VERSI 1.1

JULI 2025

PEMBELAJARAN MESIN

MODUL 1 - PENGENALAN ARTIFICIAL NEURAL NETWORK

DISUSUN OLEH:

Yufis Azhar, S.Kom., M.Kom.

Alviya Laela Lestari

Kiara Azzahra

TIM LABORATORIUM INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PENDAHULUAN

TUJUAN

Modul ini bertujuan untuk memberikan pemahaman dasar tentang *Artificial Neural Network* kepada praktikan, baik yang memiliki latar belakang teknis maupun non-teknis. Praktikan akan memahami konsep dasar *Artificial Neural Network* (ANN), struktur, dan jenis-jenisnya, serta bagaimana *Artificial Neural Network* digunakan dalam berbagai konteks.

TARGET MODUL

Mahasiswa dapat memahami konsep dari materi modul dan membuat model *Artificial Neural Network* sederhana untuk melakukan klasifikasi biner dengan dataset jenis tabular serta menyelesaikan kegiatan modul praktikum.

PERSIAPAN

1. Google Collaboratory.
2. Library NumPy, Pandas, Matplotlib, Sklearn, dan Tensorflow.
3. Source Code:
   https://colab.research.google.com/drive/1kYLfRL6eShi_PE8aNjUClOTi2Fr2OyLl?usp=sharing

TABLE OF CONTENTS

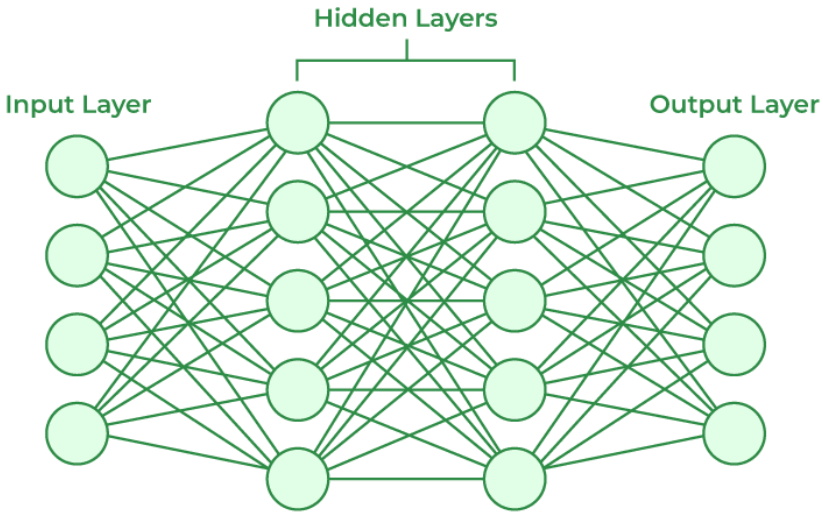
MATERI POKOK

Kenapa kita harus mempelajari *Artificial Neural Network? Artificial Neural Network* atau jaringan neural adalah fondasi dari banyak teknologi kecerdasan buatan *(artificial intelligence)* yang kita gunakan sehari-hari, seperti *face recognition,* sistem rekomendasi, penerjemah bahasa otomatis, dan lain-lain. *Artificial Neural Network* memiliki kemampuan luar biasa dalam mengenali pola dari data kompleks.

A. ARTIFICIAL NEURAL NETWORK (ANN)

Artificial Neural Network atau jaringan syaraf tiruan adalah model *machine learning* yang dapat membantu komputer untuk membuat keputusan *(decision making)* dengan cara kerja yang mirip dengan otak manusia. Proses ini meniru cara neuron biologis bekerja sama untuk mengidentifikasi adanya fenomena, mempertimbangkan pilihan, hingga tahap pembuatan keputusan atau kesimpulan.

B. STRUKTUR DASAR ARTIFICIAL NEURAL NETWORK (ANN)



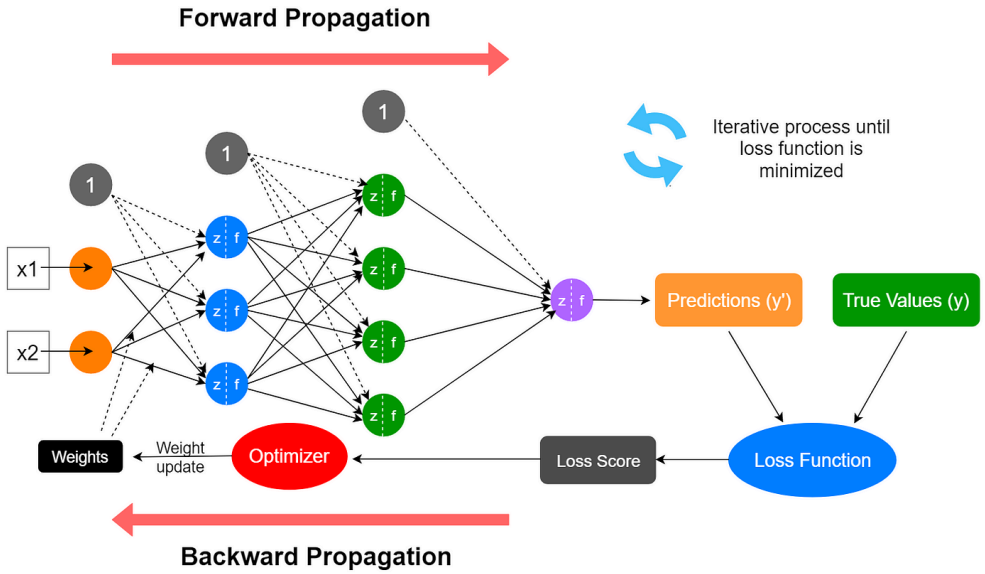
Source: Artificial Neural Networks and its Applications - GeeksforGeeks

Artificial Neural Network terdiri beberapa lapisan atau layer, dengan tiga jenis utama:

1. **Input layer:** Lapisan yang bertanggung jawab untuk menerima data mentah atau fitur-fitur dalam dataset. JIka kita ingin membuat *face recognition,* maka inputnya bisa berupa citra wajah.
2. **Hidden layer:** Lapisan yang memproses data dari input layer. Semakin banyak hidden layer, maka semakin pola dari jaringan akan semakin kompleks untuk dipelajari dan dipahami. Setiap hidden layer akan bertransformasi menjadi informasi yang lebih kompleks dan abstrak.

3. **Output layer:** Lapisan ini akan menghasilkan keputusan atau prediksi final. Output layer menerjemahkan informasi yang diproses oleh hidden layer menjadi bentuk yang dapat dimengerti dan digunakan. Sebagai contoh, setelah memproses sebuah citra, output layer akan memutuskan apakah hasilnya adalah gambar kucing atau gambar anjing.

C. TRAINING PROCESS



Source: [Overview of a Neural Network's Learning Process | by Rukshan Pramoditha | Data Science 365 | Medium]

Proses *training (learning) artificial neural network* merupakan proses berulang *(iterative)* di mana perhitungan dilakukan maju dan mundur melalui setiap lapisan dalam jaringan hingga meminimalkan *loss function*. Seluruh rangkaian proses ini dibagi menjadi tiga bagian:

1. **Forward Propagation**

   Artificial Neural network terbuat dari beberapa neuron yang ditumpuk menjadi beberapa lapisan. Lapisan ini dihubungkan oleh parameter jaringan (yang diwakili oleh tanda panah). Parameter tersebut adalah bobot *(weights)* dan bias. *Weights* akan mengontrol tingkat kepentingan setiap *input*. Sedangkan bias akan menentukan seberapa mudah neuron diaktifkan.

   Hal yang dilakukan pertama kali adalah menginisialisasi parameter jaringan dengan nilai acak bukan nol untuk *weight* dan bias. Berdasarkan nilai input dan nilai yang telah diinisialisasikan, akan dihitung fungsi linear neuron dan fungsi aktivasi neuron. Perhitungan ini akan terjadi di seluruh jaringan dan mengeluarkan hasil akhir dari *forward propagation* pada iterasi pertama.

Dalam *forward propagation,* perhitungan dilakukan dari *input layer* ke *output layer* (kiri ke kanan) melalui jaringan.

2. **Perhitungan Loss Function**

   Output hasil *forward propagation* disebut *predicted value*. Value ini harus dibandingkan dengan *real value* untuk mengukur kinerja jaringan syaraf tiruan. Di sinilah *loss function* (disebut juga *objective function* atau *cost function*) berperan.

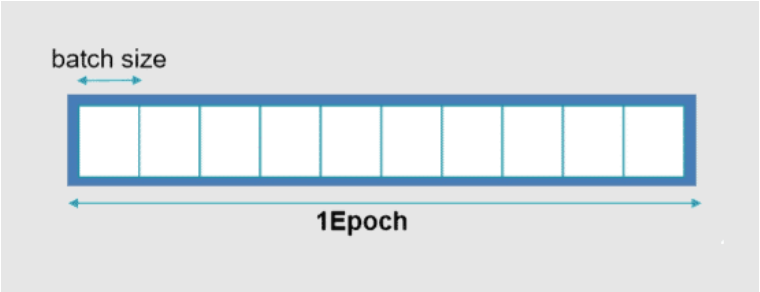
   Loss function menghitung *loss score* antara value yang diprediksi dengan value sebenarnya. Ini juga dikenal sebagai *error* dari sebuah model. *Loss function* mengukur seberapa baik kinerja model di setiap iterasi. *Loss function* digunakan sebagai *feedback* untuk perbaikan parameter di proses *backward propagation.* Nilai ideal *loss function* adalah 0. Semakin mendekati 0, maka prediksi model akan menjadi lebih baik.

3. **Backward Propagation**

   Pada iterasi pertama, *predicted value* akan jauh dari *truth value.* Hal ini disebabkan oleh penetapan parameter (*weight* dan bias) yang sembarangan. Nilai tersebut bukanlah nilai optimal. Oleh karena itu, perlu pembaruan nilai parameter untuk meminimalkan *loss function.* Proses pembaruan parameter dilakukan menggunakan *optimizer* yang menerapkan *backward propagation.* Tujuan *optimizer* adalah menemukan titik minimum global *loss function.*

   Dalam proses *backward propagation,* gradien dari fungsi *loss* terhadap parameter model dihitung menggunakan aturan rantai *(chain rule).* Gradien ini menunjukkan arah untuk memperbarui parameter model. Keras (library dalam Python) secara otomatis menghitung turunan ini melalui *automatic differentiation.* Perhitungan dilakukan dari layer *output* ke layer *input* (dari kanan ke kiri).

D. BATCH SIZE DAN EPOCH



Source: Epoch : An essential notion in real-time programming

Saat berhadapan dengan proses *training* dan *testing*, umumnya terdapat parameter yang bisa kita *tuning* (sesuaikan), di antaranya adalah *epoch* dan *batch size. Batch size* adalah jumlah sampel data yang biasanya melewati jaringan saraf pada satu waktu. *Batch size* menentukan jumlah sampel yang harus dikerjakan sebelum memperbarui parameter model internal. Misalnya, kita memiliki dataset yang berisi 1 juta sampel citra untuk *training*. Meskipun jaringan saraf yang kita gunakan tidak terlalu

kompleks, tetapi 1 juta data *training* masih terlalu banyak dan berat untuk diproses secara bersamaan.

Epoch merupakan *hyperparameter* yang menentukan berapa kali algoritma bekerja melewati seluruh dataset baik secara *forward* maupun *backward*. Satu epoch tercapai ketika semua *batch* telah berhasil melewati jaringan saraf sebanyak satu kali.

E. JENIS-JENIS ARTIFICIAL NEURAL NETWORK

Ada berbagai jenis arsitektur *Artificial Neural Network* (ANN), diantaranya:

1. **Feedforward Neural Network (FNN):** Jenis *artificial neural network* yang paling sederhana di mana informasi bergerak dalam satu arah dari *input layer,* melalui *hidden layer,* dan menuju *output layer.* Tidak ada *loop* atau *feedback* dalam jaringan ini.
2. **Convolutional Neural Network (CNN):** Dirancang khusus untuk pemrosesan data *grid* seperti citra. CNN menggunakan lapisan konvolusi yang dapat mengenali pola visual yang kompleks dengan menerapkan filter pada *input*.
3. **Recurrent Neural Network (RNN):** Memiliki koneksi berulang yang memungkinkan informasi untuk dilanjutkan sepanjang urutan data. Ini membuat RNN sangat berguna untuk data yang memiliki ketergantungan temporal atau sekuensial.
4. **Long Short-Term Memory (LSTM):** Jenis khusus dari RNN yang dirancang untuk mengatasi masalah *vanishing gradient* dengan menggunakan sel memori yang dapat mempertahankan informasi dalam jangka waktu yang lebih panjang.

SIMULASI SEDERHANA

Mari kita buat simulasi sederhana dari sebuah *Artificial Neural Network* (ANN) untuk tugas klasifikasi biner menggunakan Python dan TensorFlow/Keras. Berikut adalah langkah-langkahnya:

1. **Persiapan Data**

   Kita akan menggunakan dataset sederhana seperti make_moons dari sklearn untuk membuat data yang mudah dipahami. Pertama, kita melakukan importing library yang diperlukan.

```
# Import library yang digunakan
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

   Selanjutnya, kita gunakan fungsi make_moons dari pustaka sklearn.datasets untuk menghasilkan dataset sintetis yang sering digunakan untuk masalah klasifikasi.

```
# Generate dataset
X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)
```

   Lalu, kita bagi datanya menjadi data latih dan data tes.

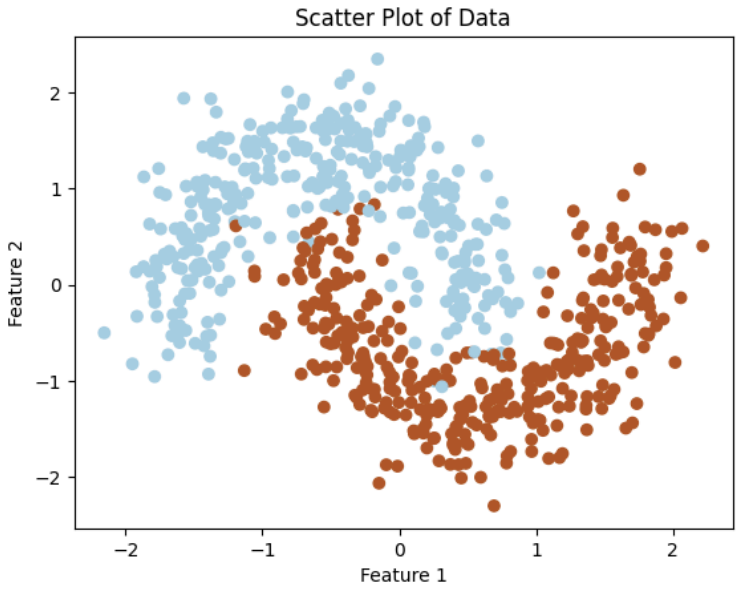
```
# Split dataset menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

   Selanjutnya, lakukan preprocessing secukupnya. Dalam simulasi ini kita terapkan normalisasi pada data menggunakan "StandardScaler" untuk memastikan bahwa fitur-fitur dalam dataset memiliki skala yang serupa, yang tentu dapat meningkatkan kinerja algoritma machine learning.

```
# Standarisasi data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

   Terakhir, sebelum membangun Model kita coba lihat distribusi datanya menggunakan Scatter Plot.

```
# Plotting data
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.Paired)
plt.title('Scatter Plot of Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```



2. Membangun Model

Model yang digunakan dalam simulasi ini adalah Feedforward Neural Network (FNN) sederhana yang dibangun menggunakan kerangka kerja Keras dari TensorFlow. Pertama, mari import library yang kita perlukan terlebih dahulu.

```
# Import library yang akan digunakan
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Selanjutnya, mari kita bangun modelnya.

```
# Membangun model
model = Sequential([
    Dense(10, input_shape=(2,), activation='relu'),
    Dense(10, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

Model ini terdiri dari tiga layer: sebuah input layer dengan 10 neuron yang menerima dua fitur input dan menggunakan fungsi aktivasi ReLU, sebuah hidden layer dengan 10 neuron yang juga menggunakan fungsi aktivasi ReLU, dan sebuah output layer dengan 1

neuron yang menggunakan fungsi aktivasi sigmoid untuk klasifikasi biner. Model ini dirancang untuk mengklasifikasikan data ke dalam dua kelas berdasarkan fitur inputnya. Setelah model telah dibuat, mari kita lakukan compiling model dengan menentukan parameter optimizer, loss, dan metrics.

```
# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Berikut adalah penjelasan dari masing - masing parameter yang digunakan:

a. Optimizer: Fungsi optimizer adalah algoritma yang digunakan untuk mengatur dan memperbarui parameter model Anda saat pelatihan. Tujuannya adalah untuk meminimalkan nilai fungsi kerugian (loss) dengan menyesuaikan bobot dan bias dalam model. Beberapa optimizer populer dalam machine learning termasuk SGD (Stochastic Gradient Descent), Adam, RMSprop, dan banyak lagi. Setiap optimizer memiliki kelebihan dan kekurangan sendiri dan dapat cocok dengan jenis masalah dan data tertentu.
b. Loss: Fungsi kerugian adalah pengukuran seberapa baik atau buruk model Anda memprediksi data pelatihan dibandingkan dengan label yang sebenarnya. Tujuannya adalah untuk mengukur seberapa besar kesalahan prediksi model. Pilihan fungsi kerugian harus disesuaikan dengan jenis masalah yang Anda hadapi. Misalnya, untuk masalah klasifikasi, Anda dapat menggunakan Cross-Entropy Loss, sedangkan untuk masalah regresi, Anda dapat menggunakan Mean Squared Error (MSE) Loss. Pemilihan yang tepat akan membantu model Anda belajar dengan lebih baik.
c. Metrics: Metrik adalah pengukuran yang digunakan untuk mengevaluasi kinerja model setelah pelatihan. Metrik ini memberikan informasi tambahan selain dari fungsi kerugian dan dapat digunakan untuk memahami sejauh mana model Anda berhasil memecahkan masalah yang diberikan. Beberapa metrik umum termasuk akurasi (accuracy) untuk masalah klasifikasi, Mean Absolute Error (MAE) atau Root Mean Squared Error (RMSE) untuk masalah regresi, dan berbagai metrik lainnya seperti F1-score, Precision, Recall, dll., tergantung pada jenis masalah yang dihadapi.

Setelah model berhasil di compile, maka model sudah siap untuk dilatih. Sebelum masuk ke tahap pelatihan, kita coba tampilkan summary dari model yang telah kita buat.

3. Melatih Model

Pada tahap ini, kita akan melatih model menggunakan data latih yang telah dibuat sebelumnya. Fungsi model.fit memerlukan input data X_train dan label y_train, dan melakukan pelatihan model selama 100 epoch. Sebanyak 20% dari data pelatihan digunakan sebagai data validasi untuk memonitor kinerja model dan membantu mencegah overfitting. Hasil pelatihan, termasuk metrik kinerja seperti akurasi dan loss

untuk setiap epoch, disimpan dalam objek history, yang dapat digunakan untuk analisis lebih lanjut dan visualisasi proses pelatihan model.

```
# Train Model
history = model.fit(X_train, y_train, epochs=100, validation_split=0.2)
```

4. **Evaluasi Model**

Setelah Model selesai dilatih, selanjutnya kita melakukan evaluasi model untuk mengetahui performa dari model.

```
# Evaluasi Model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9822 - loss: 0.0741
Test Loss: 0.0850
Test Accuracy: 0.9700
```

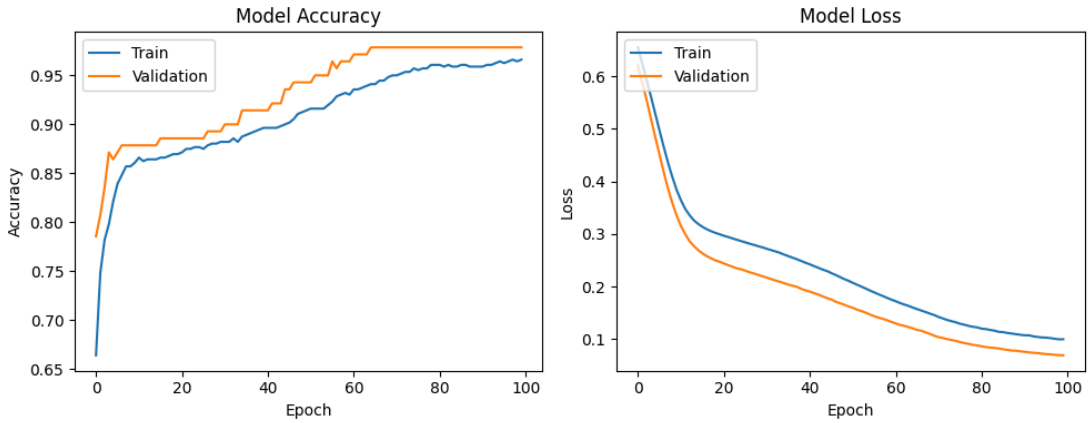
Kita juga dapat menampilkan grafik akurasi dan loss untuk data pelatihan dan validasi selama proses pelatihan berlangsung dengan cara seperti berikut.

```
plt.figure(figsize=(10,4))

# Plot akurasi
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.tight_layout()
plt.show()
```



Tampilkan juga Classification Report nya dengan cara berikut.

```
from sklearn.metrics import classification_report

# Prediksi pada data test
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int)
print(classification_report(y_test, y_pred_classes))
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       156
           1       0.98      0.96      0.97       144

    accuracy                           0.97       300
   macro avg       0.97      0.97      0.97       300
weighted avg       0.97      0.97      0.97       300
```

Uji model pada data baru

```
# Data baru untuk diuji (contoh 2 titik data)
X_new = np.array([[0.5, 0.3], [1.2, -0.8]])
proba = model.predict(scaler.transform(X_new))
kelas = (proba > 0.5).astype(int)

for d, p, k in zip(X_new, proba, kelas):
    print(f"Data: {d}, Prob kelas 1: {p[0]:.4f}, Prediksi: {k[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Data: [0.5 0.3], Prob kelas 1: 0.3757, Prediksi: 0
Data: [ 1.2 -0.8], Prob kelas 1: 0.9987, Prediksi: 1
```

Selamat! Dengan ini kita berhasil membuat sebuah model sederhana untuk klasifikasi biner.

LATIHAN PRAKTIKUM

Lengkapi kode di bawah berdasarkan instruksi yang telah diberikan!
Dataset:
https://drive.google.com/file/d/1nS5kiA_X9FuiTJjnc8_TiL_GPCvdeF7I/view?usp=sharing

1. **Import Library yang Dibutuhkan**

```
import ... as np
import ... as pd
import ... as plt
from ... import MinMaxScaler

from tensorflow.keras.layers import ...
from tensorflow.keras.models import ...
from sklearn.model_selection import ...
```

2. **Load Dataset**

```
pd.read_csv('...')

# Tampilkan 10 data awal
...

# Tampilkan informasi ringkas data
...

# Tampilkan statistik deskriptif data
...
```

3. **Preprocessing Data**

```
# Drop kolom yang tidak diperlukan
...(columns=['customer name', 'customer e-mail', 'country', 'gender'], inplace=True)

# Definisikan X dan y
X = df.drop('car purchase amount',axis=1)
y = df['car purchase amount']
```

4. **Standarisasi Data**

```
scaler = ...
X = scaler...(X)
y = scaler...(y.values.reshape(-1, 1))
```

5. **Splitting Data**
   Gunakan test size 20% dan random state 0

```
X_train, X_test, y_train, y_test = train_test_split(..., ..., ..., ...)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

6. **Buat Model**
   Ketentuan:
   a. Gunakan model Sequential
   b. Ada 3 layer dengan jenis layer Dense
   c. Layer pertama jumlah neuron 10, aktivasi ReLU
   d. Layer kedua jumlah neuron 10, aktivasi ReLU
   e. Layer ketiga jumlah neuron 1, aktivasi linear
   f. Compile model menggunakan optimizer Adam dan loss function mean_squared_error

```
model = ...([
    ...(..., activation='...', input_dim=4),
    ...(..., activation='...'),
    ...(..., activation='...')
])

model.compile(optimizer='...', loss='...')
model.summary()
```

7. **Training Model**
   a. Gunakan 100 epoch
   b. Validation data 20%

```
history = model.fit(X_train, y_train, epochs=..., validation_split=...)
```

8. **Evaluasi Model**

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

9. **Plotting Akurasi dan Loss dari Training dan Validation**
   Buatlah seluruh kodenya sendiri.

HASIL PENGERJAAN LATIHAN PRAKTIKUM

DETAIL	POIN	SELESAI
Import Library	10	☐
Load Dataset	10	☐
Preprocessing Data	10	☐
Standarisasi Data	10	☐
Splitting Data	10	☐
Membuat Model	10	☐
Training Model	10	☐
Evaluasi Model	10	☐
Plotting Akurasi dan Loss dari Training dan Validation	20	☐
TOTAL	**100**	

TUGAS PRAKTIKUM

1. **Import Library dan Load Data**
    a. Import semua library yang dibutuhkan.
    b. Muat dataset dari link berikut: Dataset Tugas
2. **Exploratory Data Analysis**
    a. Tampilkan 5 data teratas dan 5 data terbawah dari dataset.
    b. Tampilkan ringkasan statistik dataset, termasuk mean, median, standar deviasi, nilai minimum, nilai maksimum, dan kuartil.
    c. Visualisasikan distribusi label dengan grafik.
3. **Preprocessing Data**
   Atasi missing value dengan ketentuan:
    a. Isi nilai kosong pada kolom 'IPK' dan 'Kehadiran_Persen' dengan mean.
    b. Isi nilai kosong pada kolom 'Tugas_Persen' dengan median.
    c. Isi nilai kosong pada kolom 'Nama' dengan 'Nama_Tidak_Diketahui'.
4. **Inisialisasi Input**
   Pisahkan data menjadi fitur dan target.
5. **Splitting Data dengan Proporsi 80:20**
6. **Bangun Model Naive Bayes**
7. **Evaluasi Model dengan Ketentuan Berikut:**
    a. Tampilkan classification report beserta keterangan labelnya.
    b. Akurasi minimal 85%.
8. **Uji dengan Data Baru:**
    a. IPK 3.2, Kehadiran_Persen 85%, Tugas_Persen 20%.
    b. IPK 2.1, Kehadiran_Persen 60%, Tugas_Persen 50%.
    c. IPK 3.8, Kehadiran_Persen 95%, Tugas_Persen 90%.

    Expected Output:

```
Tidak Lulus
Tidak Lulus
Lulus
```

BOBOT PENILAIAN TUGAS PRAKTIKUM

DETAIL		POIN	SELESAI
Latihan Praktikum		10	☐
Tugas Praktikum	Import Library dan Load Data	5	☐
	Exploratory Data Analysis	5	☐
	Preprocessing Data	5	☐
	Inisialisasi Input	15	☐
	Splitting Data dengan Proporsi 80:20	5	☐
	Bangun Model Naive Bayes	5	☐
	Evaluasi Model	5	☐
	Uji dengan Data Baru	20	☐
Pemahaman Materi		25	☐
TOTAL		**100**	