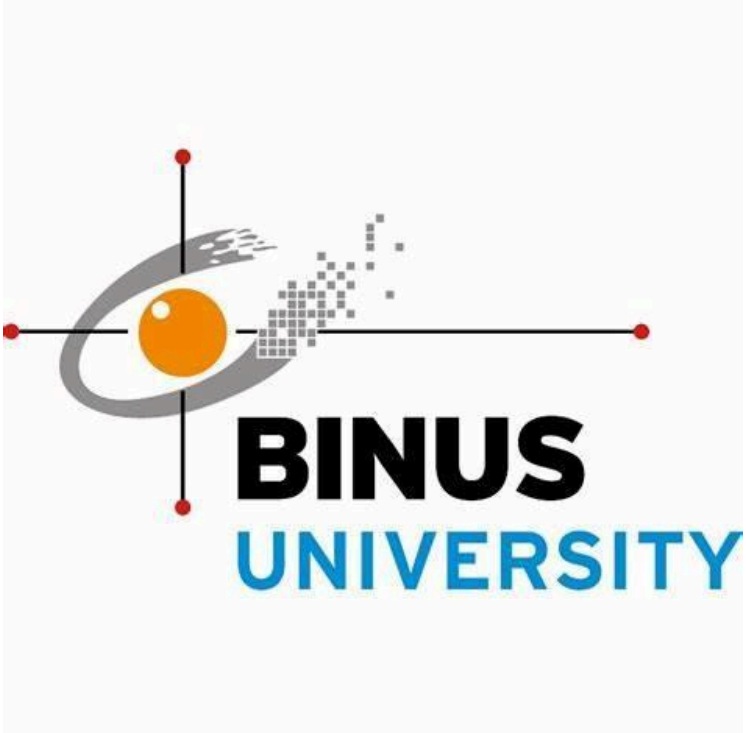
Pemantauan dan Pemeliharaan Model Machine Learning Dalam Data dan Model Drift untuk Menjamin Akurasi Prediksi Obesitas

Model Deployment



Disusun oleh :

Aaron Winston Gho (2702210522)

25 Juni 2025

Bina Nusantara University Anggrek Campus

Jalan Raya Kebon Jeruk Nomor 27, RT.1/RW.9, Kemanggisan, Kecamatan Palmerah

Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta 11530

2025

Problem 1:

[LO 1, LO 4 – 15 Points] 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.

Answer:

- **Why Do We Need Testing?**

Sebelum sebuah model machine learning dapat diandalkan dalam sistem produksi, perlu dilakukan serangkaian pengujian yang komprehensif untuk memastikan model dapat berjalan dengan baik, stabil, dan memberikan hasil prediksi yang akurat. Pengujian ini bertujuan untuk mengidentifikasi potensi masalah sejak dini, memastikan bahwa model mampu menangani variasi data nyata, serta memverifikasi integrasi antar komponen sistem. Dengan demikian, testing menjadi langkah krusial untuk menjamin kualitas dan keberlangsungan layanan prediksi obesitas di lingkungan klinik atau aplikasi pengguna akhir.

- **What Testing We Need to Do ?**

Untuk memastikan model prediksi obesitas machine learning yang udah saya deploy ini dapat berfungsi dengan optimal di lingkungan produksi, beberapa jenis pengujian perlu dilakukan, antara lain:

1. Input Validation Testing

a. Alasan

Testing Input atau Validasi input penting karena balik lagi kita harus pastiin kalau sistem tidak menerima data yang salah atau aneh yang bisa menyebabkan error runtime, hasil prediksi tidak valid, atau bahkan crash. Hal ini membuat model menjaga kestabilan dan keamanan aplikasi saat dipakai pengguna nyata. Selain itu testing ini juga memastikan kalau streamlit atau app kita benar benar dapat digunakan oleh user.

b. Contoh

Ada beberapa cara yang sebenernya bisa dilakukan seperti:

1. Pertama bisa dengan kirim request dengan nilai Age = -5 atau Weight = 300
2. Kedua bisa juga dengan input SMOKE = "sometimes" yang tidak valid.

Kalau sistem bener seharusnya menolak input ini dan memberikan error validasi.

2. Robustness Testing terhadap Nilai Ekstrim

a. Alasan

Testing Kedua Adalah Testing untuk menguji model dengan data pada batas minimal dan maksimal agar sistem tahan terhadap nilai outlier atau ekstrim. Pengujian ini memastikan model tahan terhadap data pada batas minimal dan maksimal (outlier) sehingga model tetap dapat bekerja tanpa gagal saat pengguna memasukkan nilai ekstrim yang valid secara domain.

b. Contoh

1. Uji dengan input Age = 1 dan Age = 100

2. Bisa juga dengan berat badan sangat rendah (misal 10 kg) dan sangat tinggi (misal 250 kg).

Kalau Model bener seharusnya model nantinya akan memproses tanpa error dan prediksi masih masuk akal.

3. Performance Testing

a. Alasan

Testing yang berikutnya adalah testing performa dari app kita. Testing performa dibutuhkan agar sistem backend dan model dapat melayani banyak permintaan secara bersamaan dengan waktu respons cepat, tanpa bottleneck atau crash, sehingga pengalaman pengguna tetap baik dan sistem scalable.

b. Contoh

Menyimulasikan 100 request prediksi per detik ke API dan ukur waktu respons rata-rata serta kita juga measure apakah dia performanya drop setiap responsnya.

4. Output Validation Testing

a. Alasan

Testing berikutnya adalah Validasi output penting untuk memastikan model tidak menghasilkan prediksi ngawur atau di luar kelas target. Ini menjaga kualitas dan kepercayaan terhadap hasil prediksi.

b. Contoh

Masukkan input yang sudah diketahui hasil prediksi benar berdasarkan data validasi dan pastikan model mengeluarkan output yang sama atau sangat mendekati, dengan probabilitas yang valid.

5. Integration Testing

a. Alasan

Testing integrasi memastikan bahwa seluruh komponen sistem seperti frontend (Streamlit), backend API (FastAPI), dan model machine learning dapat bekerja sama tanpa masalah, termasuk error handling yang baik agar pengguna mendapat pengalaman lancar.

b. Contoh

Kita bisa menguji pengiriman data dari frontend ke backend, terima hasil prediksi, dan juga uji respons error saat input tidak valid dikirim.

6. Feature Manipulation Testing

a. Alasan

Testing ini menguji sensitivitas model terhadap perubahan fitur utama, memastikan model bereaksi secara logis dan sesuai domain medis terhadap variasi input. Hal ini membantu validasi interpretabilitas dan relevansi model.

b. Contoh

Untuk Feature manipulation testing ini as simple as

1. Age dari 20 ke 70 tahun dan cek perubahan prediksi.
2. Weight dari 60 kg ke 120 kg.

3. SMOKE dari "no" ke "yes".
4. MTRANS dari "Public_Transportation" ke "Automobile" atau "Others".
   Amati apakah prediksi berubah sesuai ekspektasi.
5. Dll

- **Testing Planning**

1. Integration Testing

Menurut saya, pengujian integrasi adalah yang paling utama karena memastikan seluruh bagian sistem mulai dari frontend, backend API, sampai model machine learning bisa bekerja sama secara lancar. Kalau integrasi ini gagal, maka walaupun modelnya bagus, pengguna tidak akan bisa mendapatkan hasil prediksi yang benar atau bahkan aplikasi bisa error. Jadi, ini adalah fondasi agar sistem bisa dipakai dengan mulus dan jadi prioritas pertama dalam pengujian.

2. Input Validation Testing

Setelah integrasi, kita harus pastikan data yang masuk ke sistem valid. Validasi input sangat penting supaya tidak ada data yang aneh atau salah masuk, yang bisa bikin model gagal prediksi atau aplikasi crash. Dengan validasi ini, kita juga memastikan pengguna frontend, misalnya lewat form Streamlit, hanya bisa memasukkan data yang sesuai aturan, sehingga menjaga kestabilan sistem.

3. Output Validation Testing

Selanjutnya, kita perlu memeriksa apakah hasil prediksi model sudah benar dan masuk akal, terutama saat diuji dengan data yang kita sudah tahu jawabannya. Ini penting untuk memastikan model tetap dapat dipercaya dan kualitas prediksi terjaga, sehingga hasil yang keluar dari sistem benar-benar bermanfaat untuk pengambilan keputusan klinis.

4. Robustness Testing terhadap Nilai Ekstrim

Kadang-kadang, pengguna bisa memasukkan data yang ekstrim, misalnya usia sangat muda atau sangat tua, atau berat badan yang sangat tinggi. Kita perlu memastikan sistem dan model tahan dengan kondisi seperti ini tanpa error atau crash, dan tetap memberikan prediksi yang masuk akal. Ini penting agar aplikasi tetap andal dalam kondisi nyata yang variatif.

5. Feature Manipulation Testing

Selain itu, kita harus mengecek bagaimana model merespons perubahan fitur utama seperti usia, berat badan, kebiasaan merokok, atau moda transportasi. Dengan mengubah fitur-fitur ini secara sistematis, kita dapat memastikan model memahami pengaruhnya terhadap risiko obesitas dan prediksi yang diberikan masuk akal sesuai domain kesehatan.

6. Performance Testing

Terakhir, kita harus menguji performa sistem terutama di sisi backend dan API, apakah mampu menangani banyak request secara bersamaan tanpa melambat atau gagal. Ini penting

untuk pengalaman pengguna agar tidak terganggu, terutama saat aplikasi digunakan secara luas di lingkungan klinik atau publik.

- **Testing On Our Features**

1. Manipulasi Fitur Age

Misalnya kita coba masukkan nilai usia yang sangat berbeda, seperti 20 tahun dan 70 tahun, dengan data lain yang sama. Biasanya, risiko obesitas cenderung berbeda di antara kedua usia ini karena metabolisme dan gaya hidup berubah seiring bertambahnya usia. Dengan memanipulasi fitur ini, kita cek apakah model mampu menangkap perubahan risiko tersebut dan mengeluarkan prediksi yang logis, misalnya prediksi risiko obesitas lebih tinggi pada usia 70 tahun dibanding 20 tahun.

2. Manipulasi Fitur Weight

Berat badan adalah fitur yang sangat menentukan dalam prediksi obesitas. Jika kita ubah berat badan dari 60 kg menjadi 120 kg dengan fitur lain tetap, kita mengharapkan model memprediksi risiko obesitas meningkat. Dengan pengujian ini, kita pastikan model tidak "buta" terhadap perubahan berat badan dan bisa memberikan hasil yang sesuai secara klinis.

3. Manipulasi Fitur SMOKE

Status merokok bisa mempengaruhi metabolisme dan faktor kesehatan lain. Kita coba ubah nilai SMOKE dari "no" ke "yes" dan lihat apakah prediksi obesitas berubah secara wajar. Pengujian ini membantu memastikan model menganggap kebiasaan merokok sebagai faktor yang relevan dalam menentukan risiko obesitas.

4. Manipulasi Fitur MTRANS

Jenis moda transportasi yang digunakan berhubungan dengan tingkat aktivitas fisik. Dengan mengubah MTRANS dari "Public_Transportation" ke "Automobile" atau ke "Others" (misalnya jalan kaki, sepeda), kita cek apakah model bisa mengadaptasi prediksi berdasarkan aktivitas yang diasosiasikan dengan moda transportasi tersebut. Misalnya, pengguna yang lebih banyak berjalan kaki atau bersepeda cenderung memiliki risiko obesitas lebih rendah, dan prediksi model harus mencerminkan hal ini.

- **Kesimpulan**

Kesimpulannya, pengujian yang sistematis dan menyeluruh sangat penting untuk memastikan model prediksi obesitas bekerja secara andal, akurat, dan tahan terhadap berbagai kondisi nyata di sistem produksi. Dengan menguji mulai dari integrasi, validasi input/output, ketahanan terhadap data ekstrim, sensitivitas fitur, hingga performa, kita dapat meminimalkan risiko kegagalan dan meningkatkan kepercayaan pengguna terhadap sistem.

Problem 2:
[LO 1, LO 4 – 10 Points] 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?

Answer:

- **Why Does The Accuracy Drop ?**

Setelah model prediksi obesitas berjalan dalam sistem selama beberapa waktu, kita mungkin melihat akurasi hasil prediksi mulai menurun. Hal ini bisa terjadi karena kondisi dunia nyata yang berubah misalnya pola gaya hidup, kebiasaan makan, atau karakteristik populasi pengguna yang mengalami perubahan dari data yang awalnya dipakai untuk melatih model. Perubahan ini menyebabkan model yang sudah dilatih tidak lagi "pas" dengan data terbaru, sehingga performanya menurun.

- **What is Data Drift & Model Drift**

Model Drift adalah kondisi di mana kemampuan model untuk memprediksi dengan benar mulai menurun karena hubungan antara data fitur dan target yang dipelajari model sudah berubah. Contohnya seperti gini awalnya model mempelajari bahwa konsumsi makanan tinggi kalori sangat berpengaruh terhadap obesitas. Namun, jika dalam beberapa bulan terakhir banyak orang mulai mengubah pola makan menjadi lebih sehat, hubungan ini berubah. Model lama yang tidak memperhitungkan perubahan ini jadi kurang akurat memprediksi.

Data Drift adalah perubahan distribusi atau karakteristik data input yang masuk ke model dibandingkan data yang digunakan saat pelatihan. Contohnya saat pelatihan, sebagian besar pengguna menggunakan "Public_Transportation" sebagai transportasi, tapi sekarang mayoritas pengguna beralih ke "Automobile" atau "Others". Data baru ini berbeda dari data pelatihan sehingga model mungkin kurang cocok atau salah prediksi.

- **Data Drift & Model Drift On Obesity**

Model drift terjadi pada case ini ketika hubungan antara fitur gaya hidup (seperti pola makan, aktivitas fisik) dengan tingkat obesitas berubah seiring waktu, sehingga model lama yang dibuat berdasarkan data sebelumnya tidak lagi akurat memprediksi kondisi sekarang. Sedangkan data drift berarti distribusi data input yang masuk ke sistem berbeda dari data pelatihan, misalnya populasi pengguna sekarang memiliki karakteristik usia, berat badan, atau kebiasaan berbeda, sehingga model kesulitan menghasilkan prediksi yang tepat. Solusinya adalah dengan rutin memantau performa model dan distribusi data, lalu melakukan retraining atau pembaruan model menggunakan data terbaru agar model tetap relevan dan akurat sesuai kondisi terkini.

Hasil Deployment Bisa diAkses di link ini:
Frontend & Backend , untuk link backend jangan lupa /docs untuk mau coba predict