

## **INFORMASI PROYEK**

Judul Proyek: Analisis Autism Spectrum Disorder (ASD) pada Anak menggunakan Model Baseline, Machine Learning, dan Deep Learning

Nama Mahasiswa: Arfan Bagus Dharmawan

NIM: 234311033

Program Studi: Teknologi Rekayasa Perangkat Lunak

Mata Kuliah: Data Science

Dosen Pengampu: Gus Nanang Syaifuddiin

Tahun Akademik: 2025/Semester 5

Link GitHub Repository:

[https://github.com/Arfanbagus/UASDS\\_234311033\\_Arfan.git](https://github.com/Arfanbagus/UASDS_234311033_Arfan.git)

Link Video Pembahasan: <https://youtu.be/WuLm3Hctk4A>

## **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):
  - Model Baseline
  - Model Machine Learning / advanced
  - 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**

Gangguan Spektrum Autisme (ASD) merupakan gangguan perkembangan neurologis yang secara signifikan memengaruhi kemampuan komunikasi, interaksi sosial, dan perilaku anak. Deteksi dini memegang peranan krusial dalam penanganan kondisi ini, karena intervensi yang dilakukan sedini mungkin dapat meningkatkan kualitas hidup dan perkembangan anak secara substansial. Namun, metode diagnosis klinis konvensional seringkali menghadapi kendala seperti waktu tunggu yang lama, biaya yang tinggi, dan ketergantungan pada penilaian subjektif melalui kuesioner manual yang kompleks, sehingga sering menyebabkan keterlambatan dalam diagnosis dan penanganan.(Aprilia et al., 2014) (Veryawan et al., 2023)

Untuk mengatasi tantangan tersebut, penerapan teknologi kecerdasan buatan melalui pemanfaatan data historis skrining perilaku menawarkan solusi yang menjanjikan untuk efisiensi deteksi dini. Proyek ini bertujuan untuk menganalisis dan membangun sistem klasifikasi otomatis dengan membandingkan performa dari tiga pendekatan berbeda, yaitu *Model Baseline* (Logistic Regression), *Machine Learning* lanjutan (Random Forest), dan *Deep Learning* (Multilayer Perceptron). Melalui perbandingan ini, diharapkan dapat ditemukan model prediktif yang paling akurat dan efisien untuk membantu tenaga medis serta orang tua dalam mengidentifikasi risiko ASD pada anak secara lebih cepat dan objektif.

### **Daftar Referensi (APA)**

Aprilia, D., Johar, A., & Hartuti, P. (2014). *SISTEM PAKAR DIAGNOSA AUTISME PADA ANAK*. 2(2).

Veryawan, A. S. I. L., Sri Inda Lestari, Indah, & Veryawan. (2023). PERILAKU ANAK

AUTIS: PERKEMBANGAN DAN PENANGAN. *Indonesian Journal of Early*

*Childhood: Jurnal Dunia Anak Usia Dini*, 5(1), 150–155.

<https://doi.org/10.35473/ijec.v5i1.1980>

### **3. BUSSINESS UNDERSTANDING / PROBLEM UNDERSTANDING**

#### **3.1 Problem Statements**

- Dibutuhkan model *machine learning* yang mampu memprediksi risiko **Autism Spectrum Disorder (ASD)** pada anak dengan akurasi tinggi berdasarkan fitur demografis, riwayat medis keluarga, dan skor perilaku dari kuesioner skrining.
- Dataset memiliki campuran fitur kategorikal (seperti etnis, gender, riwayat penyakit) dan numerik (usia, skor tes) sehingga diperlukan proses *preprocessing* yang tepat, termasuk *encoding*, *scaling*, dan *feature engineering* agar model dapat belajar secara optimal dari data medis tersebut.
- Belum diketahui model *baseline*, *advanced machine learning*, atau *deep learning* yang memberikan performa terbaik dan paling stabil dalam mengklasifikasikan diagnosis ASD (Yes/No) pada data tabular yang tersedia.
- Diperlukan model *deep learning* yang mampu mempelajari pola *nonlinear* yang kompleks dari interaksi antara jawaban kuesioner perilaku dan faktor karakteristik anak dalam menentukan indikasi autisme.

#### **3.2 Goals**

- Membangun model klasifikasi untuk memprediksi label Class/ASD (YES/NO).
- Membandingkan performa tiga algoritma: Logistic Regression, Random Forest, dan Deep Learning (MLP).
- Mendapatkan model dengan akurasi dan F1-Score yang optimal untuk data medis yang tersedia.

#### **3.3 Solution Approach**

##### **Model 1 – Baseline Model (Logistic Regression)**

Alasan Pemilihan:

Dikarenakan data saya relative kecil dengan jumlah data 290 baris dan model ini mudah diinterpretasikan

##### **Model 2 – Advanced / MI Model (Random Forest Classifier)**

Alasan Pemilihan:

karena kemampuan *ensemble*-nya yang baik dalam menangani *overfitting* dan kemampuannya menangkap hubungan non-linear antar fitur (seperti hubungan skor A1-A10 dengan hasil diagnosis).

##### **Model 3 – Deep Learning Model (WAJIB)**

Alasan Pemilihan:

Karena MLP yang saya pilih cocok untuk data tabular dan model ini mencoba menangkap pola kompleks yang mungkin terlewatkan oleh model konvensional, meskipun dataset relatif kecil.

## 4. DATA UNDERSTANDING

### 4.1 Informasi Dataset

Sumber Dataset:

<https://archive.ics.uci.edu/dataset/419/autistic+spectrum+disorder+screening+data+for+children>

Deskripsi Dataset:

- Jumlah Baris: 292 baris
- Jumlah kolom: 21 kolom
- Tipe Data: Tabular (Campur numerik dan kategorikal)
- Ukuran Dataset: 24.7 KB

### 4.2 Deskripsi Fitur

Jumlah fitur dalam dataset saya 20, akan tetapi saya hanya mengambil 2 fitur yang relevan yaitu age dan result

Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
Age	Float64	Age adalah usia individu	4-11 years
Result	Float64	Result adalah jumlah orang yang mengisi kuesioner	5.0, 4.0, 10.0

### 4.3 Kondisi Data

Berdasarkan hasil pemeriksaan dataset, berikut adalah kondisi data pada fitur yang digunakan:

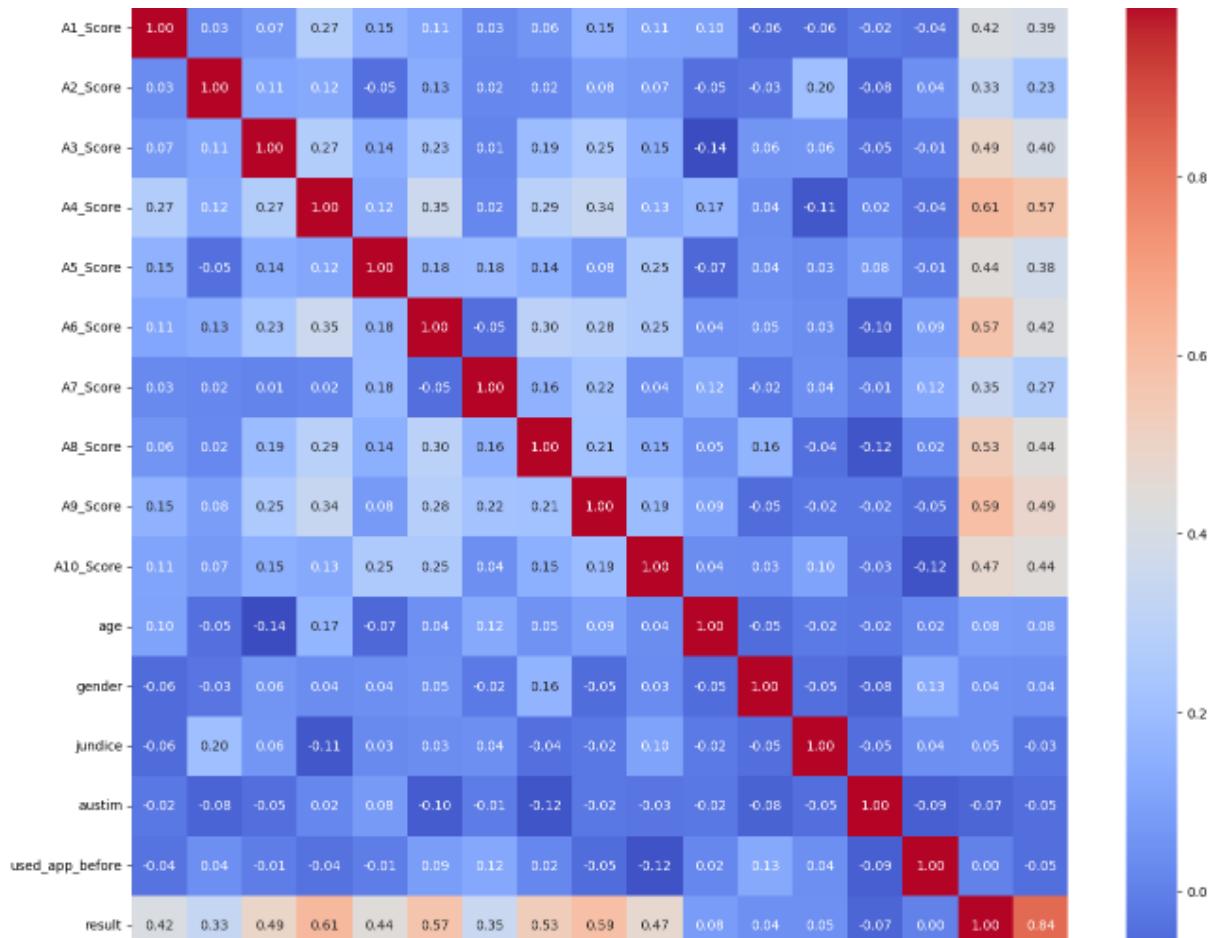
- o Missing Values: 4
- o Duplicate Data: 2
- o Outliers: Terdeteksi pada kolom age dan result (berdasarkan visualisasi Boxplot).
- o Imbalanced Data: Distribusi kelas cukup seimbang (NO: 151, YES: 141).
- o Data Quality Issues: Beberapa kolom kategorikal memiliki format string yang tidak bersih (contoh: b'YES'), perlu *cleaning*.

### 4.4 Exploratory Data Analysis (EDA) – (OPSIONAL)

Requirement: Minimal 3 visualisasi yang bermakna dan insight-nya. Contoh jenis Visualisasi yang dapat digunakan:

- o Histogram (Distribusi data)
- o Boxplot (Deteksi Outliers)
- o Heatmap Korelasi (Hubungan antar fitur)

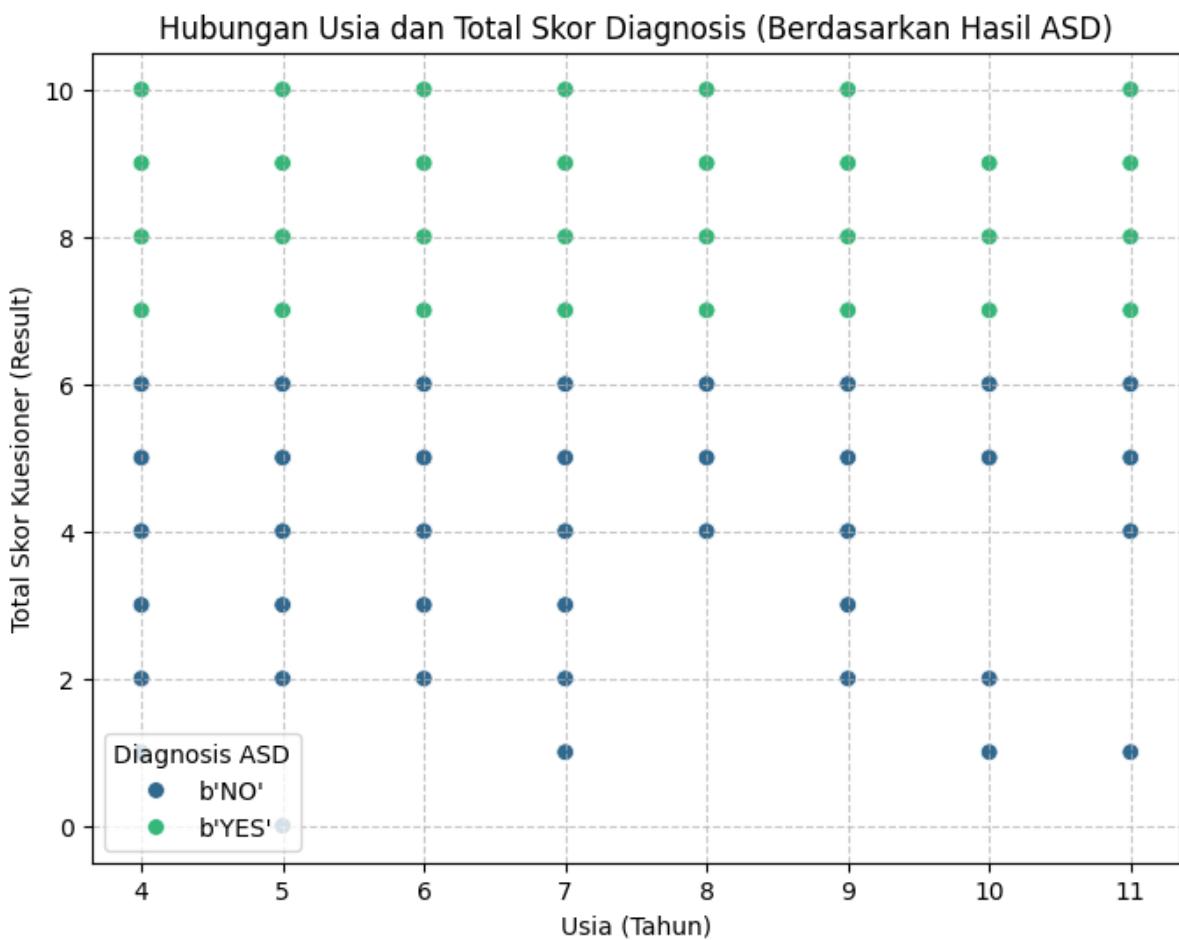
**Visualisasi 1: Heatmap**



Insight:

Skor A1–A10 (terutama A4, A6, A9) adalah indikator utama dalam menentukan result. Ini menunjukkan bahwa hasil sangat dipengaruhi oleh skor penilaian/pertanyaan, bukan oleh faktor demografis.

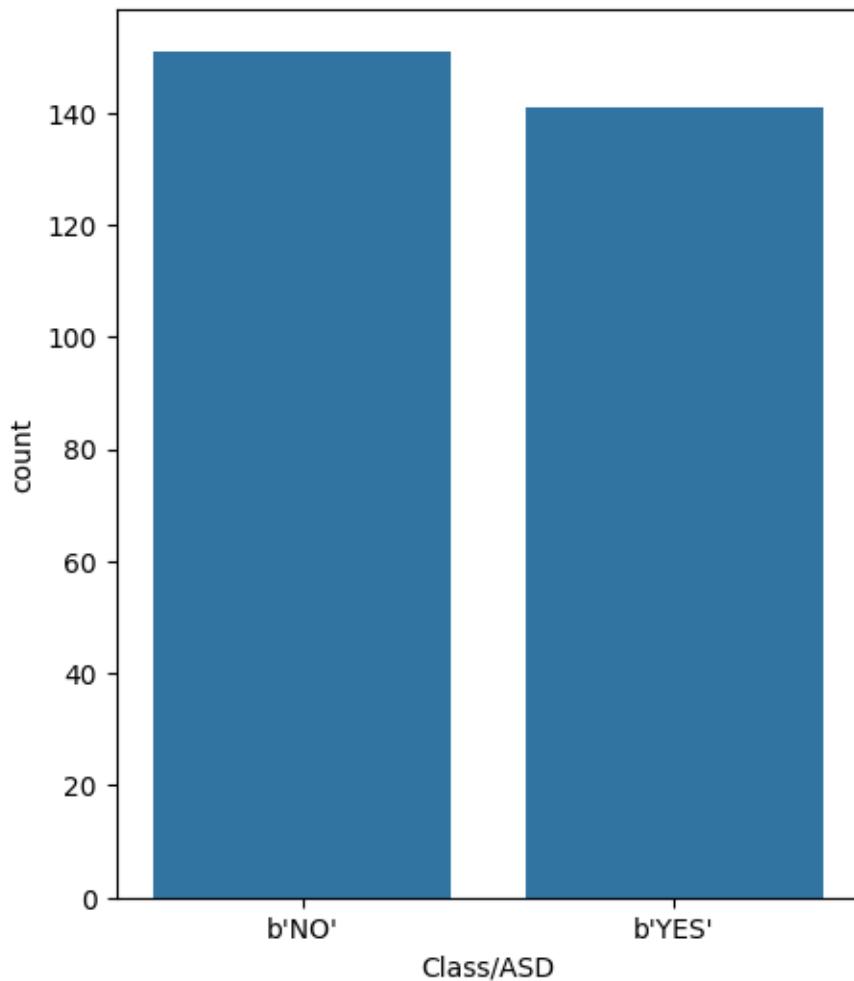
## Visualisasi 2: Scatter Plot



Insight:

Grafik menunjukkan bahwa usia tidak memiliki hubungan yang signifikan dengan total skor diagnosis ASD. Skor tinggi secara konsisten berkorelasi dengan hasil diagnosis ASD positif pada semua kelompok usia, menegaskan bahwa instrumen penilaian bersifat robust dan independen terhadap faktor usia.

### Visualisasi 3: BloxPlot



Insight:

Grafik distribusi kelas menunjukkan bahwa jumlah data pada kelas ASD “YES” dan “NO” relatif seimbang, sehingga dataset layak digunakan untuk pemodelan klasifikasi tanpa perlakuan khusus terhadap ketidakseimbangan kelas.

## 5. DATA PREPARATION

Bagian ini Menjelaskan semua proses transformasi dan preprocessing yang dilakukan

### 5.1 Data Cleaning

Aktivitas:

```
# check missing values
missing_count = autism.isna().sum()
print("Jumlah Missing Values: ",missing_count)

# check data duplikat
duplicate_count = autism.duplicated().sum()
print("Jumlah duplikasi ditemukan:", duplicate_count)
```

```

# Load data
df = pd.read_csv('/content/Autism-Child-Data-Exact.csv')

# 1. Bersihkan missing values pada kolom 'age'
df['age'] = df['age'].fillna(df['age'].median())

# 2. Hapus data duplikat
df = df.drop_duplicates()

# Cek hasil
print("Jumlah data setelah cleaning:", df.shape)
print(df.isna().sum())

```

### 1. Handling Missing Values:

Langkah yang pertama yaitu check nilai yang hilang pada data dengan menggunakan perintah isna().sum(). Disini terdapat nilai yang hilang pada data saya sebanyak 4 yaitu di kolom age dan saya bersihkan menggunakan perintah .fillna(df ['age'].median())

### 2. Removing duplicates

Setelah itu dilakukan pengecekan duplikasi pada data dengan perintah .duplicated().sum(). Disini di data saya ada duplikasi sebanyak 2 dan disini saya membersihkan data yang terduplikasi dengan perintah .drop\_duplicates()

### 3. Data Type Conversion:

Dilakukan pengecekan tipe data untuk memastikan setiap kolom memiliki tipe yang sesuai, seperti:

- Kolom Numerik: Age, result, A1\_score – A10\_Score
- Kolom Kategorikal: gender, Etnicity, jaundice, austim, country\_of\_res, age\_desc, relation, Class/ASD

## 5.2 Feature Engineering

Aktivitas:

```

# Feature Engineering: Age Group
df['age_group'] = pd.cut(
    df['age'],
    bins=[0, 5, 10, 15, 20],
    labels=['Balita', 'Anak', 'Remaja Awal', 'Remaja']
)

# Feature Engineering: Total Skor (jika ingin eksplisit)
score_cols = [f"A{i}_Score" for i in range(1, 11)]
df['total_score'] = df[score_cols].sum(axis=1)

# Feature Engineering: Flag Risiko Tinggi
df['high_risk'] = df['total_score'].apply(lambda x: 1 if x >= 7 else 0)

print(df[['age', 'age_group', 'total_score', 'high_risk']].head())

```

### 1. Disini saya melakukan penambahan fitur pada:

- Age\_group:  
Fitur Kategorikal hasil pengelompokan usia (Balita, Anak, Remaja Awal, Remaja)
- Total\_score:  
Fitur numerik hasil penjumlahan skor A1\_score sampai A10\_score
- High\_risk:  
Fitur biner(0/1) yang menandai risiko tinggi berdasarkan total\_score >\_7

### 5.3 Data Transformation

Kegunaan data transformation yaitu untuk data tabular:

Di tahap ini saya melakukan transformasi data agar data siap dipakai dengan menggunakan Encoding pada data saya

```
#@title data transformation

numeric_cols = ['total_score']

categorical_cols = [
    'age_group',
    'gender',
    'ethnicity',
    'jundice',
    'austim',
    'contry_of_res',
    'used_app_before',
    'relation'
]

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_cols)
    ]
)

X_train_transformed = preprocessor.fit_transform(X_train)
X_val_transformed = preprocessor.transform(X_val)
X_test_transformed = preprocessor.transform(X_test)

print("Train:", X_train_transformed.shape)
print("Val : ", X_val_transformed.shape)
print("Test : ", X_test_transformed.shape)
```

Transformasi data yang saya lakukan dengan memisahkan fitur numerik dan kategorikal. Fitur numerik distandarisasi menggunakan StandardScaler untuk menyamakan skala data, sedangkan fitur kategorikal diubah menjadi bentuk

numerik menggunakan One-Hot Encoding. Proses ini dilakukan menggunakan ColumnTransformer agar transformasi dapat diterapkan secara konsisten pada data latih, validasi, dan uji, sehingga data siap digunakan oleh model machine learning.

## 5.4 Data Splitting

Pada data splitting saya membagi data (training 70% dan temp 30%) dan (Validation 15% dan Test 15%)

```
#@title Split data

# Pisahkan fitur dan target
X = df.drop(columns=['Class/ASD'])
y = df['Class/ASD']

# SPLIT 1: Train (70%) dan Temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y,
    test_size=0.30,
    random_state=42,
    stratify=y
)

# SPLIT 2: Validation (15%) dan Test (15%)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp,
    test_size=0.50,
    random_state=42,
    stratify=y_temp
)

# Cek hasil pembagian
print("Total data      :", df.shape)
print("Training set    :", X_train.shape)
print("Validation set  :", X_val.shape)
print("Test set        :", X_test.shape)

print("\nProporsi label:")
print("Train:\n", y_train.value_counts(normalize=True))
print("Validation:\n", y_val.value_counts(normalize=True))
print("Test:\n", y_test.value_counts(normalize=True))
```

Pembagian ini bertujuan untuk melatih model pada data latih dan mengevaluasi performa model pada data uji yang belum pernah dilihat sebelumnya, sehingga dapat mengukur kemampuan generalisasi model serta mencegah overfitting.

## **5.5 Data Balancing**

Dikarenakan data saya sudah balance secara alami maka tidak diperlukannya balancing pada data

## **5.6 Ringkasan Data Preparation**

### **1. Data Cleaning**

Apa yang dilakukan:

- Check Missing Values, Duplicate Data, dan tipe data
- Mengatasi data jika terdapat missing values, duplikasi pada data yang mengalami missing values

Mengapa Penting:

- Memastikan kualitas data tetap terjaga baik sebelum dimasukkan ke model.
- Menghindari error saat training

### **2. Feature Engineering**

Apa yang dilakukan:

Feature engineering berfungsi untuk menambah dan membuat fitur baru

Mengapa Penting:

Penambahan fitur dapat membantu model untuk memahami pola yang tidak terlihat

### **3. Data Transformation**

Apa yang dilakukan:

Digunakan untuk membuat fitur kategorikal menjadi angka

Mengapa Penting:

Karena di model Deep Learning tidak bisa membaca data dengan adanya data transformation kita perlu mengubah kategorikal menjadi angka agar dapat terbaca

### **4. Data Splitting**

Apa yang dilakukan:

Membagi dataset menjadi training dan set dengan 70% untuk training dan 15% untuk set

Mengapa Penting:

Untuk menjaga distribusi kelas agar tetap seimbang dan memastikan evaluasi model jadi akurat

## 6. MODELING

### 6.1 Model 1 – Baseline Model (Logistic Regression)

Deskripsi Model:

Logistic Regression adalah algoritma machine learning yang digunakan untuk masalah klasifikasi, terutama klasifikasi biner. Model ini bekerja dengan menghitung probabilitas suatu data termasuk ke dalam kelas tertentu menggunakan fungsi logistik (sigmoid). Hasil probabilitas tersebut kemudian digunakan untuk menentukan label kelas.

Alasan Pemilihan:

Karena model ini sederhana dan mudah diimplementasikan yang dimana memberikan hasil awal sebagai pembanding dengan model yang lebih kompleks

Hyperparameter:

1. Max\_iter = 1000, fungsinya untuk ngepastiin model memiliki iterasi yang cukup

Implementasi:

```
# model Logistic Regression

# Train Model (Baseline)
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_transformed, y_train)

# prediksi Validation Set
y_val_pred = model.predict(X_val_transformed)

# Evaluasi Metrik
print("Accuracy : ", accuracy_score(y_val, y_val_pred))
print("\nClassification Report (Ringkas) :")
print(classification_report(y_val, y_val_pred))

# Confusion Matrix
cm = confusion_matrix(y_val, y_val_pred)

plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['NO', 'YES'],
            yticklabels=['NO', 'YES'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Baseline Logistic Regression')
plt.show()
```

Hasil:

Accuracy : 0.5348837209302325

Classification Report (Ringkas) :

	precision	recall	f1-score	support
b'NO'	0.54	0.59	0.57	22
b'YES'	0.53	0.48	0.50	21
accuracy			0.53	43
macro avg	0.53	0.53	0.53	43
weighted avg		0.53	0.53	43

## 6.2 Model 2 – MI / Advanced Model (Random Forest Classifier)

Deskripsi Model:

Random Forest adalah algoritma machine learning berbasis **ensemble** yang menggabungkan banyak **decision tree** untuk menghasilkan prediksi yang lebih akurat. Setiap pohon keputusan dilatih menggunakan subset data dan fitur yang berbeda, kemudian hasil prediksi digabungkan melalui proses voting untuk menentukan kelas akhir.

Alasan Pemilihan:

karena mampu menangkap pola non-linear pada data dengan menggabungkan banyak pohon keputusan. Model ini memiliki performa yang lebih stabil dan umumnya lebih akurat dibandingkan model baseline, sehingga cocok digunakan sebagai pembanding terhadap model deep learning.

Hyperparameter:

- N\_estimators = 100, jumlah pohon
- Max\_depth = 10, kedalaman maksimum pada setiap pohon
- Min\_samples\_split = 5, minimal data untuk split
- Random\_state = 42, memastikan hasil pelatihan

Implementasi:

```
# Model Random Forest

# 1. Train Model (Random Forest)
rf_model = RandomForestClassifier(
    n_estimators=100,          # jumlah pohon (cukup, tidak berlebihan)
    max_depth=10,             # batasi kedalaman → cegah overfitting
    min_samples_split=5,      # minimal data untuk split
    random_state=42
)

rf_model.fit(X_train_transformed, y_train)

# 2. Prediksi Validation Set
y_val_pred_rf = rf_model.predict(X_val_transformed)
# 3. Evaluasi Metrik Penting
print("Accuracy :", accuracy_score(y_val, y_val_pred_rf))
print("\nClassification Report (Ringkas):")
print(classification_report(y_val, y_val_pred_rf))
```

```

# 4. Confusion Matrix
cm = confusion_matrix(y_val, y_val_pred_rf)

plt.figure(figsize=(5, 4))
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=['NO', 'YES'],
    yticklabels=['NO', 'YES']
)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest')
plt.show()

```

Hasil:

Accuracy : 0.6046511627906976

Classification Report (Ringkas):				
	precision	recall	f1-score	support
b'NO'	0.60	0.68	0.64	22
b'YES'	0.61	0.52	0.56	21
accuracy			0.60	43
macro avg	0.61	0.60	0.60	43
weighted avg	0.61	0.60	0.60	43

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

Deskripsi Model:

Nama Model: MLP

\*\* (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 saya berbentuk tabular, sehingga arsitektur ini yang saya pilih, MLP juga mampu mempelajari hubungan non-linear antar fitur gaya hidup

Arsitektur Model:

Deskripsi Layer:

Layer	Detail
Input	Shape=input_dim
Dense layer 1	128, activation = 'relu'
Dropout layer 1	0.3 = rate
Dense layer 2	64, activation = 'relu'
Dropout layer 2	0.3 = rate
Dense layer 3	1, activation = 'sigmoid'

Input & Preprocessing Khusus:

Input Shape:

- Input\_dim = artinya jumlah neuron input itu sama dengan jumlah fitur hasil transformasi data, fitunya berasal dari 1 fitur numerik yaitu total\_score

Preprocessing Khusus:

- Konversi Sparse ke Dense = Karena One-hot Encoding itu menghasilkan sparse matrix sedangkan Model MLP membutuhkan dense array, oleh karena itu data dikonversi ke array biasa
- Konversi tipe data ke Float32 = agar kompatibel dan efisien saat training di tensorflow
- Encoding Label Target = label kategorikalnya diubah menjadi 0 dan 1
- Standarisasi fitur numerik = untuk menjaga skala fitur agar MLP lebih stabil
- One-hot Encoding = untuk mencegah bias urutan kategori

Hyperparameter

Training Configuration:

- Optimizer: Adam
- Learning rate: 0.001 (default adam)
- Loss function: binary\_crossentropy
- Metrics: accuracy
- Batch size: 32
- Epochs: 10 epoch
- Validation Split: 0.5
- Callback:tidak

Implementasi:

```
from tensorflow import keras
```

```
from tensorflow.keras import layers
# 1. PERSIAPAN DATA (WAJIB)

# X: sparse -> dense
X_train_processed = X_train_transformed.toarray().astype('float32')
X_val_processed    = X_val_transformed.toarray().astype('float32')

# y: category -> 0 / 1
y_train_processed = y_train.cat.codes.astype('int32')
y_val_processed   = y_val.cat.codes.astype('int32')

# Input dimension
input_dim = X_train_processed.shape[1]

# 2. EARLY STOPPING
early_stopping = keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

# 3. MODEL MLP
model_deep_learning = keras.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

# COMPILE MODEL
model_deep_learning.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy', keras.metrics.AUC(name='auc')]
)

# TRAINING
history = model_deep_learning.fit(
    X_train_processed, y_train_processed,
    epochs=10,
    batch_size=32,
    validation_data=(X_val_processed, y_val_processed),
    callbacks=[early_stopping],
    verbose=1
)
```

```

# CONFUSION MATRIX
y_val_pred = (model_deep_learning.predict(X_val_processed) >
0.5).astype(int)

cm = confusion_matrix(y_val_processed, y_val_pred);

plt.figure(figsize=(4, 3))
sns.heatmap(
    cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=['NO ASD', 'ASD'],
    yticklabels=['NO ASD', 'ASD']
)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Deep Learning')
plt.show()

```

### Training Process:

#### Training Time:

Waktu training model deep learning rata-rata sekitar 2 detik per epoch, sehingga total waktu training untuk 10 epoch adalah sekitar 20 detik atau 0,33 menit.

#### Computational Resource:

- Platform = Google Collab
- Hardware = CPU
- GPU = tidak
- Library = Tensorflow/keras

#### Training History Visualization:

```

Epoch 1/10
7/7 [██████████] 9s 529ms/step - accuracy: 0.5122 -
auc: 0.5084 - loss: 0.7081 - val_accuracy: 0.4651 - val_auc: 0.5195 -
val_loss: 0.6926
Epoch 2/10 [██████████] 0s 63ms/step - accuracy: 0.5279 -
auc: 0.5343 - loss: 0.6966 - val_accuracy: 0.5116 - val_auc: 0.5519 -
val_loss: 0.6885
Epoch 3/10 [██████████] 0s 65ms/step - accuracy: 0.6088 -
auc: 0.5913 - loss: 0.6825 - val_accuracy: 0.5581 - val_auc: 0.5584 -
val_loss: 0.6863
Epoch 4/10 [██████████] 1s 70ms/step - accuracy: 0.6289 -
auc: 0.6619 - loss: 0.6600 - val_accuracy: 0.5116 - val_auc: 0.5952 -
val_loss: 0.6846
Epoch 5/10 [██████████]

```

```

7/7 ━━━━━━━━━━ 1s 77ms/step - accuracy: 0.6362 -
auc: 0.6894 - loss: 0.6528 - val_accuracy: 0.5116 - val_auc: 0.5996 -
val_loss: 0.6841
Epoch 6/10

7/7 ━━━━━━━━━━ 0s 54ms/step - accuracy: 0.6095 -
auc: 0.6472 - loss: 0.6629 - val_accuracy: 0.5814 - val_auc: 0.6017 -
val_loss: 0.6813
Epoch 7/10

7/7 ━━━━━━━━━━ 1s 79ms/step - accuracy: 0.6189 -
auc: 0.6305 - loss: 0.6642 - val_accuracy: 0.6047 - val_auc: 0.6104 -
val_loss: 0.6793
Epoch 8/10

7/7 ━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5712 -
auc: 0.6368 - loss: 0.6609 - val_accuracy: 0.6047 - val_auc: 0.6212 -
val_loss: 0.6780
Epoch 9/10

7/7 ━━━━━━━━━━ 0s 49ms/step - accuracy: 0.6799 -
auc: 0.7518 - loss: 0.6137 - val_accuracy: 0.6279 - val_auc: 0.6245 -
val_loss: 0.6774
Epoch 10/10

7/7 ━━━━━━━━━━ 0s 56ms/step - accuracy: 0.6765 -
auc: 0.7497 - loss: 0.6172 - val_accuracy: 0.6279 - val_auc: 0.6180 -
val_loss: 0.6780
2/2 ━━━━━━━━━━ 0s 207ms/step

```

### Analisis Training:

- Apakah model mengalami Overfitting? tidak
- Apakah model sudah converge? Mendekati tetapi belum sepenuhnya optimal karena tidak ada lonjakan besar pada performa setelah epoch ke 8
- Apakah Perlu lebih banyak epoch? Ya, akan tetapi sedikit tambahan epoch karena validation loss masih berfluktuasi ringan

### Model Summary:

Model: "sequential_2"		
Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 128)	8,832
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: 51,461 (201.02 KB)  
Trainable params: 17,153 (67.00 KB)  
Non-trainable params: 0 (0.00 B)

## 7. EVALUATION

Metrik Evaluasi

Klasifikasi :

- Accuracy  
Accuracy menunjukkan persentase prediksi yang benar dari seluruh data uji. Nilai accuracy yang tinggi menandakan bahwa model secara umum mampu melakukan prediksi dengan baik. Namun, accuracy saja belum cukup jika data memiliki distribusi kelas yang tidak seimbang.
- Precision  
Precision mengukur ketepatan prediksi kelas positif, yaitu seberapa banyak prediksi positif yang benar. Metrik ini penting untuk mengetahui apakah model sering salah dalam memprediksi suatu kelas.
- Recall  
Recall mengukur kemampuan model dalam menemukan seluruh data positif. Nilai recall yang rendah menunjukkan bahwa masih banyak data positif yang tidak berhasil terdeteksi oleh model.
- F1-Score  
F1-Score merupakan rata-rata harmonik dari precision dan recall. Metrik ini digunakan untuk memberikan gambaran performa model yang lebih seimbang, terutama ketika precision dan recall memiliki nilai yang berbeda.
- ROC-AUC  
ROC-AUC digunakan untuk mengukur kemampuan model dalam membedakan dua kelas, yaitu kelas *NO ASD* dan *ASD*. Metrik ini dihitung berdasarkan probabilitas prediksi yang dihasilkan oleh model, bukan hanya label prediksi akhirnya.  
Pada implementasi ini, nilai ROC-AUC dihitung menggunakan probabilitas kelas positif (*YES*) yang diperoleh dari fungsi `predict_proba()`.
- Confusion Matrix  
Confusion matrix digunakan untuk melihat perbandingan antara hasil prediksi dan data sebenarnya. Dari confusion matrix dapat diketahui jenis kesalahan prediksi yang sering terjadi dan seberapa baik model membedakan setiap kelas.

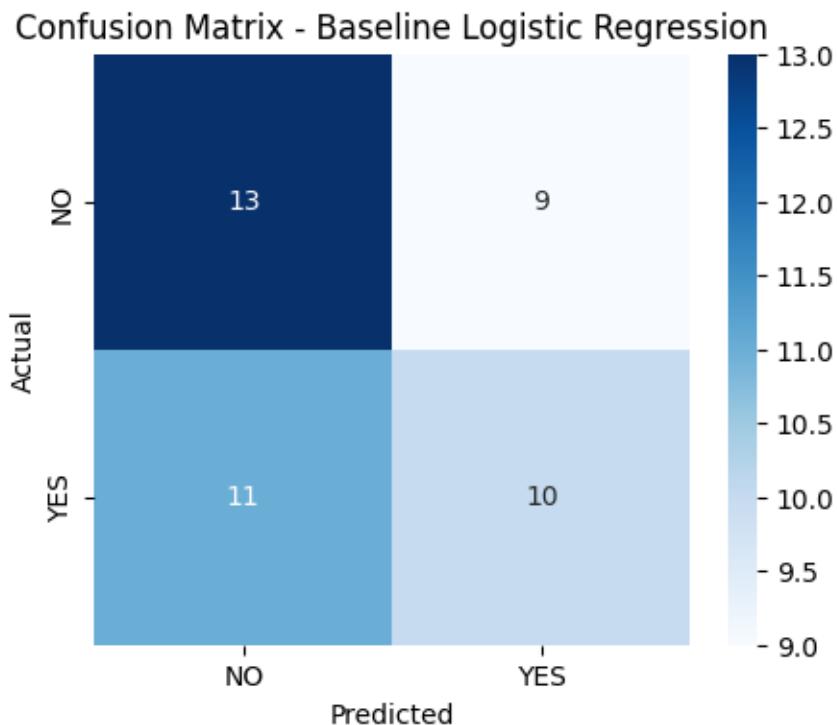
Hasil Evaluasi Model

Model 1 – Logistic Regression

Metrik:

- ROC-AUC : 0.5269151138716356
- Accuracy : 0.52
- Precision : 0.55
- Recall : 0.52
- F1\_Score : 0.53

Confusion Matrix / Visualization:

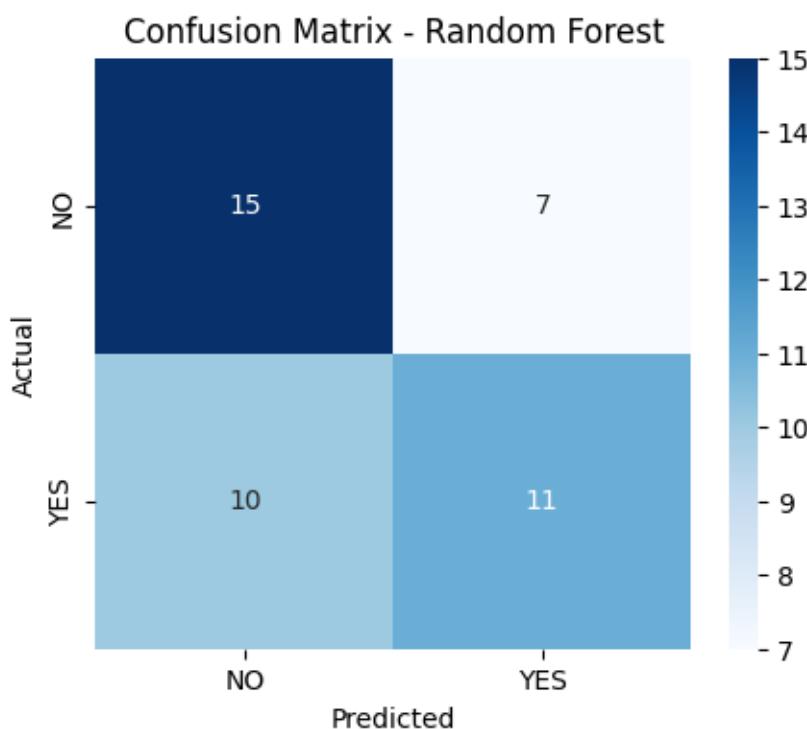


Model 2 - ML / Advanced (Random Forest Classifier)

Metrik:

- ROC-AUC : 0.5757575757575758
- Accuracy : 0.60
- Precision : 0.60
- Recall : 0.68
- F1\_Score : 0.64

Confusion Matrix / Visualization:



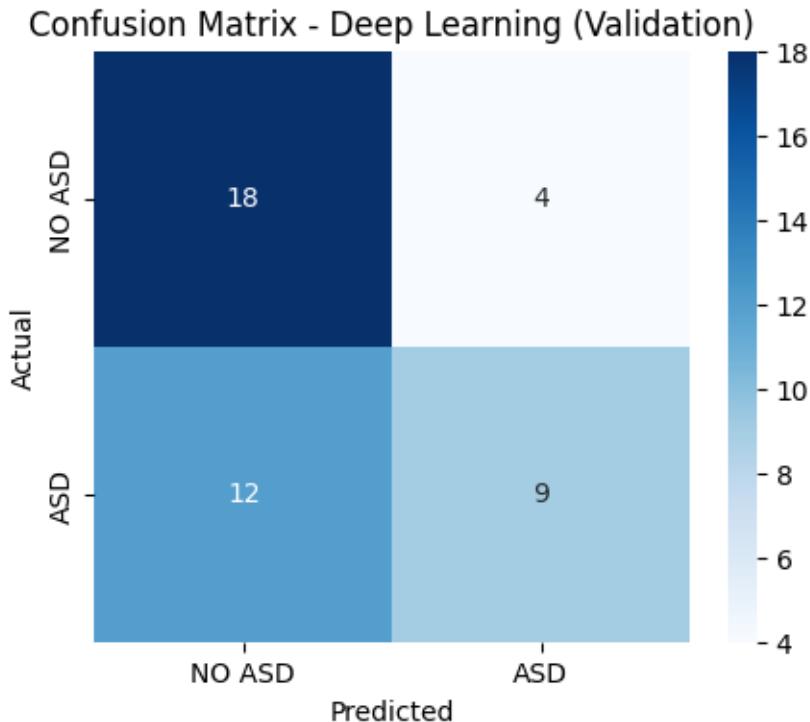
### Model 3 – Deep Learning

Metrik:

**2/2** ━━━━━━ **0s** 45ms/step

- ROC-AUC : 0.6255411255411255
- Accuracy : 0.63
- Precision : 0.60
- Recall : 0.82
- F1\_Score : 0.69

Confusion Matrix / Visualization:



#### Training History:

Epoch 1/10

```
7/7 ━━━━━━━━ 9s 529ms/step - accuracy: 0.5122 -
-auc: 0.5084 - loss: 0.7081 - val_accuracy: 0.4651 - val_auc: 0.5195 -
-val_loss: 0.6926
```

Epoch 2/10

```
7/7 ━━━━━━━━ 0s 63ms/step - accuracy: 0.5279 -
auc: 0.5343 - loss: 0.6966 - val_accuracy: 0.5116 - val_auc: 0.5519 -
-val_loss: 0.6885
```

Epoch 3/10

```
7/7 ━━━━━━━━ 0s 65ms/step - accuracy: 0.6088 -
auc: 0.5913 - loss: 0.6825 - val_accuracy: 0.5581 - val_auc: 0.5584 -
-val_loss: 0.6863
```

Epoch 4/10

```
7/7 ━━━━━━━━ 1s 70ms/step - accuracy: 0.6289 -
auc: 0.6619 - loss: 0.6600 - val_accuracy: 0.5116 - val_auc: 0.5952 -
-val_loss: 0.6846
```

Epoch 5/10

```
7/7 ━━━━━━━━ 1s 77ms/step - accuracy: 0.6362 -
auc: 0.6894 - loss: 0.6528 - val_accuracy: 0.5116 - val_auc: 0.5996 -
-val_loss: 0.6841
```

Epoch 6/10

```
7/7 ━━━━━━━━ 0s 54ms/step - accuracy: 0.6095 -
auc: 0.6472 - loss: 0.6629 - val_accuracy: 0.5814 - val_auc: 0.6017 -
-val_loss: 0.6813
```

Epoch 7/10

```
7/7 ━━━━━━━━ 1s 79ms/step - accuracy: 0.6189 -
auc: 0.6305 - loss: 0.6642 - val_accuracy: 0.6047 - val_auc: 0.6104 -
-val_loss: 0.6793
```

Epoch 8/10

```

7/7 ━━━━━━━━ 0s 62ms/step - accuracy: 0.5712 -
auc: 0.6368 - loss: 0.6609 - val_accuracy: 0.6047 - val_auc: 0.6212 -
val_loss: 0.6780
Epoch 9/10

7/7 ━━━━━━━━ 0s 49ms/step - accuracy: 0.6799 -
auc: 0.7518 - loss: 0.6137 - val_accuracy: 0.6279 - val_auc: 0.6245 -
val_loss: 0.6774
Epoch 10/10

7/7 ━━━━━━━━ 0s 56ms/step - accuracy: 0.6765 -
auc: 0.7497 - loss: 0.6172 - val_accuracy: 0.6279 - val_auc: 0.6180 -
val_loss: 0.6780
2/2 ━━━━━━━━ 0s 207ms/step

```

Perbandingan Ketiga model

Tabel Perbandingan:

Model			
Logistic Regression	0.477273	0.646154	
Random Forest	0.488372	0.656250	
Deep Learning	0.627907	0.529412	

Analisis Hasil:

InterpretasiL

Model Terbaik: Deep Learning (MLP)

Model ini memiliki nilai accuracy tertinggi (0.6279) dibandingkan Logistic Regression dan Random Forest, sehingga menunjukkan kemampuan prediksi yang paling baik dalam membedakan kelas ASD dan non-ASD.

Perbandingan dengan baseline:

Logistic Regression digunakan sebagai baseline model dengan performa paling rendah.

Random Forest menunjukkan peningkatan kecil dibandingkan baseline, namun tidak signifikan.

Deep Learning memberikan peningkatan performa yang paling jelas, yang menandakan bahwa model ini lebih mampu menangkap pola kompleks pada data dibandingkan dua model lainnya.

Trade-off:

- Logistic Regression  
Paling sederhana dan cepat, tetapi performanya rendah.
- Random Forest  
Kompleksitas lebih tinggi dari baseline dengan peningkatan performa terbatas.
- Deep Learning  
Memberikan performa terbaik, namun memiliki kompleksitas model dan waktu training yang lebih besar dibandingkan model lainnya.

Error Analysis:

Kesalahan prediksi masih sering terjadi pada data dengan karakteristik yang mirip antara kelas ASD dan non-ASD. Hal ini menyebabkan model sulit membedakan beberapa sampel, terutama pada kasus borderline yang memiliki nilai fitur yang saling tumpang tindih.

Overfitting / Underfitting:

Model tidak menunjukkan overfitting yang signifikan, karena performa pada data validasi masih sejalan dengan data latih.

Namun, Logistic Regression dan Random Forest cenderung mengalami underfitting, sedangkan Deep Learning mampu mempelajari pola data dengan lebih baik.

## 8. CONCLUSION

### Kesimpulan Utama

Model Terbaik:

Berdasarkan hasil evaluasi, model Deep Learning (MLP) merupakan model terbaik karena menghasilkan nilai akurasi tertinggi dibandingkan Logistic Regression dan Random Forest.

Alasan:

Model Deep Learning lebih unggul karena mampu menangkap pola non-linear dan hubungan kompleks antar fitur dengan lebih baik. Dibandingkan model baseline dan Random Forest, MLP memberikan peningkatan performa yang lebih signifikan tanpa menunjukkan indikasi overfitting yang berarti.

Pencapaian Goals:

Seluruh tujuan proyek telah tercapai. Proyek ini berhasil membangun tiga model machine learning (baseline, advanced, dan deep learning), melakukan preprocessing data secara lengkap, serta mengevaluasi performa model menggunakan metrik yang sesuai. Model terbaik menunjukkan performa yang lebih baik dibandingkan baseline, sehingga tujuan analisis klasifikasi ASD dapat terpenuhi.

### Key Insights

Insight dari data:

1. **Fitur numerik dan kategorikal memiliki pengaruh penting terhadap klasifikasi ASD.**  
Hasil preprocessing dan transformasi data menunjukkan bahwa kombinasi fitur numerik dan kategorikal berkontribusi dalam menentukan hasil klasifikasi.
2. **Terdapat kemiripan karakteristik antar kelas.**  
Beberapa data ASD dan non-ASD memiliki pola yang saling mendekati, sehingga menyebabkan kesalahan prediksi pada model dengan kompleksitas rendah.

## Insight dari Modelling

1. **Model non-linear memberikan performa lebih baik dibandingkan model linear.**  
Random Forest dan Deep Learning menghasilkan performa yang lebih baik dibandingkan Logistic Regression, yang cenderung underfitting karena sifatnya yang linear.
2. **Deep Learning mampu menangani kompleksitas data dengan lebih baik.**  
MLP memberikan hasil paling stabil dan akurat karena mampu mempelajari hubungan fitur yang kompleks melalui beberapa layer tersembunyi.

## Kontribusi Proyek

### Manfaat Praktis:

Proyek ini dapat digunakan sebagai alat bantu deteksi awal Autism Spectrum Disorder (ASD) berdasarkan data kuesioner. Model terbaik dapat dimanfaatkan sebagai sistem pendukung keputusan untuk membantu proses screening awal secara lebih cepat dan objektif.

### Pembelajaran yang didapat:

Melalui proyek ini, diperoleh pemahaman menyeluruh tentang alur kerja machine learning, mulai dari preprocessing data, pemilihan model, proses training, evaluasi performa, hingga analisis hasil. Proyek ini juga memberikan wawasan tentang perbedaan karakteristik dan performa antara model baseline, model machine learning lanjut, dan model deep learning.

## 9. FUTURE WORK

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 (WAJIB)

GitHub Repository  
Link Repository :

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)

Environment & Dependencies

Python Version : [3.8 / 3.9 / 3.10 / 3.11]

Main Libraries & Version:

numpy==1.24.3  
pandas==2.0.3  
scikit-learn==1.3.0  
matplotlib==3.7.2  
seaborn==0.12.2

# Deep Learning Framework (pilih salah satu)

```
tensorflow==2.14.0

# Additional libraries (sesuaikan)
xgboost==1.7.6
lightgbm==4.0.0
opencv-python==4.8.0 # untuk computer vision
nltk==3.8.1      # untuk NLP
transformers==4.30.0 # untuk BERT, dll
```