

**LAPORAN UJIAN AKHIR SEMESTER**  
**PROYEK DATA MINING**  
**Anna Baita, S.Kom., M.Kom.**



**Disusun Oleh:**

Nama : Rizky Nanda Anggia

Kelas : 22S1IF-ProyekD4(ST167)

NIM : 22.11.4825

Nomor Kursi : 137

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS AMIKOM YOGYAKARTA**  
**2025**

# **SELURUH ANALISA KODE PROGRAM PEMBENTUKAN MODEL DATA MINING**

## **Analisis Kode Program Pembentukan Model Data Mining**

**Judul Penelitian:** Klasifikasi Diabetes Tipe 2 Berbasis Faktor Klinis dan Gaya Hidup: Optimalisasi Kinerja Menggunakan Algoritma Random Forest dan Teknik SMOTE

Dokumen ini menyajikan analisis lengkap dari kode program Python yang digunakan untuk membangun model klasifikasi Diabetes Tipe 2. Analisis ini mencakup setiap tahapan, mulai dari persiapan lingkungan, pra-pemrosesan data, pemilihan fitur, pemodelan, hingga evaluasi dan penyimpanan model.

### **Blok 1: Library**

- **Tujuan:** Pada blok ini, saya mengimpor semua pustaka Python yang menjadi fondasi untuk seluruh proses pemodelan. Pustaka-pustaka ini akan membantu saya dalam setiap langkah, dari manipulasi data hingga evaluasi model.
- **Analisis:** Saya mengimpor pustaka-pustaka standar seperti pandas, numpy, seaborn, dan matplotlib.pyplot untuk kebutuhan manipulasi data dan visualisasi. Untuk pemodelan, saya menggunakan

RandomForestClassifier dari sklearn.ensemble, dan metrik evaluasi seperti classification\_report, confusion\_matrix, dan accuracy\_score dari sklearn.metrics. Pustaka khusus imblearn.over\_sampling.SMOTE juga diimpor untuk menangani data yang tidak seimbang. Terakhir, saya menyertakan pickle untuk keperluan menyimpan model yang sudah terlatih.

### **Blok 2: Load Dataset**

- **Tujuan:** Memuat dataset awal ke dalam lingkungan kerja saya.
- **Analisis:** Saya menggunakan kode df = pd.read\_csv("/content/diabetes\_data.csv") untuk membaca data dari file diabetes\_data.csv dan menyimpannya dalam sebuah DataFrame bernama df. Ini adalah langkah pertama untuk membuat data siap diproses.

### **Blok 3: Preprocessing dan EDA**

- **Tujuan:** Membersihkan, menyiapkan, dan menganalisis data secara eksploratif untuk mendapatkan pemahaman awal.

- **Analisis:**
  - **Cek Missing Value:** Saya memulai dengan memeriksa apakah ada data yang hilang menggunakan `df.isnull().sum()`. Hasil dari kode ini menunjukkan bahwa tidak ada *missing value* yang terdeteksi.
  - **Penanganan Missing Value:** Meskipun tidak ada data yang hilang, saya menyertakan logika untuk mengisi nilai yang hilang, di mana kolom kategorikal akan diisi dengan modus dan numerik dengan median. Ini adalah praktik yang baik jika data baru nanti memiliki nilai yang hilang.
  - **Drop Kolom Tidak Relevan:** Kolom PatientID dan DoctorInCharge dihapus karena tidak relevan untuk tujuan prediksi.
  - **Pemisahan Fitur dan Target:** Saya memisahkan data menjadi fitur (X) yang terdiri dari 10 kolom terpilih dan variabel target (y) yaitu kolom Diagnosis.
  - **Normalisasi Fitur:** Saya menggunakan `StandardScaler()` untuk menormalisasi fitur numerik yang ada di X.
  - **Korelasi Matriks:** Saya membuat *heatmap* korelasi untuk semua fitur. Ini membantu saya memvisualisasikan hubungan antar fitur dalam dataset.

#### Blok 4: SMOTE untuk Penyeimbangan Data

- **Tujuan:** Menangani ketidakseimbangan kelas dalam dataset, yang sangat krusial untuk mencegah model bias dalam prediksinya.
- **Analisis:**
  - Sebelum SMOTE, distribusi kelas sangat tidak seimbang: Kelas 0 (Tidak Diabetes) memiliki 1127 data dan Kelas 1 (Diabetes) hanya memiliki 752 data.
  - Saya menggunakan `SMOTE(random_state=42)` untuk menyeimbangkan data, menghasilkan jumlah data yang sama untuk kedua kelas, yaitu 1127 data.
  - Visualisasi *countplot* yang saya buat menunjukkan dengan jelas bahwa data telah berhasil diseimbangkan.

#### Blok 5: Split Data

- **Tujuan:** Membagi data menjadi set pelatihan dan pengujian untuk melatih dan mengevaluasi model secara objektif.

- **Analisis:**
  - Data yang sudah seimbang (`X_res`, `y_res`) dibagi menggunakan `train_test_split` dengan `test_size=0.2`, artinya 20% data digunakan untuk pengujian dan sisanya untuk pelatihan.
  - `random_state=42` memastikan bahwa pembagian data ini akan konsisten setiap kali program dijalankan.

## Blok 6: Feature Selection

- **Tujuan:** Mengidentifikasi dan memilih 10 fitur yang paling penting, karena tidak semua fitur memiliki kontribusi yang sama dalam prediksi.
- **Analisis:**
  - Saya melatih model `RandomForestClassifier` awal pada seluruh fitur yang tersedia.
  - Berdasarkan tingkat kepentingan fitur yang dihasilkan, saya memilih 10 fitur teratas.
  - Fitur-fitur tersebut adalah: `FastingBloodSugar`, `HbA1c`, `QualityOfLifeScore`, `FatigueLevels`, `MedicationAdherence`, `SleepQuality`, `CholesterolHDL`, `DiastolicBP`, `CholesterolLDL`, dan `Age`.
  - Saya membuat ulang *heatmap* korelasi khusus untuk 10 fitur terbaik ini, termasuk hubungan mereka dengan `Diagnosis`, untuk analisis lebih lanjut.

## Blok 7: Modelling

- **Tujuan:** Mengoptimalkan model *Random Forest* untuk mendapatkan kinerja terbaik.
- **Analisis:**
  - Saya menggunakan `GridSearchCV` untuk mencari kombinasi *hyperparameter* terbaik dengan mencoba berbagai nilai untuk `n_estimators`, `max_depth`, dan lainnya.
  - Model terbaik ditemukan dengan parameter: `{'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}`.
  - Saya juga melakukan validasi silang (*cross-validation*) 5-fold pada model terbaik, yang menghasilkan rata-rata akurasi sebesar 0.9007. Ini menunjukkan model saya cukup andal.

## Blok 8: Evaluasi Model

- **Tujuan:** Menilai kinerja model terbaik pada data pengujian yang belum pernah dilihat sebelumnya.
- **Analisis:**
  - **Akurasi:** Model terbaik berhasil mencapai akurasi sebesar 0.8980 pada data pengujian.
  - **Confusion Matrix:** Saya mendapatkan matriks kebingungan yang sangat informatif, di mana 206 data diprediksi benar sebagai kelas 0 (Tidak Diabetes), dan 199 data diprediksi benar sebagai kelas 1 (Diabetes).
  - **Laporan Klasifikasi:** Laporan ini memberikan metrik *precision*, *recall*, dan *f1-score* yang seimbang untuk kedua kelas, menegaskan bahwa model saya bekerja dengan baik.

## Blok 9: Menyimpan Model

- **Tujuan:** Menyimpan model yang sudah terlatih dan objek StandardScaler ke dalam file untuk penggunaan di masa mendatang.
- **Analisis:**
  - Saya menggunakan pickle.dump untuk menyimpan model terbaik (best\_model) dan objek penskalaan (scaler) ke Google Drive.
  - Proses ini sangat penting agar model dapat digunakan kembali untuk prediksi data baru tanpa perlu melatihnya dari awal. Saya juga memuat ulang model yang disimpan dan memverifikasi akurasinya, memastikan proses penyimpanan berhasil.

Dibawah Ini Merupakan Full Kode Serta Penjelasannya Di Setiap Baris

## Blok 1: Library

```
# 1. Import Library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pickle # untuk menyimpan dan memuat model

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from imblearn.over_sampling import SMOTE
```

## Blok 2: Load Dataset

```
# 2. Load Dataset
df = pd.read_csv("/content/diabetes_data.csv")
```

## Blok 3: Preprocessing dan EDA

```
# 3. Cek Missing Value
missing = df.isnull().sum()
print("Missing Values:\n", missing[missing > 0])

# 4. Tangani Missing Value
for col in df.columns:
    if df[col].isnull().sum() > 0:
        if df[col].dtype == 'object':
            df[col].fillna(df[col].mode()[0], inplace=True)
        else:
            df[col].fillna(df[col].median(), inplace=True)

# 5. Drop Kolom Tidak Relevan
columns_to_drop = ['PatientID', 'DoctorInCharge']
existing_cols = [col for col in columns_to_drop if col in df.columns]
df.drop(columns=existing_cols, inplace=True)

# 6. Pisahkan fitur dan target
selected_features = [
```

```

'FastingBloodSugar', 'HbA1c', 'SleepQuality', 'CholesterolHDL',
'FatigueLevels', 'CholesterolLDL', 'MedicationAdherence',
'QualityOfLifeScore', 'DiastolicBP', 'Age'
]

X = df[selected_features]
y = df['Diagnosis']

# 7. Normalisasi Fitur Numerik
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

# 8. EDA - Heatmap Korelasi (Teradaptasi)
df_corr = df.copy()

# Encoding kolom kategorikal (untuk korelasi)
for col in df_corr.select_dtypes(include='object').columns:
    df_corr[col] = LabelEncoder().fit_transform(df_corr[col])

# Matriks Korelasi
correlation_matrix = df_corr.corr()

# Heatmap antar semua fitur (tanpa angka)
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix,
            annot=False,
            cmap='coolwarm',
            linewidths=0.5,
            square=True,
            cbar_kws={"shrink": 0.8})
plt.title("Matriks Korelasi antar Seluruh Fitur (Tanpa Angka)")
plt.tight_layout()
plt.show()

```

#### Blok 4: SMOTE untuk Penyeimbangan Data

```

# 9. SMOTE untuk Penyeimbangan Data
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_scaled, y)

# LANGKAH TAMBAHAN: VISUALISASI SETELAH SMOTE
# -----

```

```

# 1. Membandingkan distribusi kelas sebelum dan sesudah SMOTE
plt.figure(figsize=(12, 5))

# Plot sebelum SMOTE
plt.subplot(1, 2, 1)
sns.countplot(x=y, palette='pastel')
plt.title('Distribusi Kelas Sebelum SMOTE')
plt.xlabel('Diagnosis (0: Tidak Diabetes, 1: Diabetes)')
plt.ylabel('Jumlah')

# Plot sesudah SMOTE
plt.subplot(1, 2, 2)
sns.countplot(x=y_res, palette='pastel')
plt.title('Distribusi Kelas Sesudah SMOTE')
plt.xlabel('Diagnosis (0: Tidak Diabetes, 1: Diabetes)')
plt.ylabel('Jumlah')

plt.tight_layout()
plt.show()

print("\nDistribusi kelas sebelum SMOTE:")
print(y.value_counts())
print("\nDistribusi kelas sesudah SMOTE:")
print(y_res.value_counts())

```

## Blok 5: Split Data

```

# 10. Split Data
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)

# 11. Fit Awal Random Forest (seluruh fitur)
rf_full = RandomForestClassifier(random_state=42)
rf_full.fit(X_train, y_train)

```

## Blok 6: Feature Selection

```

# 12. Feature Selection - Ambil 10 fitur terpenting
feature_importances = pd.Series(rf_full.feature_importances_,
index=X.columns)

```

```

top_10_features =
feature_importances.sort_values(ascending=False).head(10).index.tolist()

print("Top 10 Fitur Terpenting:\n", top_10_features)

# 13. Korelasi Matriks Khusus Top 10 Fitur
df_top10 = pd.DataFrame(X_res[top_10_features])
df_top10['Diagnosis'] = y_res # gabungkan target untuk korelasi dengan diagnosis

plt.figure(figsize=(12, 8))
sns.heatmap(df_top10.corr(), annot=True, cmap='YlGnBu', fmt='.2f',
square=True)
plt.title("Matriks Korelasi Top 10 Fitur dengan Diagnosis")
plt.tight_layout()
plt.show()

# 14. Re-split data hanya dengan 10 fitur terbaik
X_top10 = X_res[top_10_features]
X_train, X_test, y_train, y_test = train_test_split(X_top10, y_res,
test_size=0.2, random_state=42)

```

## Blok 7: Modelling

```

# 15. Optimasi dan Training Final Model
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [True, False]
}

grid_search =
GridSearchCV(estimator=RandomForestClassifier(random_state=42),
            param_grid=param_grid,
            cv=3,
            n_jobs=-1,
            verbose=1,
            scoring='accuracy')

grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

```

```

# LANGKAH TAMBAHAN: VALIDASI SILANG (CROSS-VALIDATION)
# -----
from sklearn.model_selection import cross_val_score

# best_model didapatkan dari cell #15 (GridSearchCV)
# X_train dan y_train adalah data latih Anda dari cell #14

# Lakukan 5-fold cross-validation
cv_scores = cross_val_score(best_model, X_train, y_train, cv=5,
scoring='accuracy')

print("Hasil Akurasi dari 5-Fold Cross-Validation:")
print(cv_scores)
print("\n=====")
print(f"Rata-rata Akurasi: {cv_scores.mean():.4f}")
print(f"Standar Deviasi Akurasi: {cv_scores.std():.4f}")

```

## Blok 8: Evaluasi Model

```

# 16. Evaluasi Model
y_pred = best_model.predict(X_test)
print("\nBest Parameters:", grid_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))

# 17. Visualisasi Confusion Matrix
from sklearn.metrics import ConfusionMatrixDisplay

# Buat confusion matrix
cm = confusion_matrix(y_test, y_pred)
labels = ['Tidak Diabetes', 'Diabetes']

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()

```

## Blok 9: Menyimpan Model

```
from google.colab import drive
drive.mount('/content/drive')

# Path tujuan penyimpanan di Google Drive
model_path = "/content/drive/MyDrive/random_forest_model.pkl"

# Simpan model ke file .pkl di Google Drive
with open(model_path, "wb") as file:
    pickle.dump(best_model, file)

print(f"☑ Model berhasil disimpan di Google Drive: {model_path}")

# --- Tambahan: Simpan objek scaler untuk digunakan di aplikasi Flask ---
scaler_path = "/content/drive/MyDrive/scaler.pkl" # Pastikan path ini sesuai dengan tempat Anda ingin menyimpan scaler
with open(scaler_path, "wb") as file:
    pickle.dump(scaler, file)
print(f"☑ Scaler berhasil disimpan di Google Drive: {scaler_path}")

# Load ulang model dari Google Drive
with open(model_path, "rb") as file:
    loaded_model = pickle.load(file)

print("☑ Model berhasil dimuat ulang dari Google Drive!")

# Contoh prediksi ulang
y_pred_loaded = loaded_model.predict(X_test)

# Contoh prediksi ulang
y_loaded_pred = loaded_model.predict(X_test)

# Evaluasi hasil prediksi dari model yang dimuat ulang
from sklearn.metrics import accuracy_score
print("Akurasi model yang dimuat ulang:", accuracy_score(y_test,
y_loaded_pred))
```