Konteks Bisnis

Anda adalah seorang AI Engineer di perusahaan kami. Salah satu rumah sakit klien terbesar kami ingin mengimplementasikan sistem peringatan dini di UGD mereka untuk mengidentifikasi pasien yang berisiko tinggi mengalami sepsis, suatu kondisi medis yang mengancam jiwa.

Tujuannya adalah membangun sebuah layanan AI yang dapat menerima data vital dan hasil lab awal dari seorang pasien di UGD, lalu memberikan prediksi risiko sepsis secara *real-time*. Layanan ini akan membantu dokter memprioritaskan pasien dan memulai perawatan lebih awal, yang secara signifikan dapat meningkatkan hasil klinis.

Data yang Disediakan

Anda akan diberikan sebuah file sepsis_emr_data.csv yang berisi data rekam medis elektronik (EMR) historis dari pasien UGD yang telah dianonimkan.

Deskripsi Kolom:

- patient_id: ID unik untuk setiap episode kunjungan pasien.
- heart_rate: Denyut jantung (denyut per menit).
- respiratory_rate: Laju pernapasan (napas per menit).
- temperature: Suhu tubuh (derajat Celsius).
- wbc_count: Jumlah sel darah putih (*White Blood Cell count*).
- lactate_level: Kadar laktat dalam darah.
- age: Usia pasien dalam tahun.
- num_comorbidities: Jumlah penyakit penyerta yang dimiliki pasien.
- sepsis_risk: **Target Variable**. 1 jika pasien didiagnosis sepsis dalam 24 jam, 0 jika tidak.

Tugas Anda

Tugas ini dibagi menjadi empat bagian yang mencerminkan siklus hidup pengembangan dan penerapan model AI.

Bagian 1: Pemodelan Lanjutan (Advanced Modeling)

Bagian ini menguji kemampuan Anda untuk membangun model yang kuat dan andal.

1. **Preprocessing Data:** Lakukan pemrosesan data yang diperlukan, termasuk menangani nilai yang hilang dan melakukan penskalaan fitur (*feature scaling*).
2. **Membangun Model Individual:** Bangun dan latih setidaknya **dua jenis model klasifikasi yang berbeda**. Contoh:
    - Model A: Model berbasis pohon (*tree-based*) seperti **Gradient Boosting** (misalnya, XGBoost, LightGBM).

- Model B: Model **Artificial Neural Network (ANN)** sederhana menggunakan TensorFlow/Keras atau PyTorch.

3. **Ensemble Learning:** Gabungkan prediksi dari kedua model Anda menggunakan teknik **soft voting ensemble**. Anda harus mengambil probabilitas prediksi dari setiap model dan menghitung rata-rata terbobot untuk menghasilkan prediksi akhir. Jelaskan mengapa Anda memilih bobot tertentu (misalnya, 50/50 atau berdasarkan performa validasi silang).
4. **Evaluasi Model:** Evaluasi performa model *ensemble* akhir Anda menggunakan **k-fold cross-validation** untuk memastikan hasilnya andal. Laporkan metrik yang relevan seperti **AUC-ROC**, **Precision**, dan **Recall**.
5. **Menyimpan Artefak:** Simpan model *ensemble* yang telah dilatih (termasuk *preprocessor* dan model-model individualnya) ke dalam file.

Bagian 2: Pembuatan API Layanan Prediksi

Bagian ini menguji kemampuan Anda untuk mengubah model menjadi layanan yang fungsional.

1. **Membangun API:** Buat sebuah API web sederhana menggunakan **Flask** atau **FastAPI**.
2. **Membuat Endpoint Prediksi:** API harus memiliki satu *endpoint*, misalnya /predict, yang:
    - Menerima data vital dan lab satu pasien baru melalui metode POST dalam format **JSON**.
    - Memuat artefak model *ensemble* Anda.
    - Melakukan *preprocessing* pada data input.
    - Mengembalikan hasil prediksi dari model *ensemble* dalam format **JSON** yang jelas, contohnya: {"sepsis_risk_prediction": 1, "risk_probability": 0.78}.

Bagian 3: Containerization dan Deployment (MLOps)

Bagian ini menguji kemampuan Anda untuk memastikan solusi Anda portabel dan siap untuk di-deploy.

1. **Menulis Dockerfile:** Buat sebuah Dockerfile yang akan mengemas aplikasi API Anda.
2. **Konfigurasi Dockerfile:** Dockerfile tersebut harus melakukan semua langkah yang diperlukan untuk menjalankan aplikasi secara mandiri, termasuk:
    - Menggunakan *base image* Python yang sesuai.
    - Menyalin semua file yang diperlukan (kode aplikasi, file model, requirements.txt).
    - Menginstal semua dependensi Python.
    - Menjalankan aplikasi API Anda saat *container* dijalankan.

Hasil yang Diharapkan (Deliverables)

Harap kumpulkan semua hasil Anda dalam sebuah repositori Git (misalnya, di GitHub atau GitLab) dan kirimkan tautannya kepada kami. Repositori tersebut harus memiliki struktur yang rapi dan berisi:

1. **Kode Sumber:**
    - Kode untuk melatih model (misalnya, dalam sebuah Jupyter Notebook atau skrip Python).
    - Kode untuk aplikasi API Anda (Flask/FastAPI).
2. **Artefak Model:**
    - File-file model yang sudah dilatih.
3. **File Konfigurasi:**
    - Dockerfile yang fungsional.
    - requirements.txt yang mencantumkan semua dependensi.
4. **Dokumentasi (README.md):**
    - File README.md yang sangat jelas, berisi:
        - Deskripsi singkat tentang proyek dan pendekatan Anda.
        - Ringkasan performa model *ensemble* Anda.
        - **Instruksi langkah-demi-langkah** tentang cara membangun *image* Docker dan menjalankan *container* secara lokal.
        - Contoh perintah curl atau skrip Python untuk mengirim permintaan ke API Anda.
        - Penjelasan singkat tentang pilihan desain atau asumsi yang Anda buat.