Tugas Modul Introduction to Machine Learning

Dalam sebuah rapat di Komunitas Peduli Diabetes Kota Ayodya, pengurus komunitas sedang mencari cara untuk meningkatkan deteksi dini diabetes pada anggota keluarga komunitas mereka. Mereka mengetahui bahwa diabetes adalah salah satu penyakit kronis paling umum dan memiliki dampak signifikan terhadap kualitas hidup. Tim medis pendamping komunitas ineign menggunakan machine learning untuk membantu mereka memprediksi kemungkinan anggota keluarga peserta yang mengalami diabetes berdasarkan pengukuran diagnostik.

Untuk mencapai tujuan ini, tim medis bekerja sama dengan sebuah tim data scientist untuk mengumpulkan data dari pusat kesehatan. Data yang dikumpulkan meliputi informasi tentang jumlah kehamilan (untuk perempuan), konsentrasi glukosa plasma, tekanan darah diastolik, ketebalan lipatan kulit trisep, insulin serum 2 jam, indeks massa tubuh (BMI), fungsi watisan diabetes, dan usia. Setelah data dikumpulkan, tim data scientist menggunakan dataset untuk melatih model machine learning mereka. Model ini nantinya akan diuji coba pada data pasien pusat kesehatan. Anda adalah data scientist pemula yang direkrut untuk membantu tim dalam mengembangkan model machine learning tersebut.

Instruksi dari pimpinan tim data scientis yang Anda terima adalah sebagai berikut:

Selamat datang untuk rekan-rekan baru tim data scientist. Berikut adalah instruksi yang dapat saya berikan untuk membuat model machine learning menggunakan algoritma Support Vector Machine (SVM) dan Random Forest (RF) :

1. Pahami dataset: Sebelum memulai pembuatan model, pastikan Anda memahami dataset yang akan digunakan. Periksa apakah terdapat missing value.
2. Tentukan fitur dan target: Sebagai awalan, kita akan menggunakan semua fitur yang sudah ditentukan oleh tim dokter. Tentukan fitur dan target yang akan digunakan untuk melatih model. Untuk nilai parameter **random_state** pada pemisahan dataset disesuaikan dengan dua digit terakhir nomor pegawai Anda (random state: **dua digit terakhir NPM**).
3. Buat model SVM: Gunakan library scikit-learn untuk membuat model SVM. Anda diperbolehkan untuk menyesuaikan parameter seperti C dan gamma agar model dapat bekerja dengan baik pada dataset. Untuk nilai parameter **random_state** pada pemisahan dataset disesuaikan dengan dua digit terakhir nomor pegawai Anda (random state: **dua digit terakhir NPM**).
4. Buat model Random Forest: Gunakan library scikit-learn untuk membuat model Random Forest. Anda diperbolehkan untuk menyesuaikan parameter seperti max_depth dan n_estimators agar model dapat bekerja dengan baik pada dataset. Untuk nilai parameter **random_state** disesuaikan dengan dua digit terakhir nomor pegawai Anda (random state: **dua digit terakhir NPM**).
5. Evaluasi model: Setelah membuat kedua model, lakukan evaluasi untuk mengetahui performa masing-masing model. Semoga instruksi ini dapat membantu anggota baru dalam membuat model machine learning menggunakan algoritma SVM dan Random Forest. Jika ada pertanyaan atau kesulitan, jangan ragu untuk menghubungi Dosen dan Asisten Dosen
6. Buat Interface Streamlit: Setelah model tersimpan, buat antarmuka yang dapat digunakan oleh tim medis dan pengurus Komunitas Peduli Diabetes Kota Ayodya untuk memprediksi kemungkinan seseorang terkena diabetes.

Dataset yang tersedia dalam bentuk CSV. Dataset dapat di download melalui link berikut : https://www.kaggle.com/datasets/mathchi/diabetes-data-set (https://www.kaggle.com/datasets/mathchi/diabetes-data-set)

- Tahap pertama adalah load dataset berdasarkan path di mana dataset disimpan.
- Karena dataset dalam bentuk file CSV, kita menggunakan fungsi `read_csv` dari Pandas.

In [1]:
```
import pandas as pd
import numpy as np

#load data
diabetes_csv = pd.read_csv(r'D:\dataset\diabetes.csv') ##Disesuaikan dengan tempat penyimpanan file csv Dataset diabet

#load dataset ke dalam dataframe
df_diabetes = pd.DataFrame(data = diabetes_csv, index = None)
df_diabetes
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

- Data yang hilang akan membuat model menjadi error, untuk itu kita perlu menanganinya.
- Kita dapat melakukan pengecekan pada data apabila ada data yang bernilai null, kosong, atau NaN dengan cara seperti berikut:

In [2]:
```
#mengecek data kosong, null, dan nan
print("data null\n", df_diabetes.isnull().sum())
print("\n")
print("data kosong \n", df_diabetes.empty)
print("\n")
print("data nan \n", df_diabetes.isna().sum())
```

```
data null
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64


data kosong
 False


data nan
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

- Dataset dapat kita deskripsikan secara statistik menggunakan fungsi `describe()` dan menghasilkan luaran seperti berikut:

In [3]:
```
df_diabetes.describe()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

2. Train-Test Split

- Sebelum kita membuat model dari dataset kita perlu membagi dataset menjadi set data training dan data testing.
- Data training merupakan bagian dari dataset yang digunakan untuk melatih algoritme pembelajaran mesin. Algoritme akan mempelajari ciri tiap class data dari masing-masing feature.
- Sedangkan test set adalah bagian dari dataset yang digunakan untuk pengujian. Test set dianggap sebagai set data yang belum pernah dilihat atau dipelajari oleh model machine learning kita.
- Secara umum rasio antara training set dan test set adalah 70%:30% dan 75%:25%.

In [4]:
```
#memuat library untuk train-test split dataset
from sklearn.model_selection import train_test_split

#memuat nilai fitur dalam variabel X, drop Outcome
#axis = 1 digunakan untuk menghapus kolom
X = df_diabetes.drop(columns=['Outcome'], axis = 1)

#memuat nilai label dalam variabel y
y = df_diabetes['Outcome']


#membuat variabel X_train, X_test, y_train, dan y_test untuk menampung hasil split
##nilai random state diganti dengan 2 digit npm terakhir
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)

print("bentuk X_train", X_train.shape)
print("bentuk X_test", X_test.shape)

print("bentuk y_train", y_train.shape)
print("bentuk y_test", y_test.shape)

print("y_train \n", y_train)
print("y_test \n", y_test)
```

```
bentuk X_train (576, 8)
bentuk X_test (192, 8)
bentuk y_train (576,)
bentuk y_test (192,)
y_train
 357    1
73     0
352    0
497    0
145    0
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 576, dtype: int64
y_test
 668    0
324    0
624    0
690    0
473    0
      ..
554    0
319    1
594    0
6      1
615    0
Name: Outcome, Length: 192, dtype: int64
```

3. Melatih Model, Melakukan Prediksi, dan Evaluasi Model

- Model dari algoritme dilatih menggunakan training set dan ada pengaturan parameter.
- Parameter yang diatur sesuai dengan algoritme yang digunakan. Untuk kali ini kita akan mencoba menggunakan model Support Vector Machine dan Random Forest.
- Setelah dilatih, kita melakukan prediksi terhadap test set dan menguji performa dari model yang sudah terbentuk.
- Berikut adalah contoh code dari pemanggilan model dan pelatihannya:

In [7]:
```
#import library untuk model machine learning
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

#membuat obyek model machine learning dan setting parameternya
##nilai random state diganti dengan 2 digit npm terakhir
SVM = SVC(C = 1, gamma= 0.01, random_state=42)
RF = RandomForestClassifier(max_depth=5, n_estimators=100, random_state=42)

#fungsi fit digunakan untuk melatih machine learning
SVM.fit(X_train, y_train)
RF.fit(X_train, y_train)
```

Out[7]:
```
RandomForestClassifier(max_depth=5, random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:
```
#membuat array untuk X baru yang akan diprediksi
X_new = np.array([[3, 197, 30, 19, 0, 44.8, 0.567, 55]])

print("X_new yang akan diprediksi", X_new.shape)

#prediksi label dari X baru
svm_predict = SVM.predict(X_new)
print("Label prediksi SVM", svm_predict)
rf_predict = RF.predict(X_new)
print("Label prediksi RF", rf_predict)
```

```
X_new yang akan diprediksi (1, 8)
Label prediksi SVM [0]
Label prediksi RF [1]

c:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, bu
t SVC was fitted with feature names
  warnings.warn(
c:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, bu
t RandomForestClassifier was fitted with feature names
  warnings.warn(
```

- Hasil prediksi tunggal seperti contoh sebelumnya belum bisa menentukan akurasi atau performa ketepatan model dalam memprediksi label suatu data.
- Evaluasi dapat kita lakukan menggunakan test set yang sudah kita simpan tadi.
- Langkah pengkodeannya adalah sebagai berikut:

In [9]:
```
#menggunakan fungsi predict untuk memprediksi label X_test
y_pred_svm = SVM.predict(X_test)
y_pred_rf = RF.predict(X_test)

print("Hasil prediksi SVM pada X_test:", y_pred_svm)
print("Hasil prediksi RF pada X_test:", y_pred_rf)
```

```
Hasil prediksi SVM pada X_test: [0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0]
Hasil prediksi RF pada X_test: [0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 1 1
 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0
 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1
 0 0 0 1 1 0 0]
```

In [11]:
```
#menggunakan fungsi score untuk mengukur akurasi prediksi model
print("Akurasi model SVM:", round(SVM.score(X_test, y_test), 3))
print("Akurasi model RF:", round(RF.score(X_test, y_test), 3))
```

```
Akurasi model SVM:  0.641
Akurasi model RF:  0.755
```

- Dapat kita lihat bahwa akurasi model Random Forest lebih baik dibandingkan SVM. Maka model yang sebaiknya dipilih untuk tugas klasifikasi adalah Random Forest.
- Setelah melakukan pelatihan dan pengujian model kita dapat menyimpan model menggunakan library Pickle.

In [13]:
```
#simpan model menggunakan library Pickle
import pickle

with open('rf_diabetes_model.pkl', 'wb') as f:
    pickle.dump((RF), f)

##File pickle(.pkl) akan tersimpan di folder yang sama dengan file notebook

print("Model RF berhasil disimpan")
```

```
Model RF berhasil disimpan
```

Dalam konteks pickle, dump berarti menyimpan objek Python(model) ke dalam file. Ini adalah bagian dari proses serialisasi di mana objek Python dikonversi menjadi format byte yang dapat disimpan ke dalam file.

Streamlit

- Sebelum membuat antarmuka, pastikan bahwa streamlit telah terinstalasi. Cek dengan membuka Anaconda Prompt dan ketik `pip install streamlit` .
- Setelah itu, buat file baru dengan ekstensi Python (.py) untuk antarmuka Streamlitm dan buatlah code berikut.

In [ ]:
```
import streamlit as st
import pandas as pd
import pickle
import os

#load model
model_directory = r'D:\models' ##diisi dengan path folder dimana file model berada

# Gunakan os.path.join() untuk menggabungkan direktori dan file model pickle
model_path = os.path.join(model_directory, 'rf_diabetes_model.pkl')

# Periksa apakah file ada di direktori yang ditentukan
if os.path.exists(model_path):
    try:
        #muat model dari file pickle
        with open(model_path, 'rb') as f:
            loaded_model = pickle.load(f)

        svm_model = loaded_model[0]
        rf_model = loaded_model[1]

        #bagian Streamlit App
        st.title("Prediksi Diabetes")

        st.write("Aplikasi ini digunakan untuk membantu memprediksi penyakit diabetes pada seseorang")

        pregnancies = st.slider("Pregnancies", min_value=0, max_value=17, step=1)
        glucose = st.slider("Glucose (mg/dL)", min_value=0.0, max_value=199.0, step=0.1)
        bloodPressure = st.slider("Blood Pressure (mmHg)", min_value=0, max_value=122, step=2)
        skinThickness =st.slider("Skin Thickness (mm)", min_value=0, max_value=99, step=2)
        insulin = st.slider("Insulin (µU/mL)", min_value=0, max_value=846, step=10)
        bmi = st.slider("BMI", min_value=0.0, max_value=67.1, step=0.1)
        diabetesPedigreeFunction = st.slider("Diabetes Pedigree Function", min_value=0.07, max_value=2.42, step=0.1)
        age = st.slider("Age", min_value=21, max_value=81, step=1)

        #prediksi diabetes berdasarkan input

        input_data = [[pregnancies, glucose, bloodPressure, skinThickness, insulin, bmi, diabetesPedigreeFunction, age

        if st.button("Prediksi!"):
```

```
            rf_model_prediction = rf_model.predict(input_data)
            outcome_names = {0: 'Tidak Diabetes', 1: 'Diabetes'}
            st.write(f"Orang tersebut diprediksi **{outcome_names[rf_model_prediction[0]]}** oleh RF")

    except Exception as e:
        st.error("Terjadi kesalahan: {e}")
else:
    print("File 'svm_rf_diabetes_model.pkl' tidak ditemukan di direktori")
```

- Buka terminal dengan CTRL+Shift+ , `kemudian ketik` conda activate base` untuk memastikan environment Anaconda sudah berjalan
- Untuk menjalankan projek Streamlit, ketik `streamlit run (path file)`
- Contoh : `streamlit run "D:\Kuliah\Semester 5\diabetes_streamlit.py"`
- Ketika dijalankan, secara otomatis akan otomatis terbuka tab baru di browser