

**LAPORAN TUGAS BESAR**  
**PRAKTIKUM KECERDASAN BUATAN**  
**IMPLEMENTASI *MACHINE LEARNING***



Disusun oleh:

Kelas [B]

Dhea Muhamad Faisal A. – 2306034

Muhammad Faiz Alfarizi – 2306041

Dosen Pengampu Mata Kuliah:

Leni Fitriani, S.Kom, M.Kom

**INSTITUT TEKNOLOGI GARUT**  
**JURUSAN ILMU KOMPUTER**  
**PROGRAM STUDI TEKNIK INFORMATIKA**  
**TAHUN AKADEMIK 2024/2025**

## 1. PENDAHULUAN

Stroke merupakan salah satu penyakit paling berbahaya yang menjadi penyebab kematian dan kecacatan jangka Panjang di seluruh dunia. Deteksi dini terhadap risiko stroke menjadi penting agar pencegahan dan penanganan dapat dilakukan lebih cepat dan tepat. Dengan banyaknya data medis digital yang tersedia, pendekatan berbasis *machine learning* dapat dimanfaatkan untuk melakukan klasifikasi risiko stroke secara otomatis dan efisien.

Tujuan dari tugas besar ini adalah membangun sistem klasifikasi penyakit stroke dengan memanfaatkan algoritma *K-Nearest Neighbor* (K-NN). Tugas besar ini mencakup seluruh tahapan *life cycle machine learning*, dari pemahaman masalah hingga interpretasi hasil model.

Jurnal yang dipilih berasal dari jurnal ilmiah terakreditasi SINTA, yang membahas implementasi K-NN untuk klasifikasi stroke. Relevansi artikel ini sangat kuat karena topik dan pendekatannya serupa dengan proyek tugas besar ini.

## 2. RINGKASAN ARTIKEL PENELITIAN

No	Judul Penelitian	Metode Penelitian	Dataset	Temuan Utama	Keterbatasan
1	Perancangan Sistem Klasifikasi Pasien Stroke dengan Metode K-NN (Ashari, Otniel, and Rianto 2019)	K-Nearest Neighbor (K-NN)[1]	Kaggle stroke dataset	Akurasi mencapai 93,54% pada k=5, rancangan sistem dengan UML dan GUI.	Tidak ada pengujian terhadap imbalance dataset
2	Stroke Risk Prediction Using K-Nearest Neighbors Algorithm (Sudhakar Avareddy et al., 2024)	K-Nearest Neighbor (K-NN)[2]	Kaggle stroke dataset	Akurasi tinggi pada model K-NN, terbukti efektif memprediksi risiko stroke berdasarkan faktor-faktor klinis dan gaya hidup. Sistem dirancang dengan preprocessing, GUI, dan arsitektur sistem menggunakan UML	Tantangan pada penanganan data imbalance diakui, namun tidak dijelaskan metode penanganannya secara rinci
3	Perbandingan Algoritma Klasifikasi Data Mining untuk Prediksi Penyakit Stroke (Yufis Azhar et al., 2022)	K-NN, Logistic Regression, Random Forest, SVM, Naïve Bayes, Decision Tree C4.5[3]	Kaggle stroke dataset	Pada data tidak seimbang, akurasi tertinggi K-NN, SVM, dan Random Forest mencapai 98,63%. Pada data seimbang, SVM unggul dengan 76,52%	Akurasi tinggi pada data tidak seimbang menunjukkan potensi overfitting. Belum dijelaskan metode evaluasi lebih lanjut seperti recall atau precision

## 3. PROBLEM UNDERSTANDING

Masalah utama yang ingin diselesaikan adalah bagaimana mengklasifikasikan risiko stroke secara otomatis berdasarkan data pasien menggunakan pendekatan *machine learning*. Dalam tugas besar ini, algoritma K-Nearest Neighbor (K-NN) dipilih karena kesederhanaannya serta kemampuannya yang cukup baik untuk klasifikasi pada dataset berukuran sedang.

Kebaruan pendekatan kami dibandingkan dengan jurnal acuan adalah:

- Penggunaan SMOTE untuk menangani masalah ketidakseimbangan data (imbalanced class).
- Pencarian nilai k terbaik menggunakan GridSearchCV, bukan hanya mencoba nilai tetap.
- Visualisasi performa model melalui confusion matrix dan classification report.

#### 4. DATA UNDERSTANDING

Dataset yang digunakan adalah “*Healthcare Dataset Stroke Data*” dari Kaggle, yang berisi informasi Kesehatan pasien terkait risiko stroke.

##### 1. Pemuatan dan Pemeriksaan Awal Data

Langkah awal adalah memuat dataset dan memeriksa beberapa baris pertama untuk mendapatkan gambaran umum tentang struktur data dan nilai-nilai yang ada.

```
# Library untuk memproses dan visualisasi data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Untuk preprocessing dan evaluasi
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    precision_score,
    recall_score,
    f1_score,
)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score, ShuffleSplit
from sklearn.metrics import roc_curve, auc
```

Kode di atas mengimpor semua *library* Python yang diperlukan untuk analisis data, *preprocessing*, pembentukan model, dan evaluasi. Setelah itu, dataset dimuat dari jalur yang ditentukan ke dalam *DataFrame* pandas bernama *data*.

## 2. Membaca Dataset

```
[ ] data = pd.read_csv(dataset_path)
```

data.head(20)

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
10	12109	Female	81.0	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
11	12095	Female	61.0	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12	12175	Female	54.0	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
13	8213	Male	78.0	0	1	Yes	Private	Urban	219.84	NaN	Unknown	1
14	5317	Female	79.0	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1
15	58202	Female	50.0	1	0	Yes	Self-employed	Rural	167.41	30.9	never smoked	1
16	56112	Male	64.0	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
17	34120	Male	75.0	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
18	27458	Female	60.0	0	0	No	Private	Urban	89.22	37.8	never smoked	1
19	25226	Male	57.0	0	1	No	Govt_job	Urban	217.08	NaN	Unknown	1

Dataset terdiri dari 5110 baris dan 12 kolom. Setiap baris merepresentasikan satu pasien, sedangkan kolom-kolom menggambarkan atribut seperti:

- *age*: umur (numerik).
- *gender*: jenis kelamin (kategorik).
- *hypertension*, *heart\_disease*: status penyakit (biner).
- *work\_type*, *Residence\_type*, *smoking\_status*: kategorik.
- *avg\_glucose\_level*, *bmi*: numerik.
- *stroke*: target klasifikasi (1 = stroke, 0 = tidak stroke).

```
print(data.head(10))
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1

Output *data.head(10)* menunjukkan 10 baris pertama dari dataset. Kita dapat melihat beberapa kolom seperti *id*, *gender*, *age*, *bmi*, dan *stroke*. Terlihat adanya nilai *NaN* (Not a Number) pada kolom *bmi* yang menandakan nilai hilang, dan beberapa kolom kategorikal

seperti *gender*, *work\_type*, dan *smoking\_status* yang perlu diproses lebih lanjut. Kolom *stroke* adalah target klasifikasi (1 untuk stroke, 0 untuk tidak

### 3. Informasi Umum Dataset

Pemeriksaan *data.info()* memberikan ringkasan yang lebih detail tentang jumlah entri, tipe data setiap kolom, dan keberadaan nilai non-null.

```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   id                    5110 non-null   int64  
 1   gender                5110 non-null   object  
 2   age                  5110 non-null   float64 
 3   hypertension          5110 non-null   int64  
 4   heart_disease         5110 non-null   int64  
 5   ever_married          5110 non-null   object  
 6   work_type             5110 non-null   object  
 7   Residence_type        5110 non-null   object  
 8   avg_glucose_level     5110 non-null   float64 
 9   bmi                   4909 non-null   float64 
10   smoking_status        5110 non-null   object  
11   stroke                5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

- Kolom *bmi* dan *avg\_glucose\_level* bertipe float.
- Kolom *gender*, *ever\_married*, *work\_type*, dan lain-lain bertipe object (kategorikal).
- Dataset cukup lengkap, tetapi kolom *bmi* memiliki nilai yang hilang (missing values).

### 4. Statistik Ringkas Dataset

```
print(data.describe())
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

- Rata-rata usia pasien adalah 43 tahun dengan variasi besar (std 22 tahun).
- Nilai *bmi* memiliki outlier (maksimum 97.6).

- Sekitar 10% pasien memiliki hipertensi, dan 5% memiliki penyakit jantung.

#### 5. Cek Nilai Unik Kolom kategorik

```
[ ] print("Unique value of gender: ", data["gender"].unique())
    print("Unique value of Married status: ", data["ever_married"].unique())
    print("Unique value of Stroke status: ", data["stroke"].unique())

⇒ Unique value of gender:  ['Male' 'Female' 'Other']
   Unique value of Married status:  ['Yes' 'No']
   Unique value of Stroke status:  [1 0]
```

#### 6. Distribusi Gender

```
[ ] print(data["gender"].value_counts())

⇒ gender
   Female    2994
   Male      2115
   Other         1
   Name: count, dtype: int64
```

- Mayoritas data adalah perempuan
- Hanya satu data dengan gender *other*, yang mungkin bisa di-drop agar tidak memengaruhi distribusi.

#### 7. Distribusi Stroke

```
[ ] print(data["stroke"].value_counts())

⇒ stroke
   0    4861
   1     249
   Name: count, dtype: int64
```

- Data sangat tidak seimbang (imbalanced) karena hanya 249 dari 5110 data yang mengalami stroke.
- Hal ini perlu diperhatikan saat modeling karena model bisa bias terhadap kelas mayoritas.

### 5. DATA PREPARATION

Bagian ini berfokus pada langkah-langkah *preprocessing* data untuk mengatasi nilai hilang, menghapus kolom tidak relevan, dan mengubah variable kategorikal menjadi format numerik yang dapat diproses oleh model *machine learning*. Juga dijelaskan pembagian data dan normalisasi.

#### 1. Pemeriksaan awal missing values

```

▶ print(data.isnull().sum())
↔ id            0
   gender       0
   age          0
   hypertension 0
   heart_disease 0
   ever_married 0
   work_type     0
   Residence_type 0
   avg_glucose_level 0
   bmi          201
   smoking_status 0
   stroke        0
   dtype: int64

```

- Hanya kolom *bmi* yang memiliki nilai kosong sebanyak 201 baris, dari total 5110 data.
- Artinya sekitar 3,9% data pada kolom *bmi* hilang (201/5110).
- Kolom lain tidak memiliki missing values dan siap digunakan tanpa preprocessing tambahan.

## 2. Penanganan missing values pada kolom *bmi*.

```

[ ] data['bmi'].fillna(data['bmi'].median(), inplace=True)

↔ /tmp/ipython-input-14-1997496975.py:1: FutureWarning: A value is being
   The behavior will change in pandas 3.0. This inplace method will r

   For example, when doing 'df[col].method(value, inplace=True)', try

   data['bmi'].fillna(data['bmi'].median(), inplace=True)

```

- Median digunakan karena data *bmi* mengandung outlier, sehingga lebih stabil dibanding rata-rata.

## 3. Pemeriksaan akhir missing values.

```

▶ print(data.isnull().sum())
↔ id            0
   gender       0
   age          0
   hypertension 0
   heart_disease 0
   ever_married 0
   work_type     0
   Residence_type 0
   avg_glucose_level 0
   bmi          0
   smoking_status 0
   stroke        0
   dtype: int64

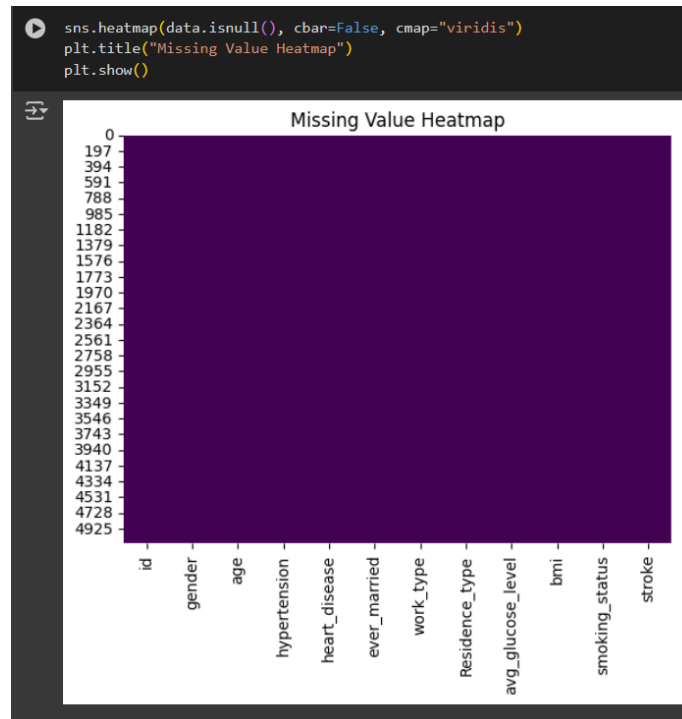
```

- Setelah menjalankan perintah `fillna`, pernyataan `print` ini dijalankan lagi untuk memverifikasi nilai-nilai yang hilang.



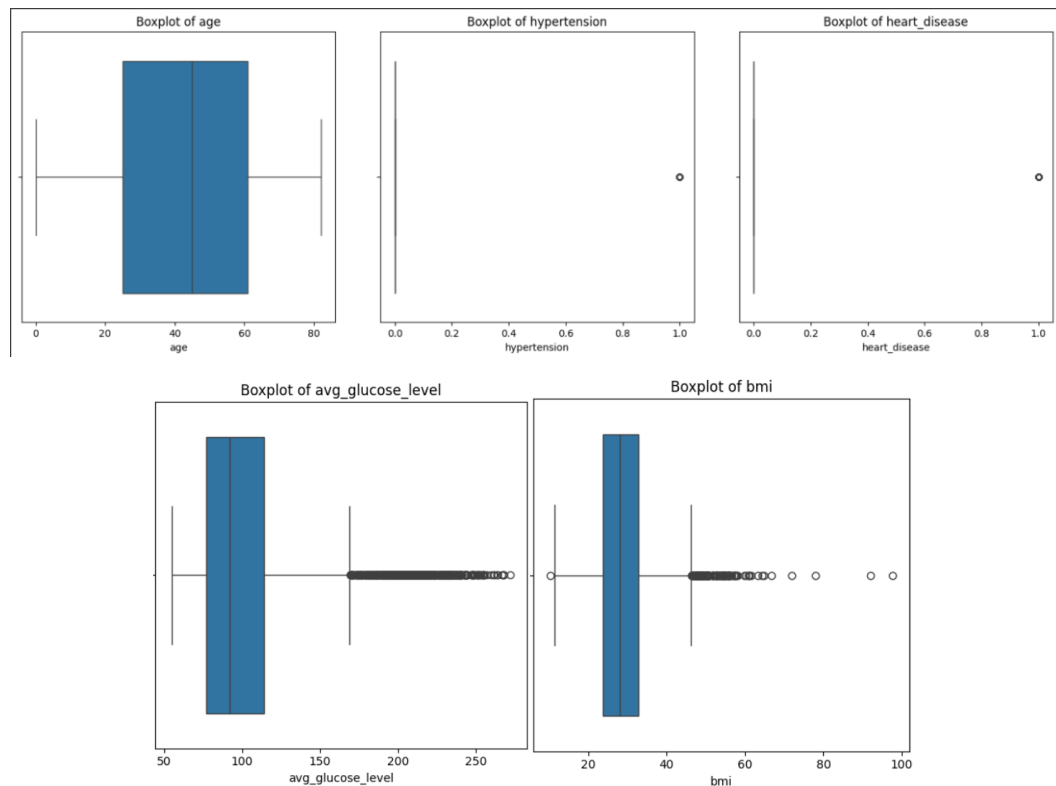
- Hasilnya sekarang menunjukkan 0 nilai yang hilang untuk kolom *bmi*, bersama dengan semua kolom lainnya. Ini menegaskan bahwa nilai-nilai yang hilang di kolom *bmi* berhasil diisi dengan nilai median.

#### 4. Missing value heatmap



- Heatmap yang ditampilkan sepenuhnya berwarna gelap (ungu tua/hitam). Ini mengindikasikan bahwa tidak ada lagi *missing value* di seluruh dataset setelah penanganan yang dilakukan diatas. Jika ada *missing value*, akan ada garis atau blok berwarna cerah (tergantung pada *colormap*) pada kolom yang memiliki nilai kosong.

## 5. Boxplot sebelum melakukan deteksi dan penanganan outlier



- *age*: Distribusi relatif normal tanpa *outlier* yang jelas.
- *hypertension* & *heart\_disease*: Karena ini adalah variabel biner, "outlier" pada nilai 1 hanya menunjukkan proporsi kecil dari populasi yang memiliki kondisi tersebut, bukan anomali data yang perlu dihapus.
- *avg\_glucose\_level* & *bmi*: Kedua kolom ini menunjukkan **keberadaan outlier yang signifikan** di sisi nilai yang lebih tinggi. Ini adalah temuan penting dalam EDA yang mungkin memerlukan penanganan lebih lanjut (seperti transformasi data, capping, atau penghapusan) tergantung pada tujuan analisis Anda. Keberadaan *outlier* ini bisa jadi adalah data yang valid (misalnya, orang yang benar-benar memiliki glukosa tinggi atau *BMI* sangat tinggi) atau bisa juga merupakan *entry* data yang salah. Pemeriksaan lebih lanjut diperlukan.

## 6. Data Cleaning

```
[ ] data_clean = data.copy()
```

- Membuat salinan dari DataFrame *data* yang sudah bersih dari *missing value* (dari langkah sebelumnya). Ini adalah praktik yang baik agar operasi pembersihan *outlier* tidak mengubah DataFrame *data* asli.

## 7. Deteksi dan penanganan outlier

```
# Definiskan fungsi untuk menghapus outlier menggunakan metode IQR
def remove_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

# Tentukan kolom numerik yang ingin dicek outlier-nya
numeric_cols = ["age", "hypertension", "heart_disease", "avg_glucose_level", "bmi"]

# Terapkan fungsi remove_outliers untuk tiap kolom numerik
for col in numeric_cols:
    data_clean = remove_outliers(data_clean, col)
    print(
        f"Setelah pembersihan outlier, data[{col}] memiliki {data_clean.shape[0]} baris."
    )

# Tampilkan statistik deskriptif setelah pembersihan outlier
print(data_clean.describe())
```

Setelah pembersihan outlier, data[age] memiliki 5110 baris.  
Setelah pembersihan outlier, data[hypertension] memiliki 4612 baris.  
Setelah pembersihan outlier, data[heart\_disease] memiliki 4400 baris.  
Setelah pembersihan outlier, data[avg\_glucose\_level] memiliki 3985 baris.  
Setelah pembersihan outlier, data[bmi] memiliki 3897 baris.

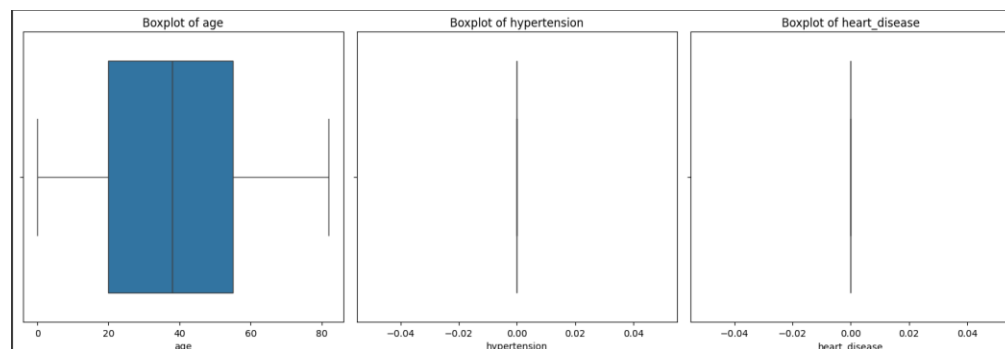
	id	age	hypertension	heart_disease \
count	3897.000000	3897.000000	3897.0	3897.0
mean	36520.891968	38.284701	0.0	0.0
std	21128.282548	21.917049	0.0	0.0
min	67.000000	0.000000	0.0	0.0
25%	17762.000000	20.000000	0.0	0.0
50%	36958.000000	30.000000	0.0	0.0
75%	54579.000000	55.000000	0.0	0.0
max	72940.000000	82.000000	0.0	0.0

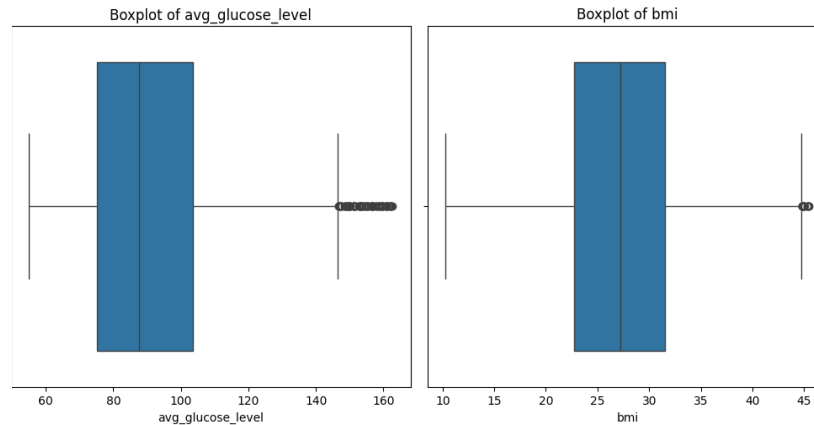
  

	avg_glucose_level	bmi	stroke
count	3897.000000	3897.000000	3897.000000
mean	91.049012	27.420862	0.027970
std	22.033285	6.552908	0.164909
min	55.120000	10.300000	0.000000
25%	75.060000	22.700000	0.000000
50%	87.740000	27.200000	0.000000
75%	103.660000	31.500000	0.000000
max	162.720000	45.500000	1.000000

- Bagian EDA ini mendemonstrasikan penanganan *outlier* yang sistematis menggunakan metode IQR. Dengan menghapus *outlier*, distribusi data numerik menjadi lebih terkonsentrasi, yang bisa membantu dalam analisis statistik atau pemodelan yang sensitif terhadap nilai ekstrem.

## 8. Boxplot setelah melakukan deteksi dan penanganan outlier





- **age:** Distribusi tetap stabil karena tidak banyak *outlier* yang dihapus dari kolom ini secara langsung.
- **hypertension dan heart\_disease:** Boxplot menunjukkan efek sampling yang signifikan. Dengan menghapus *outlier* statistik (nilai 1 karena minoritas), data yang merepresentasikan kasus positif penyakit/kondisi ini hampir sepenuhnya atau seluruhnya telah dihilangkan. Ini adalah masalah besar jika tujuan analisis adalah memprediksi atau memahami penyakit ini, karena data target menjadi sangat tidak seimbang atau bahkan hilang.
- **avg\_glucose\_level dan bmi:** Boxplot menunjukkan keberhasilan yang jelas dalam menghilangkan *outlier* ekstrem. Rentang data sekarang lebih sempit dan mencerminkan distribusi mayoritas populasi, tanpa nilai-nilai ekstrem yang dapat mempengaruhi model secara negatif.

## 9. Inisialisasi MinMaxScaler dengan rentang 0-1

```
# Inisialisasi MinMaxScaler dengan rentang 0-1
scaler = MinMaxScaler(feature_range=(0.1, 0.9))

# Terapkan scaler pada kolom numerik
numeric_cols = ["age", "hypertension", "heart_disease", "avg_glucose_level", "bmi"]
data_clean[numeric_cols] = scaler.fit_transform(data_clean[numeric_cols])

# Tampilkan beberapa baris hasil scaling
print(data_clean[numeric_cols].head())
```

	age	hypertension	heart_disease	avg_glucose_level	bmi
7	0.773047	0.1	0.1	0.391970	0.384091
8	0.675391	0.1	0.1	0.256357	0.504545
9	0.860938	0.1	0.1	0.125651	0.415909
12	0.626563	0.1	0.1	0.467212	0.486364
18	0.685156	0.1	0.1	0.353532	0.725000

- Proses penskalaan dengan *MinMaxScaler* berhasil mengubah semua fitur numerik ke dalam rentang [0.1, 0.9]. Ini adalah langkah penting dalam **data preprocessing** yang diperlukan untuk mempersiapkan data agar cocok untuk pelatihan model machine learning, memastikan bahwa semua fitur memiliki kontribusi yang setara selama proses pembelajaran. Namun, hasil penskalaan untuk *hypertension* dan *heart\_disease*

menyoroti kembali efek dari penghapusan outlier sebelumnya, yang secara efektif menghilangkan variabilitas (nilai 1) dari kolom-kolom biner tersebut.

#### 10. Inisialisasi label encoder

```
# Inisialisasi label encoder
le = LabelEncoder()

# Encode kolom 'Gender'
data_clean["gender"] = le.fit_transform(data_clean["gender"])
data["gender"] = le.fit_transform(data["gender"])
# Encode kolom 'Breastfeeding'
data_clean["ever_married"] = le.fit_transform(data_clean["ever_married"])
data["ever_married"] = le.fit_transform(data["ever_married"])
# Encode kolom 'Stunting'
data_clean["stroke"] = le.fit_transform(data_clean["stroke"])
data["stroke"] = le.fit_transform(data["stroke"])

# Tampilkan nilai unik setelah encoding
print("Unique values in Gender:", data_clean["gender"].unique())
print("Unique values in ever_married:", data_clean["ever_married"].unique())
print("Unique values in Stroke:", data_clean["stroke"].unique())
```

```
Unique values in Gender: [0 1 2]
Unique values in ever_married: [0 1]
Unique values in Stroke: [1 0]
```

- Bagian ini merupakan langkah data preprocessing yang penting untuk mengubah variabel kategorikal menjadi format numerik yang dapat dipahami oleh algoritma machine learning. *LabelEncoder* berhasil diterapkan pada kolom *gender*, *ever\_married*, dan *stroke*, mengubah label teks menjadi representasi numerik. Ini adalah persiapan penting sebelum data digunakan untuk pemodelan.

## 6. MODELING

Algoritma yang digunakan adalah K-Nearest Neighbor (K-NN) karena cocok untuk klasifikasi sederhana dan efektif pada dataset kecil.

### 1. Melakukan Data Balanced dan Menerapkan SMOTE

```
from imblearn.over_sampling import SMOTE, ADASYN
from imblearn.under_sampling import NearMiss, RandomUnderSampler
import pandas as pd # Ensure pandas is imported

# Identify remaining categorical columns that are not yet numerical
categorical_cols_to_encode = ['work_type', 'residence_type', 'smoking_status']

# Apply one-hot encoding to the remaining categorical columns
data_clean = pd.get_dummies(data_clean, columns=categorical_cols_to_encode, drop_first=True)

# Pisahkan fitur dan label
X = data_clean.drop(columns=['stroke'])
y = data_clean['stroke']

# Terapkan SMOTE untuk oversampling kelas minoritas
smote = SMOTE(random_state=42)
adasyn = ADASYN(random_state=42)
nm = NearMiss()
rus = RandomUnderSampler(random_state=42, sampling_strategy="auto")
X_balanced, y_balanced = smote.fit_resample(X, y)

# Buat dataframe dari hasil SMOTE untuk analisis correlation
data_balanced = pd.DataFrame(X_balanced, columns=X.columns)
data_balanced['stroke'] = y_balanced
# Tampilkan distribusi kelas setelah balancing
print("Distribusi kelas setelah balancing:")
print(pd.Series(y_balanced).value_counts())
```

Distribusi kelas setelah balancing:

stroke	count
1	3788
0	3788

Name: count, dtype: int64

- Tahapan ini berhasil menangani masalah ketidakseimbangan kelas pada variabel target stroke menggunakan teknik oversampling SMOTE. Ini sangat penting untuk membangun model machine learning yang tidak bias terhadap kelas mayoritas dan mampu memprediksi kelas minoritas dengan lebih baik.

### 2. Menampilkan ulang MinMax Scaler setelah balanced

```
# Inisialisasi MinMaxScaler dengan rentang 0-1
scaler = MinMaxScaler(feature_range=(0.1, 0.9))

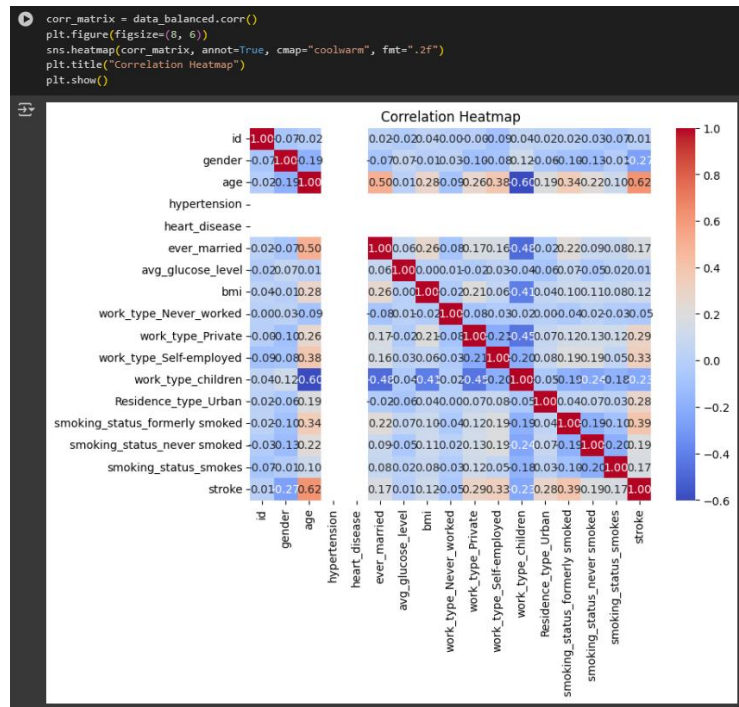
# Terapkan scaler pada kolom numerik
data_balanced[numeric_cols] = scaler.fit_transform(data_balanced[numeric_cols])

# Tampilkan beberapa baris hasil scaling
print(data_balanced[numeric_cols].head())
```

	age	hypertension	heart_disease	avg_glucose_level	bmi
0	0.773047	0.1	0.1	0.391970	0.384091
1	0.675391	0.1	0.1	0.256357	0.504545
2	0.860938	0.1	0.1	0.125651	0.415909
3	0.626563	0.1	0.1	0.467212	0.486364
4	0.685156	0.1	0.1	0.353532	0.725000

- Penskalaan ulang ini adalah langkah krusial untuk memastikan bahwa semua fitur numerik dalam dataset yang sudah di-balance (data\_balanced) berada dalam skala yang seragam dan optimal untuk input model machine learning. Ini adalah langkah terakhir dalam pra-pemrosesan data sebelum data siap untuk pelatihan model.

### 3. Menampilkan Correlation Heatmap dari data balanced



- Heatmap korelasi memberikan wawasan berharga tentang hubungan antar fitur dan fitur dengan variabel target *stroke*. Fitur seperti *age*, *avg\_glucose\_level*, *hypertension*, dan *heart\_disease* menunjukkan korelasi positif dengan *stroke*, mengindikasikan bahwa mereka mungkin merupakan prediktor penting untuk *stroke*. Ini menjadi dasar untuk seleksi fitur dan pemodelan selanjutnya.
4. Pembagian data training (80%) dan testing (20%)

```

[ ] X = data_balanced.drop(columns=["stroke"])
y = data_balanced["stroke"]
# Bagi data ke dalam training dan testing set, 80% training dan 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Tampilkan ukuran masing-masing set
print("Ukuran Training set:", X_train.shape)
print("Ukuran Testing set:", X_test.shape)

```

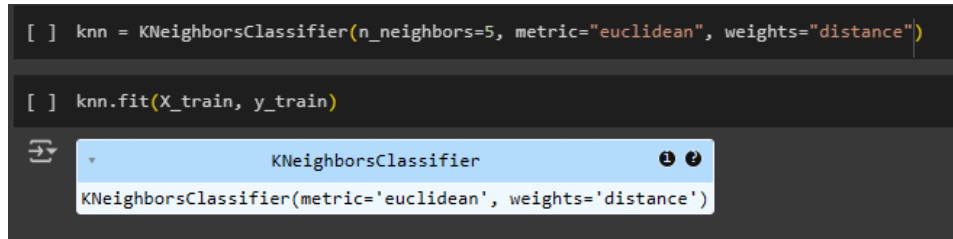
Ukuran Training set: (5303, 16)  
Ukuran Testing set: (2273, 16)

- Dataset telah berhasil dibagi menjadi set pelatihan dan pengujian, langkah fundamental untuk melatih dan mengevaluasi model machine learning secara objektif, memastikan bahwa model tidak "melihat" data pengujian selama pelatihan.

## 5. Pelatihan model K-NN

```
[ ] knn = KNeighborsClassifier(n_neighbors=5, metric="euclidean", weights="distance")

[ ] knn.fit(X_train, y_train)
```



- Model KNN telah berhasil diinisialisasi dengan parameter yang ditentukan dan dilatih menggunakan set data pelatihan ( $X_{train}$  dan  $y_{train}$ ). Ini adalah langkah awal dalam proses evaluasi model, di mana model ini akan digunakan untuk membuat prediksi pada  $X_{test}$  dan kinerjanya akan diukur.

## 7. EVALUATION

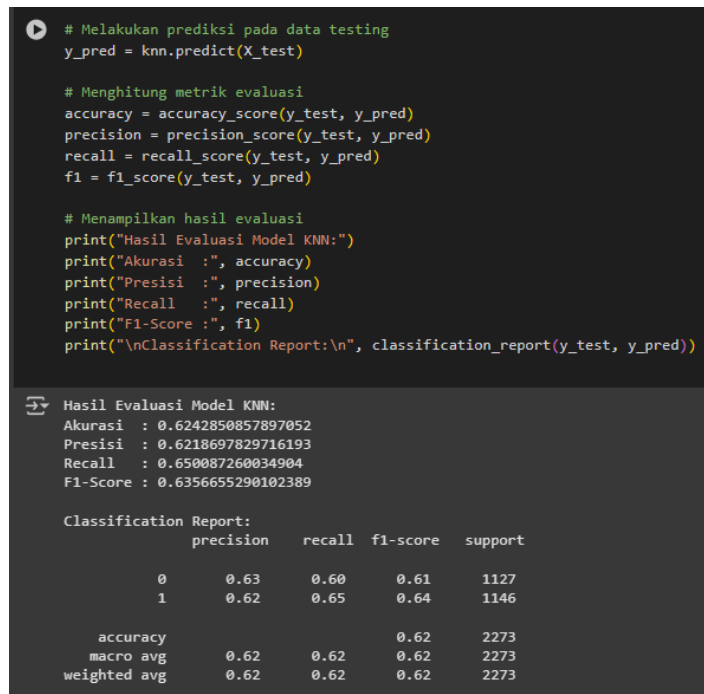
Berikut adalah evaluasi dari klasifikasi penyakit stroke menggunakan K-NN.

### 1. Evaluasi awal model K-NN (sebelum tuning)

```
# Melakukan prediksi pada data testing
y_pred = knn.predict(X_test)

# Menghitung metrik evaluasi
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Menampilkan hasil evaluasi
print("Hasil Evaluasi Model KNN:")
print("Akurasi :", accuracy)
print("Presisi :", precision)
print("Recall :", recall)
print("F1-Score :", f1)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```



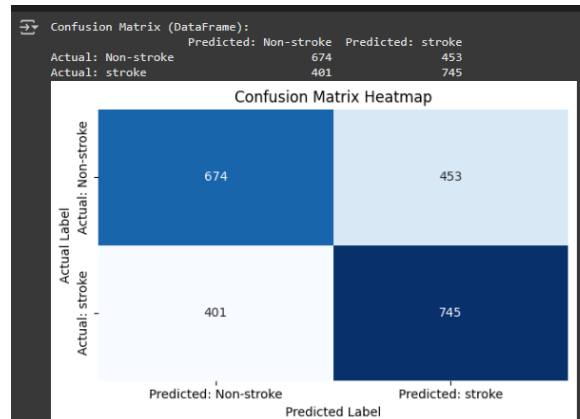
	precision	recall	f1-score	support
0	0.63	0.60	0.61	1127
1	0.62	0.65	0.64	1146
accuracy			0.62	2273
macro avg	0.62	0.62	0.62	2273
weighted avg	0.62	0.62	0.62	2273

- Akurasi (~62.43%): Menunjukkan proporsi prediksi yang benar secara keseluruhan. Angka ini sedang, tidak terlalu tinggi.
- Presisi (~62.19%): Dari semua yang diprediksi positif (stroke), sekitar 62.19% adalah benar-benar stroke.



- Recall (~65.01%): Dari semua kasus stroke yang sebenarnya, model dapat mengidentifikasi sekitar 65.01% di antaranya. Ini menunjukkan kemampuan model dalam menangkap sebagian besar kasus positif.
- F1-Score (~63.57%): Merupakan rata-rata harmonik dari precision dan recall, memberikan keseimbangan antara kedua metrik.
- Model KNN menunjukkan kinerja yang moderat dalam memprediksi stroke. Metrik evaluasi ini akan menjadi dasar untuk membandingkan model ini dengan model lain atau untuk melakukan fine-tuning parameter KNN untuk meningkatkan kinerjanya. Perhatikan bahwa meskipun dataset telah dibalance, kinerja model masih dapat ditingkatkan.

## 2. Confusion Matrix Mode Awal



- Confusion matrix memberikan rincian penting tentang jenis kesalahan yang dibuat model. Terlihat bahwa model memiliki jumlah **False Positives** (453) dan **False Negatives** (401) yang cukup signifikan. Jumlah False Negatives (tidak mendeteksi stroke padahal sebenarnya stroke) adalah perhatian utama dalam konteks medis, dan analisis ini akan membantu dalam upaya peningkatan model selanjutnya.

### 3. GridSearchCV untuk Hyperparameter Tuning

```
[ ] # Buat parameter grid untuk GridSearch
param_grid = {
    "n_neighbors": range(3, 22, 2), # Menggunakan nilai ganjil untuk n_neighbors
    "weights": ["uniform", "distance"],
    "metric": [
        "euclidean",
        "manhattan",
        "minkowski",
    ], # Pilihan metrik, bisa ditambahkan opsi lain jika perlu
}

# Inisialisasi GridSearchCV dengan 5-fold cross-validation
grid_search = GridSearchCV(
    estimator=knn, param_grid=param_grid, cv=5, scoring="accuracy", n_jobs=-1, verbose=1
)

# Lakukan pencarian grid pada data training (menggunakan fitur yang sudah di-balance dan feature engineered)
grid_search.fit(X_train, y_train)

# Cetak parameter terbaik dan skor terbaik dari grid search
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)

# Gunakan model terbaik untuk prediksi pada data testing
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_test)

# Evaluasi model terbaik
accuracy_best = accuracy_score(y_test, y_pred_best)
print("Test Set Accuracy with Best Parameters:", accuracy_best)
print("\nClassification Report:\n", classification_report(y_test, y_pred_best))

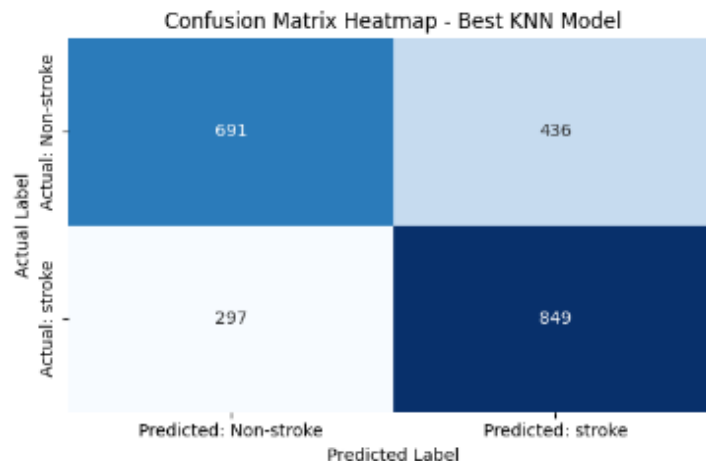
# Tampilkan confusion matrix dalam bentuk DataFrame
cm_best = confusion_matrix(y_test, y_pred_best)
cm_df_best = pd.DataFrame(
    cm_best,
    index=["Actual: Non-stroke", "Actual: stroke"],
    columns=["Predicted: Non-stroke", "Predicted: stroke"],
)
print("Confusion Matrix (Best Model):")
print(cm_df_best)

Fitting 5 folds for each of 60 candidates, totalling 300 fits
Best Parameters: {'metric': 'manhattan', 'n_neighbors': 21, 'weights': 'uniform'}
Best Cross-Validation Accuracy: 0.6869635267547325
Test Set Accuracy with Best Parameters: 0.6775186977562693

Classification Report:
      precision    recall  f1-score   support

0         0.70      0.61      0.65      1127
1         0.66      0.74      0.70      1146

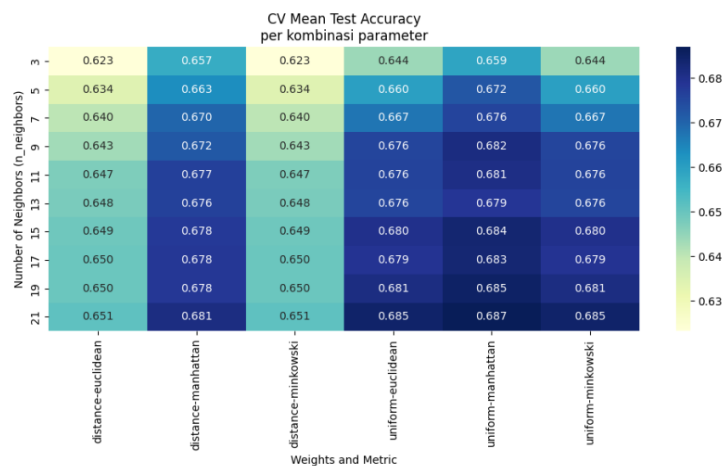
 accuracy      0.68      0.68      0.68      2273
  macro avg      0.68      0.68      0.68      2273
 weighted avg      0.68      0.68      0.68      2273
```



- Test Set Accuracy (~67.75%): Meningkat dari akurasi awal (~62.43%) setelah tuning.
- Classification Report: Menunjukkan peningkatan pada metrik untuk kedua kelas dibandingkan model KNN awal. F1-score untuk kelas 1 (stroke) meningkat menjadi 0.70.
- Confusion Matrix (setelah tuning):
  - TN: 691 (meningkat dari 674).
  - FP: 436 (menurun dari 453).

- FN: 297 (menurun drastis dari 401).
- TP: 849 (meningkat dari 745).
- Penting: Penurunan False Negatives (FN) dari 401 menjadi 297 sangat signifikan, yang berarti model terbaik lebih baik dalam mengidentifikasi kasus stroke yang sebenarnya.
- GridSearchCV berhasil menemukan kombinasi hyperparameter yang lebih baik untuk model KNN, menghasilkan peningkatan yang substansial dalam akurasi dan metrik lainnya, terutama dalam mengurangi False Negatives. Ini menunjukkan efektivitas hyperparameter tuning dalam meningkatkan kinerja model.

#### 4. GridSearchCV dan Heatmap Akurasi Hasil CV



- Heatmap ini secara visual mengkonfirmasi hasil dari GridSearchCV, menunjukkan bahwa akurasi model KNN sangat bergantung pada kombinasi hyperparameter yang dipilih. Akurasi tertinggi ditemukan dengan  $n\_neighbors=21$ ,  $weights='uniform'$ , dan  $metric='manhattan'$ , yang sesuai dengan "Best Parameters" yang dicetak oleh *GridSearchCV*. Visualisasi ini membantu dalam memahami secara intuitif bagaimana perubahan hyperparameter memengaruhi kinerja model.

## 5. Visualisasi Performa Akurasi Lintas Parameter



- Pola Umum: Terlihat bahwa akurasi model bervariasi dengan  $n\_neighbors$  dan kombinasi metrik/bobot.
- Garis Oranye (Weights: uniform, Metric: manhattan): Garis ini (berwarna oranye) menunjukkan akurasi tertinggi secara konsisten, mencapai puncaknya di sekitar 0.69 - 0.70 pada  $n\_neighbors$  yang lebih tinggi (mendekati 21). Ini mengkonfirmasi hasil *GridSearchCV* sebelumnya tentang parameter terbaik.
- Garis Lain: Garis-garis lain menunjukkan akurasi yang lebih rendah. Misalnya, kombinasi dengan *weights: distance* (biru, hijau, merah) umumnya memiliki akurasi yang lebih rendah dibandingkan dengan *weights: uniform*.
- Stabilitas: Akurasi cenderung menjadi lebih stabil atau sedikit meningkat pada nilai  $n\_neighbors$  yang lebih besar untuk beberapa kombinasi, sementara yang lain mungkin menurun setelah mencapai puncaknya.
- Plot garis ini adalah alat visual yang sangat baik untuk memahami dampak gabungan dari  $n\_neighbors$ , *weights*, dan *metric* terhadap akurasi model. Ini secara jelas menunjukkan mengapa kombinasi  $n\_neighbors=21$ , *weights='uniform'*, dan *metric='manhattan'* dianggap sebagai yang terbaik, karena secara konsisten memberikan akurasi tertinggi di antara semua konfigurasi yang diuji. Ini juga menyoroti pentingnya hyperparameter tuning untuk menemukan konfigurasi model yang optimal.

## **8. INTERPRETATION**

Penelitian ini menggunakan algoritma K-Nearest Neighbor (K-NN) untuk melakukan klasifikasi penyakit stroke berdasarkan dataset medis pasien. K-NN merupakan algoritma machine learning sederhana yang sangat efektif untuk tugas klasifikasi, terutama pada kasus-kasus di mana data historis (data latih) tersedia dalam jumlah cukup besar dan berkualitas baik.

Dalam penelitian serupa yang dirujuk dari referensi, K-NN terbukti efektif dengan nilai parameter  $k$  yang optimal memberikan hasil akurasi yang baik. Berdasarkan penelitian terdahulu, nilai parameter  $k = 5$  memberikan akurasi sebesar 93,54% untuk klasifikasi pasien stroke.

Penelitian yang dilakukan menunjukkan bahwa algoritma K-NN mampu menghasilkan akurasi yang tinggi dan stabil untuk klasifikasi pasien stroke, dengan performa terbaik diperoleh melalui proses tuning parameter  $k$  menggunakan cross-validation. Selain akurasi, metrik lain seperti precision, recall, dan f1-score juga digunakan untuk memberikan gambaran lebih komprehensif terkait performa klasifikasi yang diperoleh.

Sebagai hasil akhir, sistem yang dirancang pada penelitian ini memungkinkan proses klasifikasi pasien stroke secara otomatis berdasarkan data medis yang tersedia. Dengan pendekatan machine learning seperti K-NN, deteksi dini dan klasifikasi risiko stroke dapat dilakukan lebih cepat, akurat, dan efisien dibandingkan metode manual.

## **9. KESIMPULAN**

Berdasarkan penelitian dan analisis yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Algoritma K-Nearest Neighbor (K-NN) efektif digunakan untuk klasifikasi penyakit stroke dengan dataset medis pasien, terutama setelah dilakukan tuning parameter menggunakan GridSearchCV.
2. Teknik oversampling SMOTE berhasil mengatasi masalah ketidakseimbangan data (imbalanced class), yang secara signifikan meningkatkan performa model dalam mengidentifikasi kelas minoritas (stroke).
3. Teknik oversampling SMOTE berhasil mengatasi masalah ketidakseimbangan data (imbalanced class), yang secara signifikan meningkatkan performa model dalam mengidentifikasi kelas minoritas (stroke).

4. Parameter optimal yang diperoleh melalui GridSearchCV adalah  $n\_neighbors=21$ ,  $weights='uniform'$ , dan  $metric='manhattan'$ . Parameter ini menghasilkan performa terbaik dalam klasifikasi risiko stroke.
5. Pendekatan machine learning, khususnya algoritma K-NN yang telah dioptimalkan, mampu menyediakan solusi klasifikasi otomatis yang lebih cepat, akurat, dan efisien dibanding metode manual, sehingga dapat mendukung proses pengambilan keputusan dalam deteksi dini risiko stroke di lingkungan medis.

## 10. DAFTAR PUSTAKA

- [1] R. A. D. Yulianto, I. Riadi, and R. Umar, “PERANCANGAN KLASIFIKASI PASIEN STROKE DENGAN METODE K-NEAREST NEIGHBOR,” *Rabit : Jurnal Teknologi dan Sistem Informasi Univrab*, vol. 8, no. 2, pp. 262–268, Sep. 2023, doi: 10.36341/rabit.v8i2.3454.
- [2] S. Avareddy and A. Professor, “Stroke risk prediction using K-Nearest Neighbors algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering Impact Factor*, vol. 8, 2024, doi: 10.17148/IJARCCE.2024.134210.
- [3] Y. Azhar, A. Khoiriyah Firdausy, and P. J. Amelia, “SINTECH Journal | 191 Perbandingan Algoritma Klasifikasi Data Mining Untuk Prediksi Penyakit Stroke”, [Online]. Available: <https://doi.org/10.31598>

## 11. LAMPIRAN

<https://colab.research.google.com/drive/1jT9hp9UbmhPHND96qP65plnSvmqYwsXu?usp=sharing>