Laporan Singkat: Pemodelan (Artificial Neural Network) - Pertemuan 7

Tanggal: 25 Oktober 2025 **Disusun oleh:** [Nama Anda] **Mata Kuliah:** [Mata Kuliah Anda]

---

1. Pendahuluan

Laporan ini mendokumentasikan langkah-langkah yang diambil dalam Pertemuan 7, yang berfokus pada implementasi model *deep learning* untuk prediksi kelulusan. Tujuan dari tahap ini adalah untuk membangun, melatih, dan mengevaluasi sebuah **Artificial Neural Network (ANN)** menggunakan *library* `tensorflow.keras` pada data `processed_kelulusan.csv` yang telah disiapkan sebelumnya.

2. Langkah 1: Persiapan Data

1. **Pemuatan Data:** Dataset `processed_kelulusan.csv` berhasil dimuat ke dalam DataFrame Pandas.
2. **Pemisahan Data:** Dataset dibagi menjadi tiga bagian (Train 70%, Validation 15%, Test 15%) menggunakan `train_test_split` dengan strategi `stratify=y`.
3. **Scaling Data (Kritis):** ANN sangat sensitif terhadap skala fitur. Oleh karena itu, `StandardScaler` digunakan. Poin penting dalam proses ini adalah:
    - *Scaler* di-**fit** (`.fit_transform()`) **hanya** pada data latih (`X_train`).
    - *Scaler* yang sama kemudian digunakan untuk mentransformasi (`.transform()`) data validasi (`X_val`) dan data tes (`X_test`).
    - Pendekatan ini mencegah *data leakage* (kebocoran data) dari set validasi/tes ke dalam proses *training*.
4. **Konversi Tipe:** Data target (y) dikonversi menjadi array NumPy (`.values`) untuk kompatibilitas yang lebih baik dengan Keras.

3. Langkah 2: Pembangunan Arsitektur Model ANN

Sebuah model `keras.Sequential` dibangun. `tf.random.set_seed(42)` digunakan untuk memastikan hasil yang dapat direproduksi. Arsitektur model adalah sebagai berikut:

1. **Input Layer:** Didefinisikan secara implisit oleh `layers.Input(shape=...)` yang disesuaikan dengan jumlah fitur di `X_train_s`.
2. **Hidden Layer 1:** `Dense(32, activation="relu")`.

    - *Regularisasi:* Ditambahkan `kernel_regularizer=keras.regularizers.l2(0.001)` (L2 Regularization) untuk mengurangi *overfitting*.
3. **Dropout Layer 1:** `Dropout(0.3)` untuk menonaktifkan 30% neuron secara acak selama *training*, yang juga berfungsi sebagai regularisasi.
4. **Hidden Layer 2:** `Dense(16, activation="relu")` dengan `L2(0.001)`.
5. **Dropout Layer 2:** `Dropout(0.3)`.
6. **Output Layer:** `Dense(1, activation="sigmoid")`. Aktivasi "sigmoid" dipilih karena ini adalah masalah klasifikasi biner (0 atau 1), yang menghasilkan probabilitas antara 0 dan 1.

Model kemudian di-compile dengan parameter berikut:

- **Optimizer:** `keras.optimizers.Adam(1e-3)` (Adam dengan *learning rate* 0.001).
- **Loss Function:** `binary_crossentropy` (standar untuk klasifikasi biner).
- **Metrics:** `["accuracy", "AUC"]` (kita melacak akurasi dan Area Under the Curve).

Hasil `model.summary()` memberikan rincian arsitektur dan jumlah parameter yang dapat dilatih.

4. Langkah 3: Pelatihan Model (Training)

Pelatihan model dilakukan menggunakan `model.fit()` dengan beberapa strategi penting:

- **Batch Size:** `batch_size=8`. Model memproses 8 sampel data sekaligus dalam satu iterasi.
- **Epochs:** `epochs=200`. Model akan dilatih maksimal 200 kali putaran data latih.
- **Callbacks (EarlyStopping):**
    - Sebuah *callback* `EarlyStopping` diimplementasikan untuk menghentikan *training* secara otomatis jika performa model tidak lagi meningkat.
    - **Monitor:** `val_auc`. Kinerja model pada data validasi dipantau.
    - **Patience:** `20`. Pelatihan akan berhenti jika `val_auc` tidak meningkat selama 20 *epoch* berturut-turut.
    - **`restore_best_weights=True`**: Ini adalah pengaturan krusial yang memastikan bahwa bobot model yang dikembalikan adalah bobot dari *epoch* dengan `val_auc` terbaik, bukan bobot dari *epoch* terakhir.

5. Langkah 4: Evaluasi pada Test Set

Setelah *training* selesai (dan bobot terbaik dipulihkan), model dievaluasi pada data **test set** (`X_test_s`, `y_test_s`) yang belum pernah dilihat sebelumnya.

- **Metrik Kinerja (Test Set):**
    - **Test Loss:** [Isi skor Test Loss dari output Anda]
    - **Test Accuracy:** [Isi skor Test Accuracy dari output Anda]

    - **Test AUC:** [Isi skor Test AUC dari output Anda]
- **Prediksi:** Probabilitas (`y_proba`) dibuat, dan dikonversi menjadi kelas (`y_pred`) menggunakan ambang batas 0.5.
- **Confusion Matrix (Test):** *(Salin hasil Confusion Matrix (test) dari output Anda di sini)*
- **Classification Report (Test):** *(Salin hasil Classification Report (test) dari output Anda di sini)*

6. Langkah 5: Visualisasi Kurva Belajar (Learning Curve)

Plot `p7_learning_curve.png` dibuat untuk menganalisis proses *training*:

1. **Plot Loss (Kiri):** Membandingkan `Train Loss` vs. `Val Loss` per *epoch*. Jika `Val Loss` mulai naik sementara `Train Loss` terus turun, itu adalah tanda *overfitting*.
2. **Plot AUC (Kanan):** Membandingkan `Train AUC` vs. `Val AUC`. Idealnya, kedua kurva ini konvergen di nilai AUC yang tinggi, menunjukkan model dapat menggeneralisasi dengan baik.