

VERSION 1.0
AGUSTUS 2024



PEMBELAJARAN MESIN

MACHINE LEARNING WEB MODEL DEPLOYMENT – MODUL 6

DISUSUN OLEH:
MOCHAMAD RAHINA BINTANG PAMBAYUN
CHOIRUL SEPTYONO

DIAUDIT OLEH:
YUFIS AZHAR, S.KOM, M.KOM.

LAB. INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

PEMBELAJARAN MESIN

PERSIAPAN MATERI

1. Pengenalan tentang penggunaan model machine learning dengan antarmuka web
2. Pengenalan dan penggunaan package manager PDM
3. Pengenalan dan penggunaan streamlit sebagai framework antarmuka web
4. Penerapan kode menggunakan streamlit untuk membangun website yang melayani model machine learning mahasiswa

TUJUAN

Modul ini bertujuan untuk memberikan pemahaman tentang bagaimana menyajikan model machine learning sebagai layanan web menggunakan streamlit. Mahasiswa akan belajar bagaimana mengimplementasikan model machine learning dengan aplikasi web.

TARGET MODUL

Mahasiswa dapat memahami konsep dari materi modul, mengimplementasikan model machine learning dengan aplikasi web menggunakan streamlit, dan memahami penggunaan PDM sebagai project manager.

PERSIAPAN SOFTWARE/APLIKASI

1. Python (version ≥ 3.10)
2. PDM
3. Streamlit
4. Model H5
5. Vscode

MATERI POKOK

Deploying the Model

Dari semua model yang sudah dibuat, kita terbiasa untuk mendapatkan hasil prediksi melalui kode kita secara langsung. Orang awam belum tentu bisa melakukan setup yang kompleks mulai dari persiapan data hingga bisa melakukan prediksi dari data tersebut. Agar model tersebut bisa digunakan oleh orang awam, kita perlu membuat sebuah aplikasi antarmuka mudah digunakan. Salah satu cara agar bisa digunakan oleh pengguna awam adalah membuat aplikasi web.

Untuk mempermudah proses pembuatan aplikasi web untuk layanan machine learning, kita akan menggunakan framework streamlit. Streamlit adalah sebuah framework yang berfokus pada pengolahan data pada web berbahasa python. Streamlit menyediakan komponen-komponen yang bisa kita pakai untuk memprediksi data baru sesuai masukan dari pengguna. Selain itu, agar mempermudah dalam pembuatan aplikasi kita akan menggunakan PDM. PDM adalah sebuah tool untuk mengatur package. Konsep PDM mirip dengan venv tetapi kita bisa memilih versi python yang digunakan, mengurangi konflik versi pada pustaka yang diinstall, dan mempermudah proses pengembangan aplikasi. Namun sebelum

mulai membuat aplikasi webnya, kita persiapan preprocessing data dan model agar bisa dipakai di aplikasi web nanti.

Preprocessing dan Menyimpan Model

Pada saat kita melakukan training data terutama pada data tabular dan teks, kita sering melakukan preprocessing pada data training. Misal pada data tabular, kita biasa melakukan encoding, normalization atau standardization. Jika kita ingin melakukan prediksi dengan data baru, kita perlu mengubah data tersebut agar punya fitur yang sama seperti saat training. Misal pada data kategori kita harus tau bagaimana nilai encoding pada data training. Lalu jika kita mengimplementasikan data standardization misal seperti standardscaler, data baru juga harus mempunyai skala yang sama seperti data training.

Untuk data kategori kita bisa mengatur nilainya pada input di web. Namun untuk standardisasi atau normalisasi data, kita perlu melakukan langkah ekstra agar dapat mengubah data baru mempunyai data dengan skala atau standardisasi yang sama dengan data training. Cara paling sederhana agar bisa mendapatkan skala yang sama seperti data training yaitu dengan menyimpan variabel standardisasi dengan menggunakan pustaka joblib.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

import joblib
joblib.dump(scaler, "scaler.joblib")
```

Dari file yang sudah disimpan ini, nanti kita gunakan pada aplikasi streamlit ketika pengguna sudah memasukkan semua data lalu kita standardisasi data menggunakan file standardisasi tersebut. Data yang distandardisasi menggunakan file tersebut akan memiliki skala yang sama dengan data training sehingga data baru bisa diprediksi dengan tepat. Untuk data teks, jika kita menggunakan representasi teks seperti TFIDF kita juga perlu mengaplikasikan hal yang sama seperti standard scaler.

Selain proses scaler, proses tokenizer juga perlu disimpan dalam bentuk file joblib. Caranya pun sama seperti kita mengubah scaler menjadi file joblib. Tinggal diimplementasikan pada variabel tokenizer pada file klasifikasi teks.

Setelah preprocessing data baru, kita perlu memasukkan data tersebut pada model. Namun ketika kita menggunakan aplikasi web tidak mungkin kita melakukan training terlebih dahulu setelah memasukkan data baru. Model yang sudah dilatih bisa kita simpan dan tensorflow sudah menyediakan cara untuk menyimpan model yang sudah dilatih. Untuk menyimpan model yang sudah dilatih caranya seperti berikut:

```
model.save("tabular.h5")
```

PDM

PDM adalah tool untuk manajemen proyek dan pustaka. Tool ini mirip seperti conda namun lebih powerful. Instalasi pustaka lebih mudah karena jika terjadi konflik versi pada pustaka yang diinstall, PDM akan mengaturnya agar tidak terjadi konflik lagi. Pustaka yang diinstall juga secara otomatis dicatat beserta versi pustaka yang digunakan, sehingga mudah ketika orang lain ingin menggunakan aplikasi kita. Proses instalasi pustaka juga lebih cepat karena ada global cache sehingga tidak harus mendownload

ulang pustaka ketika kita ingin membuat aplikasi baru. Silakan install PDM dengan membuka powershell lalu masukkan perintah :

(Invoke-WebRequest -Uri <https://pdm-project.org/install-pdm.py> -UseBasicParsing).Content | python -

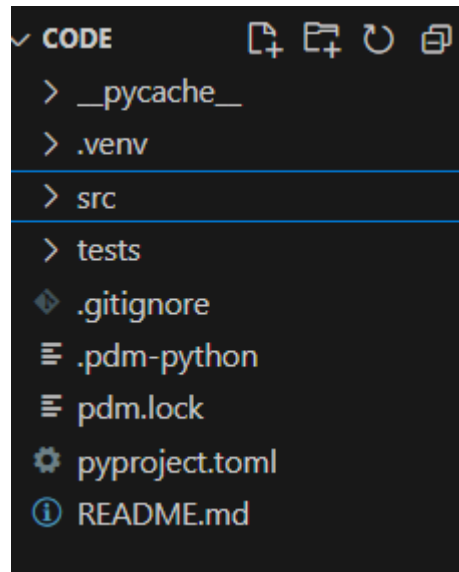
Jika PDM sudah diinstall maka kita siap untuk membuat sebuah projek baru untuk aplikasi web machine learning. Silakan masuk ke folder anda di Vscode, lalu buka terminal dan masukkan perintah “pdm init”. Setelah itu anda akan diberikan prompt untuk memilih python interpreter yang terinstall pada PC anda. Pastikan untuk projek ini anda menggunakan python dengan versi **diatas atau sama dengan 3.10**. Pada projek ini saya menggunakan versi 3.10. pada opsi nomor 5

```
PS [redacted] \code> pdm init
Creating a pyproject.toml for PDM...
Please enter the Python interpreter to use
0. cpython@3.9 ( [redacted] \python3.bat)
1. cpython@3.9 ( [redacted] \python.BAT)
2. cpython@3.10 ( [redacted] sions\3.10.11\python3.10.exe)
3. cpython@3.10 ( [redacted] sions\3.10.11\python310.exe)
4. cpython@3.10 ( [redacted] sions\3.10.11\python3.exe)
5. cpython@3.10 ( [redacted] sions\3.10.11\python.exe)
6. cpython@3.9 ( [redacted] ions\3.9.13\python3.9.exe)
7. cpython@3.9 ( [redacted] ions\3.9.13\python39.exe)
8. cpython@3.9 ( [redacted] ions\3.9.13\python3.exe)
9. cpython@3.9 ( [redacted] ions\3.9.13\python.exe)
10. cpython@3.9 ( [redacted] s\python3.9.bat)
11. cpython@3.9 ( [redacted] s\python39.bat)
Please select (0): 5
```

Setelah itu anda diminta untuk mengisi data lainnya. Anda bisa membiarkan secara default dengan tekan enter pada keyboard atau anda bisa ubah sesuai dengan kebutuhan anda. Setelah itu anda bisa menggunakan project anda.

```
Virtualenv is created successfully at [redacted] \code\.venv
Project version (0.1.0):
Do you want to build this project for distribution(such as wheel)?
If yes, it will be installed by default when running `pdm install`. [y/n] (n):
Author name (irultyo):
Author email (irul.tyo.7@gmail.com):
Python requires('*' to allow any) (==3.10.*):
```

PDM secara default akan membuat beberapa file dan folder secara otomatis. Kita akan menggunakan folder `src` sebagai tempat menyimpan kode yang akan dibuat. Lalu ada file bernama **pyproject.toml** yang berisi pustaka yang kita install pada proyek yang dibuat dan kita bisa menambahkan script untuk mempermudah kita menjalankan aplikasi kita.

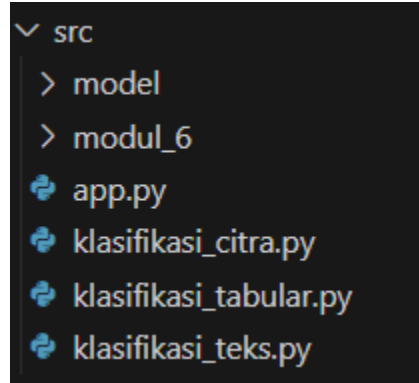


Instalasi pustaka menggunakan PDM tidak terlalu berbeda dengan menggunakan PIP. Pastikan terminal anda berada pada direktori tempat proyek web anda berada. Jika masih belum yakin, bisa coba aktivasi venv di direktori dengan cara menjalankan perintah berikut pada terminal powershell `“./.venv/Scripts/activation.ps1”`. Lalu untuk menginstall pustaka yang digunakan bisa menggunakan perintah `“pdm add <nama pustaka>”`. Pada aplikasi ini, kita membutuhkan beberapa pustaka untuk diinstall yaitu:

- Streamlit (versi 1.37.0)
- Tensorflow
- Joblib
- Scikit-learn

Streamlit

Streamlit adalah framework web python yang berfokus pada data. Dengan framework ini, kita bisa membuat antarmuka yang mudah digunakan oleh pengguna awam. Pada direktori aplikasi web yang dibuat, kita menggunakan folder `src` sebagai tempat untuk menyimpan file aplikasi. Untuk aplikasi utama kita menggunakan file `app.py`. Lalu ada file `klasifikasi_citra.py`, `klasifikasi_tabular.py`, `klasifikasi_teks.py`, dan folder `model` untuk menempatkan model dan scaler.



Sebagai contoh kita buat aplikasi untuk klasifikasi masing-masing data. Jika anda nanti ingin membuat satu klasifikasi saja, kita hanya menggunakan file app.py saja agar lebih mudah. Pada file app.py ini berisi navigasi yang mengarahkan ke tiga halaman aplikasi kita.

```

app.py
src > app.py > ...
1  import streamlit as st
2
3  pg = st.navigation([
4      st.Page("klasifikasi_tabular.py", title="Klasifikasi Tabular"),
5      st.Page("klasifikasi_citra.py", title="Klasifikasi Citra"),
6      st.Page("klasifikasi_teks.py", title="Klasifikasi Teks"),
7  ])
8  pg.run()

```

Untuk menjalankan aplikasi streamlit kita bisa masukkan perintah “streamlit run src/app.py”. Dengan menggunakan PDM, kita bisa membuat sebuah script custom untuk menjalankan perintah ini. Perintah custom tersebut bisa kita masukkan pada file pyproject.toml. Sebagai contoh kita membuat sebuah script dengan menggunakan perintah start untuk menjalankan perintah menjalankan aplikasi streamlit. Kita tinggal menambahkan pada baris terakhir di file pyproject.toml seperti berikut:

```

[tool.pdm.scripts]
start = "streamlit run ./src/app.py"

```

Dengan menambahkan script tersebut, kita tinggal menjalankan perintah “pdm run start” untuk menjalankan aplikasi streamlit. Lalu pada masing masing file kita buat antarmuka untuk input dan logika klasifikasinya.

Klasifikasi Tabular

Untuk klasifikasi tabular menggunakan referensi dari simulasi modul 1. Terdapat dua fitur yang digunakan oleh model untuk menghasilkan luaran label biner. Untuk memberi judul pada halaman web, kita gunakan komponen title dari streamlit. Pustaka streamlit mempunyai banyak komponen untuk input. Untuk input fitur klasifikasi ini menggunakan slider dengan range pada fitur1 (-1, 2.5) dan fitur1 (-1, 1.5). Range fitur ini sudah disesuaikan dengan nilai terkecil dan terbesar dari kedua fitur. Jika fitur dataset

memiliki range yang berbeda dari contoh ini, silakan ubah menyesuaikan range nilai dari dataset anda. Streamlit juga mempunyai komponen input lain seperti file, textbox, selectbox, dan lain lain. Silakan dipelajari untuk jenis input lain [disini](#).

```
st.title("Klasifikasi Tabular")
f1 = st.slider("Fitur 1", -1.0, 2.5)
f2 = st.slider("Fitur 2", -1.0, 1.5)
```

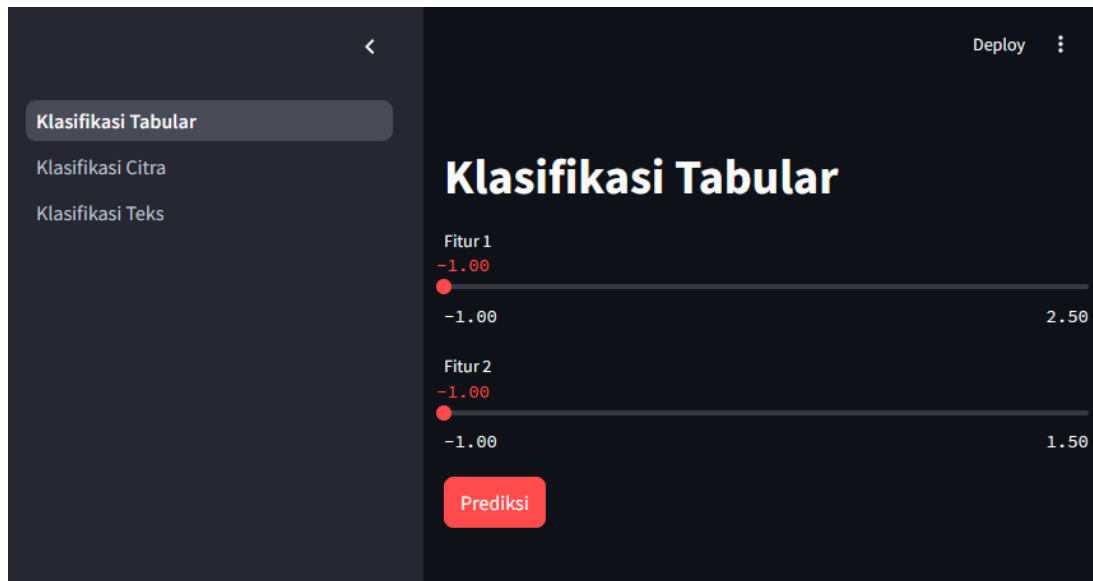
Kemudian kita buat sebuah fungsi untuk melakukan prediksi. Alur untuk memprediksi data baru adalah membuat sebuah list yang berisi data berurutan seperti data training. Misal disini kita sudah membuat dua variabel untuk f1 dan f2, maka bentuk data adalah [f1, f2]. Karena kita melatih model dalam batch, kita perlu mengubah list menjadi list 2D dengan cara memasukkan list data tadi dalam list, hasil akhirnya jadi [[f1,f2]]. Dari data baru yang diinput pengguna, kita ubah data tersebut menjadi dalam bentuk standardisasi atau normalisasi. Dari file standardisasi yang sudah disimpan sebelumnya menggunakan pustaka joblib, kita load file tersebut lalu kita lakukan transformasi data baru dengan cara yang sama seperti ketika melakukan transform data waktu training. Data yang sudah ditransformasi sudah siap untuk dilakukan prediksi oleh model dengan fungsi predict dari model yang sudah dilatih. Load model yang sudah dilatih lalu panggil fungsi predict dan masukkan data ke parameter. Fungsi akan mengembalikan sebuah nilai 0 atau 1 menyesuaikan dengan hasil prediksi oleh model.

```
def prediction():
    scaler = joblib.load(Path(__file__).parent / "model/scaler.joblib")
    model = tf.keras.models.load_model(Path(__file__).parent /
    | "model/tabular.h5")
    data = scaler.transform([[f1,f2]])
    result = (model.predict(data) > 0.5).astype("int32")
    return result[0][0]
```

Kemudian untuk menjalankan prediksi kita buat sebuah tombol untuk menjalankan fungsi prediction. Pada kode berikut, kita menjalankan fungsi prediksi ketika tombol prediksi ditekan. Pada bagian with st.spinner... berfungsi untuk menampilkan animasi spinner ketika melakukan prediksi. Animasi spinner ini akan hilang ketika proses prediction selesai dan muncul label dari data yang dimasukkan.

```
if st.button("Prediksi", type="primary"):
    st.subheader("Hasil prediksi: ")
    classes = ["Label 1", "Label 2"]
    with st.spinner('Memproses data untuk prediksi..'):
        result = prediction()
    st.write(classes[result])
```

Antarmuka yang sudah dibuat untuk klasifikasi tabular seperti berikut:



Ketika kita ubah nilainya lalu tekan tombol prediksi, maka hasilnya sebagai berikut:



Klasifikasi Teks

Proses klasifikasi untuk data teks ini tidak berbeda jauh dengan klasifikasi data tabular. Sempelnya kita perlu membuat sebuah input teks, lalu teks tersebut diubah dalam bentuk token, diubah lagi dengan fungsi `pad_sequences`, lalu kita panggil fungsi untuk memprediksi teks tersebut menggunakan model yang sudah dilatih. Model simulasi pada yang digunakan pada percobaan ini adalah pada simulasi modul 3. Sehingga untuk file tokenizer dan model diambil dari simulasi modul 3.

Input yang digunakan adalah textbox. Streamlit menyediakan input text ini menggunakan fungsi `text_input`.

```
st.title("Klasifikasi Teks")
text = st.text_input("Teks")
```

Lalu untuk proses mengubah data teks baru sama seperti pada simulasi modul 3. Tetapi tokenizer yang digunakan menggunakan file joblib hasil dari proses tokenizer ketika training model. Setelah teks sudah ditokenisasi, data diubah dengan fungsi `pad_sequence` dari tensorflow. Pastikan untuk paramter `pad_sequence` ini sama seperti pada waktu training model. Data yang sudah diproses tinggal dilakukan prediksi dengan memanggil fungsi prediksi dari model yang sudah dilatih.

```
def prediction():
    tokenizer = joblib.load(Path(__file__).parent /
                             "model/tokenizer.joblib")
    model = tf.keras.models.load_model(Path(__file__).parent /
                                         "model/teks.h5")
    sequences = tokenizer.texts_to_sequences(text)
    pad_seq = tf.keras.preprocessing.sequence.pad_sequences(
        sequences,
        maxlen=10,
        padding='post')
    result = (model.predict(pad_seq) > 0.5).astype("int32")
    return result[0][0]
```

Dengan logika yang sama seperti pada klasifikasi data tabular, kita buat tombol untuk menjalankan fungsi prediksi seperti berikut:

```
if st.button("Prediksi", type="primary"):
    st.subheader("Hasil prediksi: ")
    classes = ["negatif", "positif"]
    with st.spinner('Memproses teks untuk prediksi..'):
        result = prediction()
    st.write("Teks: "+text)
    st.subheader("Hasil prediksi: ")
    st.write(classes[result])
```

Berikut adalah hasil ketika sudah melakukan prediksi ke teks baru:

Klasifikasi Teks

Teks

Prediksi

Hasil prediksi:

Teks: The film is very good

Hasil prediksi:

positif

Klasifikasi Citra

Pada klasifikasi citra, kita menggunakan input file citra. Dari file citra yang diunggah, kita ubah preprocess data tersebut dengan menggunakan fungsi dari tensorflow. Alur preprocessing mulai dari load citra yang diunggah, lalu ubah citra menjadi array, lalu ubah dimensi array, dan terakhir kita lakukan prediksi. Kita tidak perlu menambahkan proses normalisasi secara manual karena kita sudah menambahkan layer normalisasi pada model secara langsung. Model yang digunakan pada klasifikasi citra ini adalah model simulasi modul 2.

Untuk menambahkan input file, kita bisa menggunakan komponen file_uploader dari streamlit. File yang diupload perlu dibatasi hanya untuk file berekstensi png atau jpg saja untuk menghindari file yang tidak sesuai

```
st.title("Klasifikasi Citra")
upload = st.file_uploader(
    'Unggah citra untuk mendapatkan hasil prediksi',
    type=['png', 'jpg'])
```

Seperti yang dijelaskan sebelumnya, citra yang diunggah perlu melewati proses pengubahan citra ke bentuk array. Setelah citra yang diunggah sudah dipreprocess, kita load model klasifikasi citra kita dan lakukan prediksi data citra yang diunggah untuk mendapatkan hasil prediksi.

```
def predict():
    class_names = ["peach", "pomegranate", "strawberry"]

    img = tf.keras.utils.load_img(upload, target_size=(300, 300))
    img_array = tf.keras.utils.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)

    model = tf.keras.models.load_model(Path(__file__).parent /
    "model/citra.h5")
    output = model.predict(img_array)
    score = tf.nn.softmax(output[0])
    return class_names[np.argmax(score)]
```


Dengan logika yang sama seperti pada kedua klasifikasi sebelumnya, kita buat tombol untuk menjalankan fungsi prediksi seperti berikut:

```
if st.button("Predict", type="primary"):
    if upload is not None:
        st.image(upload)
        st.subheader("Hasil prediksi: ")
        with st.spinner('Memproses citra untuk prediksi..'):
            result = predict()
        st.write(result)
    else:
        st.write("Unggah citra terlebih dahulu!!")
```

Berikut adalah hasil percobaan prediksi citra stawberry:

Klasifikasi Citra


Unggah citra untuk mendapatkan hasil prediksi




Drag and drop file here

Limit 200MB per file • PNG, JPG, JPEG

Browse files

 fresh_strawberry_2.jpg 15.9KB ×

Predict




Hasil prediksi:

strawberry

Reka Ulang Projek

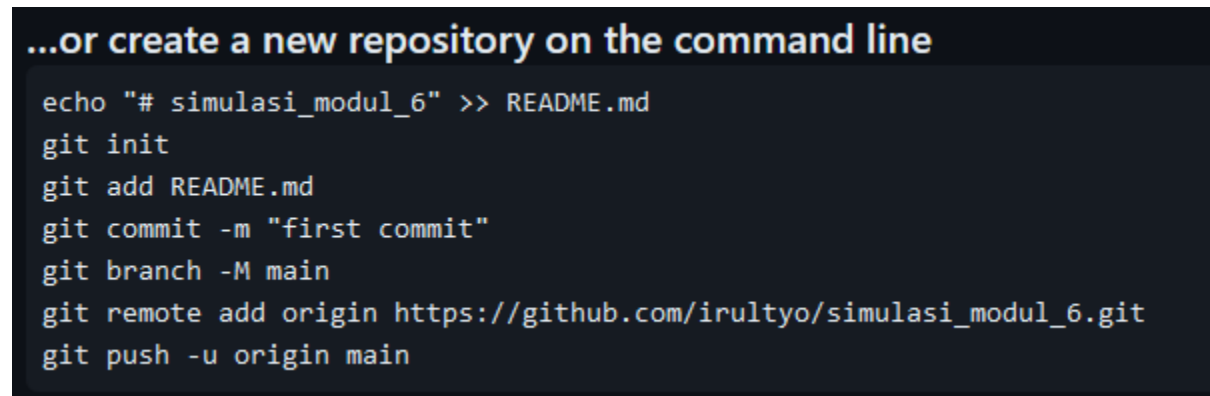
Aplikasi klasifikasi sudah selesai dibuat. Agar orang lain bisa mencoba aplikasi ini sendiri, kita bisa unggah projek ini ke github. Namun sebelum kita unggah ke github, pastikan folder model tidak ikut diunggah ke github dan silakan berikan alternatif untuk mendownload model atau biarkan orang lain menggunakan model mereka sendiri dengan kode yang sudah kita buat. Cara agar folder model tidak ikut diunggah ke

github dengan cara menambahkan path model ke file .gitignore. Tambahkan path folder model pada baris terakhir di file .gitignore seperti berikut:



```
.gitignore
164  src/model/*
```

Untuk mempermudah unggah data ke github, silakan buat repository di github terlebih dahulu, lalu copy paste perintah yang disediakan oleh github. Tapi pada baris **“git add readme.md”** silakan ganti dengan **“git add .”** dan pastikan terminal anda berada pada folder proyek anda.



```
...or create a new repository on the command line

echo "# simulasi_modul_6" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/irultyo/simulasi_modul_6.git
git push -u origin main
```

Silakan jika anda ingin mengunduh repository bisa melalui link [ini](#). Setelah anda selesai mendownload, buka terminal pada folder proyek ini lalu jalankan perintah “pdm install” untuk menginstall pustaka yang dibutuhkan. Setelah itu buat folder model di dalam folder src dan masukkan file untuk model dan joblib seperti materi yang sudah dijelaskan sebelumnya.

LATIHAN PRAKTIKUM

Agar hasil pada minggu praktikum lebih bagus, pada pekan latihan ini silakan lakukan instalasi PDM dan coba untuk membuat project streamlit dengan PDM. Ketika aplikasi streamlit berjalan, pastikan agar halaman utama menampilkan nama dan NIM lalu tunjukkan ke asisten masing-masing. Jika mengalami kendala ketika instalasi, silakan bertanya ke asisten atau teman yang sudah berhasil.

LEMBAR KERJA PRAKTIKUM

Buatlah sebuah tampilan untuk model yang dibuat dari modul 1 hingga modul 5. Pilih salah satu model saja yang dibuat untuk dibuat tampilan klasifikasinya. Silakan buat sekreatif mungkin memanfaatkan komponen yang disediakan oleh streamlit. Lalu jelaskan ke asisten bagaimana proses pembuatan proyek hingga dapat melakukan klasifikasi dengan model yang sudah dilatih.

KRITERIA & DETAIL PENILAIAN

Detail	Bobot Nilai
Pemahaman Materi	40
Ketepatan Jawaban	30
Program berjalan tanpa error	15
Tugas pekan materi	15