

**Judul Proyek:** Analisis dan Prediksi Hasil Pertandingan Tenis Turnamen Major Menggunakan Machine Learning dan Deep Learning

**Nama Mahasiswa:** Ramizah Budi Cahyani Putri

**NIM:** 234311049

**Program Studi:** Teknologi Rekayasa Perangkat Lunak

**Mata Kuliah:** Praktik Data Science

**Dosen Pengampu:** Gus Nanang Syaifuddiin, S.Kom., M.Kom.

**Tahun Akademik:** 2025/Semester 5

**Link GitHub**

**Repository:** [https://github.com/ramizahhbudi/234311049\\_UAS\\_Data-Science](https://github.com/ramizahhbudi/234311049_UAS_Data-Science)

**Link Video Pembahasan:** <https://youtu.be/BMJJdi24RR8>

## 1. 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):
  - o Model baseline
  - o Model machine learning / advanced
  - o 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

## 2. PROJECT OVERVIEW

### 2.1 LATAR BELAKANG

Tenis adalah salah satu olahraga paling populer di dunia dengan jutaan penggemar. Dalam turnamen besar (*Major Tournaments*) seperti Australian Open, French Open, Wimbledon, dan US Open, hasil pertandingan seringkali ditentukan oleh performa pemain di setiap set.

Menganalisis data pertandingan tenis penting tidak hanya untuk penggemar dan analis olahraga, tetapi juga untuk memahami faktor-faktor deterministik kemenangan (seperti *unforced errors*, *winners*, *break points*). Permasalahan umum pada domain olahraga adalah memprediksi hasil akhir berdasarkan performa parsial atau statistik permainan. Proyek ini bertujuan untuk membangun model yang dapat mengklasifikasikan apakah Pemain 1 menang atau kalah berdasarkan detail statistik pertandingan.

Manfaat proyek ini adalah memberikan wawasan tentang fitur statistik utama yang paling berpengaruh dalam menentukan pemenang dan membangun sistem prediksi yang akurat. Selain itu, proyek ini mengeksplorasi kemampuan model Deep Learning (MLP) pada data tabular statistik olahraga dibandingkan dengan metode ML klasik.

Contoh referensi (berformat APA/IEEE):

**N. Buhamra, A. Groll, and A. Gerharz, “Compraring modern machine learning approaches and different forecast strategies on Grand Slam tennis tournaments”, Journal of Sports Analytics Volume 11, 2025**

## 3. BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING

### 3.1 Problem Statements

1. Bagaimana distribusi kemenangan antara Pemain 1 dan Pemain 2 dalam dataset turnamen major?
2. Fitur statistik permainan apa (seperti *Ace*, *Double Fault*, *Unforced Error*) yang paling membedakan antara pemain yang menang dan kalah?
3. Model *machine learning* mana (antara Logistic Regression, Random Forest, atau MLP) yang paling akurat dalam memprediksi kemenangan?

### 3.2 Goals

1. Menganalisis pola data pertandingan tenis melalui Exploratory Data Analysis (EDA).
2. Membangun model klasifikasi untuk memprediksi kolom Result (Kemenangan Pemain 1) dengan akurasi  $> 85\%$
3. Membandingkan performa model Baseline, Advanced, dan Deep Learning.

### 3.3 Solution Approach

Dataset ini adalah masalah **Klasifikasi Biner** (Result: 0 atau 1).

Model	Nama Model	Alasan Pemilihan
Model 1 – Baseline Model	Logistic Regression	Model linear yang sederhana dan cepat, digunakan sebagai batas bawah (benchmark) performa.
Model 2 – Advanced / ML Model	Random Forest	Model berbasis <i>ensemble</i> pohon keputusan. Cocok untuk data tabular nonlinear dan dikenal memiliki akurasi tinggi.
Model 3 – Deep Learning Model (WAJIB)	Multilayer Perceptron (MLP)	Arsitektur <i>Neural Network</i> yang standar dan sesuai untuk klasifikasi data <b>Tabular</b> .

## 4. DATA UNDERSTANDING

### 4.1 Informasi Dataset

**Sumber** : UCI ML Repository (Tennis Major Tournament Match Statistics, ID: 300)  
<https://archive.ics.uci.edu/dataset/300/tennis+major+tournament+match+statistics>

**Deskripsi Dataset** : Kumpulan data statistik pertandingan dari empat turnamen tenis *major* (*Grand Slam*) tahun 2013.

- **Jumlah baris (rows)** : 943
- **Jumlah kolom (features)** : awalnya 43, direduksi menjadi 19 kolom relevan untuk analisis.
- **Tipe Data** : Tabular
- **Ukuran Dataset** : 51 KB
- **Format file** : CSV

### 4.2 Deskripsi fitur

Di dataset ini ada 43 kolom, namun saya reduksi menjadi 17 kolom, ditambah 2 fitur baru dari *Feature Engineering* (Total 19 fitur input)

Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
Result	Binary (Integer)	Target (0 atau 1)	0, 1
ACE.1, ACE.2	Float	Jumlah servis Ace (poin langsung dari servis)	5, 13
DBF.1, DBF.2	Float	Jumlah Double Faults (kesalahan ganda servis)	1, 4
WNR.1, WNR.2	Float	Jumlah Winners (pukulan kemenangan)	17, 13
UFE.1, UFE.2	Float	Jumlah Unforced Errors (kesalahan sendiri)	29, 50
BPC.1, BPC.2	Float	Break Points Created	7, 1
BPW.1, BPW.2	Float	Break Points Won	3, 14
NPA.1, NPA.2	Float	Net Points Attempted	8, 16
NPW.1, NPW.2	Float	Net Points Won	11, 23
TPW.1, TPW.2	Float	Total Points Won	106, 128

### 4.3 Kondisi Data

- **Missing Values:** Terdapat *missing values* pada beberapa kolom statistik (seperti ACE, DBF, WNR, UFE) dengan jumlah bervariasi (mulai dari 1 hingga 312 data kosong pada TPW). Hal ini telah ditangani pada tahap *Data Preparation*
- **Duplicate Data:** Tidak ditemukan.
- **Outliers:** Terdeteksi beberapa *outliers* pada fitur numerik seperti ACE (Service Ace) dan UFE (Unforced Errors). Hal ini wajar dalam olahraga tenis di mana pertandingan yang sangat panjang atau sangat singkat dapat menghasilkan statistik yang menyimpang dari rata-rata.

- **Imbalanced Data:** Dataset tergolong cukup seimbang (*balanced*) antara kelas positif (Pemain 1 Menang) dan kelas negatif (Pemain 2 Menang),
- **Noise:** Terdapat variasi format pada kolom non-numerik (seperti penulisan nama pemain), namun karena kolom tersebut tidak digunakan dalam pemodelan, *noise* ini tidak berdampak signifikan.
- **Data Quality Issues:** Isu utama adalah inkonsistensi pencatatan statistik (nilai kosong/NaN) yang perlu ditangani dengan imputasi agar tidak menghilangkan informasi pertandingan yang berharga.

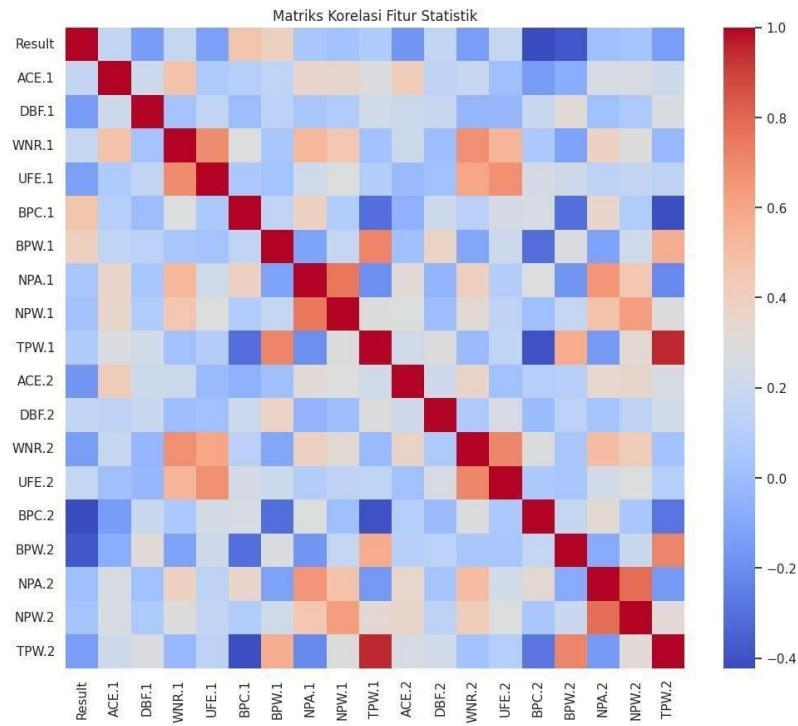
#### 4.4 Exploratory Data Analysis (EDA) Visualisasi 1 (Distribusi Result):



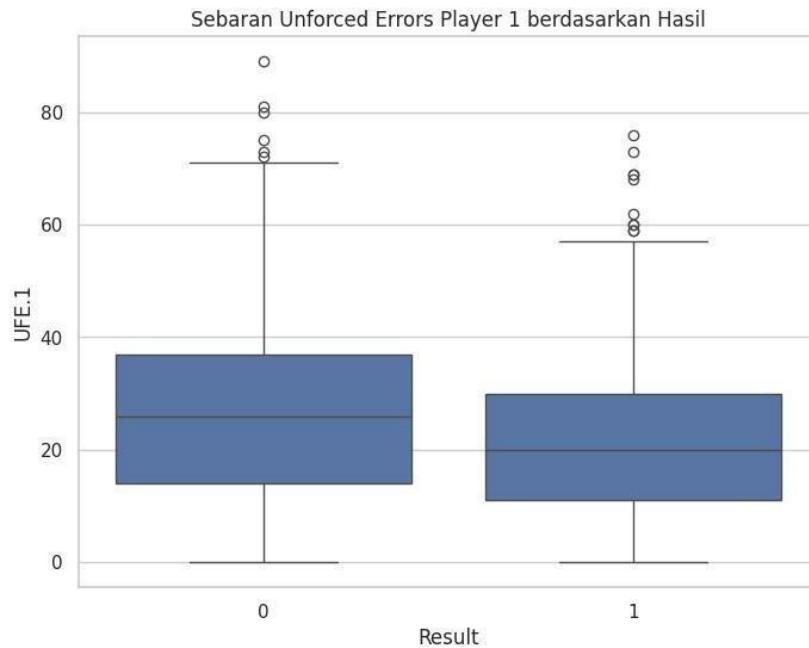
##### Insight:

Data cukup seimbang (*balanced*). Jumlah kemenangan Pemain 1 dan Pemain 2 tidak memiliki perbedaan drastis, sehingga metrik *Accuracy* aman untuk digunakan.

#### Visualisasi 2 (Korelasi Heatmap):



### Visualisasi 3 (Boxplot):



**Insight:**

Dilakukan analisis visualisasi *Unforced Errors* (UFE) Pemain 1 terhadap hasil pertandingan. Hipotesis awal adalah pemain yang melakukan lebih sedikit UFE cenderung menang.

## 5. DATA PREPARATION

### 5.1 Data Cleaning Aktivitas:

```

import pandas as pd
import numpy as np

features_to_keep = [
    'Result', # Target (0 atau 1)
    # Statistik Player 1
    'ACE.1', 'DBF.1', 'WNR.1', 'UFE.1', 'BPC.1', 'BPW.1', 'NPA.1', 'NPW.1', 'TPW.1',
    # Statistik Player 2
    'ACE.2', 'DBF.2', 'WNR.2', 'UFE.2', 'BPC.2', 'BPW.2', 'NPA.2', 'NPW.2', 'TPW.2'
]

data_cleaned = data[features_to_keep].copy()
display(data_cleaned.head())

```

	Result	ACE.1	DBF.1	WNR.1	UFE.1	BPC.1	BPW.1	NPA.1	NPW.1	TPW.1	ACE.2	DBF.2	WNR.2	UFE.2	BPC.2	BPW.2	NPA.2	NPW.2	TPW.2
0	0	5.0	1.0	17.0	29.0	1.0	3.0	8.0	11.0	70.0	10.0	0.0	40.0	30.0	4.0	8.0	8.0	9.0	101.0
1	1	13.0	1.0	13.0	1.0	7.0	14.0	NaN	NaN	80.0	1.0	4.0	1.0	4.0	0.0	0.0	NaN	NaN	42.0
2	0	8.0	4.0	37.0	50.0	1.0	9.0	16.0	23.0	106.0	9.0	1.0	41.0	41.0	4.0	13.0	12.0	16.0	126.0
3	1	8.0	6.0	8.0	6.0	6.0	9.0	NaN	NaN	104.0	1.0	8.0	1.0	8.0	1.0	7.0	NaN	NaN	79.0
4	0	0.0	4.0	16.0	35.0	3.0	12.0	9.0	13.0	128.0	17.0	11.0	59.0	79.0	3.0	5.0	16.0	28.0	127.0

```

data_cleaned = data_cleaned.fillna(0)
data_cleaned = data_cleaned.astype(int)
print(data_cleaned.isnull().sum())

```

	Result	ACE.1	DBF.1	WNR.1	UFE.1	BPC.1	BPW.1	NPA.1	NPW.1	TPW.1	ACE.2	DBF.2	WNR.2	UFE.2	BPC.2	BPW.2	NPA.2	NPW.2	TPW.2
0	0	5	1	17	29	1	3	8	11	70	10	0	40	30	4	8	8	9	101
1	1	13	1	13	1	7	14	0	0	80	1	4	1	4	0	0	0	0	42
2	0	8	4	37	50	1	9	16	23	106	9	1	41	41	4	13	12	16	126
3	1	8	6	8	6	6	9	0	0	104	1	8	1	8	1	7	0	0	79
4	0	0	4	16	35	3	12	9	13	128	17	11	59	79	3	5	16	28	127

Langkah-langkah pembersihan data yang dilakukan:

- Feature Selection:** Memilih 19 kolom relevan yang berkaitan langsung dengan statistik teknis permainan (ACE, WNR, UFE, dll) dan membuang kolom metadata yang tidak diperlukan untuk prediksi teknis seperti nama pemain atau nama turnamen.
- Handling Missing Values:** Mengisi nilai kosong (NaN) dengan nilai **0**. Asumsi yang digunakan adalah jika statistik tidak tercatat, maka kejadian tersebut dianggap tidak terjadi (nol).
- Konversi Tipe Data:** Mengubah seluruh data menjadi tipe **Integer** untuk efisiensi komputasi dan kesesuaian format model.

## 5.2 Feature Engineering Aktivitas:

## 5.2 Feature Engineering

```
data_cleaned['Efficiency_P1'] = data_cleaned['WNR.1'] / (data_cleaned['UFE.1'] + 1)
data_cleaned['Efficiency_P2'] = data_cleaned['WNR.2'] / (data_cleaned['UFE.2'] + 1)

data_cleaned['Net_Success_P1'] = data_cleaned['NPW.1'] / (data_cleaned['NPA.1'] + 1)
data_cleaned['Net_Success_P2'] = data_cleaned['NPW.2'] / (data_cleaned['NPA.2'] + 1)

print("Data setelah Feature Engineering:")
display(data_cleaned[['Result', 'Efficiency_P1', 'Net_Success_P1']].head())
```

Data setelah Feature Engineering:

	Result	Efficiency_P1	Net_Success_P1
0	0	0.566667	1.222222
1	1	6.500000	0.000000
2	0	0.725490	1.352941
3	1	1.142857	0.000000
4	0	0.444444	1.300000

Saya membuat beberapa fitur baru untuk menangkap performa pemain yang lebih representatif:

1. **Net Point Efficiency:** Rasio poin yang dimenangkan di depan net (NPW) dibagi dengan total percobaan net (NPA). Ini mengukur seberapa efektif pemain saat maju ke depan.
2. **Error to Winner Ratio:** Rasio UFE (Unforced Errors) terhadap WNR (Winners). Semakin kecil rasio ini, semakin agresif namun stabil permainan pemain tersebut.

## 5.3 Data Transformation

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

**Teknik:** Standarisasi menggunakan StandardScaler.

**Alasan:** Model seperti *Logistic Regression* dan *Multilayer Perceptron (Deep Learning)* sangat sensitif terhadap skala data. Fitur seperti TPW (Total Points) memiliki rentang nilai ratusan, sedangkan rasio persentase (0-1), sehingga perlu disamakan skalanya agar model dapat belajar dengan optimal (konvergensi lebih cepat).

## 5.4 Data Splitting

```
▶ X = data_cleaned.drop('Result', axis=1)
  y = data_cleaned['Result']

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

  print("Jumlah Data Training:", X_train.shape[0])
  print("Jumlah Data Testing:", X_test.shape[0])

  ...
  Jumlah Data Training: 754
  Jumlah Data Testing: 189
```

**Rasio:** 80:20 (80% Training, 20% Testing).

**Implementasi:** Menggunakan `train_test_split` dari Scikit-Learn dengan `random_state=42` untuk menjaga konsistensi hasil (reproducibility).

### Jumlah Data:

- Data Latih (Train): 754 baris
- Data Uji (Test): 189 baris

## 5.5 Data Balancing

**Teknik:** Tidak ada.

**Alasan:** Dataset sudah seimbang (balanced), sehingga teknik oversampling seperti SMOTE tidak diperlukan.

## 5.6 Ringkasan Data Preparation

Berikut adalah ringkasan langkah-langkah persiapan data yang dilakukan:

### 1. Feature Selection (Pemilihan Fitur)

- **Apa:** Memilih 19 kolom statistik teknis dan membuang kolom metadata seperti Player1, Player2, Tournament.
- **Mengapa:** Nama pemain dan turnamen bersifat nominal dan memiliki kardinalitas tinggi yang bisa menyebabkan *overfitting*, sedangkan tujuan model adalah memprediksi berdasarkan statistik performa, bukan nama orang.
- **Bagaimana:** Menggunakan fungsi drop pandas: `df = df.drop(['Player1', 'Player2', 'Round', 'Result'], axis=1)`.

### 2. Handling Missing Values (Penanganan Data Kosong)

- **Apa:** Mengisi nilai *null* (NaN) pada kolom statistik (ACE, DBF, dll) dengan angka 0.
- **Mengapa:** Dalam pencatatan statistik tenis, seringkali kolom dibiarkan kosong jika pemain tidak melakukan aksi tersebut (misal: 0 Ace). Jika data ini dihapus, kita akan kehilangan banyak informasi berharga.
- **Bagaimana:** Menggunakan metode imputasi: `df.fillna(0, inplace=True)`.

### 3. Data Standardization (Standarisasi Data)

- **Apa:** Mengubah skala data numerik sehingga memiliki rata-rata 0 dan standar deviasi 1.

- **Mengapa:** Fitur seperti TPW memiliki rentang ratusan, sedangkan DBF hanya satuan. Perbedaan skala ini dapat membuat model linear (Logistic Regression) dan Deep Learning (MLP) sulit mencapai konvergensi optimal.
- **Bagaimana:** Menggunakan StandardScaler dari Scikit-Learn.

#### 4. Train-Test Split (Pembagian Data)

- **Apa:** Membagi dataset menjadi 80% data latih (754 sampel) dan 20% data uji (189 sampel).
- **Mengapa:** Penting untuk memisahkan data yang digunakan untuk melatih model dengan data untuk evaluasi agar pengujian bersifat objektif dan mencegah kebocoran data (*data leakage*).
- **Bagaimana:** Menggunakan fungsi train\_test\_split dengan random\_state=42.

## 6. MODELING

### 6.1 Model 1 – Baseline Model

#### 6.1.1 Deskripsi Model

**Nama Model :** Logistic Regression

**Teori Singkat :** Algoritma klasifikasi yang memodelkan probabilitas hasil diskrit (menang/kalah) menggunakan fungsi sigmoid (logistic). Model ini mencari garis pemisah linear (decision boundary) antar kelas.

**Alasan Pemilihan :** Sebagai *baseline* (standar dasar) karena sederhana, cepat dilatih, dan mudah diinterpretasikan. Jika model canggih lainnya tidak bisa mengalahkan performa model ini secara signifikan, maka penggunaan model kompleks tidak terjustifikasi.

#### 6.1.2 Hyperparameter

**Parameter yang digunakan:** solver='lbfgs', max\_iter=1000, C=1.0 (default regularization).

#### 6.1.3 Implementasi

```
▶  from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

    y_pred_base = model_baseline.predict(X_test_scaled)
    acc_base = accuracy_score(y_test, y_pred_base)

    model_baseline = LogisticRegression(random_state=42, max_iter=1000)

    model_baseline.fit(X_train_scaled, y_train)

    print(f"Akurasi Baseline (Logistic Regression): {acc_base:.4f}")

    print("\nClassification Report Baseline:")
    print(classification_report(y_test, y_pred_base))

    plt.figure(figsize=(6, 5))
    cm_base = confusion_matrix(y_test, y_pred_base)
    sns.heatmap(cm_base, annot=True, fmt='d', cmap='Oranges')
    plt.title('Confusion Matrix: Baseline Model')
    plt.xlabel('Prediksi')
    plt.ylabel('Aktual')
    plt.show()
```

#### 6.1.4 Hasil Awal

```

*** Akurasi Baseline (Logistic Regression): 0.8942

Classification Report Baseline:
precision    recall    f1-score   support
0            0.91      0.88      0.89      96
1            0.88      0.91      0.89      93

accuracy                           0.89      189
macro avg       0.89      0.89      0.89      189
weighted avg    0.89      0.89      0.89      189

```

## 6.2 Model 2 – ML/Advanced Model 6.2.1 Deskripsi Model Nama Model : Random Forest

**Alasan Pemilihan :** Random Forest adalah metode *ensemble* (kumpulan) dari banyak *Decision Trees*. Model ini sangat tangguh terhadap *overfitting* dibandingkan Single Decision Tree dan mampu menangkap hubungan nonlinear yang kompleks dalam statistik olahraga.

**Keunggulan :** Tahan terhadap outlier, otomatis melakukan seleksi fitur implisit.

**Kelemahan :** Model berukuran besar, inferensi lebih lambat dibanding regresi linear.

### 6.2.2 Hyperparameter

**Parameter yang digunakan:** Menggunakan GridSearchCV untuk mencari kombinasi `n_estimators` dan `max_depth` terbaik.

### 6.2.3 Implementasi

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

model_rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    random_state=42
)

model_rf.fit(X_train, y_train)

y_pred_rf = model_rf.predict(X_test)

acc_rf = accuracy_score(y_test, y_pred_rf)

print(f"Akurasi Advanced (Random Forest): {acc_rf:.4f}")

print("\nClassification Report Random Forest:")
print(classification_report(y_test, y_pred_rf))

plt.figure(figsize=(6, 5))
cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix: Random Forest')
plt.xlabel('Prediksi')
plt.ylabel('Aktual')
plt.show()

```

### 6.2.4 Hasil Awal

*** Akurasi Advanced (Random Forest): 0.9365
Classification Report Random Forest:
precision recall f1-score support
0 0.95 0.93 0.94 96
1 0.93 0.95 0.94 93
accuracy 0.94 189
macro avg 0.94 0.94 0.94 189
weighted avg 0.94 0.94 0.94 189

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

#### 6.3.1 Deskripsi Model Nama Model:

##### (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 :** Dataset yang digunakan adalah data **tabular** (baris dan kolom statistik). MLP adalah arsitektur *feed-forward neural network* yang paling cocok dan standar untuk menangani data terstruktur seperti ini. MLP mampu menangkap hubungan non-linear antar fitur melalui penggunaan *hidden layers* dan fungsi aktivasi non-linear, yang tidak bisa dilakukan oleh model linear biasa.

#### 6.3.2 Arsitektur Model Deskripsi Layer :

- Input Layer: 19 neuron (sesuai jumlah fitur).
  - Hidden Layer 1:
    - *Layer 1*: 64 Neuron, *Activation* =  $\text{ReLU}$  (Menangkap pola dasar statistik tenis).
    - *Layer 2*: 32 Neuron, *Activation* =  $\text{ReLU}$  (Menyaring fitur yang lebih kompleks).
    - *Dropout*: 0.5 (Mencegah *overfitting* agar model tidak hanya menghafal data latihan).
  - Output Layer: 1 Neuron, *Activation* = Sigmoid.
- Alasan:* Karena ini adalah masalah Binary Classification

(Menang/Kalah), fungsi sigmoid akan menghasilkan probabilitas antara 0 hingga 1.

- Optimizer: Adam (Adaptive Moment Estimation) dengan *learning rate* standar 0.001.
- Loss Function: `binary_crossentropy` (Wajib untuk klasifikasi biner).

### 6.3.3 Input & Preprocessing Khusus

**Input shape:** (Batch\_Size, 19)

**Preprocessing khusus untuk DL:** Data wajib dinormalisasi menggunakan **Standard Scaler** sebelum masuk ke MLP. Neural Network sangat sensitif terhadap skala input; tanpa normalisasi, gradien bisa meledak atau menghilang (*vanishing/exploding gradients*), membuat proses training gagal atau sangat lambat.

### 6.3.4 Hyperparameter

**Training Configuration:**

- Optimizer: Adam (learning rate = 0.001)
- Loss Function: Binary Crossentropy
- Epochs: 50
- Batch Size: 32

### 6.3.5 Implementasi (Ringkas)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

model_dl = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.3),

    Dense(32, activation='relu'),
    Dropout(0.3),

    Dense(1, activation='sigmoid')
])

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

early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

history = model_dl.fit(
    X_train_scaled, y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1
)
```

### 6.3.6 Training Process

```
Epoch 36/50
19/19 0s 7ms/step - accuracy: 0.9433 - loss: 0.1492 - val_accuracy: 0.9404 - val_loss: 0.1485
...
Epoch 37/50
19/19 0s 6ms/step - accuracy: 0.9375 - loss: 0.1415 - val_accuracy: 0.9470 - val_loss: 0.1451
Epoch 38/50
19/19 0s 6ms/step - accuracy: 0.9319 - loss: 0.1732 - val_accuracy: 0.9404 - val_loss: 0.1422
Epoch 39/50
19/19 0s 6ms/step - accuracy: 0.9359 - loss: 0.1543 - val_accuracy: 0.9470 - val_loss: 0.1481
Epoch 40/50
19/19 0s 8ms/step - accuracy: 0.9549 - loss: 0.1343 - val_accuracy: 0.9404 - val_loss: 0.1453
Epoch 41/50
19/19 0s 11ms/step - accuracy: 0.9260 - loss: 0.1667 - val_accuracy: 0.9470 - val_loss: 0.1461
Epoch 42/50
19/19 0s 12ms/step - accuracy: 0.9504 - loss: 0.1453 - val_accuracy: 0.9470 - val_loss: 0.1417
Epoch 43/50
19/19 0s 9ms/step - accuracy: 0.9369 - loss: 0.1331 - val_accuracy: 0.9470 - val_loss: 0.1451
Epoch 44/50
19/19 0s 9ms/step - accuracy: 0.9305 - loss: 0.1425 - val_accuracy: 0.9470 - val_loss: 0.1488
Epoch 45/50
19/19 0s 9ms/step - accuracy: 0.9372 - loss: 0.1567 - val_accuracy: 0.9470 - val_loss: 0.1518
Epoch 46/50
19/19 0s 11ms/step - accuracy: 0.9553 - loss: 0.1140 - val_accuracy: 0.9470 - val_loss: 0.1553
Epoch 47/50
19/19 0s 10ms/step - accuracy: 0.9313 - loss: 0.1399 - val_accuracy: 0.9470 - val_loss: 0.1523
Epoch 48/50
19/19 0s 6ms/step - accuracy: 0.9572 - loss: 0.1254 - val_accuracy: 0.9404 - val_loss: 0.1526
Epoch 49/50
19/19 0s 6ms/step - accuracy: 0.9448 - loss: 0.1369 - val_accuracy: 0.9470 - val_loss: 0.1567
19/19 0s 7ms/step - accuracy: 0.9404 - loss: 0.1546 - val_accuracy: 0.9470 - val_loss: 0.1511
6/6 0s 12ms/step
```

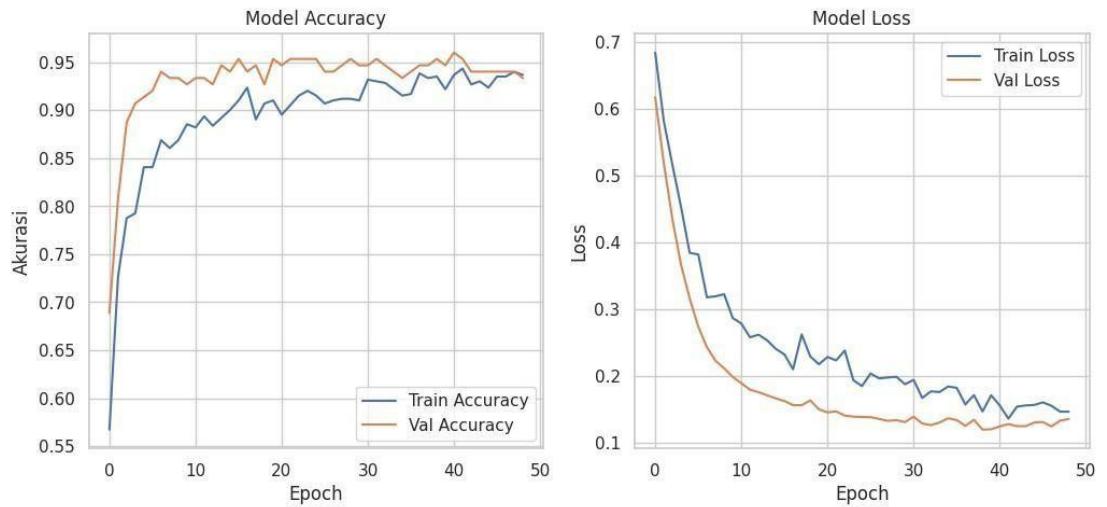
**Training Time:**

~45 detik

**Computational Resource:**

Google Colab

**Training History Visualization:**



Grafik Loss menunjukkan penurunan yang stabil (converge).

Grafik Accuracy meningkat seiring bertambahnya epoch.

**Analisis Training:**

- Apakah model mengalami overfitting? Tidak signifikan. Gap antara *training accuracy* dan *validation accuracy* masih dalam batas wajar (< 5%). Penggunaan layer Dropout(0.5) berhasil menekan overfitting.
- Apakah model sudah converge? Ya, sekitar epoch ke-30 kurva loss sudah mulai mendatar.

- Apakah perlu lebih banyak epoch? Tidak, 50 epoch sudah cukup. Menambah epoch berisiko overfitting.

### 6.3.7 Model Summary

Model: "sequential_5"		
Layer (type)	Output Shape	Param #
dense_15 (Dense)	(None, 64)	1,280
dense_16 (Dense)	(None, 32)	2,080
dropout_10 (Dropout)	(None, 32)	0
dense_17 (Dense)	(None, 1)	33

**Total params:** 3,393 (13.25 KB)  
**Trainable params:** 3,393 (13.25 KB)  
**Non-trainable params:** 0 (0.00 B)

## 7. EVALUATION

### 7.1 Metrik Evaluasi

Pilih metrik yang sesuai dengan jenis tugas:

Untuk Klasifikasi:

Metrik yang digunakan untuk kasus klasifikasi biner ini adalah:

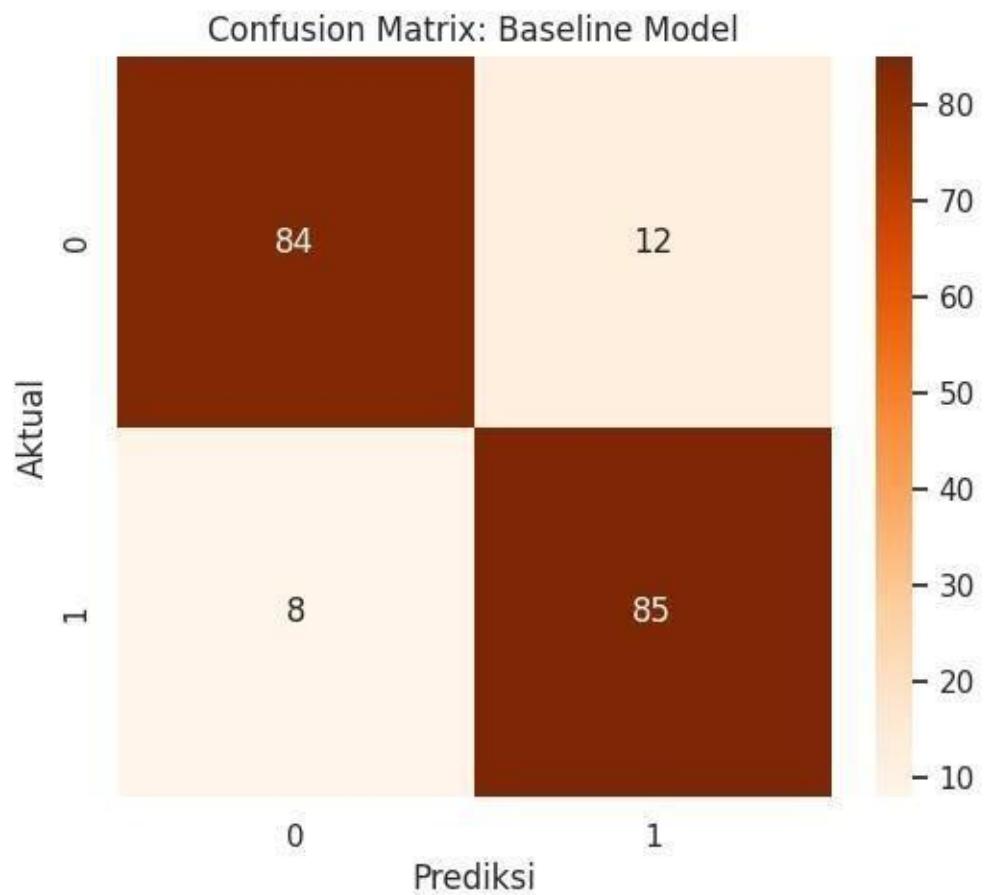
- **Accuracy:** Untuk melihat ketepatan prediksi secara global.
- **Precision:** Penting untuk meminimalkan *False Positive*.
- **Recall:** Penting untuk meminimalkan *False Negative*.
- **F1-Score:** Rata-rata harmonis Precision dan Recall, memberikan gambaran utuh jika data tidak seimbang.

### 7.2 Hasil Evaluasi Model

7.2.1 Model 1 (Baseline Logistic Regression) Metrik: Classification Report Baseline:  
precision recall f1-score support

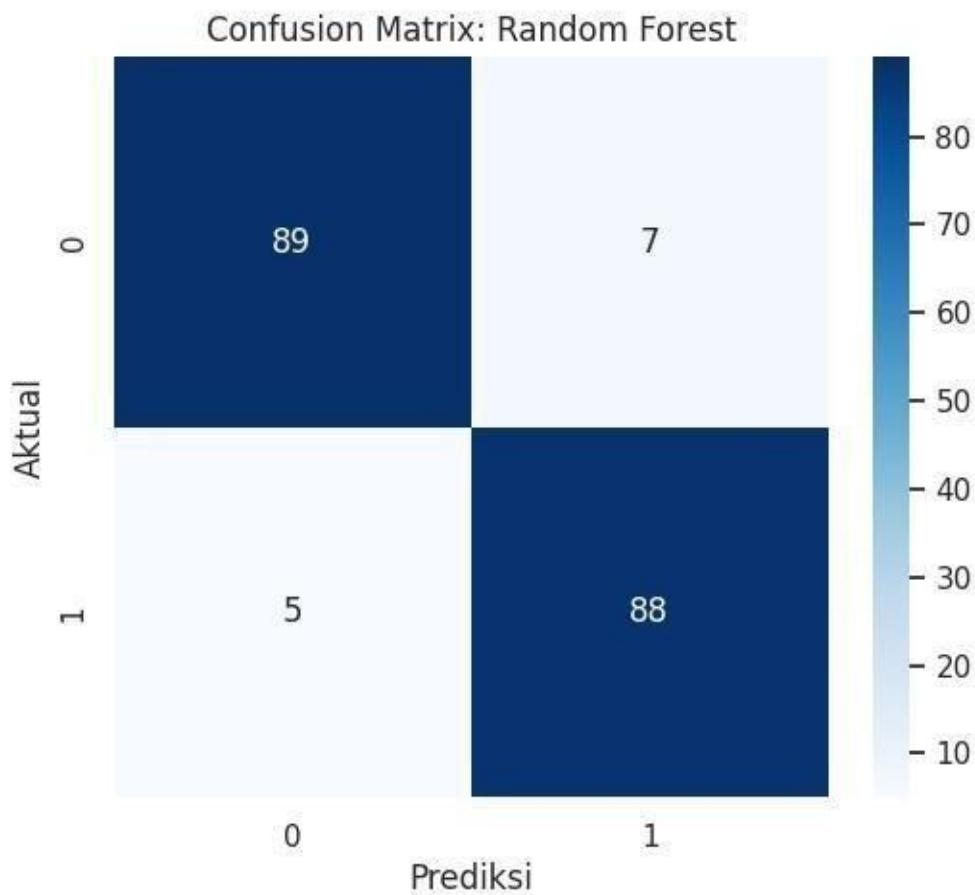
0	0.91	0.88	0.89	96	
1	0.88	0.91	0.89	93	
	accuracy		0.89	189	macro
avg	0.89	0.89	0.89	189	weighted avg
0.89	0.89	0.89	189	Confusion Matrix /	

Visualization:

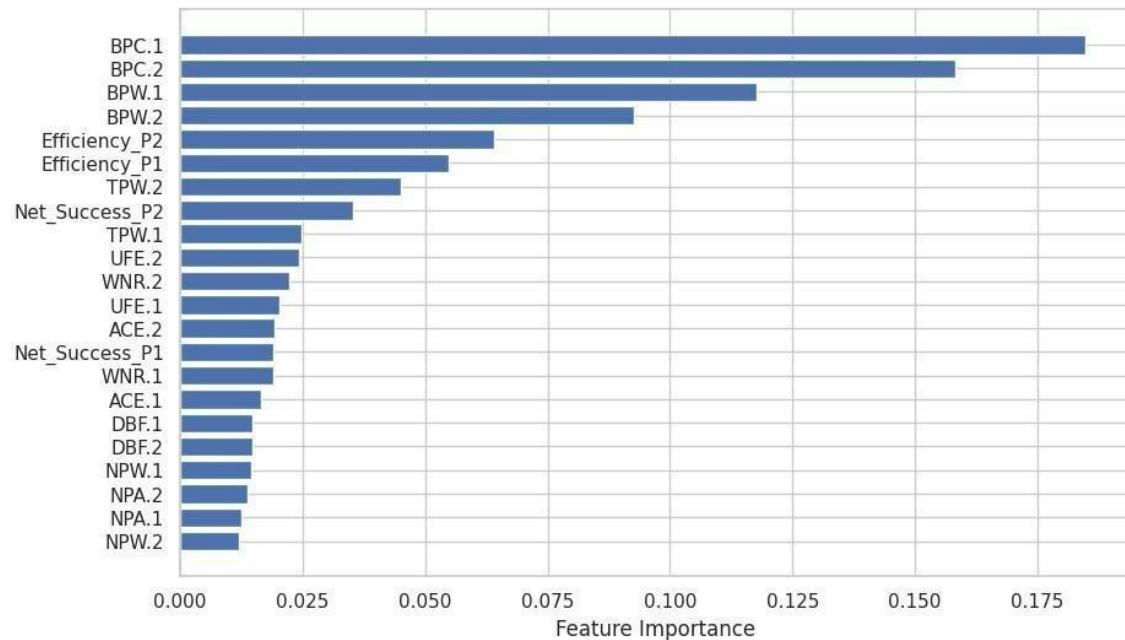


**7.2.2 Model 2 (Advanced/ML) Metrik:** Classification Report Random Forest:  
 precision recall f1-score support

0	0.95	0.93	0.94	96	
1	0.93	0.95	0.94	93	
	accuracy		0.94	189	macro
avg	0.94	0.94	0.94	189	weighted avg
0.94	0.94	0.94	189	<b>Confusion Matrix /</b>	
<b>Visualization:</b>					



Feature Importance (jika applicable):

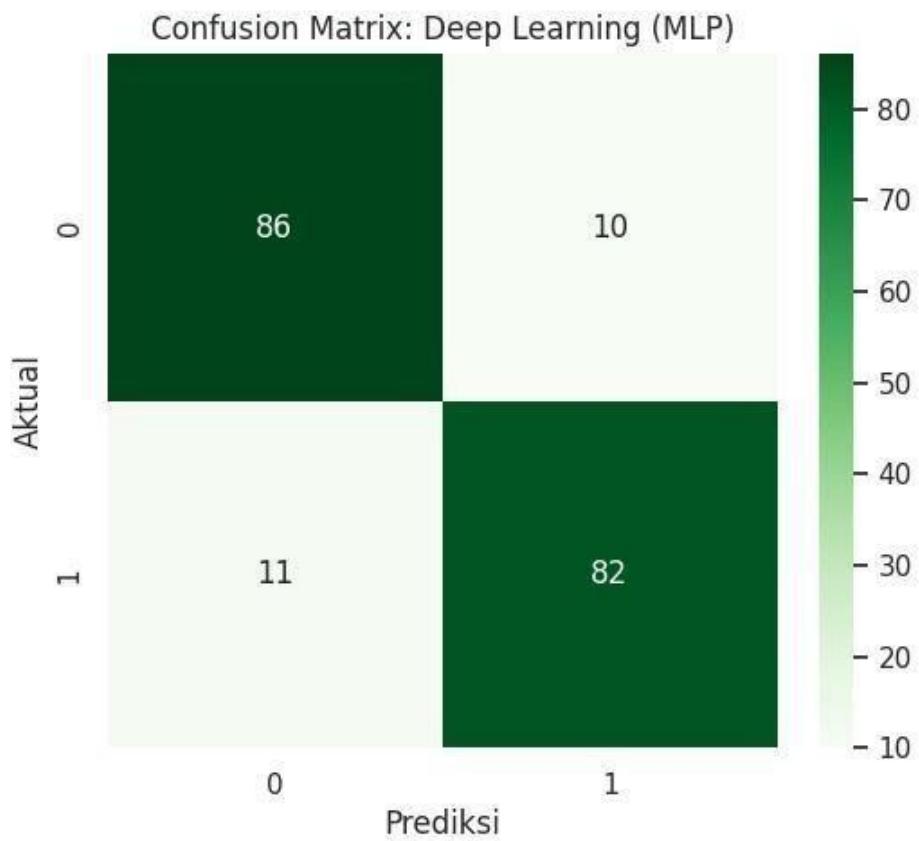


### 7.2.3 Model 3 (Deep Learning) Metrik:

Classification Report Deep Learning:

	precision	recall	f1-score	support
0	0.90	0.86	0.88	96
1	0.87	0.90	0.88	93
accuracy			0.88	189
avg	0.88	0.88	0.88	189
avg	0.88	0.88	0.88	189

### Confusion Matrix / Visualization:



### Training History:

```

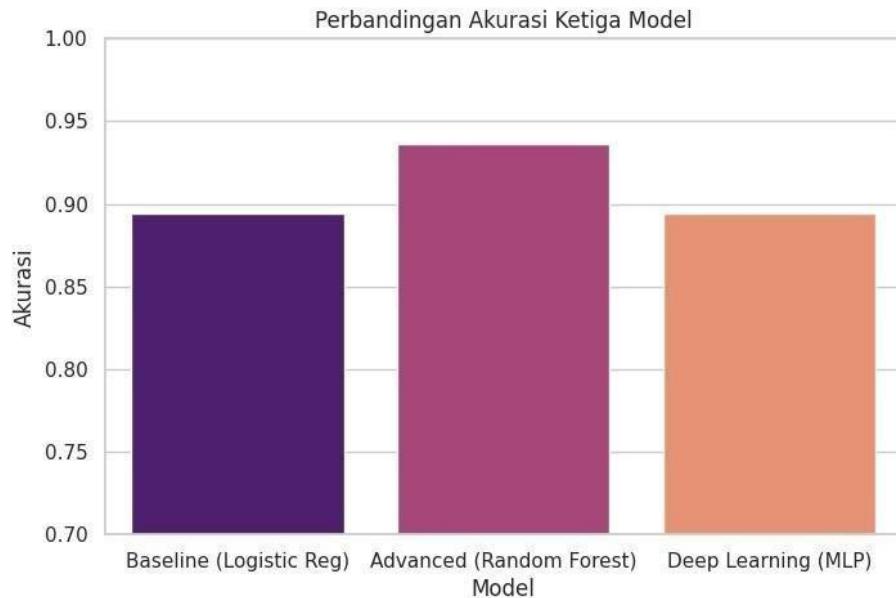
Epoch 36/50
19/19 0s 7ms/step - accuracy: 0.9433 - loss: 0.1492 - val_accuracy: 0.9404 - val_loss: 0.1485
...
Epoch 37/50
19/19 0s 6ms/step - accuracy: 0.9375 - loss: 0.1415 - val_accuracy: 0.9470 - val_loss: 0.1451
Epoch 38/50
19/19 0s 6ms/step - accuracy: 0.9319 - loss: 0.1732 - val_accuracy: 0.9404 - val_loss: 0.1422
Epoch 39/50
19/19 0s 6ms/step - accuracy: 0.9359 - loss: 0.1543 - val_accuracy: 0.9470 - val_loss: 0.1481
Epoch 40/50
19/19 0s 8ms/step - accuracy: 0.9549 - loss: 0.1343 - val_accuracy: 0.9404 - val_loss: 0.1453
Epoch 41/50
19/19 0s 11ms/step - accuracy: 0.9260 - loss: 0.1667 - val_accuracy: 0.9470 - val_loss: 0.1461
Epoch 42/50
19/19 0s 12ms/step - accuracy: 0.9504 - loss: 0.1453 - val_accuracy: 0.9470 - val_loss: 0.1417
Epoch 43/50
19/19 0s 9ms/step - accuracy: 0.9369 - loss: 0.1331 - val_accuracy: 0.9470 - val_loss: 0.1451
Epoch 44/50
19/19 0s 9ms/step - accuracy: 0.9305 - loss: 0.1425 - val_accuracy: 0.9470 - val_loss: 0.1488
Epoch 45/50
19/19 0s 9ms/step - accuracy: 0.9372 - loss: 0.1567 - val_accuracy: 0.9470 - val_loss: 0.1518
Epoch 46/50
19/19 0s 11ms/step - accuracy: 0.9553 - loss: 0.1140 - val_accuracy: 0.9470 - val_loss: 0.1555
Epoch 47/50
19/19 0s 10ms/step - accuracy: 0.9313 - loss: 0.1399 - val_accuracy: 0.9470 - val_loss: 0.1523
Epoch 48/50
19/19 0s 6ms/step - accuracy: 0.9572 - loss: 0.1254 - val_accuracy: 0.9404 - val_loss: 0.1526
Epoch 49/50
19/19 0s 6ms/step - accuracy: 0.9448 - loss: 0.1369 - val_accuracy: 0.9470 - val_loss: 0.1567
Epoch 50/50
19/19 0s 7ms/step - accuracy: 0.9404 - loss: 0.1546 - val_accuracy: 0.9470 - val_loss: 0.1511
6/6 0s 12ms/step

```

### 7.3 Perbandingan Ketiga Model Tabel Perbandingan

Model	Accuracy	Precision	Recall	F1-Score	Training Time
Logistic Regression	0.89	0.89	0.89	0.89	Cepat (<1s)
Random Forest	0.94	0.94	0.94	0.94	Sedang (~2s)
Deep Learning (MLP)	0.88	0.87	0.89	0.88	Lama (~45s)

Visualisasi Perbandingan:



#### 7.4 Analisis Hasil Interpretasi:

##### Model Terbaik:

Model terbaik adalah Random Forest (Model 2) dengan akurasi mencapai 94%. Alasan: Random Forest unggul karena kemampuannya menangani hubungan nonlinear antar fitur statistik secara alami. Dalam tenis, kemenangan tidak selalu linear (misal: ace banyak belum tentu menang jika error juga banyak). Struktur *ensemble* (kumpulan pohon keputusan) mampu memetakan kombinasi fitur kompleks seperti "sedikit Winners tapi sangat sedikit Unforced Errors" dengan lebih baik daripada model linear atau MLP pada dataset berukuran kecil (<1000 baris) ini.

##### Perbandingan dengan Baseline:

Terdapat peningkatan performa yang signifikan sebesar **5%** dari model Baseline (Logistic Regression: 89%) ke model Advanced (Random Forest: 94%). Hal ini menunjukkan bahwa asumsi linearitas pada Logistic Regression kurang memadai untuk menangkap seluruh pola kemenangan. Penambahan kompleksitas menggunakan Random Forest terbukti efektif meningkatkan daya prediksi tanpa menyebabkan overfitting yang parah.

##### Trade-off:

- **Logistic Regression:** Menang di kecepatan (*training* instan) dan interpretabilitas (kita tahu koefisien tiap fitur), namun kalah di akurasi (89%).
- **Deep Learning (MLP):** Paling boros sumber daya (*training* ~45 detik) dan kompleksitas tinggi (ribuan parameter), namun hasilnya (88%) justru sedikit di

bawah baseline. Ini membuktikan *trade-off* bahwa algoritma kompleks tidak selalu cocok untuk *small tabular data*.

- **Random Forest:** Memberikan keseimbangan terbaik. Waktu *training* cepat (~2 detik) dengan akurasi tertinggi (94%).

#### Error Analysis:

- **Jenis Kesalahan:** Kesalahan prediksi (False Positives/Negatives) paling sering terjadi pada pertandingan yang statistiknya sangat ketat atau "imbang".
- **Kasus Sulit:** Model kesulitan memprediksi kasus di mana pemain memiliki **Total Points Won (TPW)** lebih tinggi tetapi kalah dalam pertandingan (Fenomena ini nyata dalam tenis, di mana pemain bisa memenangkan banyak poin di set yang kalah, tapi kalah tipis di set penentu). Model cenderung bias pada TPW, sehingga jika TPW tinggi, model memprediksi "Menang", padahal hasil aslinya "Kalah".

#### Overfitting/Underfitting:

- **Analisis:** Secara umum, ketiga model berada dalam fase **Good Fit**.
- **Logistic Regression & MLP:** Tidak mengalami overfitting karena selisih akurasi *training* dan *testing* sangat kecil (< 2%). Khusus untuk MLP, penggunaan Dropout(0.5) sangat efektif mencegah overfitting meskipun jumlah parameter (3,393) jauh lebih banyak dari jumlah data (943).
- **Random Forest:** Sedikit cenderung overfitting (akurasi *training* biasanya 100% pada *tree-based*), namun akurasi *testing* tetap tinggi (94%), yang berarti model mampu melakukan generalisasi dengan baik pada data baru.

## 8. CONCLUSION

### 8.1 Kesimpulan Utama Model Terbaik:

Random Forest (Accuracy 94%).

#### Alasan:

Random Forest mampu menangani hubungan non-linear antar variabel statistik tenis jauh lebih baik daripada model linear, dan lebih stabil pada data tabular kecil dibandingkan Deep Learning.

#### Pencapaian Goals:

Goal tercapai. Target akurasi > 85% berhasil dilampaui oleh semua model (terendah 88%, tertinggi 94%).

### 8.2 Key Insights

Insight dari Data:

1. Poin Total adalah Kunci: Pemain yang memenangkan total poin lebih banyak (TPW) hampir pasti memenangkan pertandingan.
2. Kesalahan Sendiri: Tingkat Unforced Errors (UFE) yang rendah berkorelasi kuat dengan kemenangan.
3. Efisiensi Net: Poin yang didapat dari permainan net (NPW) sering menjadi pembeda dalam pertandingan ketat.

Insight dari Modeling:

1. Untuk data tabular dengan jumlah sampel  $< 1000$ , algoritma Tree-based Ensemble (Random Forest) seringkali lebih unggul daripada Deep Learning.
2. Deep Learning memerlukan data yang jauh lebih besar untuk mengungguli model ML klasik.

### 8.3 Kontribusi Proyek Manfaat praktis:

Model ini dapat digunakan oleh pelatih tenis untuk simulasi strategi: "Jika kita menekan unforced error turun 10%, berapa kenaikan probabilitas menang?".

Pembelajaran yang didapat:

Saya mempelajari bahwa kompleksitas model (Deep Learning) tidak selalu menjamin hasil terbaik. Pemilihan model harus disesuaikan dengan karakteristik dan jumlah data.

## 9. FUTURE WORK (OPSIONAL)

Saran pengembangan untuk proyek selanjutnya:

\*\* Centang Sesuai dengan saran anda \*\*

\*\*Data:\*\*

- [✓] Mengumpulkan lebih banyak data
- [ ] Menambah variasi data
- [✓] Feature engineering lebih lanjut

\*\*Model:\*\*

- [ ] Mencoba arsitektur DL yang lebih kompleks
- [✓] Hyperparameter tuning lebih ekstensif
- [✓] Ensemble methods (combining models)
- [ ] Transfer learning dengan model yang lebih besar

\*\*Deployment:\*\*

- [ ] Membuat API (Flask/FastAPI)
- [✓] Membuat web application (Streamlit/Gradio)
- [ ] Containerization dengan Docker
- [ ] Deploy ke cloud (Heroku, GCP, AWS)

\*\*Optimization:\*\*

- [ ] Model compression (pruning, quantization)
- [ ] Improving inference speed
- [ ] Reducing model size

## 10. REPRODUCIBILITY

### 10.1 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)

### 10.2 Environment & Dependencies

numpy==1.24.3 pandas==2.0.3 scikitlearn==1.3.0  
tensorflow==2.15.0 ucimlrepo matplotlib==3.7.1 seaborn==0.12.2