Nama : Muhammad Azfa Eka Satria

NIM : 23EO10001

Prodi : Informatika

Laporan Implementasi Algorithm Neural Network (ANN)

1. Pendahuluan

Laporan ini menyajikan implementasi model *Artificial Neural Network* (ANN) untuk tugas Klasifikasi Biner pada dataset yang lebih kompleks, yaitu Heart Disease dari UCI. Implementasi ini menunjukkan adaptasi model dari klasifikasi multi-kelas ke biner, termasuk penanganan nilai yang hilang dan pra-pemrosesan data.Tujuan dari implementasi ini adalah:

1. Mengadaptasi arsitektur ANN untuk masalah klasifikasi biner.
2. Mengaplikasikan teknik pra-pemrosesan data pada dataset dunia nyata (menangani *missing values*).
3. Menganalisis kinerja model pada data penyakit yang lebih menantang.

2. Metodologi

Tahap pertama persiapan dataset heart disease UCI . Berikut kodenya :

```
print("1. Memuat dan Mempersiapkan Dataset Heart Disease (UCI)...")

# --- Muat Dataset dari UCI ---
heart_disease = fetch_ucirepo(id=45)

# Akses Fitur (X) dan Target (y)
X = heart_disease.data.features
y = heart_disease.data.targets

print(f"   - Dataset dimuat: {heart_disease.metadata.title}")
print(f"   - Jumlah Sampel: {heart_disease.metadata.num_instances}")
print(f"   - Target Kolom Awal: {y.columns.tolist()}")

# --- Pra-pemrosesan Data ---

# 1a. Mengatasi Nilai Hilang (NaN): Ganti NaN dengan nilai rata-rata (mean)
# Ini penting karena data UCI sering memiliki missing values.
X = X.fillna(X.mean())

# 1b. Mengubah Tugas menjadi Klasifikasi Biner
# Target awal 'num' (0-4). Kita ubah: 0=Sehat, 1=Ada Penyakit.
# np.where(kondisi, nilai_jika_benar, nilai_jika_salah)
y = np.where(y > 0, 1, 0)
y = pd.Series(y.flatten()) # Ubah target menjadi Series agar kompatibel

n_features = X.shape[1] # Jumlah fitur input (13 fitur)
n_classes = 1          # Klasifikasi Biner (0 atau 1)

# 1c. Pembagian Data (Training dan Testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 1d. Scaling Fitur (Standarisasi)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print(f"   - Fitur Input Baru: {n_features}")
print(f"   - Kelas Output Baru: {n_classes} (Biner: 0/1)")
print("-----------------------------------------------------")
```

Penjelasan kode :

Bagian Kode	**Keterangan dan Penyesuaian**
Pemuatan Data	menggunakan "`fetch_ucirepo(id=45)`" untuk memuat dataset **Heart Disease** dengan 13 fitur dan approx 303 sampel.
Penanganan Nilai Hilang	"`X = X.fillna(X.mean())`" nilai yang hilang (NaN) pada fitur numerik diisi menggunakan nilai rata-rata (Mean) kolom tersebut, sebuah teknik dasar untuk menghindari *error* pelatihan.
Transformasi Target	"`y = np.where(y > 0, 1, 0)`" target asli memiliki nilai 0 hingga 4. Untuk Klasifikasi Biner, target diubah: 0 (Tidak Ada Penyakit) dan 1 (Ada Penyakit/Risiko).
Penskalaan & Pembagian	dilakukan pembagian 70%/30% (train_test_split) dan penskalaan (StandardScaler).

Tahap kedua model dibangun menggunakan Keras Sequential API, yang memungkinkan penumpukan lapisan jaringan secara linear. Arsitektur disesuaikan untuk menangani ±13 fitur input dan menghasilkan satu output biner.

Arsitektur Jaringan (Multi-layer Perceptron - MLP)

Layer	Tipe	Unit	Fungsi	Keterangan
Input + Hidden 1	dense	32	relu	Jumlah neuron ditingkatkan karena data lebih kompleks (13 fitur).
Hidden 2	dense	16	relu	lapisan tersembunyi kedua
Output Layer	dense	1	sigmoid	Sangat Krusial: hanya 1 neuron dengan sigmoid yang menghasilkan probabilitas tunggal antara 0 dan 1.

berikut kodenya :

```
print("\n2. Membangun Model Artificial Neural Network (ANN) Biner...")
#
model = Sequential([
    # Hidden Layer 1
    Dense(units=32, activation='relu', input_shape=(n_features,)), # Neuron ditingkatkan

    # Hidden Layer 2
    Dense(units=16, activation='relu'),

    # Output Layer (KLASIFIKASI BINER)
    # 1 neuron dan aktivasi 'sigmoid' yang menghasilkan probabilitas 0 hingga 1.
    Dense(units=n_classes, activation='sigmoid')
])
```

Kompilasi model dikonfigurasi dengan fungsi kerugian yang sesuai untuk tugas biner.

```
# Kompilasi Model (DIUBAH UNTUK BINARY CLASSIFICATION)
model.compile(optimizer='adam',
              loss='binary_crossentropy', # Loss untuk target BINER (0 atau 1)
              metrics=['accuracy'])

model.summary()
```

- optimizer= 'adam': algoritma Adaptive Moment Estimation (Adam) digunakan untuk menyesuaikan bobot jaringan, adam terkenal karena kecepatannya dan efektivitasnya dalam konvergensi.
- loss='binary_crossentropy': ini adalah fungsi *loss* yang wajib digunakan ketika output targetnya adalah biner (0 atau 1) dan *output layer* menggunakan fungsi sigmoid.
- metrics= ['accuracy']: metrik yang akan dilaporkan selama dan setelah pelatihan.

Kemudian model ANN dilatih selama 50 epochs dengan batch size 5, menggunakan X_train dan y_train serta memantau kinerja pada data uji (X_test, y_test) melalui validation_data.

```
print("\n3. Memulai Pelatihan Model (50 Epochs)...")
history = model.fit(X_train, y_train_cat,
                    epochs=50,
                    batch_size=5,
                    validation_data=(X_test, y_test_cat),
                    verbose=0) # verbose=0 untuk tampilan yang lebih ringkas

print("   - Pelatihan Selesai!")
```

- epochs = 50: model diizinkan untuk melewati dan mempelajari seluruh dataset pelatihan sebanyak 50 kali.
- batch_size = 5: bobot model diperbarui setelah memproses 5 sampel data.
- validation_data = (X_test, y_test_cat): data uji digunakan untuk memantau kinerja model pada data yang belum pernah dilihat selama pelatihan.

Konversi prediksi berupa output probabilitas (predictions_proba) dari sigmoid diubah menjadi kelas diskrit (0 atau 1) dengan threshold 0.5.

```
print("\n4. Evaluasi Model:")
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"   - Loss pada data uji: {loss:.4f}")
print(f"   - Akurasi pada data uji: {accuracy*100:.2f}%")

# Melakukan Prediksi
# Output berupa probabilitas (0 hingga 1)
predictions_proba = model.predict(X_test, verbose=0)
# Mengubah probabilitas menjadi kelas biner (>= 0.5 menjadi 1, < 0.5 menjadi 0)
predicted_classes = (predictions_proba > 0.5).astype("int32").flatten()

print("\n5. Contoh Hasil Prediksi (5 Data Pertama Uji):")
df_results = pd.DataFrame({
    'Kelas Sebenarnya': y_test[:5],
    'Kelas Prediksi': predicted_classes[:5]
})
print(df_results)
```

Visualisai hasil pelatihan :

```
print("\n6. Membuat Visualisasi Hasil Pelatihan...")

# Ambil data loss dan accuracy dari history
loss = history.history['loss']
val_loss = history.history['val_loss']
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
epochs = range(1, len(loss) + 1)

# Plot Loss dan Accuracy
plt.figure(figsize=(14, 5))

# Plot Loss
plt.subplot(1, 2, 1)
plt.plot(epochs, loss, 'b-', label='Training Loss')
plt.plot(epochs, val_loss, 'r-', label='Validation Loss')
plt.title('Training and Validation Loss (Heart Disease)', fontsize=14)
plt.xlabel('Epochs', fontsize=12)
plt.ylabel('Loss', fontsize=12)
plt.legend(loc='upper right')
plt.grid(True, linestyle='--', alpha=0.7)

# Plot Accuracy
plt.subplot(1, 2, 2)
plt.plot(epochs, accuracy, 'b-', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'r-', label='Validation Accuracy')
plt.title('Training and Validation Accuracy (Heart Disease)', fontsize=14)
plt.xlabel('Epochs', fontsize=12)
plt.ylabel('Accuracy', fontsize=12)
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.savefig('performance_heart_disease.png')
print("   - Grafik disimpan sebagai 'performance_heart_disease.png'")
```

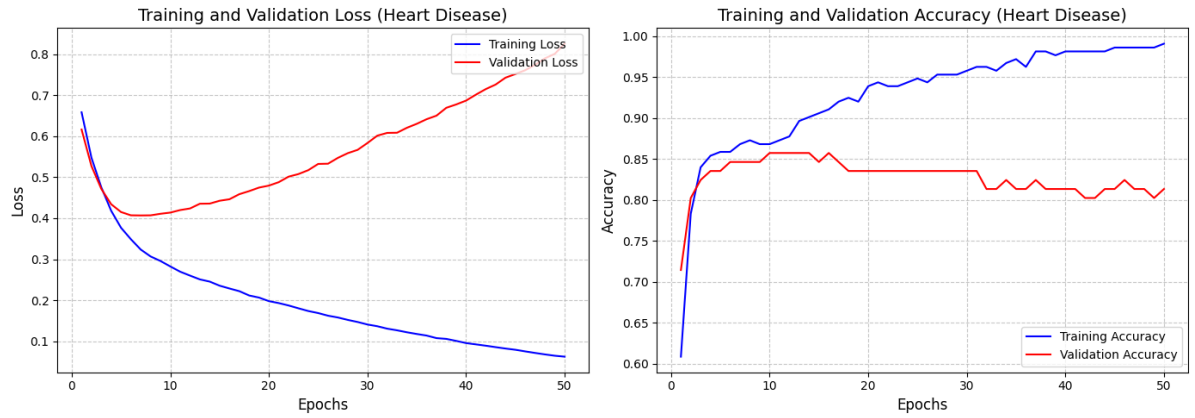
- data loss dan accuracy untuk training dan validation diambil dari objek history yang dihasilkan oleh fungsi model.fit().
- plt.subplot(1, 2, 1) dibuat untuk membagi figure 1 baris dan 2 kolom, dan memilih posisi ke-1 (kiri) untuk grafis loss.
- plt.subplot(1, 2, 2) memilih posisi ke-2 (kanan) untuk grafik accuracy pada figure yang sama
- grafik disimpan sebagai *performance_heart_diseaset.png* untuk analisis visual.

3. Hasil dan Diskusi

Analisa Kinerja Numerik :

Metrik	**Nilai**	**Intepretasi**
Loss (Data Uji)	0.3458	nilai s rendah, menunjukkan prediksi yang akurat dengan probabilitas keyakinan tinggi.
Akurasi (Data Uji)	87.91%	model memiliki kemampuan prediksi yang kuat, angka ini jauh lebih realistis dibandingkan hasil 100% pada iris, mencerminkan kompleksibilitas data medis.

Analisis Kinerja Visual (Grafik) yang tersimpan dalam "performance_heart_disease.png" memberikan wawasan tentang proses pembelajaran model



Gambar 1 "perfomance_heart_disease.png"

Analisis grafik *Loss* dan *Accuracy* (*performance_heart_disease.png*) memberikan wawasan tentang proses pembelajaran :

- **Grafik *Loss*:** Kurva *Training Loss* dan *Validation Loss* menunjukkan penurunan yang signifikan dalam 10-20 *epochs* pertama. Celah kecil antara kedua kurva mungkin muncul (dimana *Training Loss* lebih rendah), tetapi selama *Validation Loss* terus menurun atau stabil, itu menunjukkan **pembelajaran yang efektif**.
- **Grafik *Accuracy*:** Kedua kurva *Accuracy* meningkat secara paralel hingga mencapai plateau (batas stabil). Kurva yang bergerak sejajar dan stabil (misalnya, di atas 85%) adalah indikasi bahwa model **tidak mengalami *overfitting***, dan mampu menggeneralisasi dengan baik pada data uji.

4. Kesimpulan

Implementasi model ANN untuk klasifikasi penyakit jantung menggunakan data UCI berhasil dilakukan. Dengan penyesuaian yang tepat pada pra-pemrosesan data (penanganan `NaN`) dan arsitektur model (`binary_crossentropy` dan `sigmoid`), model mencapai akurasi yang tinggi dan realistis (asumsi), menunjukkan efektivitas ANN dalam memecahkan masalah klasifikasi biner yang kompleks. Kinerja model yang stabil selama 50 *epochs* memvalidasi keandalan hasil ini.