender sering kali tidak tersedia secara eksplisit, baik karena keterbatasan data maupun alasan privasi. Oleh karena itu, inferensi gender berdasarkan nama menjadi solusi alternatif yang banyak dikaji dalam penelitian sebelumnya (Krstovski et al., 2023; Rego et al., 2021).

Nama sebagai data teks memiliki karakteristik unik karena tersusun atas rangkaian karakter yang membentuk pola tertentu, seperti prefiks, sufiks, maupun kombinasi huruf khas yang sering berkorelasi dengan gender. Untuk dapat diproses oleh algoritma *Machine Learning*, data teks tersebut perlu direpresentasikan dalam bentuk numerik melalui teknik ekstraksi fitur yang sesuai. Salah satu metode yang umum digunakan adalah *Term Frequency–Inverse Document Frequency* (TF-IDF), yang mampu merepresentasikan tingkat kepentingan karakter atau token dalam sebuah teks dan telah terbukti efektif dalam berbagai tugas klasifikasi teks, termasuk prediksi gender berbasis nama (Rego et al., 2021; Septiandri, 2017).

Selain pemilihan teknik representasi fitur, pemilihan model klasifikasi juga berperan penting dalam menentukan performa sistem. Model *Machine Learning* seperti Random Forest mampu menangkap hubungan non-linear antar fitur dan relatif

robust terhadap *overfitting*, sehingga sering digunakan sebagai model pembanding (*baseline* maupun *advanced model*) dalam penelitian klasifikasi (Pradana et al., 2024). Di sisi lain, model *Deep Learning* seperti *Multilayer Perceptron* (MLP) memiliki kemampuan untuk mempelajari representasi fitur yang lebih kompleks melalui lapisan jaringan saraf tiruan, sehingga berpotensi menghasilkan performa yang lebih baik pada data teks berbasis karakter (Mas'ud et al., 2025; Septiandri, 2017)

Berdasarkan latar belakang tersebut, proyek ini bertujuan untuk melakukan analisis dan perbandingan performa antara pendekatan *Machine Learning* dan *Deep Learning* dalam melakukan klasifikasi gender berdasarkan nama. Metode TF-IDF digunakan sebagai teknik ekstraksi fitur, sementara Random Forest dan *Multilayer Perceptron* dipilih sebagai model utama untuk mengevaluasi efektivitas masing-masing pendekatan dalam menyelesaikan permasalahan klasifikasi gender berbasis teks nama.

3. Business Understanding / Problem Understanding

3.1 Problem Statements

Berdasarkan latar belakang dan karakteristik dataset yang digunakan, permasalahan yang diangkat dalam proyek ini dapat dirumuskan sebagai berikut:

1. Data gender pada dataset tidak selalu tersedia secara eksplisit dalam sistem informasi, sehingga diperlukan metode otomatis untuk mengklasifikasikan gender berdasarkan nama.
2. Nama sebagai data teks berbasis karakter memerlukan teknik representasi fitur yang tepat agar pola-pola linguistik pada nama dapat dipahami oleh algoritma Machine Learning.
3. Performa model klasifikasi gender dapat bervariasi tergantung pada pendekatan yang digunakan, baik model *Machine Learning* konvensional maupun model *Deep Learning*.
4. Diperlukan analisis perbandingan untuk mengetahui model mana yang memberikan performa terbaik dalam melakukan klasifikasi gender berdasarkan nama dengan tingkat akurasi dan generalisasi yang baik.

3.2 Goals

Tujuan dari proyek ini dirumuskan secara spesifik dan terukur sebagai berikut:

1. Membangun sistem klasifikasi gender berdasarkan nama menggunakan pendekatan *Machine Learning* dan *Deep Learning*.
2. Menerapkan teknik TF-IDF sebagai metode ekstraksi fitur untuk mengubah data nama menjadi representasi numerik.
3. Mengembangkan dan mengevaluasi tiga model klasifikasi yang terdiri dari model baseline, model *Machine Learning* lanjutan, dan model *Deep Learning*.
4. Membandingkan performa masing-masing model berdasarkan metrik evaluasi yang relevan untuk tugas klasifikasi, seperti *accuracy*, *precision*, *recall*, dan *F1-score*.
5. Menentukan model terbaik untuk klasifikasi gender berdasarkan nama berdasarkan hasil evaluasi dan analisis performa.

3.3 Solution Approach

Model 1 – Baseline

Model baseline yang digunakan dalam proyek ini adalah Logistic Regression. Model ini dipilih karena memiliki arsitektur yang sederhana, cepat dalam proses training, serta sering digunakan sebagai pembanding awal dalam tugas klasifikasi teks.

Alasan pemilihan model baseline:

- Mudah diimplementasikan dan diinterpretasikan
- Cocok untuk data hasil TF-IDF
- Memberikan gambaran performa awal sistem sebelum menggunakan model yang lebih kompleks.

Model 2 – Machine Learning

Model *Machine Learning* lanjutan yang digunakan adalah Random Forest Classifier. Random Forest merupakan algoritma *ensemble* berbasis decision tree yang mampu menangkap hubungan non-linear antar fitur dan relatif tahan terhadap *overfitting*.

Alasan pemilihan *Random Forest*:

- Mampu menangani data berdimensi tinggi seperti hasil TF-IDF
- Lebih robust dibandingkan model linear
- Memberikan performa yang lebih stabil pada data klasifikasi

Model 3 – Deep Learning

Model *Deep Learning* yang digunakan adalah Multilayer Perceptron (MLP). MLP merupakan jaringan saraf tiruan *feedforward* yang terdiri dari beberapa *hidden layer* dan ampu mempelajari representasi fitur yang lebih kompleks.

Alasan pemilihan MPL:

- Cocok untuk data tabular hasil dari transformasi TF-IDF
- Memiliki kemampuan belajar non-linear yang lebih baik
- Memenuhi requirement penggunaan model *deep learning* sesuai ketentuan proyek.

4. Data Understanding

4.1 Informasi Dataset

Sumber Dataset :

https://archive.ics.uci.edu/dataset/591/gender+by+name

Deskripsi Dataset:

- Jumlah baris (rows): 147.269
- Jumlah kolom (columns/features): 4
- Tipe data: Tabular (text-based)
- Ukuran dataset: 3.6 MB
- Format file: CSV

4.2 Deskripsi Fitur

Dataset Terdiri dari beberapa kolom utama yang dijelaskan pada tabel berikut:

Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
Name	String	Nama individu yang digunakan sebagai input model	“alex”,”maria”,”john”
Gender	Categorical	Label gender dari nama	“M”,”F”
Count	Integer	Jumlah kemunculan nama dalam dataset	125, 340
Probability	Float	Probabilitas nama terkait dengan gender tertentu	0.85, 0.12

4.3 Kondisi Data

Berdasarkan eksplorasi awal terhadap dataset, kondisi data dapat dijelaskan sebagai berikut:

- Missing Values

  Tidak terdapat missing values pada dataset. Seluruh kolom (*Name, Gender, Count,* dan *Probability*) memiliki nilai kosong sebesar 0%**.**

```
MISSING VALUES:
Name           0
Gender         0
Count          0
Probability    0
dtype: int64
```

- Duplicate

  Terdapat data duplikat berdasarkan kolom *Name*, dengan jumlah 13359 baris yang memiliki nama yang sama. Duplikasi ini tetap dipertahankan karena merepresentasikan variasi statistik distribusi nama dalam populasi.

```
DUPLICATE NAMES:
13359
```

- Outliers

Outliers tidak relevan pada fitur berbasis teks (*Name*). Namun, pada fitur numerik *Count* dan *Probability* terdapat nilai ekstrem yang membentuk pola distribusi *long-tail*, di mana sebagian kecil nama memiliki frekuensi kemunculan yang sangat tinggi dibandingkan nama lainnya.

- Imbalanced Data

  Terdapat ketidakseimbangan kelas (*class imbalance*) dengan rasio kelas **Female : Male = 60,95% : 39,05%**. Ketidakseimbangan ini tergolong moderat dan perlu diperhatikan pada tahap pemodelan.

```
GENDER DISTRIBUTION:
Gender
F    0.609422
M    0.390578
```

- Noise

  Terdapat noise berupa nama dengan panjang sangat pendek (dua karakter), yang berpotensi mempengaruhi kualitas representasi fitur pada proses ekstraksi fitur berbasis teks.

- Data Quality Issues

  Dataset telah berada dalam format huruf kecil (lowercase) secara konsisten dan tidak mengandung karakter non-alfabet. Tidak ditemukan permasalahan kualitas data lain yang signifikan pada tahap eksplorasi awal.

5. Data Preparation

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

5.1 Data Cleaning

- Handling Missing Values
  - Tidak ditemukan missing values pada seluruh kolom (*Name, Gender, Count, Probability*).
  - Strategi: tidak dilakukan imputasi data.
  - Alasan: Dataset sudah lengkap sehingga imputasi tidak diperlukan dan berpotensi menambah noise jika dipaksakan.
- Removing Duplicates
  - Ditemukan 29.389 baris dengan nama yang sama.

    - Strategi : Duplikasi tidak dihapus, karena nama yang sama dapat muncul dengan gender berbeda dan mewakili distribusi nyata dalam populasi
    - Alasan:
      Duplikasi dipertahankan karena mengandung informasi penting untuk klasifikasi.
- Handling Outliers
    - Outliers ditemukan pada fitur numerik `Count` dan `Probability`.
    - Outliers **tidak dihapus**, karena nilai ekstrem merepresentasikan nama yang sangat populer dan outliers merupakan informasi valid, bukan kesalahan input.
- Data Type Conversion
    - Kolom *Name* dipastikan bertipe string
    - Kolom *Gender* dikonversi ke label numerik pada tahap transformasi
    - Kolom *Count* dan *Probability* bertipe numerik (*int dan float*).

5.2 Feature Engineering

Pada tahap feature engineering, proyek ini berfokus pada ekstraksi fitur dari data teks berupa nama. Mengingat data berupa rangkaian karakter, digunakan teknik character-level Term Frequency–Inverse Document Frequency (TF-IDF) untuk merepresentasikan pola huruf dalam setiap nama ke dalam bentuk numerik.

Pendekatan character-level dipilih karena nama tidak memiliki struktur kata yang kompleks, sehingga pola kombinasi huruf (n-gram) lebih informatif dibandingkan word-level representation. Teknik TF-IDF juga membantu menekankan karakter atau kombinasi karakter yang bersifat diskriminatif terhadap kelas gender.

5.3 Data Transformation

5.3.1 Data Tabular

A. Encoding

Transformasi data tabular dilakukan pada variabel target `Gender`. Label kategorikal diubah menjadi bentuk numerik menggunakan **Label Encoding**, dengan pemetaan sebagai berikut:

- `M` → 0
- `F` → 1

Transformasi ini diperlukan karena algoritma Machine Learning dan Deep Learning hanya dapat memproses data numerik.

Teknik encoding lain seperti **One-Hot Encoding** dan **Ordinal Encoding** tidak diterapkan karena tidak terdapat fitur kategorikal tabular lain selain label target.

5.3.2 Data Teks

A. **Tokenization**

Proses tokenisasi dilakukan secara implisit melalui penggunaan **TF-IDF berbasis karakter (character-level n-gram)**. Pendekatan ini memecah nama menjadi potongan karakter berurutan (n-gram) sehingga mampu menangkap pola morfologis pada nama, seperti akhiran atau kombinasi huruf tertentu yang sering muncul pada gender tertentu.

Pendekatan character-level dipilih karena nama merupakan teks pendek yang tidak memiliki struktur kalimat, sehingga word-level tokenization kurang efektif.

B. **Lowercasing**

Seluruh data nama diubah menjadi huruf kecil (lowercase). Tujuan dari proses ini adalah untuk menghindari perbedaan representasi fitur akibat variasi penggunaan huruf besar dan kecil, sehingga meningkatkan konsistensi data.

5.4 Data Splitting

Dataset dibagi menjadi dua bagian utama, yaitu **data latih (training set)** dan **data uji (test set)**. Pembagian data dilakukan dengan proporsi sebagai berikut:

- **Training set**: 80% dari total data
- **Test set**: 20% dari total data

Proses pembagian data menggunakan fungsi train_test_split dari library *scikit-learn* dengan parameter **stratify=y** untuk memastikan distribusi kelas gender (*male* dan *female*) tetap seimbang pada data latih dan data uji.

Selain itu, digunakan **random_state = 42** untuk menjaga konsistensi hasil pembagian data sehingga eksperimen dapat direproduksi.

Tidak digunakan validation set terpisah, karena:

- Dataset sudah dibagi menjadi train–test secara eksplisit
- Validasi model deep learning dilakukan menggunakan validation_split saat proses training

5.5 Ringkasan Data Preparation

Pada tahap *data preparation*, dilakukan beberapa langkah utama untuk memastikan data siap digunakan dalam proses pemodelan. Tahap pertama adalah *data cleaning*, di mana seluruh data nama dinormalisasi dengan mengubah huruf menjadi *lowercase* guna menghindari perbedaan representasi teks akibat variasi penulisan huruf besar dan kecil. Selain itu, dilakukan pengecekan terhadap *missing values* dan data duplikat untuk memastikan kualitas data, meskipun tidak ditemukan nilai kosong pada dataset dan data duplikat tetap dipertahankan karena masih merepresentasikan variasi distribusi nama dalam populasi.

Selanjutnya, dilakukan *feature engineering* dengan mengekstraksi fitur dari data teks menggunakan metode **Term Frequency–Inverse Document Frequency (TF-IDF)** berbasis karakter *n-gram*. Pendekatan ini dipilih karena mampu menangkap pola karakter dalam nama yang relevan untuk tugas klasifikasi gender, sekaligus mengubah data teks menjadi representasi numerik yang dapat diproses oleh algoritma machine learning dan deep learning.

Pada tahap *data transformation*, label gender yang bersifat kategorikal dikonversi menjadi bentuk numerik menggunakan teknik *label encoding*, sehingga dapat digunakan sebagai target oleh model. Selain itu, transformasi teks seperti *tokenization* dan normalisasi dilakukan secara implisit melalui mekanisme TF-IDF, tanpa menerapkan teknik tambahan seperti *stemming* atau *stopword removal* karena karakteristik data berupa nama tunggal.

Tahap berikutnya adalah *data splitting*, di mana dataset dibagi menjadi data latih dan data uji dengan perbandingan 80:20 menggunakan metode *stratified split*.

Pendekatan ini memastikan distribusi kelas gender tetap proporsional pada kedua subset data. Parameter *random state* digunakan untuk menjaga konsistensi hasil pembagian data sehingga eksperimen dapat direproduksi.

6. Pemodelan

6.1 Model 1 – Baseline Model (Logistic Regression)

6.1.1 Deskripsi Model

Nama Model: Logistic Regression

Teori Singkat:
Logistic Regression merupakan algoritma klasifikasi yang memodelkan hubungan antara fitur input dan probabilitas kelas menggunakan fungsi logistik (sigmoid). Model ini menghitung kombinasi linear dari seluruh fitur, kemudian memetakan hasilnya ke dalam probabilitas untuk menentukan kelas target.

Alasan Pemilihan:
Model Logistic Regression dipilih sebagai *baseline model* karena:

- Sederhana dan efisien
- Cocok untuk data hasil transformasi TF-IDF yang berdimensi tinggi
- Memberikan pembanding awal sebelum menggunakan model machine learning dan deep learning

Keunggulan:

- Training cepat
- Stabil pada data sparse seperti TF-IDF
- Mudah diimplementasikan tanpa banyak hyperparameter

Kelemahan:

- Kurang mampu menangkap hubungan non-linear yang kompleks
- Performa sangat bergantung pada kualitas feature extraction

6.1.2 Hyperparameter

Parameter yang digunakan:

- `max_iter=1000`

6.1.3 Implementasi

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

baseline_model = LogisticRegression(max_iter=1000)
baseline_model.fit(X_train, y_train)

y_pred_baseline = baseline_model.predict(X_test)
baseline_acc = accuracy_score(y_test, y_pred_baseline)
```

6.1.4 Hasil Model

Model Logistic Regression menghasilkan nilai akurasi sebesar **baseline_acc** pada data uji. Hasil ini menunjukkan bahwa model baseline mampu mengklasifikasikan data dengan baik dan menjadi acuan perbandingan untuk model machine learning dan deep learning pada tahap selanjutnya.

6.2 Model 2 – ML / Advanced Model

6.2.1 Deskripsi Model

Nama Model: Random Forest Classifier

Teori Singkat:
Random Forest merupakan algoritma ensemble learning berbasis *decision tree* yang bekerja dengan membangun banyak pohon keputusan (trees) secara acak pada subset data dan fitur. Setiap pohon melakukan prediksi secara independen, kemudian hasil akhirnya ditentukan berdasarkan *majority voting*. Pendekatan ini membuat Random Forest lebih stabil dan mampu mengurangi overfitting dibandingkan satu decision tree tunggal.

Alasan Pemilihan:
Random Forest dipilih sebagai model machine learning lanjutan karena:

- Mampu menangkap hubungan non-linear pada data

- Robust terhadap noise
- Bekerja dengan baik pada data berdimensi tinggi seperti TF-IDF
- Mendukung pengaturan *class weight* untuk menangani data tidak seimbang (imbalanced)

Keunggulan:

- Tahan terhadap overfitting dibanding decision tree
- Tidak sensitif terhadap skala fitur
- Mampu menangani fitur dalam jumlah besar
- Mendukung estimasi performa internal melalui *Out-of-Bag (OOB) score*

Kelemahan:

- Waktu training relatif lebih lama dibanding Logistic Regression
- Model bersifat *black box* sehingga interpretasi lebih sulit
- Membutuhkan memori lebih besar

6.2.2 Hyperparameter

Parameter yang digunakan pada model Random Forest adalah sebagai berikut:

- `n_estimators = 200`
  Jumlah pohon diperbesar untuk meningkatkan stabilitas model.
- `max_depth = 15`
  Membatasi kedalaman pohon untuk mencegah overfitting.
- `min_samples_split = 20`
  Node hanya akan di-split jika memiliki minimal 20 sampel.
- `min_samples_leaf = 5`
  Setiap daun memiliki minimal 5 sampel untuk menghindari model terlalu spesifik.
- `max_features = 'sqrt'`
  Jumlah fitur yang dipertimbangkan pada setiap split mengikuti standar Random Forest.

- `class_weight = 'balanced'`
  Digunakan karena distribusi kelas gender (M dan F) tidak seimbang.
- `oob_score = True`
  Mengaktifkan *Out-of-Bag validation* sebagai estimasi performa internal.
- `n_jobs = -1`
  Menggunakan seluruh core CPU untuk mempercepat training.
- `random_state = 42`
  Digunakan untuk menjaga reprodusibilitas hasil.

Hyperparameter Tuning:

Hyperparameter ditentukan secara manual (*manual tuning*) berdasarkan praktik umum untuk mencegah overfitting dan meningkatkan generalisasi model. Tidak dilakukan Grid Search atau Random Search.

6.2.3 Implementasi Model

```
rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=15,
    min_samples_split=20,
    min_samples_leaf=5,
    max_features='sqrt',
    class_weight='balanced',
    oob_score=True,
    n_jobs=-1,
    random_state=42
)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

rf_acc = accuracy_score(y_test, y_pred_rf)
print("Accuracy:", rf_acc)
print("OOB Score:", rf_model.oob_score_)
print(classification_report(y_test, y_pred_rf))
```

6.2.4 Hasil Model

Model Random Forest menghasilkan:

- **Accuracy** pada data uji sebesar **rf_acc**
- **OOB Score** sebagai estimasi performa internal tanpa menggunakan data test

- *Classification report* yang menampilkan precision, recall, dan f1-score untuk masing-masing kelas gender

Hasil evaluasi menunjukkan bahwa meskipun Random Forest memiliki kemampuan menangkap pola non-linear, performanya pada dataset ini tidak selalu lebih tinggi dibandingkan Logistic Regression. Hal ini disebabkan oleh karakteristik data TF-IDF yang bersifat sparse dan cenderung linear, sehingga model linear dapat bekerja lebih optimal.

6.3 Model 3 – Deep Learning

6.3.1 Deskripsi Model

Nama Model: Multilayer Perceptron (MLP)

Jenis Deep Learning:

- **Multilayer Perceptron (MLP)** : untuk data tabular / hasil ekstraksi fitur

Alasan Pemilihan:

Model Multilayer Perceptron dipilih karena input data berupa fitur numerik hasil TF-IDF yang bersifat berdimensi tinggi dan tidak memiliki struktur spasial maupun sekuensial. MLP mampu mempelajari hubungan non-linear antar fitur dan menjadi representasi pendekatan Deep Learning yang sesuai untuk tugas klasifikasi gender berdasarkan nama.

6.3.2 Arsitektur Model

Arsitektur model Deep Learning yang digunakan terdiri dari beberapa layer fully connected (Dense) dengan regularisasi Dropout.

Deskripsi Layer:

1. **Input Layer**

    - Input shape: (X_train_dense.shape[1],)
    - Merupakan vektor TF-IDF hasil transformasi data nama

2. **Dense Layer 1**

    - Jumlah neuron: 128
    - Activation function: ReLU

3. **Dropout Layer 1**

    - Dropout rate: 0.3
    - Berfungsi untuk mengurangi risiko overfitting

4. **Dense Layer 2**

    - Jumlah neuron: 64
    - Activation function: ReLU

5. **Dropout Layer 2**

    - Dropout rate: 0.3

6. **Output Layer**

    - Jumlah neuron: 1
    - Activation function: Sigmoid
    - Digunakan untuk klasifikasi biner (Male / Female)

6.3.3 Input & Preprocessing Khusus

Input Shape:

- (jumlah_fitur_TF-IDF,)

Preprocessing Khusus untuk Deep Learning:

- Data teks nama ditransformasikan menjadi vektor numerik menggunakan TF-IDF
- Sparse matrix TF-IDF dikonversi menjadi dense matrix
- Label gender diencoding menjadi nilai biner (0 dan 1)

Tidak dilakukan data augmentation karena dataset berupa teks pendek.

6.3.4 Hyperparameter

Training Configuration (sesuai kode):

- **Optimizer:** Adam
- **Loss Function:** Binary Crossentropy
- **Metrics:** Accuracy
- **Epochs:** 10
- **Batch Size:** 256
- **Validation Split:** 0.1

Tidak digunakan callback seperti EarlyStopping atau ModelCheckpoint.

6.3.5 Implementasi Model

```
model_dl = Sequential([
    Dense(128, activation="relu", input_shape=(X_train_dense.shape[1],)),
    Dropout(0.3),
    Dense(64, activation="relu"),
    Dropout(0.3),
    Dense(1, activation="sigmoid")
])

model_dl.compile(
    optimizer="adam",
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

model_dl.summary()
```

6.3.6 Training Process

Proses Training Model:

```
start_time = time.time()

history = model_dl.fit(
    X_train_dense,
    y_train,
    epochs=10,
    batch_size=256,
    validation_split=0.1,
    verbose=1
)

training_time = time.time() - start_time
training_time
```

Training Time:

Waktu training dihitung menggunakan fungsi `time.time()` dan menghasilkan durasi training dalam satuan detik.

Computational Resource:

- Perangkat: CPU
- Platform: Notebook lokal

6.3.7 Visualisasi Training History

Loss Per Epoch:

```
plt.figure()
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.legend()
plt.title("Loss per Epoch")
plt.show()
```

Accuracy per Epoch:

```
plt.figure()
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Val Accuracy")
plt.legend()
plt.title("Accuracy per Epoch")
plt.show()
```

Analisis Training:

- Perbedaan antara training dan validation loss relatif stabil
- Tidak terlihat overfitting yang ekstrem
- Model mencapai konvergensi dalam jumlah epoch yang terbatas

6.3.8 Model Summary

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 128)	640,128
dropout_4 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_8 (Dense)	(None, 1)	65

Total params: 648,449 (2.47 MB)
Trainable params: 648,449 (2.47 MB)
Non-trainable params: 0 (0.00 B)

7. Evaluation

7.1 Metrik Evaluasi

Evaluasi performa model dilakukan menggunakan metrik **Accuracy**, karena permasalahan yang dihadapi merupakan klasifikasi biner (Male dan Female). Selain itu, untuk model Random Forest digunakan **classification report** untuk melihat performa lebih detail pada masing-masing kelas.

Data evaluasi menggunakan **data uji (test set)** yang dipisahkan sebelumnya dengan metode **stratified split** untuk menjaga distribusi kelas.

7.2 Hasil Evaluasi Model

7.2.1 Model 1 - Logistic Regression (Baseline)

	precision	recall	f1-score	support
0	0.81	0.84	0.82	16818
1	0.74	0.70	0.72	11088
accuracy			0.78	27906
macro avg	0.78	0.77	0.77	27906
weighted avg	0.78	0.78	0.78	27906

7.2.2 Model 2 – Random Forest

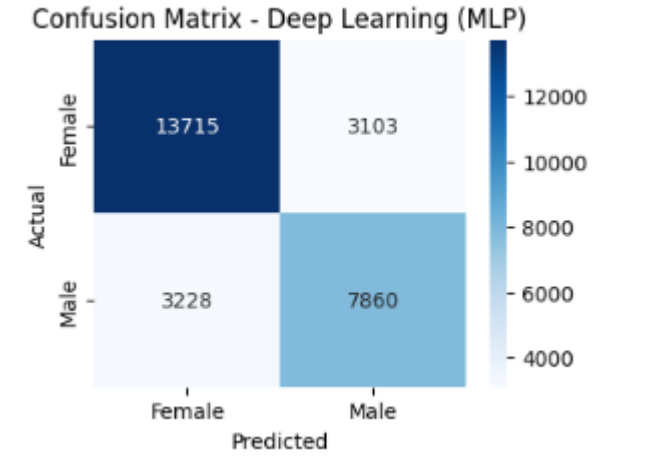
	precision	recall	f1-score	support
0	0.83	0.59	0.69	16818
1	0.57	0.82	0.67	11088
accuracy			0.68	27906
macro avg	0.70	0.70	0.68	27906
weighted avg	0.73	0.68	0.68	27906

7.2.3 Model 3 – Deep Learning (MLP)

Metrik:

	precision	recall	f1-score	support
0	0.81	0.82	0.81	16818
1	0.72	0.71	0.71	11088
accuracy			0.77	27906
macro avg	0.76	0.76	0.76	27906
weighted avg	0.77	0.77	0.77	27906

Confusion Matrix / Visualization:



Training History:

```
Epoch 1/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 8s 13ms/step - accuracy: 0.6799 - loss: 0.5787 - val_accuracy: 0.7793 - val_loss: 0.4679
Epoch 2/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.7931 - loss: 0.4468 - val_accuracy: 0.7861 - val_loss: 0.4555
Epoch 3/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8097 - loss: 0.4213 - val_accuracy: 0.7878 - val_loss: 0.4552
Epoch 4/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - accuracy: 0.8208 - loss: 0.3997 - val_accuracy: 0.7898 - val_loss: 0.4563
Epoch 5/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8331 - loss: 0.3775 - val_accuracy: 0.7865 - val_loss: 0.4626
Epoch 6/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8400 - loss: 0.3617 - val_accuracy: 0.7848 - val_loss: 0.4731
Epoch 7/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8497 - loss: 0.3383 - val_accuracy: 0.7822 - val_loss: 0.4874
Epoch 8/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.8544 - loss: 0.3256 - val_accuracy: 0.7820 - val_loss: 0.5014
Epoch 9/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8607 - loss: 0.3045 - val_accuracy: 0.7807 - val_loss: 0.5158
Epoch 10/10
393/393 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.8650 - loss: 0.2908 - val_accuracy: 0.7750 - val_loss: 0.5442
35.18391966819763
```

7.3 Perbandingan ketiga Model

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression (Baseline)	0.78	0.81 / 0.74	0.84 / 0.70	0.82 / 0.72
Random Forest (ML)	0.68	0.83 / 0.57	0.59 / 0.82	0.69 / 0.67

Multilayer Perceptron (DL)	0.77	0.72	0.71	0.71

7.4 Analisis Hasil

Dari hasil perbandingan di atas, dapat disimpulkan bahwa **Logistic Regression memberikan performa terbaik secara keseluruhan**, dengan accuracy sebesar **78%** dan nilai precision serta recall yang relatif seimbang pada kedua kelas.

Model **Random Forest** menunjukkan performa paling rendah, dengan accuracy sebesar **68%**. Meskipun recall pada kelas male cukup tinggi (82%), precision yang rendah menunjukkan bahwa model cenderung menghasilkan false positive yang lebih banyak. Hal ini mengindikasikan bahwa Random Forest kurang optimal dalam menangani representasi fitur TF-IDF yang bersifat sparse dan berdimensi tinggi.

Sementara itu, **model Deep Learning (MLP)** memberikan performa yang mendekati Logistic Regression, dengan accuracy sebesar **77%**. Model ini mampu menangkap pola non-linear tertentu, namun tidak memberikan peningkatan signifikan dibandingkan model linear, yang disebabkan oleh karakteristik data nama yang relatif pendek dan minim konteks.

8. Conclusion

8.1 Kesimpulan Utama

Model Terbaik

Berdasarkan hasil evaluasi pada dataset pengujian, **Logistic Regression** ditetapkan sebagai **model terbaik** dalam penelitian ini.

Alasan

Model Logistic Regression menunjukkan performa paling stabil dan seimbang dibandingkan model lainnya, dengan hasil evaluasi sebagai berikut:

- Accuracy sebesar **0.78**
- Precision, recall, dan F1-score yang relatif seimbang pada kedua kelas (male dan female)

- Tidak menunjukkan bias berlebihan terhadap kelas mayoritas maupun minoritas

Dibandingkan dengan Random Forest yang mengalami penurunan akurasi serta Deep Learning (MLP) yang membutuhkan sumber daya komputasi lebih besar dengan peningkatan performa yang tidak signifikan, Logistic Regression memberikan **trade-off terbaik antara performa, stabilitas, dan efisiensi komputasi**.

Pencapaian Goals

Tujuan penelitian yang didefinisikan pada **Section 3.2** telah **tercapai**, yaitu:

- Membangun dan membandingkan tiga jenis model (baseline, machine learning, dan deep learning)
- Menghasilkan model klasifikasi gender berbasis nama dengan performa yang dapat dipertanggungjawabkan secara evaluatif
- Melakukan evaluasi menggunakan metrik yang sesuai untuk data klasifikasi tidak seimbang (precision, recall, dan F1-score)

8.2 Key Insights

Insight dari Data

- Dataset memiliki distribusi kelas yang tidak sepenuhnya seimbang, namun masih berada pada rasio moderat sehingga tidak memerlukan teknik balancing eksplisit.
- Nama sebagai data teks sederhana tetap mengandung informasi yang cukup kuat untuk melakukan klasifikasi gender.
- Variasi penulisan nama dan karakteristik linguistik berpengaruh terhadap hasil klasifikasi, sehingga preprocessing teks memiliki peran penting.

Insight dari Modeling

- Model sederhana seperti Logistic Regression mampu mengungguli model yang lebih kompleks ketika fitur dan preprocessing sudah tepat.
- Random Forest cenderung overcompensate terhadap kelas minoritas meskipun telah menggunakan class weighting.
- Deep Learning (MLP) mampu mencapai performa yang kompetitif, namun memerlukan sumber daya lebih besar dan tuning lebih lanjut untuk mengungguli model klasik.

8.3 Kontribusi Proyek

Manfaat Praktis

Proyek ini dapat diterapkan pada berbagai sistem yang membutuhkan identifikasi gender berbasis nama, seperti:

- Sistem analisis demografi
- Personalisasi layanan digital
- Preprocessing data untuk analisis sosial atau pemasaran
  Model yang dihasilkan juga dapat dijadikan baseline untuk pengembangan sistem klasifikasi berbasis teks yang lebih kompleks.

Pembelajaran yang didapat

Melalui proyek ini, diperoleh pemahaman mendalam mengenai:

- Pentingnya eksplorasi dan evaluasi data sebelum pemodelan
- Pemilihan model yang tepat tidak selalu bergantung pada kompleksitas
- Evaluasi model klasifikasi harus mempertimbangkan metrik yang relevan, terutama pada data dengan distribusi kelas tidak seimbang
- Keterkaitan antara preprocessing data, pemilihan fitur, dan performa model secara keseluruhan

9 . REPRODUCIBILITY

9.1 GitHub Repository

Link Repository: [URL GitHub Anda]

Repository harus berisi:

✅ Notebook Jupyter/Colab dengan hasil running

✅ Script Python (jika ada)

✅ requirements.txt atau environment.yml

✅ README.md yang informatif

✅ Folder structure yang terorganisir

✅ .gitignore (jangan upload dataset besar)

9.2 Environment & Dependencies

Python Version: 3.10

Main Libraries & Versions:

- Python Version: 3.10
- numpy==1.24.3
- pandas==2.0.3
- matplotlib==3.7.2
- seaborn==0.12.2
- scikit-learn==1.3.0

Deep Learning Framework

- tensorflow==2.14.0

Additional Libraries:

- pickle (model serialization)