Nama: Eirene Michella Tjhan
NIM: 2702256630

Code Link

https://drive.google.com/drive/folders/1UB3EGYDJSo0tncrBfdYb3d1FrpPvI1hp?usp=sharing

Halo selamat malam, di file ini akan saya lampirkan **Video Link, Essay,** beserta bukti **Test Case (screenshot)**

Video Link

https://drive.google.com/drive/folders/1QCl6FGGuhpafk0U-e5S07OAuyzzA8XdA?usp=sharing

Essay

Testing apa saja yang perlu Anda lakukan untuk memastikan bahwa model prediksi tingkat obesitas ini akan berjalan dengan baik dan tidak akan mengalami gangguan ketika berada di sistem produksi? Jelaskan rencana pengujian Anda secara sistematis dan sertakan contoh berdasarkan fitur dalam dataset obesitas, seperti bagaimana jika nilai pada Age, Weight, SMOKE, atau MTRANS dimanipulasi.

Tujuan testing adalah untuk memastikan model prediksi obesitas berjalan dengan benar, aman, dan stabil saat digunakan di sistem, serta mampu menangani berbagai jenis input tanpa error agar bisa diandalkan dalam pengambilan keputusan clinic.

1) Input Validation Test
Mengecek apakah data input sudah benar dari segi format, tipe, dan nilai. Jika data yang masuk salah, model bisa error atau menghasilkan prediksi yang tidak masuk akal. Validasi input mencegah sistem crash dan menjaga kualitas prediksi. Oleh karena itu, kita harus memastikan semua input yang masuk ke API sesuai dengan tipe dan rentang nilai yang valid.

Contoh:

- Age = -5 → Tidak valid. Usia tidak mungkin negatif.
- Weight = "berat banget" → Tidak valid. Harus berupa angka.
- SMOKE = "kadang" → Tidak valid. Hanya boleh "yes" atau "no".
- MTRANS = "teleport" → Tidak valid. Bukan opsi yang disediakan.

Ini juga harus difokuskan untuk memastikan setiap tahapan preprocessing (OrdinalEncoder, OneHotEncoder, StandardScaler) bekerja sesuai harapan. Maka kita bisa membuat data dummy yang ukurannya kecil dan spesifik untuk setiap jenis fitur (ordinal, nominal, numerik) dan memverifikasi output dari setiap transformer.

Contoh untuk consistency mapping:

1. Fitur FCVC (Ordinal):

- Input: ['Jarang', 'Sedang', 'Sering', 'Jarang']
- Expected Output (setelah ordinal encoding): [0, 1, 2, 0]

2. Fitur Gender (Nominal):
    - Input: ['Laki-laki', 'Wanita', 'Laki-laki']
    - Expected Output (setelah one-hot encoding): [[1, 0], [0, 1], [1, 0]]
3. Fitur Age (Numeric):
    - Input: [20, 30, 40]
    - Expected Output (setelah standard scaling): Nilai terstandardisasi yang memiliki mean 0 dan standar deviasi 1.

Solusi: Gunakan validasi dengan Pydantic agar API otomatis menolak input yang salah.

2) Extreme Value and Anomaly (Edge Case Test)
Menguji data anomali ke model untuk memastikan bahwa model dapat menangani input ekstrem tanpa error. Jadi kita memerikan nilai yang ekstrem, tidak valid, atau bermasalah pada fitur-fitur penting. Ini sangat penting untuk sistem produksi karena data dunia nyata jarang sempurna.

Contoh:
- Manipulasi Age:
    - Input: Age = 0 (bayi) atau Age = 150 (sangat tua).
    - Model harus memberikan prediksi yang masuk akal atau setidaknya tidak crash.
- Manipulasi Weight:
    - Input: Weight = 10 kg (sangat kurus) atau Weight = 300 kg (sangat obesitas).
    - Prediksi harus sesuai dengan nilai ekstrem ini (misalnya, Insufficient Weight atau Obesity Type III). Ini juga menguji apakah StandardScaler menangani outlier dengan baik.
- Manipulasi SMOKE:
    - Input: SMOKE = " " (string kosong) atau SMOKE = 'maybe' (nilai tidak dikenal/salah ketik).
    - Model harus menangani handle_unknown='ignore' di OneHotEncoder dengan baik, mungkin menghasilkan representasi nol untuk kategori yang tidak dikenal atau idealnya, sistem validasi input di lapisan aplikasi akan menangkap ini. Model tidak boleh crash.
- Manipulasi MTRANS:
    - Input: MTRANS = "-" atau MTRANS = 'Pesawat' (nilai tidak dikenal).
    - Model harus tetap berfungsi tanpa error berkat handle_unknown='ignore'. Prediksi untuk input yang tidak dikenal ini mungkin kurang akurat atau default ke kelas mayoritas.

Solusi: Tambahkan batas logis pada nilai-nilai fitur.
- $10 \leq \text{Age} \leq 100$
- $40 \leq \text{Weight} \leq 200$
- $1 \leq \text{FCVC} \leq 3$

3) Tampered Data Test (test data yang dimanipulasi)

Menguji apakah sistem tahan terhadap input yang sengaja dimanipulasi oleh user.Dalam dunia nyata, orang bisa saja memberikan jawaban yang tidak jujur untuk terlihat lebih sehat. Sistem harus bisa mencatat hal ini agar tidak menghasilkan analisis yang menyesatkan.

Contoh manipulasi:

- Mengatur SMOKE = "no" padahal merokok, untuk menghindari diagnosis.
- Memasukkan MTRANS = "Walking" padahal sebenarnya naik mobil, agar terlihat lebih sehat.

Solusi:

- Logging semua data input untuk keperluan audit.
- Bisa juga tambahkan “confidence score” atau edukasi pengguna untuk mengisi data dengan benar.

4) Functional Test

Menguji apakah prediksi model sesuai dengan logika dan data yang diberikan. Model harus memberikan hasil yang masuk akal. Jika input mengarah ke obesitas, maka hasilnya juga harus mengarah ke obesitas. Ini membuktikan bahwa model tidak hanya “asal tebak”.

Contoh Skenario:

- Kasus 1 (obesitas):
  Gender: 'Laki-laki', Age: 45, Height: 1.70, Weight: 120, family_history_with_overweight: 'ya', FAVC: 'ya', FCVC: 'Jarang', NCP: 'Berlebih', CAEC: 'Selalu', SMOKE: 'ya', CH2O: 'Kurang', SCC: 'tidak', FAF: 'Tidak Aktif', TUE: 'Sering', CALC: 'Sering', MTRANS: 'Mobil' → Expected prediction: Obesity Type III atau Obesity Type II (karena ini adalah tingkat risiko tertinggi yang harus terdeteksi).
- Kasus 2 (normal):
  Gender: 'Wanita', Age: 28, Height: 1.60, Weight: 55, family_history_with_overweight: 'tidak', FAVC: 'tidak', FCVC: 'Sering', NCP: 'Cukup', CAEC: 'Tidak pernah', SMOKE: 'tidak', CH2O: 'Banyak', SCC: 'ya', FAF: 'Sangat Aktif', TUE: 'Jarang', CALC: 'Tidak pernah', MTRANS: 'Jalan Kaki' → Expected prediction: Normal Weight

Untuk melatih modelnya, kita bisa mengambil sampel data di mana model membuat prediksi yang salah (berdasarkan hasil classification_report dan confusion_matrix sebelumnya) dan menganalisis mengapa kesalahan itu terjadi. Ini bisa memunculkan kebutuhan akan lebih banyak data untuk kelas tertentu atau fitur rekayasa.

Solusi: Buat data uji yang disusun manual (terkontrol) untuk melihat apakah model memprediksi dengan benar.

5) Performance Test

Menguji apakah API tetap cepat dan tidak lemot walaupun banyak yang mengakses. Kalau API digunakan oleh banyak user secara bersamaan, kita harus pastikan tetap responsif dan tidak crash. Ini sangat penting untuk aplikasi klinik atau publik. Jadi kita mengukur waktu yang dibutuhkan model untuk memprediksi satu atau sejumlah besar data. Melakukan pengujian beban (load testing) dengan banyak permintaan serentak jika model akan diakses oleh banyak pengguna.

Contoh:

- Kirim 100 permintaan secara bersamaan.

- Target waktu respon: di bawah 1 detik.
- Tidak boleh ada error 500 (server crash).

Solusi: Gunakan alat seperti Apache JMeter atau Locust untuk uji beban (load testing).

6) Integration Test
Memastikan bahwa preprocessing pipeline terintegrasi dengan baik dengan model klasifikasi (XGBoost) dan seluruh pipeline dapat memproses data dari input mentah hingga prediksi akhir. Jadi integration test ini penting untuk menguji apakah API bisa digunakan dengan lancar di aplikasi frontend/web. Kalau API tidak bisa menerima data dari form pengguna atau hasilnya tidak bisa dibaca oleh sistem lain, maka prediksi tidak akan bisa digunakan.

Contoh:
- Aplikasi form di web mengirim JSON ke endpoint /predict (API) dan hasilnya ditampikan ke user.
- Masukkan satu baris data mentah lengkap ke dalam xgb_pipeline.predict() dan pastikan outputnya adalah salah satu kategori NObeyesedad yang valid.

Solusi: Tes menggunakan Postman, Swagger /docs, atau dari frontend langsung untuk memastikan aur komunikasi berhasil.

7) Monitoring dan Logging (Post-deployment Test)
Mengecek performa model setelah digunakan secara nyata. Setelah model berjalan di sistem nyata, perlu dicek apakah masih akurat dan apakah ada input aneh atau error yang sering muncul.

Contoh:
- Log data input dan output setiap prediksi.
- Jika banyak input mendadak berubah drastis (misalnya banyak data dengan Weight = 0), itu pertanda error atau manipulasi.
- Cek jika distribusi input berubah drastis (tanda model perlu retraining).

Solusi: Buat sistem logging & alert jika model mulai error atau prediksinya aneh. Lalu buat mekanisme retraining jika model mulai tidak akurat.

Anda telah berhasil membangun dan mendistribusikan model machine learning ke dalam sistem informasi klinik untuk memprediksi tingkat obesitas seseorang berdasarkan data gaya hidup. Setelah berjalan selama beberapa bulan, Anda mulai memperhatikan bahwa hasil prediksi mulai menunjukkan penurunan akurasi. Jelaskan apa yang dimaksud dengan model drift dan data drift dalam konteks sistem prediksi obesitas ini?

Waktu model preksis obesity level kita menunjukan penurunan akurasi setelah berjaan beberapa bulan di industri, ini adalah indikasi kuat terjadinya drift. Ada dua jenis drift utama yang mengakibatkan ini, Data Drift dan Model Drift. Keduanya saling terkait dan seringkali merupakan penyebab satu sama lain.

Data Drift
Data Drift mengacu pada perubahan karakteristik data input (fitur-fitur yang digunakan model untuk membuat prediksi) seiring waktu. Dalam konteks prediksi obesity level, ini berarti distribusi dari

fitur-fitur gaya hidup (FCVC, FAVC, FAF, MTRANS, etc.) yang masuk ke model saat ini sudah berbeda secara signifikan dengan distribusi data yang digunakan model saat dilatih.

1. **Perubahan demografi pasien**
   Klinik mungkin mulai menarik populasi pasien yang lebih tua, atau dari wilayah geografis dengan kebiasaan makan/aktivitas fisik yang berbeda. Misalnya, rata-rata Age pasien yang masuk sekarang adalah 50 tahun, padahal saat pelatihan data Age dominan di 30-an.

2. **Perubahan tren gaya hidup**
   Mungkin ada kampanye kesehatan publik yang berhasil meningkatkan kesadaran tentang makan sehat, sehingga frekuensi konsumsi sayuran (FCVC) atau asupan air (CH2O) pasien secara umum meningkat. Atau, sebaliknya, ada tren peningkatan penggunaan teknologi (TUE) dan penurunan aktivitas fisik (FAF) karena gaya hidup yang lebih menetap.

3. **Perubahan lingkungan**
   Pandemi atau krisis ekonomi dapat mengubah kebiasaan transportasi (MTRANS), konsumsi makanan (FAVC, CAEC), atau tingkat stres yang memengaruhi berat badan.

4. **Perubahan data collection method**
   Jika ada perubahan dalam cara data Height atau Weight diukur (misalnya, beralih dari timbangan manual ke digital yang lebih akurat), ini bisa menyebabkan pergeseran distribusi nilai numerik.

Ini penting untuk diperhatikan karena model di *train* berdasarkan pola lama, jadi kalau pola input berubah, prediksi bisa jadi tidak akurat.

Model Drift
Model Drift ini adalah perubahan hubungan antara fitur input dan variabel target (tingkat obesitas itu sendiri) seiring waktu. Artinya, meskipun mungkin tidak ada perubahan besar pada data input, definisi atau "konsep" obesitas itu sendiri (atau bagaimana gaya hidup memengaruhinya) telah bergeser. Model yang dulunya akurat dalam memahami hubungan ini, kini tidak lagi.

1. **Perubahan kriteria medis/gaya hidup**
   Mungkin ada penelitian baru yang menunjukkan bahwa ambang batas tertentu untuk BMI (yang mendasari tingkat obesitas saat ini) atau kombinasi kebiasaan tertentu kini dianggap lebih berisiko atau kurang berisiko dari sebelumnya. Model yang ada sekarang, dilatih berdasarkan definisi lama, bukan yang baru.

2. **Faktor eksternal**
   Mungkin ada obat baru yang memengaruhi metabolisme atau berat badan, atau perubahan pola makan populer yang mengubah bagaimana FAVC atau NCP berkorelasi dengan NObeyesedad. Model sekarang tidak pernah melihat efek dari faktor-faktor baru ini saat pelatihan.

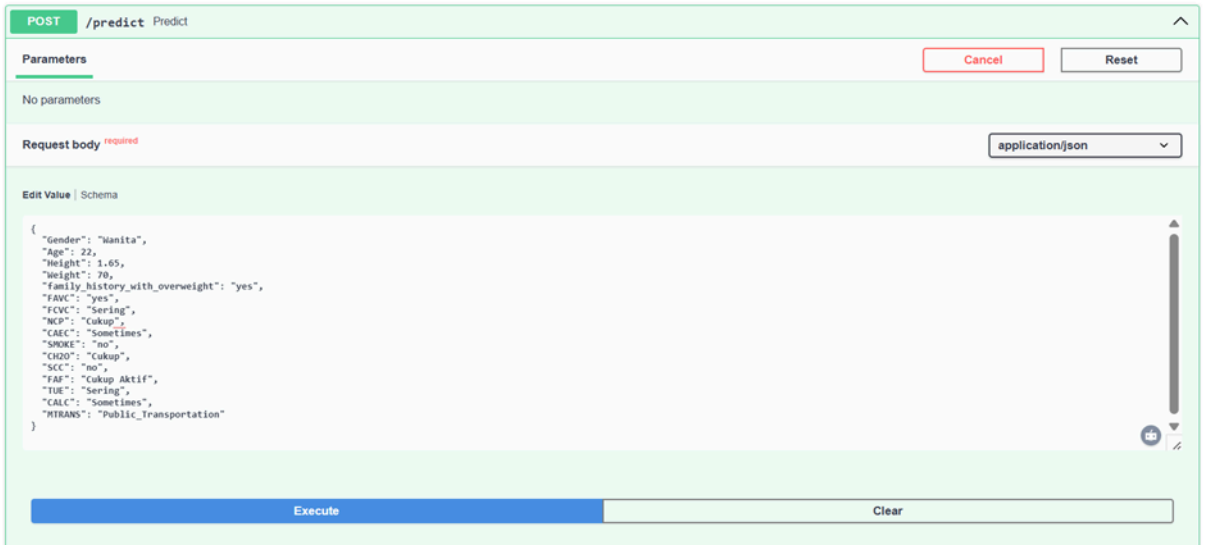
3. **Adaptasi masyarakat**
   Masyarakat mungkin telah beradaptasi dengan kampanye kesehatan sebelumnya, sehingga korelasi antara, misalnya, FAF dan NObeyesedad berubah. Orang mungkin menjadi lebih sadar tentang dampak aktivitas fisik, sehingga dampaknya pada obesitas menjadi lebih kompleks atau tidak sesederhana dulu.

4. **Bias yang mucul**
   Model mungkin mulai mengembangkan bias terhadap kelompok tertentu jika hubungan fitur-target berubah lebih cepat untuk sub-populasi tertentu.

Hubungan antara Data Drift dan Model Drift
Data drift seringkali menjadi pemicu model drift. Jika data yang masuk berubah (data drift), dan model tidak dilatih ulang untuk memahami pola baru tersebut, maka kemampuan model untuk

memprediksi target dengan benar akan menurun (model drift). Namun, model drift juga bisa terjadi tanpa perubahan drastis pada input data jika hubungan inheren antara fitur dan target berubah karena faktor eksternal atau "konsep" yang mendasarinya bergeser. Untuk mengatasi kedua jenis drift ini, sistem ML di produksi perlu memiliki strategi pemantauan drift dan pelatihan ulang (retraining) model secara berkala menggunakan data terbaru.

Test Case #1



Prediksi Obesitas

Isi form di bawah untuk memprediksi kategori obesitas.

Profil & Riwayat

Jenis Kelamin
Perempuan

Umur
22
10 80

Tinggi badan (m)
1.65
1.00 2.00

Berat badan (kg)
70.00
20.00 180.00

Ada riwayat obesitas di keluarga?
Ya

Pola Makan & Minum

Sering konsumsi makanan tinggi kalori?
Ya

Frekuensi makan sayur
Sering

Frekuensi makan harian
Cukup

Sering ngemil di luar waktu makan?
Kadang-kadang

Konsumsi air putih
Cukup

Aktivitas Fisik & Lainnya

Aktivitas fisik mingguan
Cukup Aktif

Durasi penggunaan teknologi setiap hari
Sering

Moda transportasi utama
Transportasi Umum

Kondisi Khusus

Merokok?
Tidak

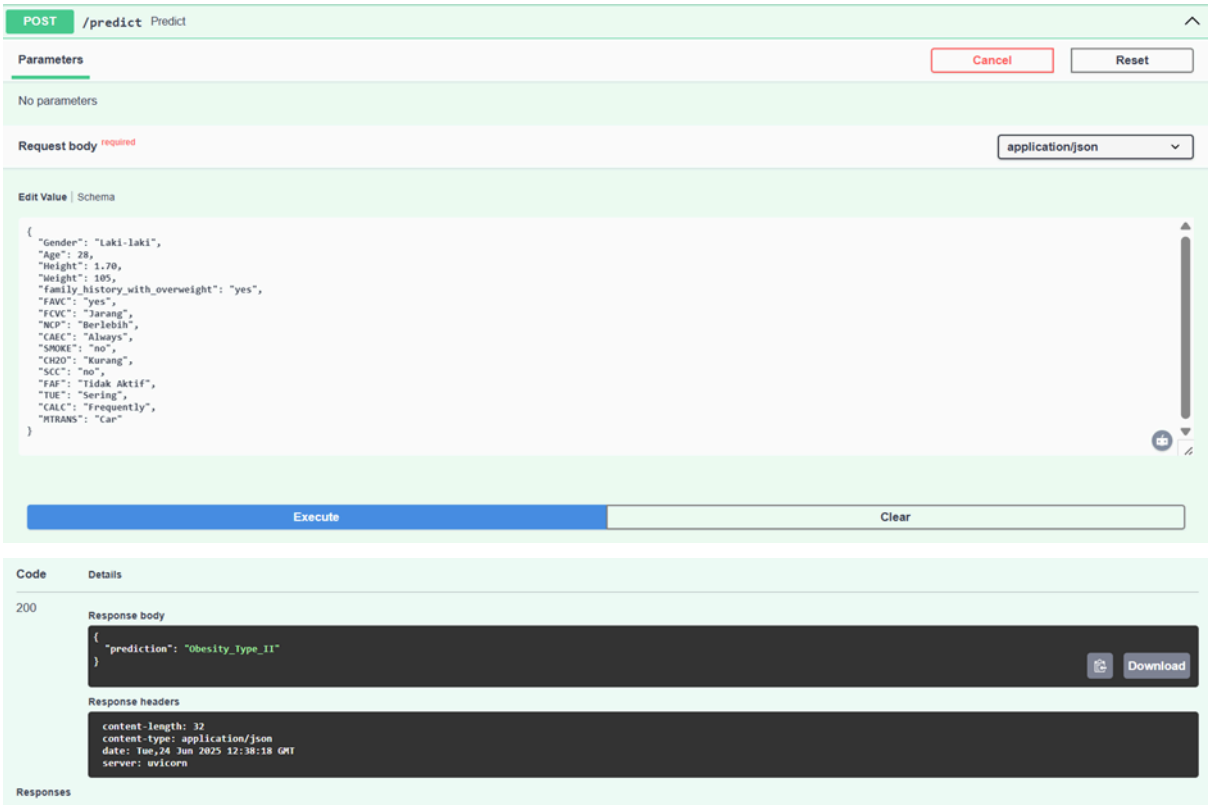
Konsumsi alkohol
Kadang-kadang

Apakah memantau asupan kalori perhari?
Tidak

Prediksi

Hasil Prediksi: Overweight_Level_I

Test Case #2


POST
/predict Predict
Parameters
Cancel
Reset
No parameters
Request body required
application/json
Edit Value | Schema
{
"Gender": "Laki-laki",
"Age": 28,
"Height": 1.70,
"Weight": 105,
"family_history_with_overweight": "yes",
"FAVC": "yes",
"FCVC": "Jarang",
"NCP": "Berlebih",
"CAEC": "Always",
"SMOKE": "no",
"CH2O": "Kurang",
"SCC": "no",
"FAF": "Tidak Aktif",
"TUE": "Sering",
"CALC": "Frequently",
"MTRANS": "Car"
}
Execute
Clear
Code
Details
200
Response body
{
"prediction": "Obesity_Type_II"
}
Download
Response headers
content-length: 32
content-type: application/json
date: Tue,24 Jun 2025 12:38:18 GMT
server: uvicorn
Responses

Prediksi Obesitas

Isi form di bawah untuk memprediksi kategori obesitas.

Profil & Riwayat

Jenis Kelamin
Laki-laki

Umur
28
10 80

Tinggi badan (m)
1.70
1.00 2.00

Berat badan (kg)
105.00
20.00 180.00

Ada riwayat obesitas di keluarga?
Ya

Pola Makan & Minum

Sering konsumsi makanan tinggi kalori?
Ya

Frekuensi makan sayur
Jarang

Frekuensi makan harian
Berlebih

Sering ngemil di luar waktu makan?
Selalu

Konsumsi air putih
Kurang

Aktivitas Fisik & Lainnya

Aktivitas fisik mingguan
Tidak Aktif

Durasi penggunaan teknologi setiap hari
Sering

Moda transportasi utama
Mobil

Kondisi Khusus

Merokok?
Tidak

Konsumsi alkohol
Sering

Apakah memantau asupan kalori perhari?
Tidak

Prediksi

Hasil Prediksi: Obesity_Type_II