LAPORAN TUGAS BESAR EKSPLORASI AUTOML MENGGUNAKAN DATASET PIMA

Anggota Kelompok:
1.Farhan Rangkuti (1304202025)
2.Bhagas Ade Pramono (1304201018)
3.Muhamad Meidy Mahardika (1304202024)

FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2024

Latar Belakang Data Set

Penelitian NIDDK (Institut Nasional Diabetes dan Penyakit Pencernaan dan Ginjal) menciptakan pengetahuan tentang dan pengobatan penyakit yang paling kronis, mahal, dan berdampak. Dataset yang digunakan dalam proyek ini berasal dari NIDDK. Tujuannya adalah untuk memprediksi apakah seorang pasien menderita diabetes atau tidak, berdasarkan pengukuran diagnostik tertentu yang disertakan dalam kumpulan data. Buat model untuk memprediksi secara akurat apakah pasien dalam kumpulan data menderita diabetes atau tidak. Kumpulan data terdiri dari beberapa variabel prediktor medis dan satu variabel target (Hasil). Variabel prediktor mencakup jumlah kehamilan yang dialami pasien, BMI, tingkat insulin, usia, dan lainnya. Variabel Deskripsi Kehamilan Berapa kali hamil Glukosa Konsentrasi glukosa plasma dalam tes toleransi glukosa oral Tekanan Darah Tekanan darah diastolik (mm Hg) Ketebalan Kulit Ketebalan lipatan kulit trisep (mm) Insulin Insulin serum dua jam BMI Indeks Massa Tubuh Diabetes Fungsi Silsilah Diabetes Fungsi silsilah Diabetes Usia Usia dalam tahun Hasil Variabel kelas (0 atau 1). 268 dari 768 nilai adalah 1, dan yang lainnya adalah 0 Inspirasi.

Formulasi Masalah

Diberikan dataset PIMA(Pima-Indians-Diabetes) yang terdiri dari beberapa variabel medis independen (seperti kehamilan, kadar glukosa, tekanan darah, ketebalan kulit, insulin, BMI, fungsi keturunan diabetes, dan usia) serta satu variabel hasil (outcome) yang menunjukkan apakah seseorang memiliki diabetes atau tidak (diwakili oleh nilai 1 untuk diabetes dan nilai 0 untuk tidak diabetes). Tujuan dari penggunaan dataset ini adalah untuk mendeteksi apakah seseorang menderita diabetes atau tidak berdasarkan data medis yang tersedia menggunakan automated machine learning.

Eksplorasi dan Persiapan Data

Data memiliki beberapa atribut, antara lain:

No	Nama atribut	Tipe atribut	Deskripsi
1	Outcome(target)	Numeric	0 (tidak memiliki diabetes), 1(memiliki diabetes)
2	Pregnancies	Numeric	Jumlah kehamilan
3	Glucose	Numeric	Konsentrasi plasma glukosa dalam 2 jam
4	BloodPressure	Numeric	Tekanan darah diastolik(mm hg)
5	SkinThickness	Numeric	Ketebalan lipatan kulit trisep (mm)
6	Insulin	Numeric	2 jam serum insulin(mu U/ml)

7	BMI	Numeric	Body Mass Index ((weight (kg)/height (m))^2)
8	Age	Numeric	Umur (tahun)
9	DiabetesPedigreeFunction	Numeric	Skor kemungkinan memiliki diabetes berdasarkan riwayat keluarga

EDA adalah proses eksplorasi dan analisis awal terhadap data untuk memahami karakteristik, pola, dan hubungan dalam dataset. Tujuan utama dari EDA adalah mengidentifikasi informasi yang berguna, mengungkapkan wawasan, serta mengidentifikasi anomali atau kesalahan dalam data. Pada EDA ada beberapa langkah tahapan yang kami lakukan antara lain:

1. Import Library

```
[ ] import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    from scipy.io import arff
    from tpot import TPOTClassifier
    from sklearn.metrics import classification_report
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import accuracy_score
```

2. Ubah data dari .arff menjadi .csv

```
# Load ARFF file
data = arff.loadarff('/content/dataset_.arff')
df = pd.DataFrame(data[0])

# Save as CSV
df.to_csv('/content/pima.csv', index=False)
```

```
[84] df = pd.read_csv('/content/pima.csv')
```

3. Eksplorasi data

```
[85] df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1.0
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	0.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0

```
[86] df.tail()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0	0.0
764	2.0	122.0	70.0	27.0	0.0	36.8	0.340	27.0	0.0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.0	0.0
766	1.0	126.0	60.0	0.0	0.0	30.1	0.349	47.0	1.0
767	1.0	93.0	70.0	31.0	0.0	30.4	0.315	23.0	0.0

```
[87] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    float64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    float64
dtypes: float64(9)
memory usage: 54.1 KB
```

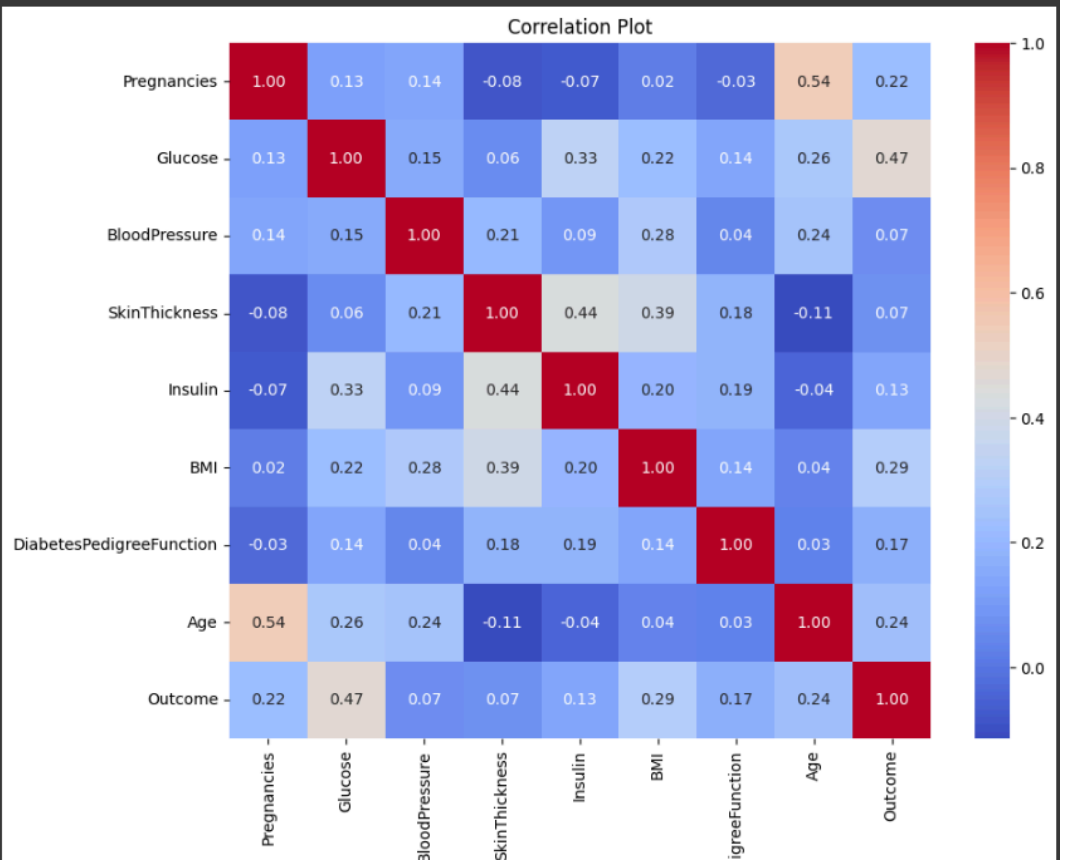
4. Pengecekan nilai null

```
[101] print(df.isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

5. Pengecekan korelasi antar atribut/features

```
[88] correlation_matrix = df.corr()

     plt.figure(figsize=(10, 8))
     sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
     plt.title('Correlation Plot')
     plt.show()
```

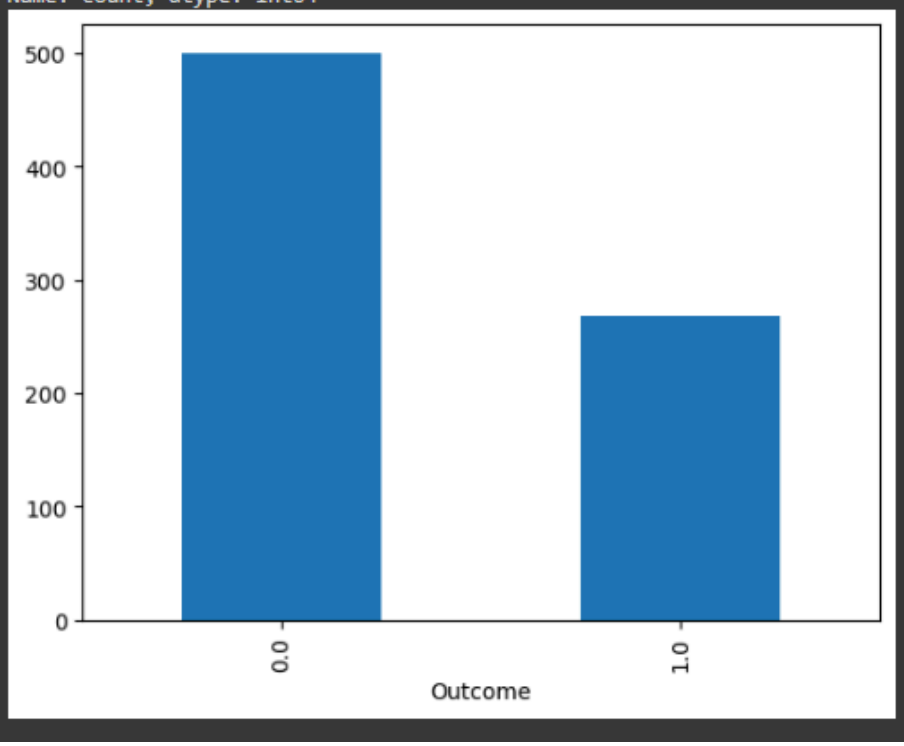


Tidak terdapat korelasi signifikan antar atribut/features.

6. Plot target class

```
color_wheel = {1: "#0392cf",
               2: "#7bc043"}
colors = df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(df.Outcome.value_counts())
p=df.Outcome.value_counts().plot(kind="bar")
```

```
Outcome
0.0    500
1.0    268
Name: count, dtype: int64
```



Target data memiliki bias dimana hasil akhir lebih condong kepada tidak memiliki diabetes atau 0. Jumlah yang tidak memiliki diabetes hampir 2 kali lipat dari jumlah yang memiliki diabetes

7. Deskripsi dataset

```
[90] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Kolom berikut memiliki nilai min 0 yang tidak masuk akal, hal ini mengindikasikan missing value :

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

Preprocessing data adalah tahap kritis dalam proses machine learning yang melibatkan transformasi dan pembersihan data mentah menjadi bentuk yang lebih sesuai untuk pemodelan dan analisis. Langkah-langkah yang kami lakukan dalam preprocessing data meliputi:

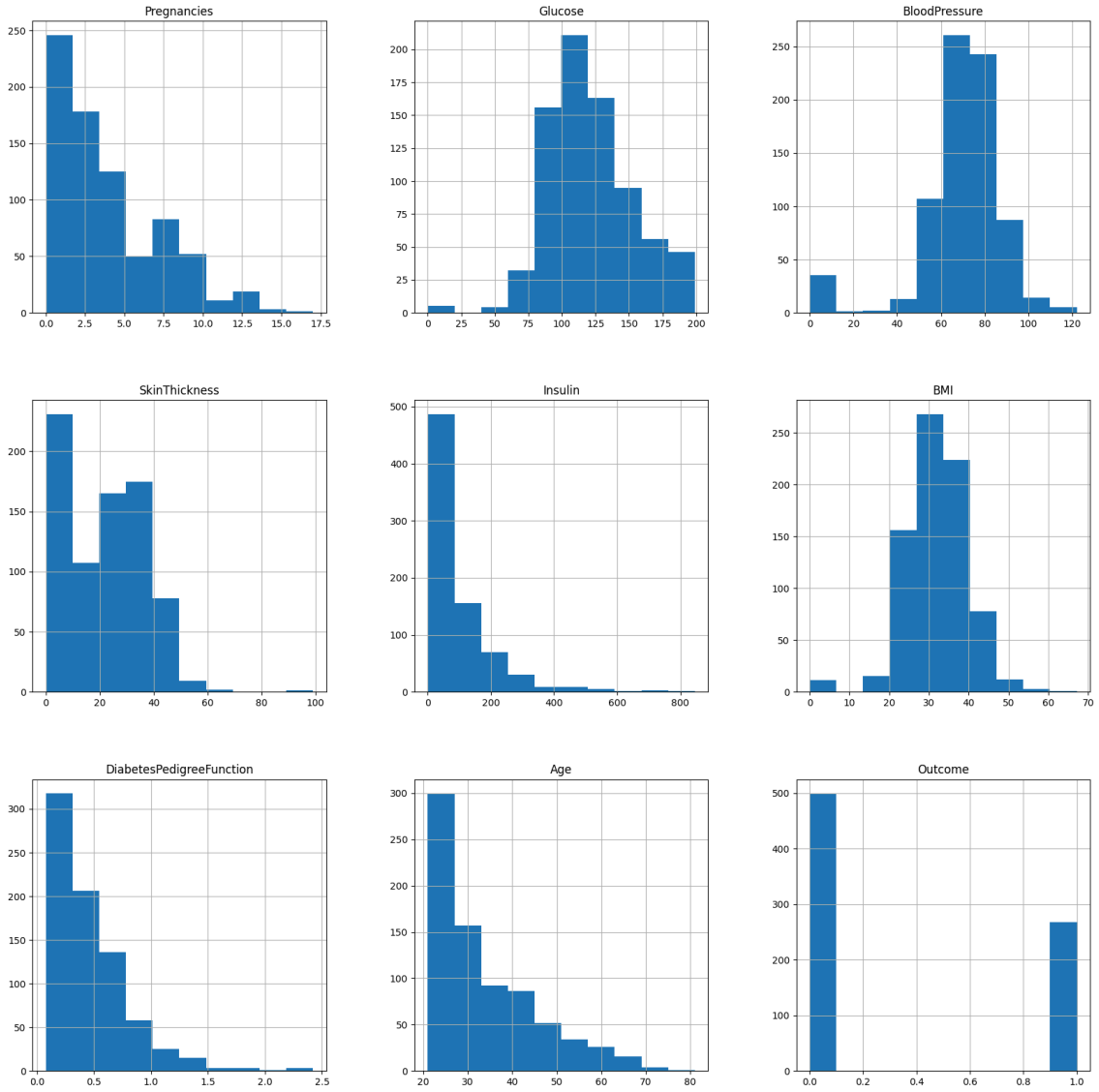
1. Ubah nilai 0 yang telah ditemukan sebelumnya menjadi NaN

Ubah nilai 0 menjadi NaN agar lebih mudah untuk menghitungnya

```
[91] df_copy = df.copy(deep = True)
     df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

     print(df_copy.isnull().sum())
```

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

2. Plot distribusi data pada setiap kolom untuk menentukan cara pengisian nilai NaN.



Ubah nilai NaN menjadi median atau mean berdasarkan distribusi data masing-masing. Data yang memiliki kecondongan ke kiri atau kanan nilai meannya akan lebih terpengaruh oleh outlier, oleh karena itu lebih baik untuk menggunakan median pada kolom tersebut.

3. Ubah nilai NaN menggunakan median atau mean

```
[93] df_copy['Glucose'].fillna(df_copy['Glucose'].mean(), inplace = True)
     df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(), inplace = True)
     df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(), inplace = True)
     df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace = True)
     df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace = True)
```

4. Split data menjadi 80% training dan 20% testing

```
[94] from sklearn.model_selection import train_test_split

[95] X = df_copy.drop("Outcome", axis=1)
     y = df_copy["Outcome"]


     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Normalisasi data

```
[96] from sklearn.preprocessing import StandardScaler

     scaler = StandardScaler()

     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

Scaling data menggunakan StandardScaler. TPOT menggunakan banyak model yang bekerja lebih baik dengan menggunakan data yang sudah discaling.

Pemodelan dan Eksperimen

- TPOT

1. Install TPOT

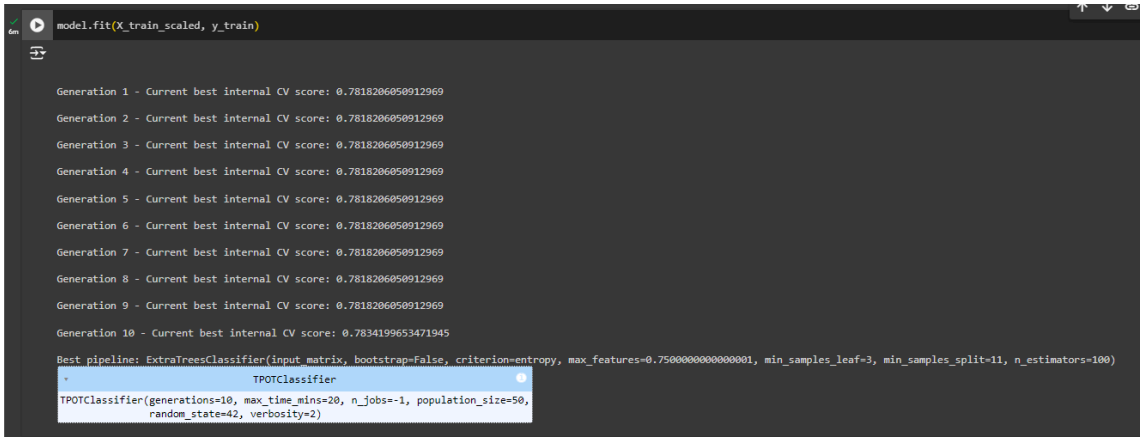
```
!pip install tpot

Collecting tpot
  Downloading TPOT-0.12.2-py3-none-any.whl (87 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 87.4/87.4 kB 2.9 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.10/dist-packages (from tpot) (1.25.2)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from tpot) (1.11.4)
Collecting scikit-learn>=1.4.1 (from tpot)
  Downloading scikit_learn-1.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.3/13.3 MB 42.5 MB/s eta 0:00:00
Collecting deap>=1.2 (from tpot)
  Downloading deap-1.4.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.4/135.4 kB 17.7 MB/s eta 0:00:00
Collecting update-checker>=0.16 (from tpot)
  Downloading update_checker-0.18.0-py3-none-any.whl (7.0 kB)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.10/dist-packages (from tpot) (4.66.4)
Collecting stopit>=1.1.1 (from tpot)
  Downloading stopit-1.1.2.tar.gz (18 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from tpot) (2.0.3)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from tpot) (1.4.2)
Requirement already satisfied: xgboost>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tpot) (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->tpot) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->tpot) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->tpot) (2024.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.4.1->tpot) (3.5.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from update-checker>=0.16->tpot) (2.31.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.24.2->tpot) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (2024.6.2)
Building wheels for collected packages: stopit
  Building wheel for stopit (setup.py) ... done
  Created wheel for stopit: filename=stopit-1.1.2-py3-none-any.whl size=11938 sha256=3bd1d1a109992a64eb7b31b5db2306342824501f18c6f841cc791610b911d595
  Stored in directory: /root/.cache/pip/wheels/af/f9/87/bf5b3d565c2a007b4dae9d8142dccc85a9f164e517062dd519
Successfully built stopit
Installing collected packages: stopit, deap, update-checker, scikit-learn, tpot
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed deap-1.4.1 scikit-learn-1.5.0 stopit-1.1.2 tpot-0.12.2 update-checker-0.18.0
```

2. Pembangunan model

```
[97] model = TPOTClassifier(verbosity = 2, max_time_mins = 20, generations = 10, population_size = 50, n_jobs=-1, random_state=42)
```

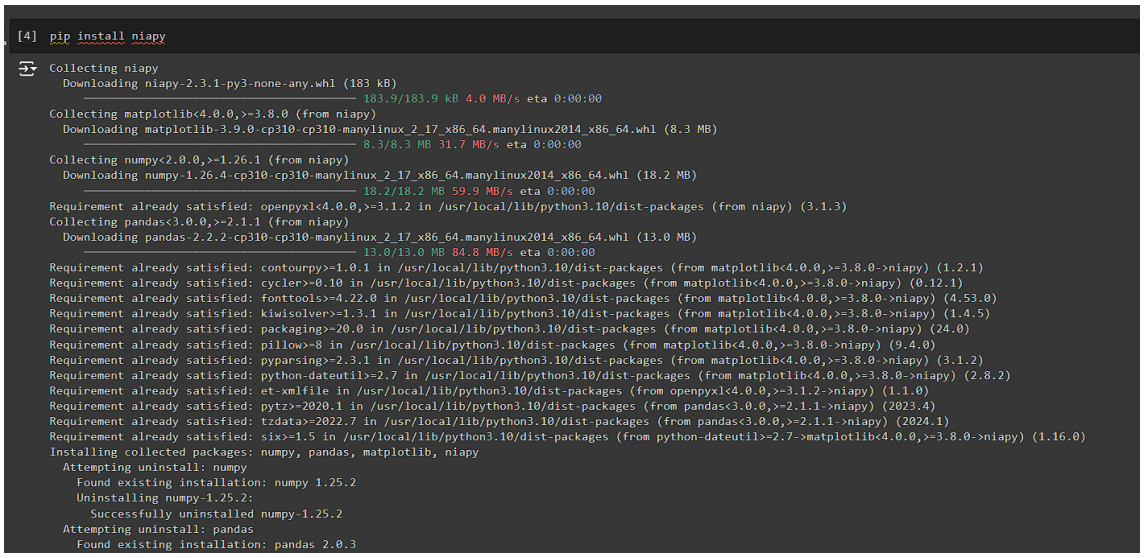
Setelah melakukan beberapa eksperimen, parameter diatas sudah optimal dikarenakan tidak ada perubahan signifikan jika menggunan parameter lain.

3. Pelatihan model



Model terbaik yang ditemukan adalah ExtraTreesClassifier dengan parameter yang tertera diatas.

- Niapy
  1. Install Tools
     Penginstalan tools Niapy sebagai libray model



  2. Model Pertama
     Model menggunakan pendekatan GeneticAlgorithm dalam pemodelannya dengan beberapa parameter seperti mencari Best C untuk menentukan akurasi, menentukan best kernel yang akan digunakan serta best gamma.

Model Niapy menggunakan pendekatan GeneticAlgorithm

```
[25] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from niapy.task import Task
     from niapy.problems import Problem
     from niapy.algorithms.basic import GeneticAlgorithm
     from sklearn.svm import SVC
     from sklearn.model_selection import cross_val_score
     from sklearn.metrics import accuracy_score
     import numpy as np

     # Define the optimization problem
     class SVMHyperparameterOptimization(Problem):
         def __init__(self):
             super().__init__(dimension=4, lower=[0.1, 0.0001, 0, 1], upper=[100, 1, 2, 5])

         def _evaluate(self, solution):
             C, gamma, kernel_idx, degree = solution
             kernels = ['linear', 'rbf', 'poly']
             kernel = kernels[int(kernel_idx)]
             if kernel == 'poly':
                 model = SVC(C=C, gamma=gamma, kernel=kernel, degree=int(degree))
             else:
                 model = SVC(C=C, gamma=gamma, kernel=kernel)
             scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
             return -np.mean(scores)

     # Create the optimization task
     task = Task(problem=SVMHyperparameterOptimization(), max_evals=300)

     # Perform the optimization using Genetic Algorithm
     algo = GeneticAlgorithm(population_size=50)
     best_solution = algo.run(task=task)

     # Extract the best hyperparameters
     best_C, best_gamma, best_kernel_idx, best_degree = best_solution[0]
     best_kernel = ['linear', 'rbf', 'poly'][int(best_kernel_idx)]
     print(f"Best C: {best_C}, Best gamma: {best_gamma}, Best kernel: {best_kernel}, Best degree: {best_degree}")
```

3. Model kedua

   Model kedua ini menggunakan pendekatan ParticleSwarmOptimization dengan parameter yang digunakan yaitu dim = 2, lower dan upper, kernel yang digunakan rbf dikarenakan sebelumnya sudah dicari kernel terbaik. serta evaluasinya = 5 dan max evaluasi = 200

Model Niapy menggunakan pendekatan ParticleSwarmOptimization

```python
from niapy.task import Task
from niapy.problems import Problem
from niapy.algorithms.basic import ParticleSwarmOptimization
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
import numpy as np

class SVMHyperparameterOptimization(Problem):
    def __init__(self):
        super().__init__(dimension=2, lower=[0.1, 0.0001], upper=[100, 1])

    def _evaluate(self, solution):
        C, gamma = solution
        model = SVC(C=C, gamma=gamma, kernel='rbf')
        scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
        return -np.mean(scores)

# Create the optimization task
task = Task(problem=SVMHyperparameterOptimization(), max_evals=200)

# Perform the optimization using Particle Swarm Optimization
algo = ParticleSwarmOptimization(population_size=30)
best_solution = algo.run(task=task)

# Extract the best hyperparameters
best_C, best_gamma = best_solution[0]
print(f"Best C: {best_C}, Best gamma: {best_gamma}")
```

```
Best C: 96.36801395535757, Best gamma: 0.0001
```

Evaluasi

1. TPOT

```
[ ] model.score(X_test_scaled, y_test)

0.7597402597402597

[ ] y_pred=model.predict(X_test_scaled)
    print(classification_report(y_test,y_pred))
    #Accuracy Score
    print('accuracy is ',accuracy_score(y_pred,y_test))
```

	precision	recall	f1-score	support
0.0	0.82	0.81	0.81	99
1.0	0.66	0.67	0.67	55
accuracy			0.76	154
macro avg	0.74	0.74	0.74	154
weighted avg	0.76	0.76	0.76	154

```
accuracy is  0.7597402597402597
```

Dikarenakan class data yang imbalance dimana jumlah yang tidak diabetes lebih banyak daripada yang memiliki diabetes, maka akurasi dari classifier tidak maksimal. Model dapat lebih baik memberikan prediksi untuk class 0 (tidak diabetes) daripada class 1 (diabetes). Meskipun telah dilakukan eksperimen dengan berbagai parameter, hasil akurasi model tetap berada di sekitar 75-76%.

2. **Niapy**

Hasil dari algoritma ini menunjukkan bahwa Best C: 1.1567437584303444, Best gamma: 0.185435851720627, Best kernel: rbf, Best degree: 4.122304955988525 Accuracy of the optimized model: 0.7337662337662337

```
# Train the model with the best hyperparameters
if best_kernel == 'poly':
    optimized_model = SVC(C=best_C, gamma=best_gamma, kernel=best_kernel, degree=int(best_degree))
else:
    optimized_model = SVC(C=best_C, gamma=best_gamma, kernel=best_kernel)

optimized_model.fit(X_train, y_train)

# Predict and evaluate the model
y_pred = optimized_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the optimized model: {accuracy}")
```

```
Best C: 1.1567437584303444, Best gamma: 0.185435851720627, Best kernel: rbf, Best degree: 4.122304955988525
Accuracy of the optimized model: 0.7337662337662337
```

Untuk algoritma kedua yang menunjukkan hasil yang lebih baik yaitu Accuracy of the optimized model: 0.7662337662337663 dengan Best C: 96.36801395535757, Best gamma: 0.0001.

```
from sklearn.metrics import accuracy_score

# Train the model with the best hyperparameters
optimized_model = SVC(C=best_C, gamma=best_gamma, kernel='rbf')
optimized_model.fit(X_train, y_train)

# Predict and evaluate the model
y_pred = optimized_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the optimized model: {accuracy}")
```

```
Accuracy of the optimized model: 0.7662337662337663
```

Bandingkan Langkah dan hasil kedua tool

TPOT:

- Langkah:
  - TPOT adalah alat otomatisasi machine learning yang melakukan pemilihan model dan tuning hyperparameter secara otomatis.
  - TPOT menggunakan algoritma genetic programming untuk mengeksplorasi berbagai kombinasi model dan parameter.
  - Dalam kasus ini, TPOT melakukan eksperimen dengan berbagai parameter untuk menemukan model terbaik.
- Hasil:
  - Model yang dihasilkan memiliki akurasi sekitar 75-76%.

    - TPOT menghadapi kesulitan dalam memberikan prediksi yang baik untuk kelas minoritas (diabetes) karena ketidakseimbangan data.

NiaPy:

- Langkah:
    - NiaPy adalah toolkit untuk optimasi yang menggunakan berbagai algoritma optimasi metaheuristik.
    - Dalam kasus ini, NiaPy digunakan untuk mengoptimalkan parameter model Support Vector Machine (SVM).
    - Parameter yang dioptimalkan termasuk C, gamma, kernel, dan degree.
- Hasil:
    - Hasil optimasi menunjukkan parameter terbaik sebagai berikut:
        - Best C: 1.1567437584303444
        - Best gamma: 0.185435851720627
        - Best kernel: RBF (Radial Basis Function)
        - Best degree: 4.122304955988525
    - Akurasi dari model yang telah dioptimasi mencapai 73.38%.

Kesimpulan

Baik TPOT maupun NiaPy menunjukkan hasil yang serupa dalam hal akurasi, yaitu sekitar 73-76%. Meskipun kedua alat ini melakukan optimasi hyperparameter yang signifikan, masalah ketidakseimbangan data tetap menjadi kendala utama. Untuk meningkatkan performa model lebih lanjut, beberapa langkah yang dapat dipertimbangkan adalah:

1. Penanganan Ketidakseimbangan Data:
    - Menggunakan teknik resampling seperti oversampling untuk kelas minoritas atau undersampling untuk kelas mayoritas.
    - Menggunakan metode Synthetic Minority Over-sampling Technique (SMOTE).
    - Menerapkan algoritma yang dirancang khusus untuk menangani data tidak seimbang seperti Balanced Random Forest atau EasyEnsemble.
2. Pemilihan Fitur:
    - Melakukan seleksi fitur untuk mengurangi redundansi dan meningkatkan relevansi fitur terhadap kelas target.
3. Penyesuaian Model:
    - Menggunakan algoritma machine learning yang lebih canggih dan kompleks yang mampu menangani ketidakseimbangan data dengan lebih baik.

Dengan langkah-langkah ini, diharapkan performa model dapat lebih ditingkatkan untuk memberikan prediksi yang lebih akurat, terutama untuk kelas minoritas (diabetes).