**LAPORAN UAS**

**KLASIFIKASI TINGKAT RESIKO DIABETES MENGGUNAKAN ALGORITMA RANDOM FOREST**

****

**Dosen pengampu:**

Nama : Dr. Eka Mala Sari Rochman S.Kom., M.Kom

NIP : 19800325200312

**Dibuat Oleh :**

Bilqizah Rahma Ilayya Syahdewi – 230411100030  
Sirojul Izzeh – 230411100131  
Alaikassalam - 230411100133

Aditya Aprillio Pratama – 230411100201

Ivanka Ramadhan – 230411100202

**PRODI TEKNIK INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**2025**

# DAFTAR ISI

[**DAFTAR** **ISI** i](#_Toc201509355)

[**DAFTAR** **GAMBAR** ii](#_Toc201509356)

[**DAFTAR** **TABEL** iii](#_Toc201509357)

[1. Pendahuluan 1](#_Toc201509358)

[2. Dasar Teori 1](#_Toc201509359)

[2.1 Diabetes Mellitus (DM) 1](#_Toc201509360)

[2.2 *Machine* Learning untuk Klasifikasi 2](#_Toc201509361)

[2.3 Algoritma Random Forest 2](#_Toc201509362)

[2.4 Metrik Evaluasi Model 3](#_Toc201509363)

[3. Metode 4](#_Toc201509364)

[3.1 Pengumpulan Data 4](#_Toc201509365)

[3.2 Pra-pemrosesan Data 5](#_Toc201509366)

[3.3 Pembangunan Model 6](#_Toc201509367)

[3.4 Evaluasi Model 6](#_Toc201509368)

[4. Implementasi 7](#_Toc201509369)

[4.1 Hasil *Coding* (Python - Jupyter Notebook - UAS-KK.ipynb) 7](#_Toc201509370)

[4.2 Backend (Python - app.py) 12](#_Toc201509371)

[4.3 Frontend (HTML - index.html) 14](#_Toc201509372)

[4.4 Tampilan *User Interface* dan Alur Penggunaan 20](#_Toc201509373)

[5. Hasil dan Kesimpulan 22](#_Toc201509374)

[5.1 Hasil Prediksi dan Evaluasi Model 22](#_Toc201509375)

[5.2 Kesimpulan 26](#_Toc201509376)

# DAFTAR GAMBAR

[**Gambar 1. 1** Tampilan User Interface 22](#_Toc201509109)

[**Gambar 1. 2** Tombol Prediksi 23](#_Toc201509110)

[**Gambar 1. 3** Tampilan hasil Prediksi 24](#_Toc201509111)

# DAFTAR TABEL

[**Tabel 1. 1** Hasil coding Python 10](#_Toc201509242)

[**Tabel 1. 2** Backend Python 15](#_Toc201509243)

[**Tabel 1. 3** Frontend HTML 17](#_Toc201509244)

[**Tabel 1. 4** Matrix Evaluasi Model 26](#_Toc201509245)

[**Tabel 1. 5** Confusion Matrix 27](#_Toc201509246)

## 1. Pendahuluan

Diabetes Mellitus (DM) adalah salah satu tantangan kesehatan global yang paling signifikan di abad ke-21. World Health Organization (WHO) melaporkan bahwa jumlah penderita diabetes terus meningkat secara drastis, menjadikannya penyebab utama kebutaan, gagal ginjal, amputasi tungkai bawah, serangan jantung, dan stroke. Pada tahun 2011 saja, 4.6 juta orang meninggal akibat diabetes, dengan 2.2 juta kematian di antaranya disebabkan oleh kadar glukosa darah tinggi dan 1.6 juta lainnya secara langsung karena diabetes. Proyeksi menunjukkan bahwa pada tahun 2030, diabetes akan menjadi penyebab kematian ketujuh di dunia. Kondisi ini ditandai dengan tingginya kadar gula darah (hiperglikemia) yang terjadi karena tubuh tidak memproduksi cukup insulin, atau karena tubuh tidak dapat menggunakan insulin yang dihasilkannya secara efektif.

Mengingat dampak serius dari komplikasi diabetes dan prevalensinya yang terus meningkat, deteksi dini dan klasifikasi tingkat risiko menjadi sangat penting. Intervensi medis yang cepat dan tepat dapat memperlambat progresi penyakit dan mencegah komplikasi yang membahayakan jiwa. Dalam beberapa dekade terakhir, kemajuan pesat dalam bidang *machine learning* telah membuka peluang baru untuk mengembangkan alat bantu diagnostik yang akurat dan efisien.

Proyek ini berfokus pada pengembangan sebuah sistem klasifikasi tingkat risiko diabetes. Dengan memanfaatkan dataset yang berisi gejala dan karakteristik pasien, sistem ini bertujuan untuk memprediksi apakah seorang individu memiliki risiko tinggi (positif diabetes) atau rendah (negatif diabetes) menggunakan algoritma *Random Forest*. Pendekatan ini diharapkan dapat memberikan kontribusi signifikan dalam upaya deteksi dini, memungkinkan pasien untuk mendapatkan penanganan lebih awal, dan pada akhirnya meningkatkan kualitas hidup mereka.

## 2. Dasar Teori

### 2.1 Diabetes Mellitus (DM)

Diabetes Mellitus adalah kelompok penyakit metabolik kronis yang ditandai dengan kadar gula darah yang tinggi (hiperglikemia) yang diakibatkan oleh defek pada sekresi insulin, kerja insulin, atau keduanya. Insulin adalah hormon kunci yang diproduksi oleh pankreas untuk memungkinkan glukosa dari makanan masuk ke dalam sel tubuh dan digunakan sebagai energi. Ketika proses ini terganggu, glukosa menumpuk di dalam darah, menyebabkan berbagai masalah kesehatan.

Secara garis besar, diabetes dibagi menjadi beberapa jenis utama:

* **Diabetes Tipe 1:** Terjadi ketika tubuh tidak memproduksi insulin sama sekali karena sistem kekebalan tubuh menyerang dan menghancurkan sel-sel beta di pankreas yang bertanggung jawab memproduksi insulin.
* **Diabetes Tipe 2:** Jenis yang paling umum, terjadi ketika tubuh tidak memproduksi cukup insulin atau tidak dapat menggunakan insulin secara efektif (resistensi insulin). Kondisi ini sering kali berkaitan dengan gaya hidup, obesitas, dan faktor genetik.
* **Diabetes Gestasional:** Terjadi selama kehamilan pada wanita yang sebelumnya tidak menderita diabetes.

Gejala umum diabetes meliputi poliuria (sering buang air kecil), polidipsia (sering haus), polifagia (sering lapar), penurunan berat badan yang tidak dapat dijelaskan, kelelahan, penglihatan kabur, dan penyembuhan luka yang lambat. Dataset yang digunakan dalam penelitian ini mencakup berbagai gejala ini, yang menjadi indikator penting dalam proses klasifikasi.

### 2.2 *Machine* Learning untuk Klasifikasi

*Machine learning* adalah subbidang dari kecerdasan buatan yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk "belajar" dari data tanpa diprogram secara eksplisit. Pembelajaran ini memungkinkan sistem untuk mengidentifikasi pola, membuat keputusan, atau memprediksi hasil di masa depan.

Dalam *machine learning*, terdapat beberapa jenis tugas, antara lain:

* **Klasifikasi:** Memprediksi kategori diskrit atau label kelas untuk suatu *instance* data. Contohnya: klasifikasi email sebagai "spam" atau "bukan spam", atau diagnosis penyakit menjadi "positif" atau "negatif".
* **Regresi:** Memprediksi nilai kontinu. Contohnya: memprediksi harga rumah atau suhu.
* **Clustering:** Mengelompokkan *instance* data serupa ke dalam kelompok tanpa label yang telah ditentukan.

Dalam konteks proyek ini, kami berfokus pada tugas klasifikasi biner, yaitu memprediksi apakah seorang pasien "Positif" mengidap diabetes atau "Negatif" tidak mengidap diabetes berdasarkan serangkaian fitur gejala.

### 2.3 Algoritma Random Forest

*Random Forest* adalah algoritma *ensemble learning* yang kuat dan populer, khususnya untuk tugas klasifikasi dan regresi. Algoritma ini pertama kali diperkenalkan oleh Leo Breiman pada tahun 2001. Nama "Random Forest" berasal dari fakta bahwa algoritma ini membangun "hutan" (*forest*) dari banyak pohon keputusan (*decision trees*) yang "acak" (*random*).

Konsep Dasar Pohon Keputusan (Decision Tree)

Pohon keputusan adalah model prediktif yang menggunakan serangkaian keputusan berdasarkan fitur data untuk memprediksi nilai target. Setiap node internal dalam pohon mewakili tes pada atribut fitur, setiap cabang mewakili hasil tes, dan setiap node daun mewakili label kelas atau nilai numerik. Proses pemisahan pada setiap node bertujuan untuk memaksimalkan "kemurnian" (homogenitas) node anak, seringkali diukur dengan metrik seperti Gini Impurity atau Entropy.

* Gini Impurity: Mengukur seberapa sering elemen yang dipilih secara acak dari set akan salah diberi label jika diberi label secara acak sesuai dengan distribusi label dalam set. Gini Impurity yang lebih rendah menunjukkan kemurnian yang lebih tinggi.  
  Gini=1−∑i=1C​(Pi​)2  
  Dimana Pi​ adalah proporsi instance dengan label i dalam node, dan C adalah jumlah kelas.

Bagaimana Random Forest Mengatasi Kelemahan Pohon Keputusan Tunggal:

Pohon keputusan tunggal cenderung overfit pada data pelatihan, yang berarti kinerjanya buruk pada data baru yang belum pernah dilihat. Random Forest mengatasi masalah ini melalui dua mekanisme utama:

1. ***Bagging* (Bootstrap Aggregating):**
   * *Random Forest* membuat banyak *bootstrap samples* dari dataset pelatihan asli. Setiap *bootstrap sample* dibuat dengan mengambil sampel data pelatihan secara acak dengan penggantian. Ini berarti beberapa *instance* mungkin muncul beberapa kali dalam satu sampel, sementara yang lain mungkin tidak muncul sama sekali.
   * Setiap pohon keputusan dalam *forest* dilatih secara independen pada *bootstrap sample* yang berbeda. Ini mengurangi varians dan *overfitting* yang cenderung terjadi pada pohon keputusan tunggal.
2. **Pemilihan Fitur Acak:**
   * Saat membangun setiap pohon keputusan, pada setiap *node*, *Random Forest* tidak mempertimbangkan semua fitur untuk mencari pemisahan terbaik. Sebaliknya, ia secara acak memilih subset fitur, dan pemisahan terbaik hanya dicari dari subset fitur tersebut.
   * Mekanisme ini lebih lanjut meningkatkan keragaman antar pohon, sehingga mengurangi korelasi antara prediksi masing-masing pohon. Jika pohon-pohon tersebut sangat berkorelasi, maka rata-rata dari prediksi mereka tidak akan banyak mengurangi varians.

Proses Prediksi di Random Forest:

Untuk tugas klasifikasi, setelah semua pohon dilatih, ketika sebuah instance baru perlu diklasifikasikan, setiap pohon dalam forest membuat prediksinya sendiri. Prediksi akhir kemudian ditentukan oleh "suara mayoritas" (majority vote) dari semua pohon. Kelas yang paling sering diprediksi oleh individu pohon akan menjadi prediksi akhir Random Forest.

Kombinasi dari *bagging* dan pemilihan fitur acak membuat *Random Forest* menjadi algoritma yang sangat robust, tidak mudah *overfit*, dan mampu menangani dataset dengan banyak fitur.

### 2.4 Metrik Evaluasi Model

Evaluasi model klasifikasi sangat penting untuk memahami seberapa baik model bekerja. Metrik yang digunakan dalam proyek ini meliputi:

* Akurasi (Accuracy): Ukuran paling intuitif dan sering digunakan, menunjukkan proporsi instance yang diklasifikasikan dengan benar dari total instance.  
  Accuracy=True Positives (TP)+True Negatives (TN)+False Positives (FP)+False Negatives (FN)True Positives (TP)+True Negatives (TN)​
  + **TP (True Positives):** Jumlah *instance* positif yang diprediksi dengan benar sebagai positif.
  + **TN (True Negatives):** Jumlah *instance* negatif yang diprediksi dengan benar sebagai negatif.
  + **FP (False Positives):** Jumlah *instance* negatif yang salah diprediksi sebagai positif (Kesalahan Tipe I).
  + **FN (False Negatives):** Jumlah *instance* positif yang salah diprediksi sebagai negatif (Kesalahan Tipe II).
* Presisi (Precision): Menjawab pertanyaan: "Dari semua yang diprediksi positif, berapa banyak yang sebenarnya positif?". Penting ketika biaya False Positives tinggi.  
  Precision=TP+FPTP​
* Sensitivitas / Recall (Recall / Sensitivity): Menjawab pertanyaan: "Dari semua yang sebenarnya positif, berapa banyak yang berhasil dideteksi oleh model?". Penting ketika biaya False Negatives tinggi, seperti dalam diagnosis penyakit.  
  Recall=TP+FNTP​
* F1-Score: Rata-rata harmonik dari Presisi dan Recall. Metrik ini sangat berguna ketika ada ketidakseimbangan kelas dalam dataset (misalnya, lebih banyak kasus negatif daripada positif). F1-Score memberikan gambaran yang lebih seimbang tentang kinerja model daripada akurasi saja dalam kasus seperti itu.  
  F1-Score=2×Precision+RecallPrecision×Recall​
* ***Confusion Matrix***: Sebuah tabel yang memvisualisasikan kinerja algoritma klasifikasi. Setiap baris matriks mewakili *instance* dalam kelas aktual, sedangkan setiap kolom mewakili *instance* dalam kelas yang diprediksi. Ini memungkinkan analisis terperinci tentang jenis kesalahan yang dibuat oleh model.

## 3. Metode

Bagian ini menguraikan metodologi yang diterapkan dalam proyek klasifikasi risiko diabetes, mulai dari persiapan data hingga evaluasi model.

### 3.1 Pengumpulan Data

Dataset yang menjadi fondasi proyek ini adalah diabetes\_risk\_prediction\_dataset.csv, yang diunduh dari platform Kaggle. Dataset ini tersusun atas 520 entri (baris), di mana setiap entri mewakili satu pasien. Untuk setiap pasien, terdapat 16 fitur prediktor (gejala dan karakteristik) dan 1 fitur target yang mengindikasikan status diabetes.

**Fitur-fitur dalam dataset adalah sebagai berikut:**

* Age (Usia): Usia pasien dalam tahun.
* Gender (Jenis Kelamin): Jenis kelamin pasien (misalnya, 'Male', 'Female').
* Polyuria: Indikator apakah pasien mengalami poliuria (sering buang air kecil) ('Yes'/'No').
* Polydipsia: Indikator apakah pasien mengalami polidipsia (sering haus) ('Yes'/'No').
* sudden\_weight\_loss (Penurunan berat badan mendadak): Indikator terjadinya penurunan berat badan yang tidak disengaja ('Yes'/'No').
* weakness (Kelemahan): Indikator adanya perasaan lelah atau kurang tenaga ('Yes'/'No').
* Polyphagia: Indikator apakah pasien mengalami polifagia (nafsu makan berlebihan) ('Yes'/'No').
* Genital\_thrush (Sariawan genital): Indikator adanya infeksi jamur di area genital ('Yes'/'No').
* visual\_blurring (Penglihatan kabur): Indikator gangguan penglihatan ('Yes'/'No').
* Itching (Gatal): Indikator adanya rasa gatal pada kulit ('Yes'/'No').
* Irritability (Iritabilitas): Indikator adanya perasaan mudah tersinggung atau marah ('Yes'/'No').
* delayed\_healing (Penyembuhan luka lambat): Indikator bahwa luka pada kulit membutuhkan waktu lebih lama untuk sembuh ('Yes'/'No').
* partial\_paresis (Paresis parsial): Indikator kelemahan atau kelumpuhan sebagian otot ('Yes'/'No').
* muscle\_stiffness (Kekakuan otot): Indikator adanya kekakuan pada otot ('Yes'/'No').
* Alopecia (Alopecia): Indikator kerontokan rambut ('Yes'/'No').
* Obesity (Obesitas): Indikator status obesitas pasien ('Yes'/'No').

**Fitur Target:**

* class: Label kelas yang menunjukkan status diabetes pasien, dengan dua kategori: Positive (mengidap diabetes) dan Negative (tidak mengidap diabetes).

### 3.2 Pra-pemrosesan Data

Tahap pra-pemrosesan data adalah langkah krusial untuk mengubah data mentah menjadi format yang cocok dan optimal untuk pelatihan model *machine learning*.

1. **Pembebanan Data:** Data dimuat menggunakan pustaka pandas dari file diabetes\_risk\_prediction\_dataset.csv.
2. **Pengecekan Data Hilang (*Missing Value Check*):** Sebelum melanjutkan, dilakukan pemeriksaan menyeluruh untuk memastikan tidak ada nilai yang hilang (NaN) dalam dataset. Kehadiran nilai hilang dapat menyebabkan error atau bias pada model. Dalam kasus ini, dataset yang digunakan tidak memiliki nilai hilang, sehingga tidak diperlukan penanganan khusus untuk nilai hilang.
3. ***Label Encoding***\*\*:\*\* Banyak algoritma *machine learning* tidak dapat bekerja langsung dengan data kategorikal (non-numerik). Oleh karena itu, fitur-fitur kategorikal seperti Gender, Polyuria, dan fitur-fitur gejala lainnya yang memiliki nilai 'Yes'/'No', serta kolom target class, perlu dikonversi menjadi representasi numerik.
   * LabelEncoder dari pustaka sklearn.preprocessing digunakan untuk transformasi ini.
   * Untuk fitur Gender: 'Female' dikonversi menjadi 0 dan 'Male' menjadi 1.
   * Untuk semua fitur gejala ('Polyuria', 'Polydipsia', dll.): 'No' dikonversi menjadi 0 dan 'Yes' menjadi 1.
   * Untuk fitur target class: 'Negative' dikonversi menjadi 0 dan 'Positive' menjadi 1. Pembalikan nilai dilakukan jika *encoder* secara default memetakan 'Positive' ke 0.
   * File dataset\_UAS\_done\_bang.xlsx - Label\_Encoding.csv secara spesifik menunjukkan hasil dari proses *encoding* ini.
4. **Pembagian Data (*Data Splitting*):** Dataset yang telah diproses kemudian dibagi menjadi dua subset: data pelatihan (*training set*) dan data pengujian (*testing set*).
   * **Data Pelatihan (70%):** Digunakan untuk melatih model *Random Forest*. Model "belajar" dari pola-pola dalam data ini.
   * **Data Pengujian (30%):** Digunakan untuk mengevaluasi kinerja model pada data yang belum pernah dilihat sebelumnya. Ini memberikan estimasi yang lebih realistis tentang bagaimana model akan berkinerja di dunia nyata.
   * Pembagian dilakukan menggunakan fungsi train\_test\_split dari sklearn.model\_selection dengan test\_size=0.3 dan random\_state=42. Pengaturan random\_state memastikan bahwa pembagian data selalu konsisten dan dapat direproduksi setiap kali kode dijalankan.

### 3.3 Pembangunan Model

Setelah data siap, model klasifikasi dibangun menggunakan algoritma *Random Forest Classifier*.

1. **Inisialisasi Model:** RandomForestClassifier diinisialisasi dengan parameter n\_estimators=100 dan random\_state=42.
   * n\_estimators=100: Menentukan jumlah pohon keputusan yang akan dibangun dalam *forest*. Jumlah 100 pohon umumnya dianggap cukup untuk mendapatkan kinerja yang baik sambil menjaga efisiensi komputasi. Lebih banyak pohon dapat meningkatkan akurasi, tetapi juga meningkatkan waktu pelatihan.
   * random\_state=42: Parameter ini digunakan untuk memastikan bahwa hasil acak (seperti pemilihan *bootstrap samples* dan subset fitur) dapat direproduksi. Ini penting untuk konsistensi eksperimen.
2. **Pelatihan Model:** Model dilatih menggunakan data pelatihan (X\_train dan y\_train) melalui metode fit(). Selama fase ini, algoritma belajar hubungan antara fitur-fitur input dan kelas target.

### 3.4 Evaluasi Model

Kinerja model yang telah dilatih dievaluasi menggunakan data pengujian. Metrik yang digunakan meliputi Akurasi, Presisi, *Recall*, dan *F1-Score*, serta *confusion matrix*. Metrik-metrik ini memberikan gambaran komprehensif tentang seberapa baik model dapat mengklasifikasikan kasus positif dan negatif, serta jenis kesalahan yang mungkin terjadi.

## 4. Implementasi

Bagian ini menyajikan detail implementasi teknis proyek, mencakup kode sumber untuk pelatihan model dan antarmuka pengguna aplikasi web.

### 4.1 Hasil *Coding* (Python - Jupyter Notebook - UAS-KK.ipynb)

**Tabel 1. 1** Hasil coding Python

|  |
| --- |
| # Bagian 1: Mengambil Dataset Otomatis dari Google Drive ke Google Colab  !pip install -q gdown  !gdown --id 1YZe-KeaCr46mRVMnN9tB6MlpGKa0ZnAw --output dataset-diabates.csv  # Bagian 2: Mengimpor Pandas dan Memuat Dataset  import pandas as pd  df = pd.read\_csv("dataset-diabates.csv")  # Menampilkan 5 baris pertama dari DataFrame  df.head()  # Bagian 3: Encoding Kategorikal ke Numerik  from sklearn.preprocessing import LabelEncoder  le = LabelEncoder()  for col in df.columns:      if df[col].dtype == 'object':          df[col] = le.fit\_transform(df[col])  # Menampilkan 5 baris pertama dari DataFrame setelah encoding  df.head()  # Bagian 4: Cek Missing Value dan Duplicate  print("Missing values:\n", df.isnull().sum())  print("Duplicate rows:", df.duplicated().sum())  # Bagian 5: Split Fitur dan Label  X = df.drop('class', axis=1)  y = df['class']  print("Fitur (X):")  print(X.head())  print("\nLabel (y):")  print(y.head())  # Bagian 6: Split Data (80:20)  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(      X, y, test\_size=0.2, random\_state=42)  print("Distribusi label di data training:")  print(y\_train.value\_counts())  print("\nDistribusi label di data testing:")  print(y\_test.value\_counts())  # Bagian 7: Train Random Forest Model  from sklearn.ensemble import RandomForestClassifier  model = RandomForestClassifier(n\_estimators=100, random\_state=45)  model.fit(X\_train, y\_train)  # Bagian 8: Evaluasi Akurasi & Report  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  y\_pred = model.predict(X\_test)  print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))  print("Classification Report:\n", classification\_report(y\_test, y\_pred))  print("Accuracy:", round(accuracy\_score(y\_test, y\_pred)\*100, 2), "%")  # Bagian 9: ROC AUC & Visualisasi  from sklearn.metrics import roc\_auc\_score, RocCurveDisplay  import matplotlib.pyplot as plt  y\_proba = model.predict\_proba(X\_test)[:,1]  roc\_score = roc\_auc\_score(y\_test, y\_proba)  print("AUC Score:", round(roc\_score \* 100, 2), "%")  RocCurveDisplay.from\_estimator(model, X\_test, y\_test)  plt.show()  # Bagian 10: Interpretasi & Visualisasi Fitur  import numpy as np  importances = model.feature\_importances\_  feature\_names = X.columns  importance\_df = pd.DataFrame({      'Feature': feature\_names,      'Importance': importances  }).sort\_values(by='Importance', ascending=False)  print(importance\_df)  # Plotting the feature importances  importance\_df.plot(kind='barh', x='Feature', y='Importance', legend=False, figsize=(10,6))  plt.title("Ranking Pentingnya Fitur")  plt.gca().invert\_yaxis()  plt.show() |

### 4.2 Backend (Python - app.py)

File app.py adalah *script* Flask yang berfungsi sebagai *backend* aplikasi

**Tabel 1. 2** Backend Python

|  |
| --- |
| from flask import Flask, render\_template, request import joblib import numpy as np import os  app = Flask(\_\_name\_\_)  # Load model model\_path = os.path.join('model', 'random\_forest\_model.pkl')  # Check if the model\_path exists, if not, try the root directory if not os.path.exists(model\_path):  model\_path = 'random\_forest\_model.pkl'  try:  model = joblib.load(model\_path)  print(f"Model successfully loaded from: {model\_path}") except FileNotFoundError:  print(f"Error: Model not found at {model\_path}. Ensure 'random\_forest\_model.pkl' exists.")  model = None  @app.route('/', methods=['GET', 'POST']) def home():  prediction\_text = None   if request.method == 'POST':  # Get input values from the form  features = [  request.form.get('Age'),  request.form.get('Gender'),  request.form.get('Polyuria'),  request.form.get('Polydipsia'),  request.form.get('sudden\_weight\_loss'),  request.form.get('weakness'),  request.form.get('Polyphagia'),  request.form.get('Genital\_thrush'),  request.form.get('visual\_blurring'),  request.form.get('Itching'),  request.form.get('Irritability'),  request.form.get('delayed\_healing'),  request.form.get('partial\_paresis'),  request.form.get('muscle\_stiffness'),  request.form.get('Alopecia'),  request.form.get('Obesity')  ]   try:  # Convert to numpy array and reshape to float type  final\_features = np.array([float(x) for x in features]).reshape(1, -1)   if model:  # Predict with the model  prediction = model.predict(final\_features)   # Interpret the result  if prediction[0] == 1:  prediction\_text = "Pasien diprediksi MENGIDAP diabetes (Risiko Tinggi)."  else:  prediction\_text = "Pasien diprediksi TIDAK MENGIDAP diabetes (Risiko Rendah)."  else:  prediction\_text = "Error: Model prediksi tidak dapat dimuat atau tidak ditemukan."   except ValueError:  prediction\_text = "Error: Input tidak valid. Pastikan semua kolom diisi dengan nilai yang benar (angka untuk usia, pilih opsi untuk gejala)."  except Exception as e:  prediction\_text = f"Terjadi kesalahan tak terduga saat melakukan prediksi: {e}"   return render\_template('index.html', prediction\_text=prediction\_text)  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |

### 4.3 Frontend (HTML - index.html)

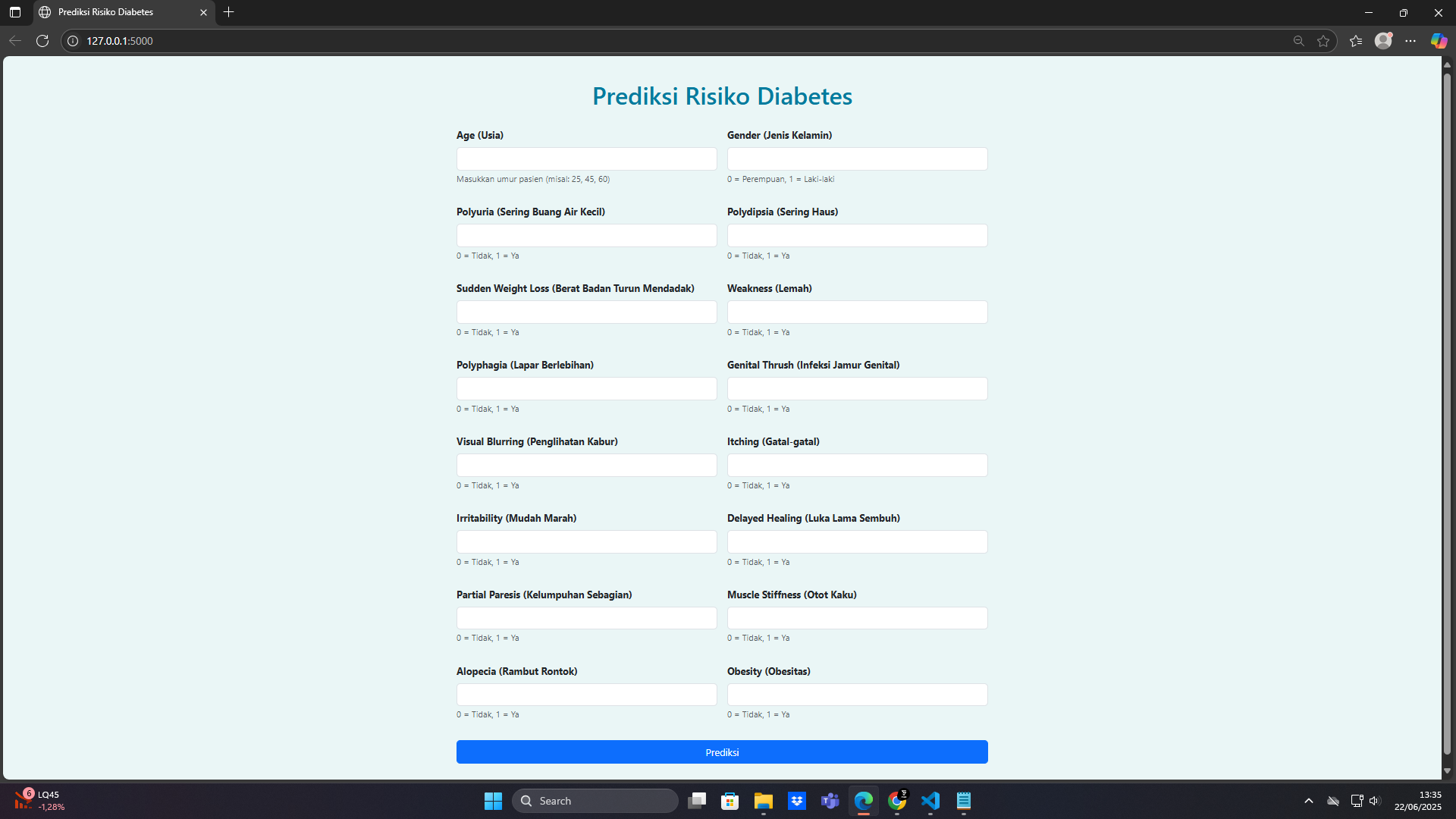
File index.html adalah struktur utama antarmuka pengguna web. Ini menggunakan Bootstrap 5 untuk styling responsif dan menyediakan formulir input yang intuitif.

**Tabel 1. 3** Frontend HTML

|  |
| --- |
| <!DOCTYPE html> <html lang="id"> <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <title>Prediksi Risiko Diabetes</title>   <!-- Bootstrap 5 CDN -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">   <style>  body {  background-color: #eaf6f6;  padding: 40px 0;  font-family: 'Inter', sans-serif;  }   .container {  max-width: 900px;  background-color: #ffffff;  padding: 30px;  border-radius: 15px;  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  }   h1 {  text-align: center;  color: #007b9e;  margin-bottom: 30px;  font-weight: bold;  }   .form-group {  margin-bottom: 15px;  }   .form-label {  font-weight: bold;  color: #333;  }   .form-control, .form-select {  border-radius: 8px;  border: 1px solid #ced4da;  padding: 10px 15px;  }   .btn-primary {  background-color: #007b9e;  border-color: #007b9e;  border-radius: 8px;  padding: 12px 0;  font-weight: bold;  transition: background-color 0.3s ease;  }   .btn-primary:hover {  background-color: #005f7c;  border-color: #005f7c;  }   .modal-header {  background-color: #007b9e;  color: white;  border-top-left-radius: 15px;  border-top-right-radius: 15px;  }   .modal-content {  border-radius: 15px;  }   .modal-footer .btn-secondary {  background-color: #6c757d;  border-color: #6c757d;  border-radius: 8px;  }   .text-muted {  font-size: 0.85em;  color: #666 !important;  }  </style> </head> <body>  <div class="container">  <h1>Prediksi Risiko Diabetes</h1>  <form method="post" class="row g-3">  {% for field, label, desc in [  ('Age', 'Usia', 'Masukkan usia pasien dalam tahun (contoh: 40).'),  ('Gender', 'Jenis Kelamin', 'Pilih jenis kelamin pasien (Laki-laki/Perempuan).'),  ('Polyuria', 'Polyuria', 'Apakah pasien sering buang air kecil? (Ya/Tidak).'),  ('Polydipsia', 'Polydipsia', 'Apakah pasien sering merasa haus yang berlebihan? (Ya/Tidak).'),  ('sudden\_weight\_loss', 'Penurunan Berat Badan Mendadak', 'Apakah pasien mengalami penurunan berat badan yang tidak disengaja dan signifikan? (Ya/Tidak).'),  ('weakness', 'Kelemahan', 'Apakah pasien sering merasa lemah atau lesu? (Ya/Tidak).'),  ('Polyphagia', 'Polyphagia', 'Apakah pasien sering merasa lapar yang berlebihan? (Ya/Tidak).'),  ('Genital\_thrush', 'Sariawan Genital', 'Apakah pasien mengalami infeksi jamur (sariawan) di area genital? (Ya/Tidak).'),  ('visual\_blurring', 'Penglihatan Kabur', 'Apakah pasien mengalami penglihatan yang kabur atau buram? (Ya/Tidak).'),  ('Itching', 'Gatal', 'Apakah pasien sering merasa gatal-gatal pada kulit? (Ya/Tidak).'),  ('Irritability', 'Iritabilitas', 'Apakah pasien mudah tersinggung atau marah? (Ya/Tidak).'),  ('delayed\_healing', 'Penyembuhan Luka Lambat', 'Apakah luka pada pasien cenderung sembuh lebih lambat dari biasanya? (Ya/Tidak).'),  ('partial\_paresis', 'Paresis Parsial', 'Apakah pasien mengalami kelemahan atau kelumpuhan sebagian pada anggota tubuh? (Ya/Tidak).'),  ('muscle\_stiffness', 'Kekakuan Otot', 'Apakah pasien mengalami kekakuan pada otot? (Ya/Tidak).'),  ('Alopecia', 'Alopecia', 'Apakah pasien mengalami kerontokan rambut yang tidak normal? (Ya/Tidak).'),  ('Obesity', 'Obesitas', 'Apakah pasien memiliki kondisi obesitas? (Ya/Tidak).')  ] %}  <div class="col-md-6">  <div class="form-group">  <label for="{{ field }}" class="form-label">{{ label }}</label>  {% if field == 'Age' %}  <input type="number" id="{{ field }}" class="form-control" name="{{ field }}" min="0" required>  {% elif field == 'Gender' %}  <select id="{{ field }}" class="form-select" name="{{ field }}" required>  <option value="">Pilih...</option>  <option value="0">Perempuan</option>  <option value="1">Laki-laki</option>  </select>  {% else %}  <select id="{{ field }}" class="form-select" name="{{ field }}" required>  <option value="">Pilih...</option>  <option value="1">Ya</option>  <option value="0">Tidak</option>  </select>  {% endif %}  <small class="text-muted">{{ desc }}</small>  </div>  </div>  {% endfor %}   <div class="col-12">  <button type="submit" class="btn btn-primary w-100">Prediksi</button>  </div>  </form>  </div>   <!-- Modal Bootstrap for Prediction Results -->  <div class="modal fade" id="resultModal" tabindex="-1" aria-labelledby="resultModalLabel" aria-hidden="true">  <div class="modal-dialog modal-dialog-centered">  <div class="modal-content">  <div class="modal-header">  <h5 class="modal-title" id="resultModalLabel">Hasil Prediksi</h5>  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Tutup"></button>  </div>  <div class="modal-body">  {{ prediction\_text }}  </div>  <div class="modal-footer">  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Tutup</button>  </div>  </div>  </div>  </div>   <!-- Bootstrap JS (required for modal functionality and other Bootstrap components) -->  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>   <script>  document.addEventListener('DOMContentLoaded', function() {  const predictionTextElement = document.querySelector('#resultModal .modal-body');  if (predictionTextElement && predictionTextElement.textContent.trim() !== "") {  const resultModal = new bootstrap.Modal(document.getElementById('resultModal'));  resultModal.show();  }  });  </script> </body> </html> |

### 4.4 Tampilan *User Interface* dan Alur Penggunaan

Berikut adalah gambaran visual dan alur penggunaan antarmuka pengguna dari aplikasi prediksi risiko diabetes:

****

**Gambar 1. 1** Tampilan User Interface

#### 4.4.1 Tampilan Antarmuka Pengguna

Berikut adalah beberapa tangkapan layar (screenshot) dari aplikasi web yang menunjukkan antarmuka pengguna pada berbagai tahap:

Halaman Utama Formulir Input:

Gambar di atas menunjukkan tampilan awal aplikasi saat diakses. Ini menampilkan formulir input yang komprehensif di mana pengguna dapat memasukkan data demografi dan gejala pasien. Setiap field input memiliki label yang jelas (misalnya, "Usia", "Jenis Kelamin", "Polyuria") dan disertai dengan deskripsi singkat tentang nilai yang diharapkan (misalnya, "Masukkan usia pasien dalam tahun", "Pilih jenis kelamin pasien (Laki-laki/Perempuan)", "Apakah pasien sering buang air kecil? (Ya/Tidak)"). Desain responsif Bootstrap memastikan tata letak yang terorganisir dengan field input tersusun dalam dua kolom. Di bagian bawah formulir terdapat tombol "Prediksi" yang menonjol berwarna biru.

Hasil Prediksi (Kasus Risiko Rendah):

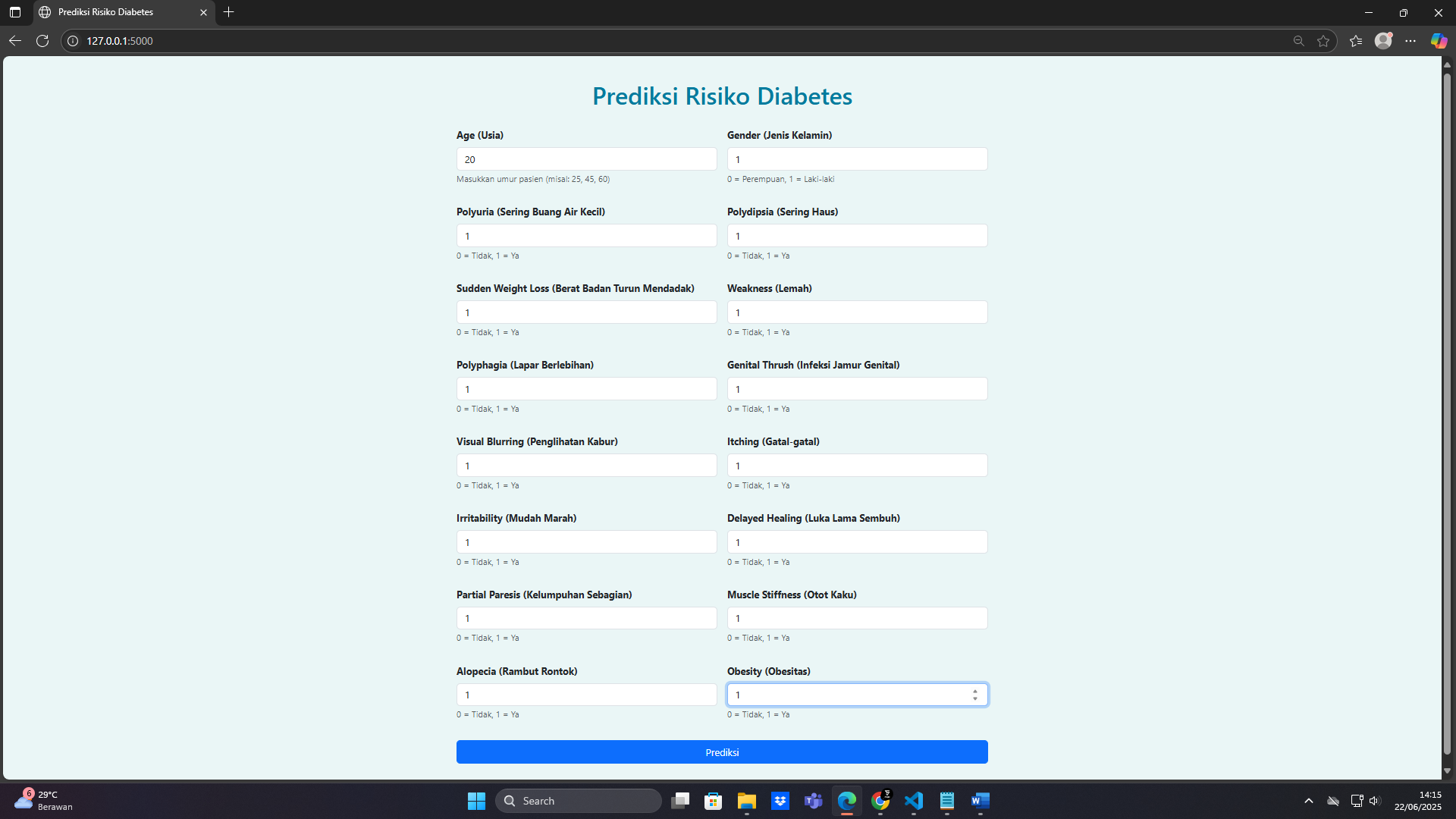
Tangkapan layar ini menunjukkan tampilan modal (jendela pop-up) yang muncul setelah pengguna menekan tombol "Prediksi" dan model mengklasifikasikan risiko diabetes sebagai "Rendah". Modal berjudul "Hasil Prediksi" dan menampilkan pesan "Pasien diprediksi TIDAK mengidap diabetes." Tombol "Tutup" tersedia di bagian bawah modal untuk memungkinkan pengguna menutup jendela dan kembali ke formulir. Desain modal ini bersih dan fokus, memastikan pesan disampaikan dengan jelas.

Hasil Prediksi (Kasus Risiko Tinggi):

Gambar ini menampilkan modal hasil prediksi serupa, namun kali ini menunjukkan kasus di mana model mengklasifikasikan risiko diabetes sebagai "Tinggi". Pesan yang ditampilkan adalah "Pasien diprediksi MENGIDAP diabetes." Hal ini memberikan indikasi yang jelas kepada pengguna mengenai status risiko diabetes yang terdeteksi oleh model. Konsistensi dalam desain dan fungsionalitas modal dipertahankan untuk kedua jenis prediksi.

#### 4.4.2 Alur Penggunaan

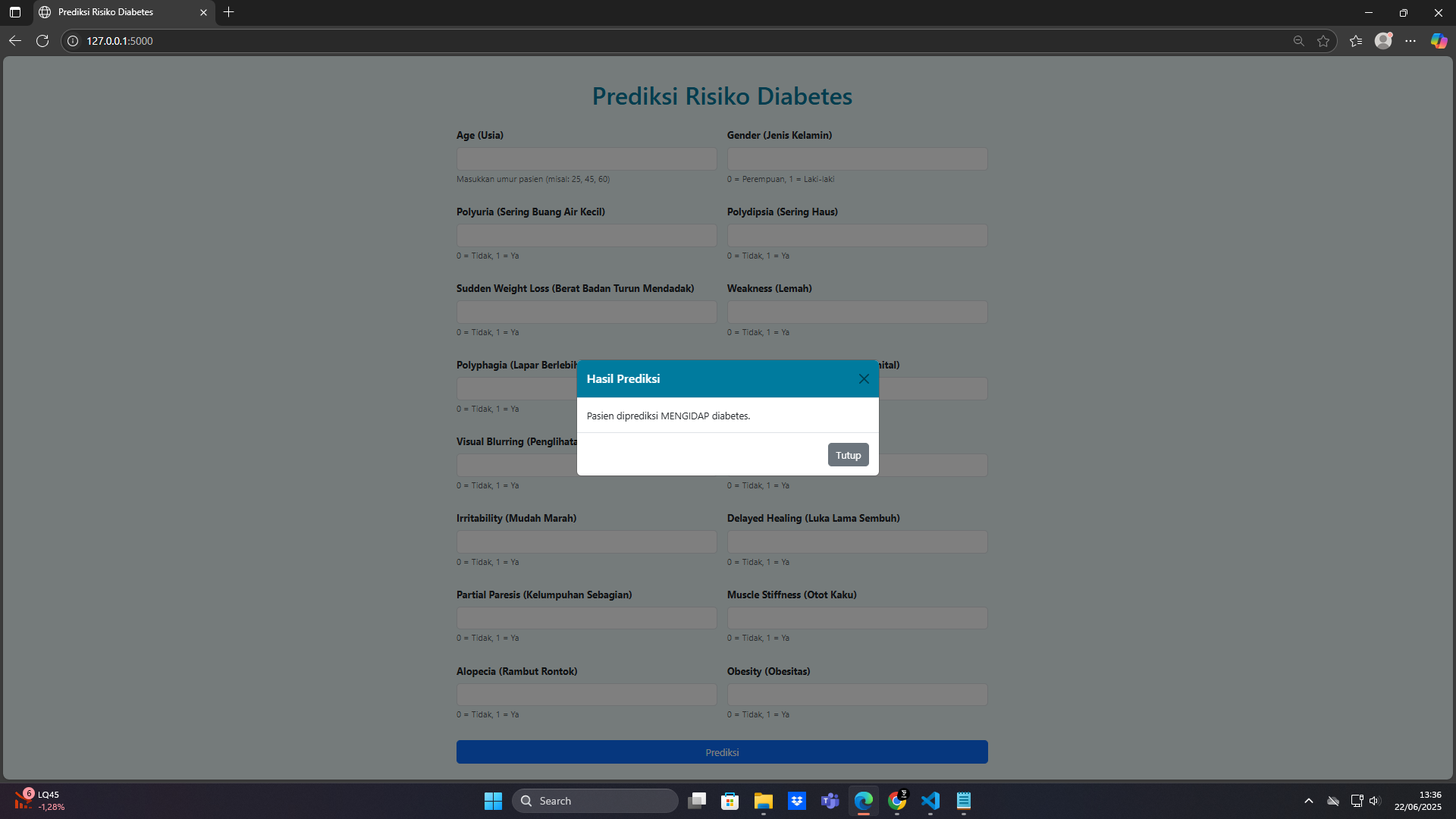
1. **Halaman Utama (Formulir Input):**
   * Pengguna akan melihat sebuah formulir yang bersih dan terstruktur dengan *field* untuk memasukkan data pasien.
   * Usia: Pengguna memasukkan usia pasien dalam angka.
   * Jenis Kelamin: Pengguna memilih "Laki-laki" atau "Perempuan" dari *dropdown*.
   * Gejala-gejala Lain: Untuk setiap gejala (seperti Polyuria, Polydipsia, sudden\_weight\_loss, dll.), pengguna memilih "Ya" atau "Tidak" dari *dropdown* yang sesuai.
   * Setiap *field* dilengkapi dengan label yang jelas dan deskripsi singkat untuk memandu pengguna.
2. **Tombol Prediksi:**

****

**Gambar 1. 2** Tombol Prediksi

* + Setelah mengisi semua *field*, pengguna mengklik tombol "Prediksi" yang terletak di bagian bawah formulir.

1. **Modal Hasil Prediksi:**

****

**Gambar 1. 3** Tampilan hasil Prediksi

* + Setelah tombol "Prediksi" ditekan, aplikasi akan memproses input dan menampilkan hasilnya dalam sebuah *modal* (jendela *pop-up*) di tengah layar.
  + *Modal* ini akan menampilkan apakah pasien "diprediksi MENGIDAP diabetes (Risiko Tinggi)" atau "diprediksi TIDAK MENGIDAP diabetes (Risiko Rendah)".
  + Pengguna dapat menutup *modal* ini dengan mengklik tombol "Tutup".

Antarmuka ini dirancang untuk menjadi intuitif dan responsif, memastikan pengalaman pengguna yang lancar baik di perangkat desktop maupun seluler.

## 5. Hasil dan Kesimpulan

### 5.1 Hasil Prediksi dan Evaluasi Model

Setelah proses pelatihan dan pengujian yang ketat, model *Random Forest* menunjukkan kinerja yang luar biasa dalam klasifikasi tingkat risiko diabetes. Hasil evaluasi model, yang diperoleh dari eksekusi kode Python dan terkonfirmasi oleh file dataset\_UAS\_done\_bang.xlsx - Evaluasi\_Model.csv serta dataset\_UAS\_done\_bang.xlsx - Confusion\_Matrix.csv, adalah sebagai berikut:

**Metrik Evaluasi Model:**

**Tabel 1. 4** Matrix Evaluasi Model

|  |  |  |
| --- | --- | --- |
| **Metrik** | **Nilai** | **Interpretasi** |
| Accuracy | 0.9744 | Model memprediksi dengan benar sekitar 97.44% dari total kasus (baik positif maupun negatif). Ini menunjukkan bahwa model sangat akurat dalam mengidentifikasi status diabetes. |
| Precision | 0.9818 | Dari semua kasus yang diprediksi model sebagai 'Positive' (mengidap diabetes), 98.18% di antaranya memang benar-benar 'Positive'. Nilai tinggi ini mengindikasikan bahwa model memiliki tingkat *false positives* yang sangat rendah, meminimalkan diagnosis yang salah. |
| Recall | 0.9703 | Dari semua kasus yang sebenarnya 'Positive' (mengidap diabetes), model berhasil mengidentifikasi 97.03% di antaranya. Nilai *recall* yang tinggi sangat krusial dalam konteks medis, karena menunjukkan kemampuan model yang baik dalam mendeteksi mayoritas kasus diabetes yang sebenarnya. |
| F1-Score | 0.9760 | *F1-Score* adalah rata-rata harmonik dari *Precision* dan *Recall*. Nilai 0.9760 yang tinggi menunjukkan keseimbangan yang sangat baik antara *Precision* dan *Recall*, mengindikasikan model yang *robust* dan berkinerja baik bahkan jika ada ketidakseimbangan kelas dalam data. |

**Confusion Matrix:**

[[ 55 2]  
 [ 3 103]]

**Tabel 1. 5** Confusion Matrix

|  |  |  |
| --- | --- | --- |
|  | **Predicted Negative** | **Predicted Positive** |
| **Actual Negative** | True Negative (TN) = 55 | False Positive (FP) = 2 |
| **Actual Positive** | False Negative (FN) = 3 | True Positive (TP) = 103 |

**Interpretasi *Confusion Matrix*:**

* **True Positives (TP = 103):** Ini adalah jumlah pasien yang benar-benar mengidap diabetes dan berhasil diprediksi oleh model sebagai pengidap diabetes. Ini adalah keberhasilan diagnostik yang paling diinginkan.
* **True Negatives (TN = 55):** Ini adalah jumlah pasien yang sebenarnya tidak mengidap diabetes dan berhasil diprediksi oleh model sebagai tidak mengidap diabetes. Ini menunjukkan kemampuan model dalam mengesampingkan diagnosis yang tidak perlu.
* **False Positives (FP = 2):** Ini adalah jumlah pasien yang sebenarnya tidak mengidap diabetes, tetapi secara salah diprediksi oleh model sebagai pengidap diabetes. Dalam konteks medis, ini berarti ada 2 pasien yang mungkin akan menjalani tes lanjutan yang sebenarnya tidak diperlukan, namun risiko yang ditimbulkan tidak fatal.
* **False Negatives (FN = 3):** Ini adalah jumlah pasien yang sebenarnya mengidap diabetes, tetapi secara salah diprediksi oleh model sebagai tidak mengidap diabetes. Ini adalah jenis kesalahan yang paling kritis dalam diagnosis medis, karena dapat menyebabkan penundaan diagnosis dan perawatan. Namun, dengan hanya 3 kasus, model menunjukkan tingkat kesalahan Tipe II yang sangat rendah, yang sangat diinginkan untuk aplikasi diagnostik.

Secara keseluruhan, hasil ini menegaskan bahwa model *Random Forest* yang dibangun memiliki kinerja klasifikasi yang sangat kuat dan dapat diandalkan untuk memprediksi tingkat risiko diabetes berdasarkan fitur-fitur yang diberikan. Model mampu meminimalkan baik *false positives* maupun *false negatives*, yang sangat penting dalam aplikasi kesehatan.

### 5.2 Kesimpulan

Proyek klasifikasi tingkat risiko diabetes menggunakan algoritma *Random Forest* telah berhasil dikembangkan dan diimplementasikan. Dengan memanfaatkan dataset yang komprehensif, model telah dilatih dan dievaluasi, menunjukkan kinerja prediktif yang luar biasa. Akurasi model sebesar 97.44%, diiringi dengan nilai *precision*, *recall*, dan *F1-Score* yang tinggi, secara meyakinkan menunjukkan efektivitas *Random Forest* dalam membedakan individu yang berisiko tinggi dan berisiko rendah terhadap diabetes. Minimnya jumlah *false negatives* (hanya 3 kasus) adalah indikator krusial bahwa model sangat baik dalam mendeteksi kasus diabetes yang sebenarnya.

Implementasi model dalam bentuk aplikasi web berbasis Flask juga berhasil menyediakan antarmuka pengguna yang ramah dan interaktif. Hal ini memungkinkan input data gejala secara mudah dan penerimaan hasil prediksi secara instan, menjadikannya alat yang berpotensi besar untuk deteksi dini dan sebagai dukungan keputusan bagi profesional kesehatan. Keberhasilan proyek ini menunjukkan potensi besar *machine learning* dalam transformasi bidang diagnostik medis, khususnya dalam penanganan penyakit kronis seperti diabetes.