Kickstart Your Machine Learning Journey: Setting Up JupyterLab for Machine Learning Development

1 Introduction

Dalam proses pengembangan Machine Learning, beberapa elemen penting diperlukan untuk memastikan kelancaran dan efektivitas pembangunan model. Salah satunya adalah **platform pemrograman** yang mendukung, seperti **JupyterLab**, dan **distributor pustaka** yang andal, seperti **Anaconda**. Tutorial ini menyajikan panduan komprehensif dari awal hingga akhir, termasuk persiapan lingkungan, impor dataset, pemisahan fitur, pembuatan fungsi untuk pelatihan dan evaluasi, serta pelatihan dan evaluasi model. Dengan pendekatan praktis, tutorial ini bertujuan memberikan pemahaman mendalam tentang konsep dan teknik pengembangan Machine Learning dan kemampuan untuk mengaplikasikannya dalam proyek nyata. Pada panduan pengembangan machine learning kali ini, kita akan menggunakan JupyterLab sebagai platform pemrograman dan Anaconda sebagai penyedia pustaka serta lingkungan untuk pengembangan machine learning. Berikut merupakan penjelasan mengenai JupyterLab dan Anaconda yang harus dipahami terlebih dahulu:

1.1 JupyterLab

JupyterLab merupakan platform yang digunakan untuk melakukan aktivitas pemrograman. Platform ini memiliki fungsi yang mirip dengan platform pemrograman lainnya seperti Visual Studio Code. Namun, alasan utama penggunaan JupyterLab adalah karena kita menggunakan Anaconda sebagai penyedia pustaka dan lingkungan pengembangan. Anaconda juga menyediakan JupyterLab sebagai alternatif platform pemrograman yang terintegrasi dengan baik, sehingga memudahkan dalam pengaturan dan penggunaan pustaka yang diperlukan untuk pengembangan machine learning.

1.2 Anaconda

Anaconda adalah distributor pustaka yang sangat terkenal. Dengan menggunakan platform ini, proses pengunduhan pustaka dan persiapan lingkungan menjadi lebih mudah. Setelah menginstal Anaconda, Python akan otomatis terinstal dan terintegrasi dengan perangkat Anda, sehingga tidak perlu repot melakukan instalasi terpisah. Anaconda juga menyediakan berbagai pustaka yang lengkap, seperti Machine Learning, NumPy, Matplotlib, Pandas, dan pustaka pendukung lainnya. Keunggulan ini membuat Anaconda menjadi pilihan utama bagi banyak pengembang dalam memulai proyek machine learning dan data science.

2 Workflow

Pada bagian ini, saya akan terlebih dahulu menjelaskan workflow yang akan kita lakukan agar lebih mudah dipahami. Alasan lainnya adalah untuk memudahkan kita mengidentifikasi secara berurutan kesalahan yang mungkin terjadi berdasarkan langkah-langkah yang kita lakukan. Berikut beberapa bagian dari workflow pada persiapan dan pengembangan machine learning:

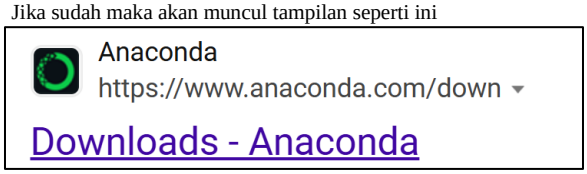
2.1 Instalasi Aplikasi

Pada bagian ini, kita akan melakukan beberapa instalasi untuk aplikasi pada perangkat yang akan kita gunakan. Karena pada penjelasan sebelumnya kita sudah menentukan bahwa kita akan menggunakan JupyterLab sebagai platform pemrograman dan Anaconda sebagai platform penyedia pustaka. Maka pada bagian ini saya akan memberikan panduan bagaimana cara melakukan instalasi pada kedua aplikasi tersebut.

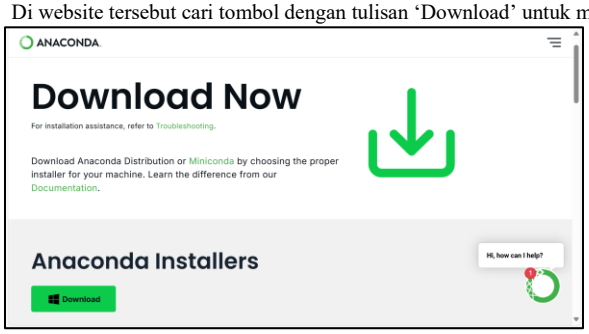
2.1.1 Anaconda

Hal yang harus kalian persiapkan adalah internet dan penyimpanan sekitar 1 GB untuk menampung aplikasi Anaconda itu sendiri. Jika sudah maka ikut Langkah-langkah berikut:

- Buka Google kemudian cari ‘Anaconda Download’
- Jika sudah maka akan muncul tampilan seperti ini



- Di website tersebut cari tombol dengan tulisan ‘Download’ untuk melakukan proses unduh pada file Anaconda.

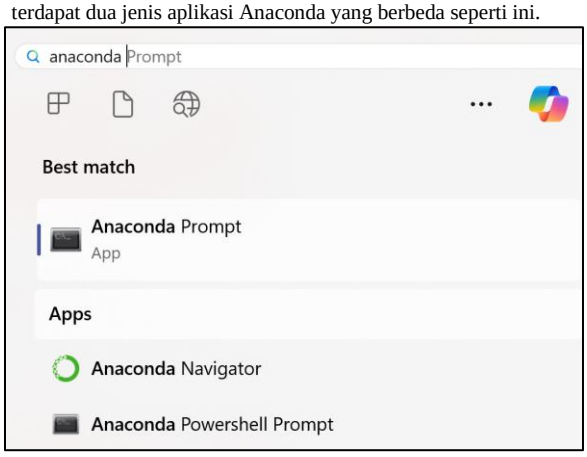


- Apabila sudah ditekan, proses pengunduhan pada file akan dimulai. File akan berupa format installer jika proses pengunduhan selesai. Pada tahap ini lakukan instalasi seperti biasa dengan membuka file yang sudah diunduh. Jika sudah dibuka akan terdapat beberapa pemberitahuan terkait proses instalasi. Lanjutkan proses dengan cara menekan ‘Next’ hingga proses instalasi selesai.

2.1.2 JupyterLab & Environment Settings

Tahap ini akan kita lakukan apabila proses pengunduhan dan instalasi pada Anaconda sudah sempurna sukses. Apabila terjadi kesalahan coba ulangi dan pahami Langkah-langkah nya kemudian hapus dan lakukan instalasi Kembali.

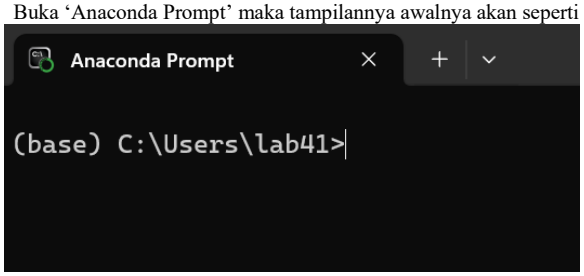
- Jika sudah sempurna dan sukses maka coba buka menu Windows kemudian ketikan Anaconda pada menu search. Nantinya terdapat dua jenis aplikasi Anaconda yang berbeda seperti ini.



- Terdapat dua cara untuk mempersiapkan lingkungan dan pengunduhan serta instalasi pustaka pada Anaconda. Di sini kita akan menyebutnya sebagai Metode 1 untuk penggunaan Anaconda Prompt, sedangkan Metode 2 untuk penggunaan Anaconda Navigator.

Metode 1

- Untuk metode ini kita akan menggunakan cara yang lebih ‘Programmer’ Dimana kita akan menggunakan prompt untuk melakukan download dan instalasi pada pustaka serta pembuatan lingkungan tersebut.
- Buka ‘Anaconda Prompt’ maka tampilannya awalnya akan seperti ini.



Keterangan:

a. (base): Merupakan lingkungan yang sedang aktif.
b. C:\Users\lab41: Merupakan Alamat atau address dari folder yang kita akses saat itu juga.

- Terdapat informasi yang harus diketahui terlebih dahulu yaitu untuk instalasi pustaka sama sekali tidak melibatkan address atau Alamat folder apapun. Pustaka akan langsung tersimpan di penyimpanan lingkungan tanpa memperdulikan posisi dari folder yang sedang diakses. Jadi mau dimanapun posisi address folder yang aktif, tidak akan disimpan ke dalam folder tersebut melainkan langsung masuk ke dalam penyimpanan lingkungan yang sedang aktif di mana lingkungan yang sedang aktif berdasarkan gambar sebelumnya yaitu adalah **'base'**.
- Kemudian jangan pernah melakukan modifikasi apapun terhadap lingkungan **'base'** dikarenakan lingkungan base digunakan sebagai lingkungan utama yang digunakan pada Anaconda. Jadi apabila terjadi sesuatu seperti kesahalan versi, bentrokan antar pustaka, ketidakcocokan antara pustaka yang berujung error maka mau tidak mau harus melakukan instalasi ulang pada Anaconda.
- Jika sudah membaca panduan sebelumnya, maka sudah saatnya kita masuk ke Langkah selanjutnya yaitu pembuatan lingkungan dengan cara memasukan kode seperti ini

```
conda create -n nama_environment (example: conda create -n machine_learning)
```

Kode di atas digunakan untuk membuat lingkungan dengan nama 'machine_learning' tanpa melakukan instalasi tambahan berupa pustaka.

```
conda create -n nama_environment nama_package (example: conda create -n machine_learning tensorflow)
```

Kode di atas digunakan untuk membuat lingkungan dengan nama 'machine_learning' dan juga sekaligus melakukan instalasi tambahan berupa pustaka yaitu pustaka tensorflow di dalamnya.

Perbedaannya yaitu kode pertama membuat lingkungan saja sedangkan kode kedua membuat lingkungan serta mengunduh package atau pustaka yang akan diinstall di dalamnya.

- Apabila ingin menghapus lingkungan yang sudah dibuat akibat terjadinya error yang tidak mungkin lagi untuk diperbaiki, gunakan kode di bawah ini untuk menghapusnya.

```
conda remove -n nama_environment -all (example: conda remove -n machine_learning -all)
```

Kode di atas digunakan untuk menghapus environtment dengan nama 'machine_learning' beserta semua package dan pustaka di dalamnya.

- Jika ingin melakukan pengecekan pada lingkungan yang sudah dibuat atau tersedia di dalam aplikasi Anaconda. Pengguna dapat menjalankan kode di bawah ini.

```
conda env list
```

kode di atas digunakan untuk mengecek environment apa saja yang sudah dibuat dan tersedia di dalam aplikasi Anaconda.

```
conda list
```

kode di atas digunakan untuk mengecek pustaka dan package apa saja yang sudah terinstall di dalam environment yang sudah dibuat.

- Biasanya apabila kita sudah berhasil membuat lingkungan pada aplikasi Anaconda. Akan terdapat perintah tambahan yang teman-teman harus ingat seperti ini.

```
conda activate nama_environment (example: conda activate machine_learning)
```

Kode di atas digunakan untuk mengaktifkan environment dengan nama 'machine_learning'.

```
conda deactivate
```

Kode di atas digunakan untuk menonaktifkan environtment apapun yang sedang aktif.

Kode-kode di atas merupakan kode untuk persiapan environment atau lingkungan. Perlu diingat bahwa teman-teman harus melakukan instalasi package apapun dengan cara membuat environment berdasarkan projek yang sedang dikerjakan. Jangan menggabungkan projek satu dengan lainnya menggunakan satu buah environtment yang sama. Kemudian jangan pernah menggunakan environment base sebagai lingkungan untuk develop machine learning. Setiap ingin melakukan instalasi perlu diingat untuk selalu mengecek lingkungan yang sedang aktif.

- Apabila persiapan lingkungan sudah siap, maka aktifkan lingkungan dengan mengikuti cara di atas kemudian ketikan kode berikut untuk melakukan instalasi untuk JupyterLab di lingkungan yang sedang aktif.

```
conda install jupyterlab
```

Kode di atas digunakan untuk melakukan instalasi jupyter lab di dalam lingkungan yang sedang aktif. Apabila berhasil maka setiap ingin mengakses jupyter lab, teman teman harus mengaktifkan dahulu environtment tempat jupyterlab dan pustaka teman-teman berada.

Setiap melakukan instalasi, aplikasi akan meminta persetujuan dengan tanda y/n. Jika teman-teman setuju untuk menambahkan pustaka atau package ke dalam environtment maka tekan y lalu enter. Sebaliknya jika tidak maka tekan n lalu enter maka eksekusi kode akan berakhir.

- Jika sudah selesai terinstall, maka teman-teman bisa pergi ke folder yang ingin teman-teman buka di jupyter lab dengan cara pindah direktori pada bagian address atau alamat folder.

nama_path: (example: D:, E:, F:)
Kode di atas digunakan untuk mengganti path pada folder, karena biasanya address folder deafult berada pada C atau tempat anaconda terinstall. Kode tersebut untuk mengganti ke path tempat penyimpanan lainnya seperti di D, E, atau F tergantung dengan tempat penyimpanan kedua teman-teman.

```
cd nama_folder (example: cd develop_ml)
```

Kode di atas digunakan untuk mengganti direktori folder teman-teman. Kurang lebih gunanya untuk mengganti folder atau masuk ke dalam folder yang ingin dituju. Pada contoh di atas, kode akan masuk atau berpindah ke folder dengan nama ‘develop_ml’.

```
cd..
```

Kode di atas digunakan apabila ingin keluar dari folder yang sedang diakses.

```
dir
```

Kode di atas digunakan apabila ingin mengecek list folder apa saja yang tersedia di dalam path tersebut.

Path atau alamat folder tidak ada hubungannya dengan penyimpanan pustaka dan lingkungan. Jadi apabila teman-teman ketika menginstall pustaka dan membuat lingkungan dan sedang mengakses folder develop_ml. Bukan berarti pustaka dan lingkungan tersebut disimpan di dalam folder develop_ml. Semua pustaka dan lingkungan akan langsung disimpan di penyimpanan env di aplikasi Anaconda. Jadi tidak perlu khawatir sama sekali apabila ingin mengakses lingkungan harus masuk ke folder tertentu.

Lingkungan dan pustaka akan langsung disimpan di folder env pada aplikasi Anaconda.
Folder dan file seperti foto, file python dll akan disimpan langsung ke path folder yang teman-teman sedang akses.

Kedua hal ini tidak saling berhubungan satu sama lain karena berbeda fungsi dan tugas. Jadi tidak perlu khawatir asalkan bener-bener mengikuti instruksi dan pemahaman yang diberikan.

- Saya akan mencontohkan dengan sebuah kasus seperti ini. Saya sudah membuat folder dengan nama ‘develop_ml’ di path D. Kemudian saya ingin mengaksesnya di jupyter lab. Maka cara yang saya lakukan adalah seperti ini

Cara 1

```
D:
```

Untuk berpindah ke path D

```
jupyter lab
```

Untuk membuka jupyter lab. Apabila sudah terbuka saya bisa langsung memilih folder develop_ml untuk saya akses.

Cara 2

```
D:
```

Untuk berpindah ke path D

```
cd develop_ml
```

Untuk pindah folder atau direktori

```
jupyter lab
```

Untuk membuka jupyterlab. Folder akan langsung diakses jika jupyterlab terbuka.

Selamat anda sudah selesai membuat environtment beserta jupyter lab di aplikasi Anaconda. Pergi ke step selanjutnya untuk melihat panduan bagaimana cara install pustaka atau package tambahan untuk develop machine learning.

Metode 2

- Untuk metode ini bisa dikatakan lebih mudah dibandingkan metode pertama. Pengguna hanya tinggal membuka 'Anaconda Navigator' pada menu search windows untuk membuka aplikasi Anaconda.
- Apabila sudah dibuka akan muncul tampilan dari aplikasi tersebut. Pada tampilan awal, anaconda sudah menampilkan beberapa platform yang dapat kita gunakan dalam pengembangan machine learning dan aktivitas pemrograman lainnya seperti aplikasi Spyder, JupyterLab, PyCharm, dan lain-lain.
- Pada bagian ini teman-teman bisa cari JupyterLab lalu tekan tombol install untuk memulai instalasi pada aplikasi. Ikuti langkah-langkah yang diberikan hingga JupyterLab terinstall.
- Kemudian apabila sudah selesai, teman-teman bisa pergi ke bagian 'environment' kemudian cari tombol create lalu isi nama lingkungan yang ingin dibuat lalu ceklis bahasa Python sebagai package utama.
- Apabila sudah selesai dibuat, masuk ke environtment dengan cara menekan lingkungan yang sudah dibuat tadi lalu apabila ingin menambahkan pustaka maka bisa dilihat di bagian sebelah kanan pustaka apa saja yang ingin diinstall di dalamnya.

Cara ini sedikit lebih ribet sejujurnya dibandingkan dengan menggunakan Anaconda Prompt. Jadi disarankan untuk menggunakan metode 1 sebagai cara yang utama.

2.2 Install Pustaka

Jika kalian sudah berada pada tahap ini tanpa mengalami error. Selamat kalian tinggal selangkah lagi untuk mendevelop sebuah machine learning. Pada bagian ini kita hanya perlu melakukan instalasi pustaka yang kita butuhkan nantinya. Terdapat beberapa pustaka yang kita butuhkan yaitu seperti berikut:

- Tensorflow: `conda install tensorflow`
  Kode di atas digunakan untuk menginstall pustaka tensorflow (machine learning). Note: apabila sudah menginstall ini jangan pernah menginstall numpy, dikarenakan takut berbentrokan versi dengan yang dibutuhkan oleh tensorflow.

- XGBoost: `conda install xgboost`
  Kode di atas digunakan untuk menginstall pustaka XGBoost (machine learning).

- LightGBM: `conda install lightgbm`
  Kode di atas digunakan untuk menginstall pustaka LightGBM (machine learning).

- CatBoost: `conda install catboost`
  Kode di atas digunakan untuk menginstall pustaka CatBoost (machine learning).

- Scikit-learn: `conda install scikit-learn`
  Kode di atas digunakan untuk menginstall pustaka sklearn (penyedia machine learning, kalkulasi metric, dan diagram plot)

- Pandas: `conda install pandas`
  Kode di atas digunakan untuk menginstall pustaka Pandas (pustaka untuk menghitung operasi mtk)

- Matplotlib: `conda install matplotlib`
  Kode di atas digunakan untuk menginstall pustaka matplotlib (penyedia grafik, diagram, dan plost)

- Seaborn: `conda install seaborn`
  Berfungsi sama seperti matplotlib

- Micromlgen: `pip install micromlgen`
  Kode di atas digunakan untuk menginstall pustaka micromlgen (export model menjadi header c). Note: conda tidak menyediakan pustaka micromlgen. Oleh karena itu kita menggunakan penyedia lain yaitu pip.

2.3 Pengembangan Machine Learning

Pada tahap ini kalian bisa mengikuti kode yang saya berikan untuk memulai develop machine learning. Di sini akan dibagi menjadi beberapa bagian yaitu:

- Header

  Bagian ini berfungsi untuk memanggil semua pustaka dan package yang telah kita install sebelumnya. Semua pustaka yang dibuhtuhkan dipanggil di bagian ini. Apabila selesai maka akan memunculkan teks sukses yang berarti semua pustaka berhasil di import.

  Tanda # merupakan komen, tidak perlu untuk diketik di program hanya digunakan sebagai penanda saja. Bagian header sudah dipisah berdasarkan fungsi dan kegunaannya. Bagian pertama yaitu machine learning yang akan kita develop atau gunakan, bagian kedua yaitu pustaka yang digunakan untuk training dan testing. Bagian ketiga digunakan untuk membuat diagram atau plot visualisasi dari hasil training dan testing. Dan bagian terakhir adalah bagian untuk mengkonversi model menjadi header c agar dapat di deploy.

header

```
# Machine Learning Methods
import tensorflow as tf
import xgboost as xgb # Extreme Gradient Boosting
import lightgbm as lgb # Light Gradient Boosting Machine
from catboost import CatBoostClassifier # CatBoost
from sklearn.ensemble import AdaBoostClassifier # AdaBoost
from sklearn.ensemble import GradientBoostingClassifier # Gradient Boosting Classifier
from sklearn.ensemble import RandomForestClassifier # Random Forest
from sklearn.tree import DecisionTreeClassifier # Decision Tree
from sklearn.linear_model import LogisticRegression # Logistic Regression
from sklearn.neighbors import KNeighborsClassifier # K-Nearest Neighbors
from sklearn.neural_network import MLPClassifier # Multilayer Perceptron
from sklearn.naive_bayes import GaussianNB # Gaussian Naive Bayes

# Additional Libraries for Training and Testing
import pandas as pd
import numpy as np
import time
from sklearn.preprocessing import LabelEncoder, StandardScaler, normalize,
label_binarize
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Additional Libraries for Plotting
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix, roc_curve, auc

# Additional Library for Porting to Microcontroller
from micromlgen import port
import os

print("Success")
```

- Train Function

  Bagian ini berfungsi untuk mempersiapkan fungsi pelatihan dan test pada model yang ingin dibuat. Pada bagian ini terdiri dari beberapa step seperti import dataset, pemisahan fitur train dan test menjadi x (data sensor) dan y (data label), mentrigger train dan test, print hasil berupa data numerik (metric).

train function

```
table = pd.read_csv('1_Percent_Data/1_Percent_Data.csv') #import dataset

table_cleaned = table.drop(columns=['EMG_chest']) #hapus kolom yang tidak ingin
digunakan

x = table_cleaned.drop(columns=['StressLevel']) #data sensor
y = table_cleaned['StressLevel'] #data label

#pemisahan fitur untuk train dan test (ukuran size data test sebesar 15% dari data
input)
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.15, random_state=42)

#fungsi train yang nanti akan dipanggil setiap melakukan train
def train_model(model):
    # Train the model
    model.fit(x_train, y_train)

    # Predict on test data
    start_time = time.time()
    y_pred = model.predict(x_test)
    y_prob = model.predict_proba(x_test) if hasattr(model, 'predict_proba') else None
    end_time = time.time()

    # Calculate results
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred, average='weighted'), 3)
    recall = round(recall_score(y_test, y_pred, average='weighted'), 3)
    f1 = round(f1_score(y_test, y_pred, average='weighted'), 3)
    test_time = round(end_time - start_time, 3)

    # Print results
    print("Accuracy score of the model: ",accuracy)
    print("Precision score of the model:",precision)
    print("Recall score of the model: ",recall)
    print("F1 score of the model: ",f1)
    print("Time taken to test the model: ",test_time)

    return y_pred, y_prob, test_time
```

Untuk model machine learning tensorflow fnn, kita harus memodifikasi sedikit bagian ini yaitu dengan cara seperti berikut.

call function

```
table = pd.read_csv('1_Percent_Data/1_Percent_Data.csv')

table_cleaned = table.drop(columns=['EMG_chest'])

x = table.drop(columns=['StressLevel'])
y = table['StressLevel']

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

scaler = StandardScaler()
x = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.15, random_state=42)

def tensorflow():
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(128, activation='relu', input_shape=
(x_train.shape[1],)),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(len(np.unique(y)), activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Train the model
    history = model.fit(x_train, y_train, epochs=50, batch_size=32,
validation_split=0.2, verbose=0)

    # Predict on test data
    start_time = time.time()
    y_prob = model.predict(x_test)
    y_pred = np.argmax(y_prob, axis=1)
    end_time = time.time()

    # Calculate results
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred, average='weighted'), 3)
    recall = round(recall_score(y_test, y_pred, average='weighted'), 3)
    f1 = round(f1_score(y_test, y_pred, average='weighted'), 3)
    test_time = round(end_time - start_time, 3)

    # Print results
    print("Accuracy score of the model: ", accuracy)
    print("Precision score of the model:", precision)
    print("Recall score of the model: ", recall)
    print("F1 score of the model: ", f1)
    print("Time taken to test the model: ", test_time)

    return y_pred, y_prob, test_time
```

- Plotting Function
  Pada bagian ini berfungsi untuk membuat fungsi berupa plot atau diagram visualisasi dari hasil training yang sudah dihasilkan. Hal ini dapat membantu dalam melihat visualisasi dari hasil model yang sudah dihasilkan.

plot function

```
# Normalized Confusion Matrix
def confusion_matrix_normalized(y_test, y_pred, model_name, save_path='plot'):
    conf_matrix = confusion_matrix(y_test, y_pred)
    conf_matrix_normalized = normalize(conf_matrix, axis=1, norm='l1')

    if not os.path.exists(save_path):
        os.makedirs(save_path)

    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix_normalized,
display_labels=np.unique(y_test))
    disp.plot(cmap='Blues', values_format='.2f')
    plt.title(f'Normalized Confusion Matrix - {model_name}')

    filename = f'normalized_confusion_matrix_{model_name}.png'
    plt.savefig(os.path.join(save_path, filename))

    plt.show()

# Non-Normalized Confusion Matrix
def confusion_matrix_non_normalized(y_test, y_pred, model_name, save_path='plot'):
    conf_matrix = confusion_matrix(y_test, y_pred)

    if not os.path.exists(save_path):
        os.makedirs(save_path)

    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=np.unique(y_test))
    disp.plot(cmap='Blues', values_format='d')
    plt.title(f'Non-Normalized Confusion Matrix - {model_name}')

    filename = f'non_normalized_confusion_matrix_{model_name}.png'
    plt.savefig(os.path.join(save_path, filename))

    plt.show()

# ROC Curve
def plot_roc_curve(y_test, y_prob, model_name, save_path='plot', n_classes=None):
    if n_classes is None:
        n_classes = len(np.unique(y_test))

    y_test_bin = label_binarize(y_test, classes=np.unique(y_test))

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_prob[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_prob.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    if not os.path.exists(save_path):
        os.makedirs(save_path)

    colors = plt.get_cmap('tab10')

    for i in range(n_classes):
        plt.plot(fpr[i], tpr[i], color=colors(i / n_classes), lw=1, label=f'Class {i}
ROC curve (area = {roc_auc[i]:.2f})')

    plt.plot(fpr["micro"], tpr["micro"], color='navy', lw=1, label=f'Micro-average ROC
curve (area = {roc_auc["micro"]:.2f})')

    plt.plot([0, 1], [0, 1], color='red', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {model_name}')
    plt.legend(loc='lower right', fontsize=8)

    filename = f'roc_curve_{model_name}.png'
    plt.savefig(os.path.join(save_path, filename))

    plt.show()
```

- Call Function

  Pada bagian ini, fungsi yang sudah kita buat sebelumnya dan sudah dijalankan kita panggil masing-masing menggunakan fungsi di bawah ini. Setiap fungsi dipanggil berdasarkan machine learning yang kita gunakan. Silahkan jalankan kode di bawah ini berdasarkan model yang ingin dipilih.

```
call function

xgboost = xgb.XGBClassifier()
y_pred, y_prob, testing_time = train_model(xgboost)

model_name = 'XGBoost'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

```
call function

lightGBM = lgb.LGBMClassifier()
y_pred, y_prob, testing_time = train_model(lightGBM)

model_name = 'LightGBM'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

```
call function

catboost = CatBoostClassifier()
y_pred, y_prob, testing_time = train_model(catboost)

model_name = 'CatBoost'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

```
call function

adaboost = AdaBoostClassifier()
y_pred, y_prob, testing_time = train_model(adaboost)

model_name = 'AdaBoost'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

```
call function

gradientboosting = GradientBoostingClassifier()
y_pred, y_prob, testing_time = train_model(gradientboosting)

model_name = 'Gradient Boosting Classifier'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

```
call function

randomforest = RandomForestClassifier()
y_pred, y_prob, testing_time = train_model(randomforest)

model_name = 'Random Forest'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
decisiontree = DecisionTreeClassifier()
y_pred, y_prob, testing_time = train_model(decisiontree)

model_name = 'Decision Tree'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
logisticregression = LogisticRegression()
y_pred, y_prob, testing_time = train_model(logisticregression)

model_name = 'Logistic Regression'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
knn = KNeighborsClassifier()
y_pred, y_prob, testing_time = train_model(knn)

model_name = 'KNN'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
mlp = MLPClassifier()
y_pred, y_prob, testing_time = train_model(mlp)

model_name = 'MLP'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
gaussiannb = GaussianNB()
y_pred, y_prob, testing_time = train_model(gaussiannb)

model_name = 'GaussianNB'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

call function

```
y_pred, y_prob, testing_time = tensorflow()

model_name = 'Tensorflow FNN'
confusion_matrix_normalized(y_test, y_pred, model_name)
confusion_matrix_non_normalized(y_test, y_pred, model_name)
plot_roc_curve(y_test, y_prob, model_name, n_classes=len(np.unique(y_test)))
```

Note: Apabila ingin menambahkan parameter pada model machine learning yang ingin dilatih. Kalian bisa mengecek parameter yang bisa digunakan di website Scikit-Learn lalu masukan nama modelnya (example: scikit learn random

forest) maka nanti akan muncul semua parameter termasuk guide nya. Lalu apabila ingin menambahkannya maka kalian bisa masukan di bagian () (example: GaussianNB(masukan_parameter)

2.4 Convert Machine Learning

Jika sudah sampai pada tahap ini, artinya teman-teman semua sudah berhasil membuat model machine learning. Nah di tahap ini, model tersebut akan dikonversi menjadi header c agar dapat dijalankan di microcontroler seperti portentah. Namun perlu diketahui tidak semua machine learning bisa dikonversi, hanya xgboost, decision tree, random forest, dan logistic regression yang bisa dikonversi. Berikut kode nya.

- **Perlu diingat bahwasanya program ini hanya akan berhasil apabila kalian menjalankan program call function terlebih dahulu dengan cara menyesuaikan model yang kalian pilih. Contohnya jalankan terlebih dahulu call function untuk xgboost, setelah proses train selesai dan hasil berupa nilai dan plot keluar, jalankan program ini ini untuk model xgboost. Maka hasil model akan langsung dikonversi ke header c. Begitu juga berlaku untuk model machine learning lainnya.**

conversion

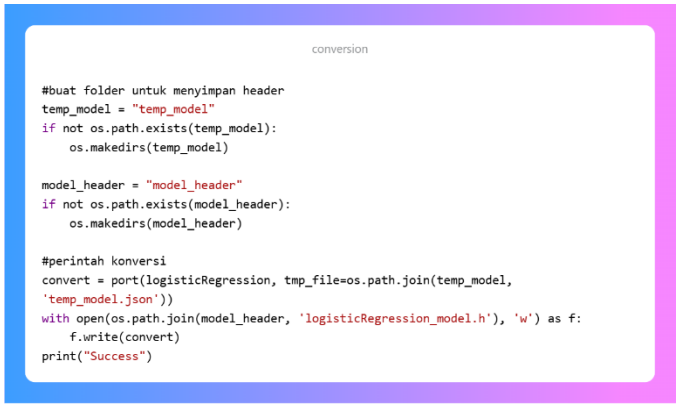
```
#buat folder untuk menyimpan header
temp_model = "temp_model"
if not os.path.exists(temp_model):
    os.makedirs(temp_model)

model_header = "model_header"
if not os.path.exists(model_header):
    os.makedirs(model_header)

#perintah konversi
convert = port(xgboost, tmp_file=os.path.join(temp_model, 'temp_model.json'))
with open(os.path.join(model_header, 'xgboost_model.h'), 'w') as f:
    f.write(convert)
print("Success")
```

conversion

```
#buat folder untuk menampung header
temp_model = "temp_model"
if not os.path.exists(temp_model):
    os.makedirs(temp_model)

model_header = "model_header"
if not os.path.exists(model_header):
    os.makedirs(model_header)

#perintah konversi
convert = port(randomforest, tmp_file=os.path.join(temp_model, 'temp_model.json'))
with open(os.path.join(model_header, 'randomForest_model.h'), 'w') as f:
    f.write(convert)
print("Success")
```

conversion

```
#buat folder untuk menyimpan header
temp_model = "temp_model"
if not os.path.exists(temp_model):
    os.makedirs(temp_model)

model_header = "model_header"
if not os.path.exists(model_header):
    os.makedirs(model_header)

#perintah konversi
convert = port(decisiontree, tmp_file=os.path.join(temp_model, 'temp_model.json'))
with open(os.path.join(model_header, 'decisionTree_model.h'), 'w') as f:
    f.write(convert)
print("Success")
```



2.5 Catatam

Perlu diingat bahwasanya untuk menjalankan program dengan benar. Pengguna harus dapat memahami struktur nya terlebih dahulu seperti langkah-langkah dan algoritma dari proses develop machine learning. Program yang sudah dimasukan dijalankan satu per satu jangan pernah berbarengan untuk dapat mengecek errornya di bagian mana. Berikut langkah yang harus selalu pengguna ikuti agar program berjalan baik.

1. Header
   Masukan header yang diperlukan dan jalankan dalam 1 cell (masukan saja semua header yang saya berikan walaupun hanya ingin mencoba xgboost saja. Hal ini dijadikan sebagai template pustaka yang kita butuhkan untuk memproses 12 machine learning berbeda)

2. Train function
   Masukan train function dan jalankan dalam 1 cell (untuk tensorflow gunakan train function yang sudah saya berikan khusus)

3. Plot Function
   Masukan plot function dan jalankan dalam 1 cell

4. Call Function
   Masukan call function dan jalankan dalam 1 cell

5. Convert to C
   Masukan convert function dan jalankan dalam 1 cell (Proses ini akan berhasil apabila model yang dikonversi sesuai dengan model yang dilatih sebelumnya. Jadi apabila ingin mengkonversi xgboost, pastikan mengeksekusi call function untuk xgboost lalu eksekusi convert function xgboost apabila sudah selesai)

   Semua itu dilakukan berurutan dan jangan berbarengan. Ikut langkah satu per satu. Apabila masih terjadi error, maka ulangi dari bagian header. Intinya apabila ingin melakukan training maka kalian bisa melakukan nya langsung secara berbarengan. Namun apabila ingin melakukan konversi (khusus konversi saja) maka kalian harus melakukan nya satu per satu dan tidak berbarengan. Contohnya seperti ini

   - **Call Function XGBoost -> Convert Function XGBoost**
   - **Call Function Decision Tree -> Convert Function Decision Tree**

   Tidak bisa seperti ini
   - **Call function XGBoost -> Call Function Decision Tree -> Convert Function XGBoost**

   Kalau seperti ini bagian convert akan mengalami error, kalau pun berhasil convert function malah akan eksekusi hasil dari decision tree atas nama xgboost.

   Untuk link guide yang mungkin bisa dijadikan pedoman, kalian bisa cek di sini
   scikit-learn: machine learning in Python — scikit-learn 1.6.0 documentation

   Kalau untuk install pustaka kalian bisa ikut guide seperti ini, contoh ingin install xgboost kalian bisa cari di google seperti ini. Install Anaconda XGBoost