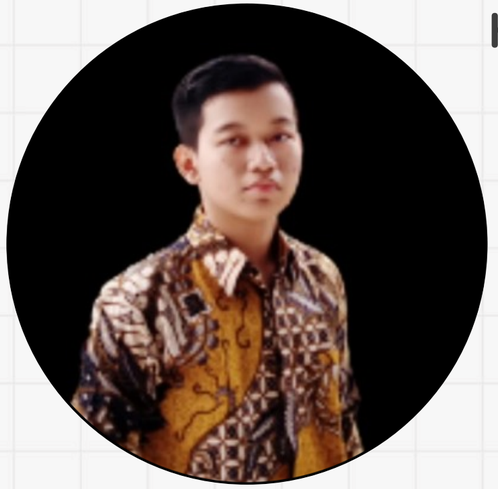
ANALYSIS OF DENGUE

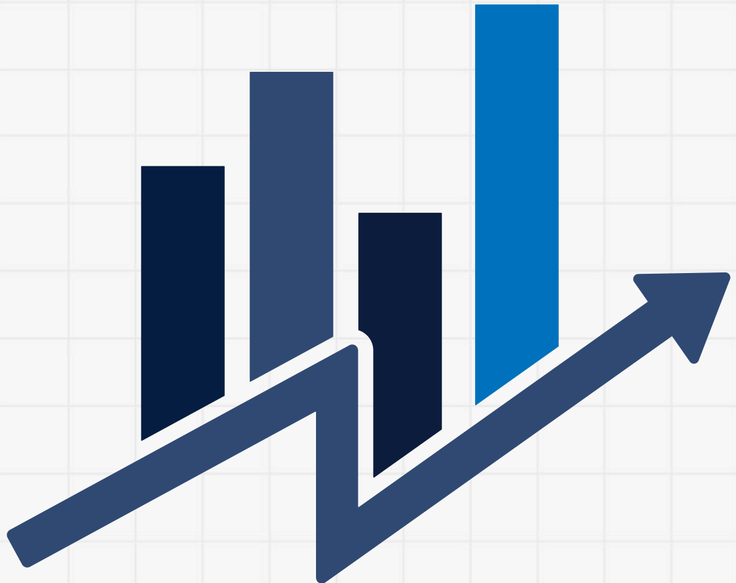
By: Raka arrayan

TABLE OF CONTENT

ABOUT THE WRITER

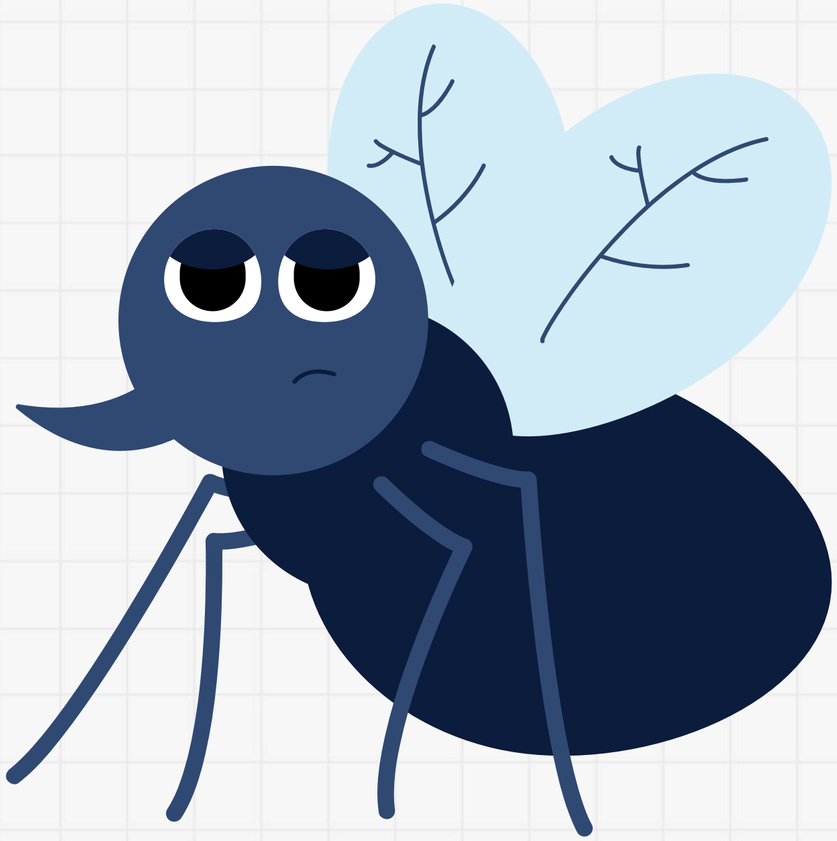


Hi! I am a thrid-semester student at University Indonesia. I am a fast learner with a strong interest in data analytics. My organizational experience has equipped me with excellent teamwork and communication skills. I enjoy being involved in various projects in the IT sector, and I am constantly looking for opportunities to learn and enhance my abilities through professional organizations.



BACKGROUND

Dengue fever is a vector-borne disease transmitted to humans through the Aedes mosquito. Bangladesh, particularly the Dhaka region, has experienced periodic Dengue outbreaks, making it a critical area for research, monitoring, and public health interventions.



PROBLEM AND GOALS

VS

PROBLEM

Dengue hemorrhagic fever (DHF) not only causes significant morbidity and mortality but also imposes a high cost burden on the healthcare system. Early detection and timely intervention are crucial to reduce disease severity and mortality rates. However, early diagnosis of DHF is often challenging due to initial symptoms that resemble those of other febrile illnesses. Therefore, an effective predictive tool is needed to aid in early diagnosis.

GOALS

This data processing aims to develop
prediction models that can help in early detection
dengue fever infections based on clinical and demographic data.
By using machine learning techniques,
This model is expected to:
1 Predict dengue fever infections with high accuracy.
2. Identify the main factors that cause a person to become infected with dengue fever.
3. Increase the efficiency of early detection of dengue cases.
4. create a prediction model that can determine whether someone suffers from dengue fever (Outcome = 1) or not (Outcome = 0) based on the data
5.Know the best Machine Learning Algorithms for data modeling and classification

TOOLS

Google colab is used as platform for analyzing data

Google Sheets is used to make data easier preprocessing is like removing duplicates

pyhton is used data for data analysis processes

ANALYSIS PROCESS

1. Data Preparation
2. Exploratory Data Analysis
3. Preprocessing Data
4. Processing Data
5. Modeling Data
6. Evaluasi

DATA PREPARATION

Data collected from The Devastator dataset on Kaggle
This dataset presents real-world data collected through surveys conducted in the Dhaka region of Bangladesh. It focuses on understanding the prevalence and characteristics of the Dengue fever phenomenon, a significant public health concern in the area. The dataset is updated monthly to reflect the evolving nature of the Dengue outbreak.

Rows

1000

Columns

10

DATA PREPARATION

Import library dan data

```
from google.colab import files
upload=files.upload()
```

Reading dataset structure

Displays concise information about the DataFrame, including the number of rows and columns, column names, number of non-null values in each column, data type for each column, and memory usage

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Gender     1000 non-null   object
 1   Age        1000 non-null   int64 
 2   NS1        1000 non-null   int64 
 3   IgG        1000 non-null   int64 
 4   IgM        1000 non-null   int64 
 5   Area       1000 non-null   object
 6   AreaType   1000 non-null   object
 7   HouseType  1000 non-null   object
 8   District   1000 non-null   object
 9   Outcome    1000 non-null   int64 
dtypes: int64(5), object(5)
memory usage: 78.2+ KB
```

Read the entire contents of the dataset

```
import pandas as pd

df = pd.read_csv('Dataset.csv')
df
```

	Gender	Age	NS1	IgG	IgM	Area	AreaType	HouseType	District	Outcome
0	Female	45	0	0	0	Mirpur	Undeveloped	Building	Dhaka	0
1	Male	17	0	0	1	Chawkbazar	Developed	Building	Dhaka	0
2	Female	29	0	0	0	Paltan	Undeveloped	Other	Dhaka	0
3	Female	63	1	1	0	Motijheel	Developed	Other	Dhaka	1
4	Male	22	0	0	0	Gendaria	Undeveloped	Building	Dhaka	0
...	...	...	...	...	...	...	...	...	...	...
995	Female	16	1	1	0	New Market	Developed	Building	Dhaka	1
996	Male	41	1	1	0	Paltan	Undeveloped	Other	Dhaka	1
997	Male	45	0	0	1	Motijheel	Developed	Building	Dhaka	0
998	Female	19	1	1	1	Paltan	Undeveloped	Building	Dhaka	1
999	Female	28	0	0	1	Adabor	Developed	Building	Dhaka	0

1000 rows × 10 columns

EXPLORATORY DATA ANALYSIS

Data Description

```
df.describe()
```

	Age	NS1	IgG	IgM	Outcome
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	36.313000	0.521000	0.537000	0.475000	0.533000
std	17.684393	0.501808	0.510776	0.499624	0.499159
min	8.000000	0.000000	0.000000	0.000000	0.000000
25%	22.000000	0.000000	0.000000	0.000000	0.000000
50%	37.000000	1.000000	1.000000	0.000000	1.000000
75%	50.000000	1.000000	1.000000	1.000000	1.000000
max	200.000000	2.000000	4.000000	1.000000	1.000000

Preprocessing Data

A broader concept that includes all stages preparation of data before or while using it for analysis or modeling, including data cleaning

Missing Value Handling

Counts the number of missing (null) values in each column in the DataFrame df

```
missing_data = df.isnull().sum()
print(missing_data)
```

```
Gender        0
Age           0
NS1           0
IgG           0
IgM           0
Area          0
AreaType      0
HouseType     0
District      0
Outcome       0
dtype: int64
```

LABEL ENCORDER

Perform label coding on the 'Gender' column in the df DataFrame

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['Gender'] = encoder.fit_transform(df['Gender'])
df
```

	Gender	Age	NS1	IgG	IgM	Area	AreaType	HouseType	District	Outcome
0	0	45	0	0	0	Mirpur	Undeveloped	Building	Dhaka	0
1	1	17	0	0	1	Chawkbazar	Developed	Building	Dhaka	0
2	0	29	0	0	0	Paltan	Undeveloped	Other	Dhaka	0
3	0	63	1	1	0	Motijheel	Developed	Other	Dhaka	1
4	1	22	0	0	0	Gendaria	Undeveloped	Building	Dhaka	0
...	...	...	...	...	...	...	...	...	...	...
995	0	16	1	1	0	New Market	Developed	Building	Dhaka	1
996	1	41	1	1	0	Paltan	Undeveloped	Other	Dhaka	1
997	1	45	0	0	1	Motijheel	Developed	Building	Dhaka	0
998	0	19	1	1	1	Paltan	Undeveloped	Building	Dhaka	1
999	0	28	0	0	1	Adabor	Developed	Building	Dhaka	0

1000 rows × 10 columns

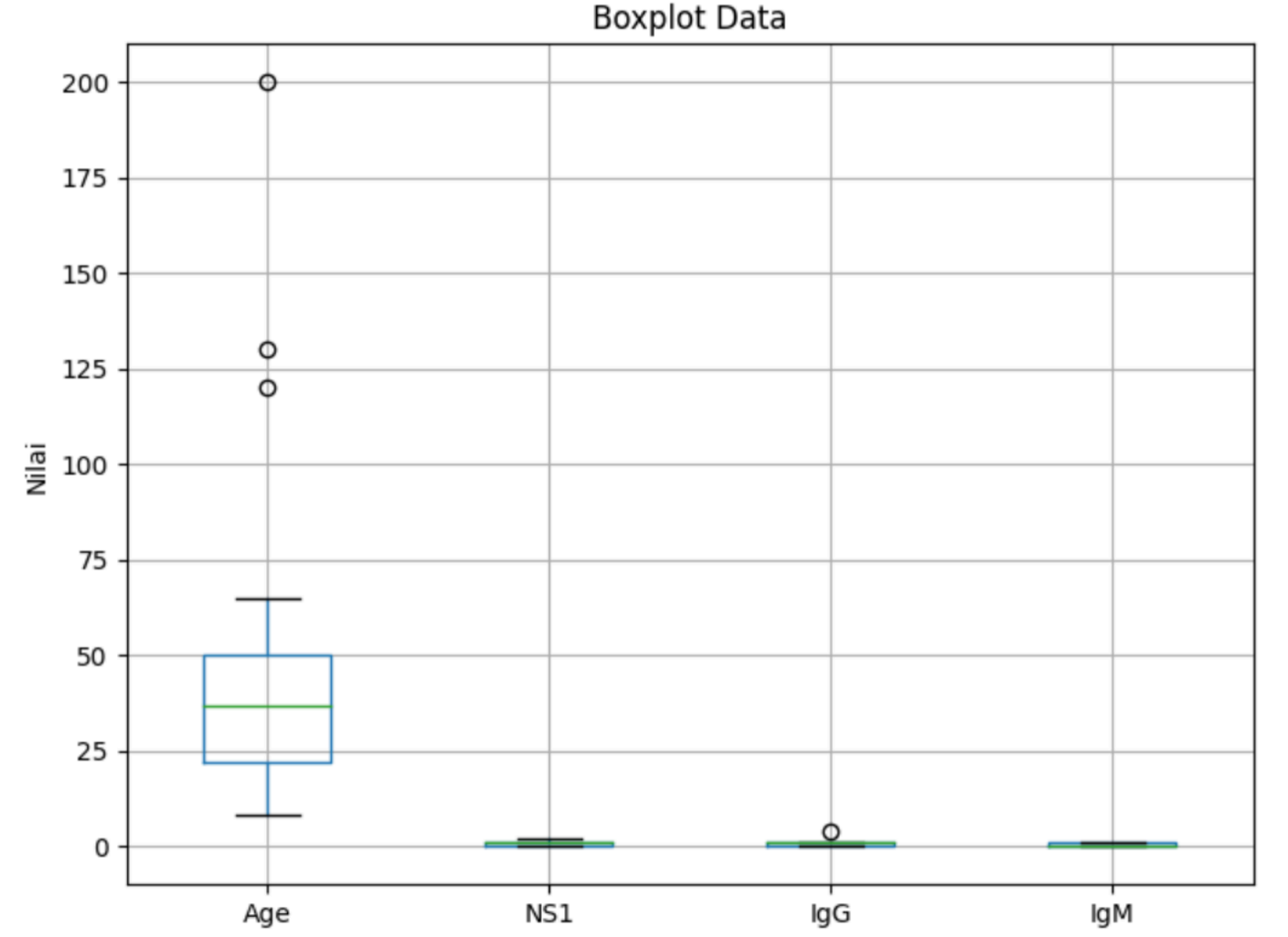
Data Outlier

visualization and outlier detection. provides a visual overview of the distribution of values and potential outliers in each column

```
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

plt.figure(figsize=(8, 6))
df[['Age', 'NS1', 'IgG', 'IgM']].boxplot()
plt.title('Boxplot Data')
plt.ylabel('Nilai')
plt.show()

# Menghitung Q1, Q3, dan IQR untuk kolom numerik saja
numeric_cols = ['Age', 'NS1', 'IgG', 'IgM']
Q1 = df[numeric_cols].quantile(0.25)
Q3 = df[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

# Menentukan outlier
outliers = ((df[numeric_cols] < (Q1 - 1.5 * IQR)) | (df[numeric_cols] > (Q3 + 1.5 * IQR))).sum()

print('Data Outlier:')
print(outliers)
```

Preprocessing Data

Data Outlier




```
Data Outlier:
Age     3
NS1     0
IgG     1
IgM     0
dtype: int64
```

Removing Outlier

Detect and remove outliers from a DataFrame
Rows containing outliers are identified and stored in the variable outliers.dataset_no_outliers is created by removing rows containing outliers from dataset_cleaned.
dataset_no_outliers contains only data without outliers and is ready to be used for further analysis

```
numeric_cols = ['Age', 'NS1', 'IgG', 'IgM']
Q1 = dataset_cleaned[numeric_cols].quantile(0.25)
Q3 = dataset_cleaned[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

#menghapus outlier
outliers = ((dataset_cleaned[numeric_cols] < (Q1 - 1.5 * IQR)) | (dataset_cleaned[numeric_cols] > (Q3 + 1.5 * IQR))).any(axis=1)
dataset_no_outliers = dataset_cleaned[~outliers]#ini untuk hapus

print("Dataframe Tanpa Outlier dan Data Kosong:")
print(dataset_no_outliers)
```

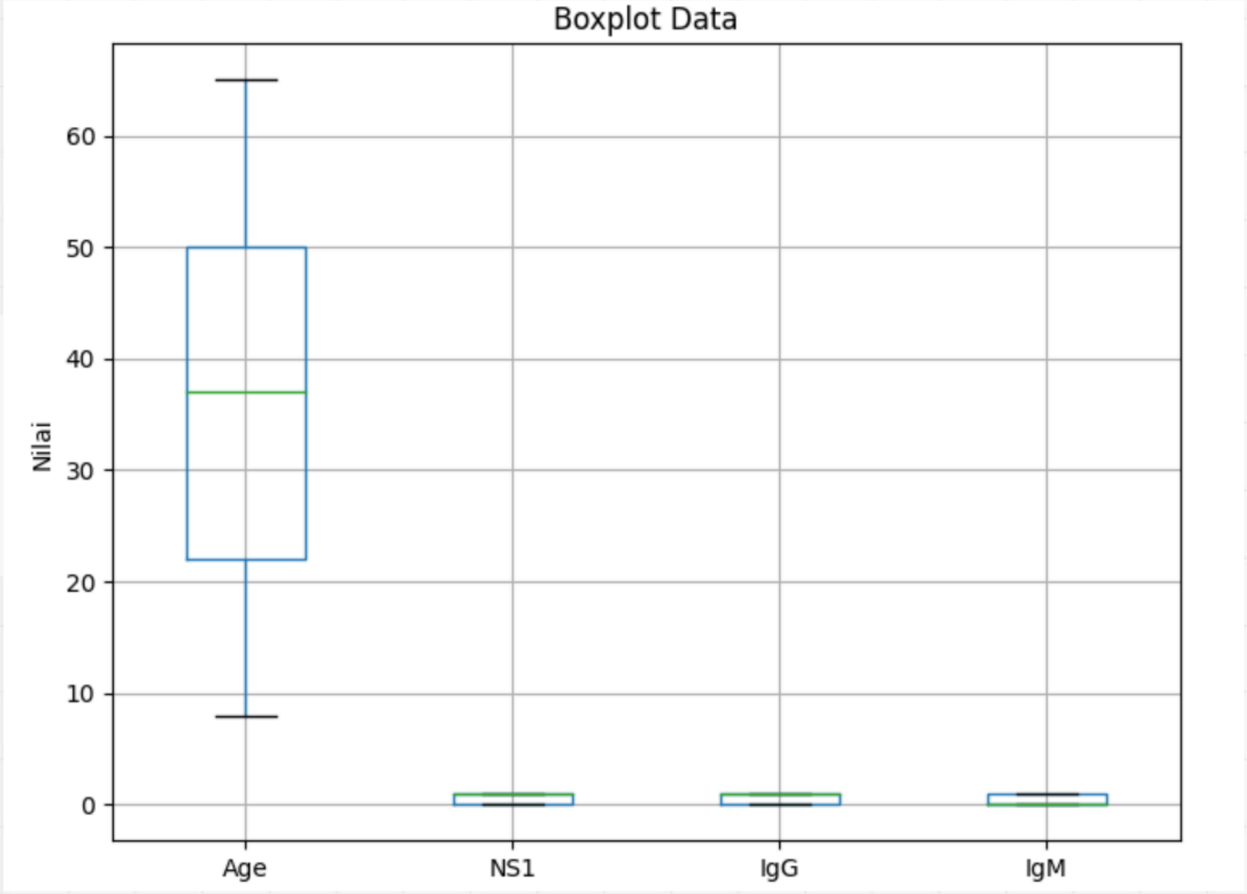
Removing Outlier

```
Dataframe Tanpa Outlier dan Data Kosong:
     Gender  Age  NS1  IgG  IgM        Area     AreaType HouseType District
0         0   45    0    0    0      Mirpur  Undeveloped  Building    Dhaka
1         1   17    0    0    1  Chawkbazar    Developed  Building    Dhaka
2         0   29    0    0    0      Paltan  Undeveloped     Other    Dhaka
3         0   63    1    1    0   Motijheel    Developed     Other    Dhaka
4         1   22    0    0    0    Gendaria  Undeveloped  Building    Dhaka
..      ...  ...  ...  ...  ...         ...          ...       ...      ...
995       0   16    1    1    0  New Market    Developed  Building    Dhaka
996       1   41    1    1    0      Paltan  Undeveloped     Other    Dhaka
997       1   45    0    0    1   Motijheel    Developed  Building    Dhaka
998       0   19    1    1    1      Paltan  Undeveloped  Building    Dhaka
999       0   28    0    0    1      Adabor    Developed  Building    Dhaka

     Outcome
0          0
1          0
2          0
3          1
4          0
..       ...
995        1
996        1
997        0
998        1
999        0

[996 rows x 10 columns]
```

Removing Outlier

Create and display a boxplot showing the distribution of data from the columns 'Age', 'NS1', 'IgG', and 'IgM' in a DataFrame dataset_no_outliers

```
plt.figure(figsize=(8, 6))
dataset_no_outliers[['Age', 'NS1', 'IgG', 'IgM']].boxplot()
plt.title('Boxplot Data')
plt.ylabel('Nilai')
plt.show()
```



Processing Data

Data Processing is a series of actions or processes is done to convert raw data into a form more useful, informative, and understandable.

```
# memisahkan atribut pada dataset dan menyimpannya pada sebuah variabel
X = df[df.columns[:9]]
#df.columns[:9] adalah cara untuk memilih kolom pertama hingga kolom kedelapan dari dataframe df.

# memisahkan label pada dataset dan menyimpannya pada sebuah variabel
y = df['Outcome']
#df['Outcome'] mengakses kolom bernama Outcome dalam dataframe df
```

```
from sklearn.model_selection import train_test_split

# memisahkan data untuk training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

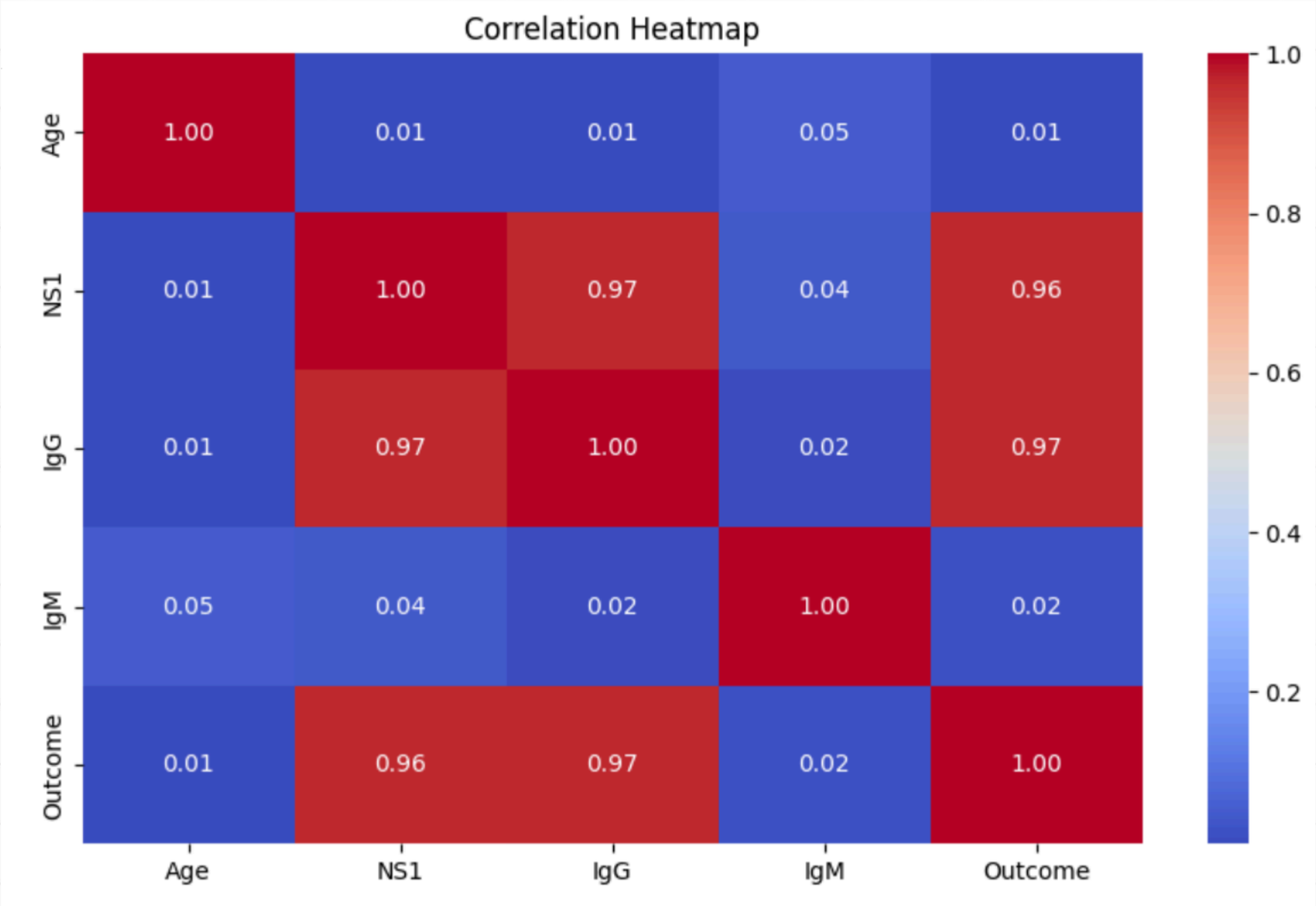
Check the correlation of each feature

analyze and visualize correlation relationships between numeric columns in a DataFrame df

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Pilih hanya kolom-kolom numerik
numeric_columns = ['Age', 'NS1', 'IgG', 'IgM', 'Outcome']
numeric_df = df[numeric_columns]

# Hitung matriks korelasi
correlation_matrix = numeric_df.corr()

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", xticklabels=numeric_df.columns, yticklabels=numeric_df.columns)
plt.title("Correlation Heatmap")
plt.show()
```

Processing Data

Check the correlation of each feature



K-Means Modeling

K-Means is choosing a random samples to be used as centroids. Centroid is a sample of the data which is the center of a cluster.

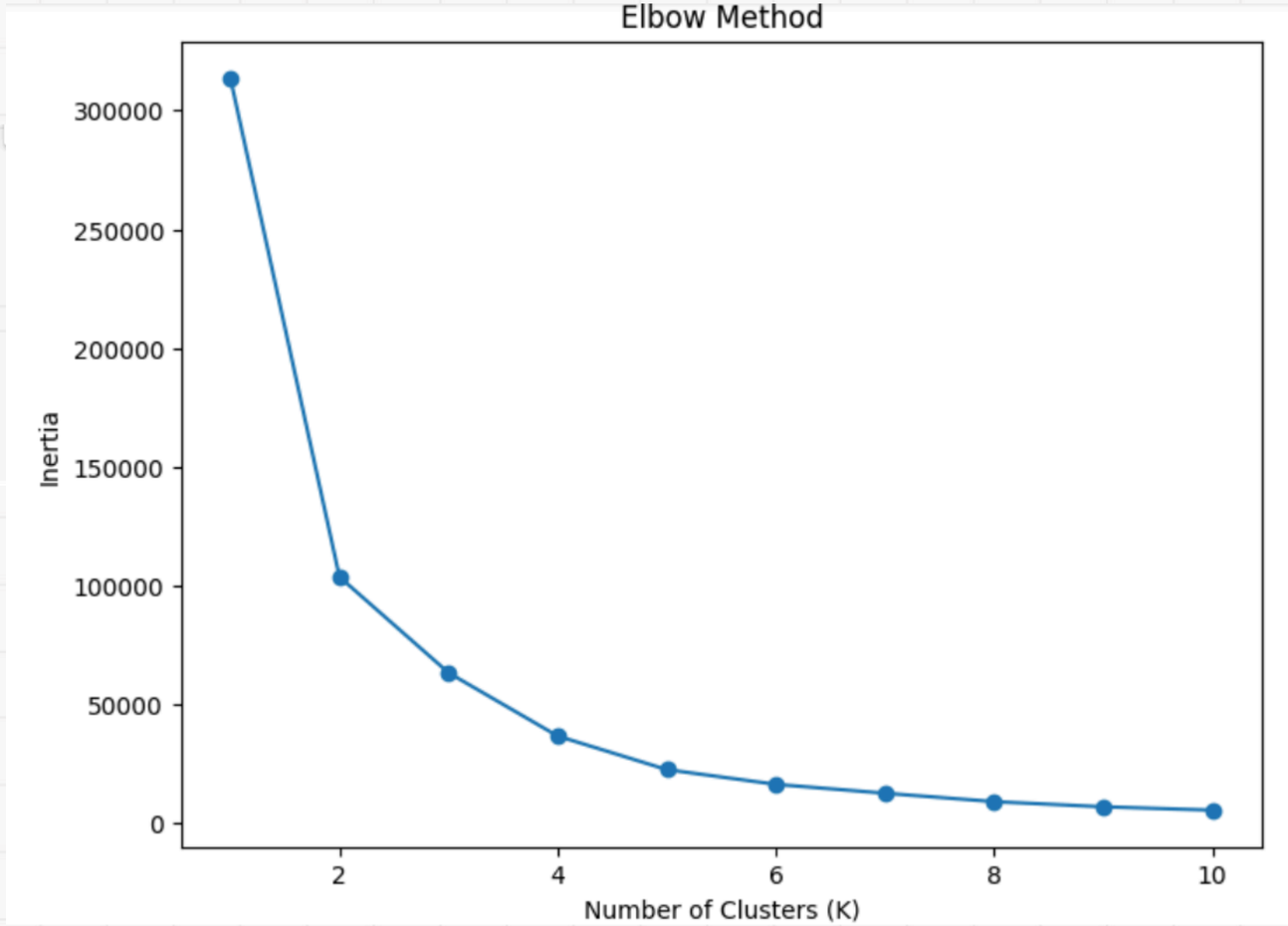
Determining the best k using the elbow technique

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Pra-pemrosesan data misalnya 'df' memiliki kolom-kolom kategorikal seperti 'Gender'

# Contoh menggunakan LabelEncoder untuk kolom 'Gender'
label_encoder = LabelEncoder()
df['Gender_encoded'] = label_encoder.fit_transform(df['Gender'])  # Mengubah 'Gender' menjadi numerik

# Sekarang hanya pilih kolom numerik untuk dilatih dengan KMeans
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()

inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df[numeric_cols])  # Melatih model KMeans pada kolom numerik dari df
    inertia.append(kmeans.inertia_)  # Menambahkan nilai inertia ke dalam list inertia
```

Processing Data

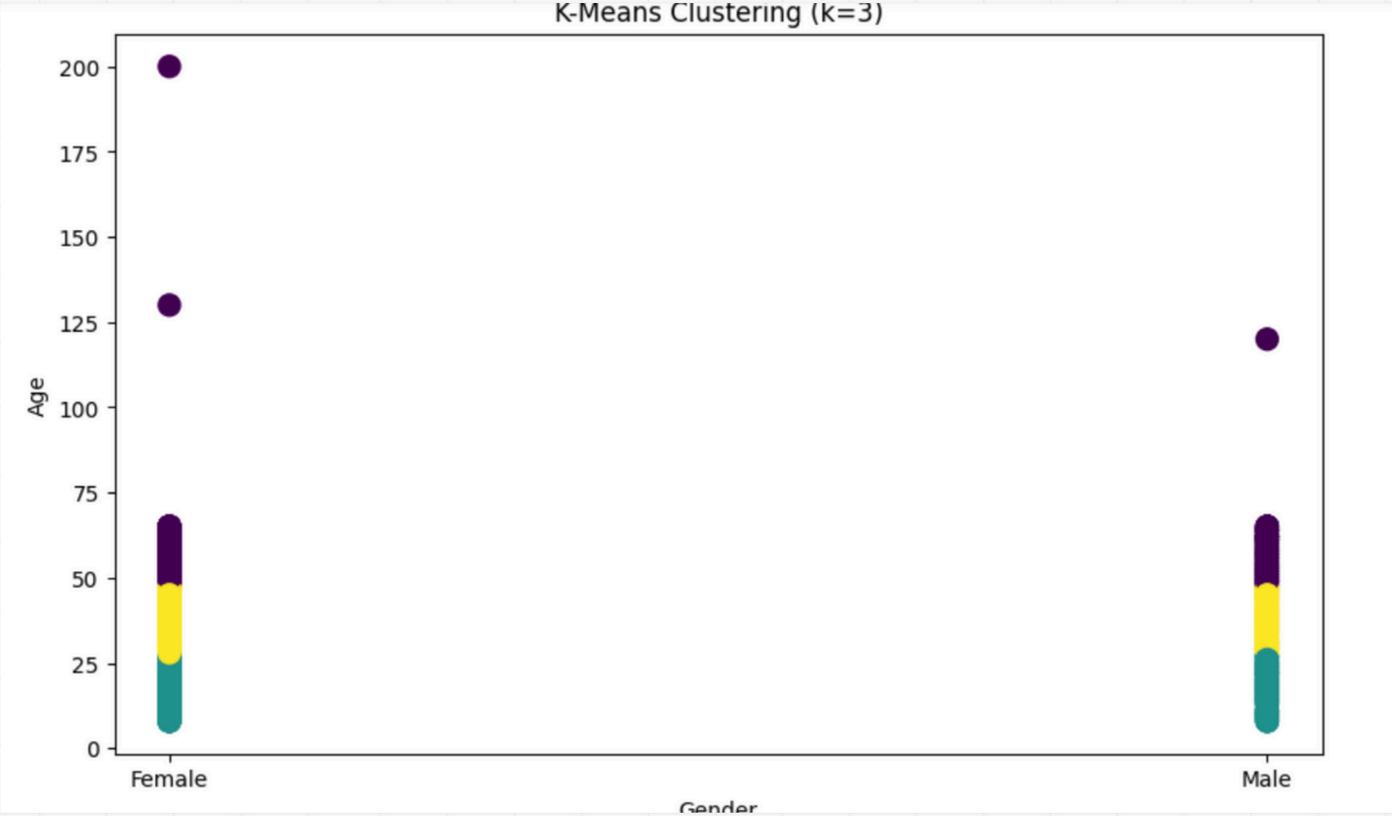
K-Means Modeling

```
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
#plt.plot yang digunakan untuk membuat plot garis.
#range(1, 11) nilai k berada dari 1 hingga 10
#inertia: Ini adalah list yang berisi nilai inertia yang dihitung untuk setiap nilai k dari 1 hingga 10.
#marker='o': Argumen ini digunakan untuk menentukan jenis marker yang digunakan pada titik-titik data dalam plot.
#Di sini, 'o' menunjukkan bahwa titik-titik akan ditandai dengan lingkaran (bulatan).

plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```



K-Means Modeling

```
plt.figure(figsize=(10, 6))
plt.scatter(df['Gender'], df['Age'], c=clusters, cmap='viridis', s=100)

plt.xlabel('Gender')
plt.ylabel('Age')
plt.title('K-Means Clustering (k=3)')
plt.show()
```



Modeling Data

- Machine Learning Models are commonly used computer programs to recognize patterns in data or make predictions.
- This machine learning model is created from a machine learning algorithm, which trained using labeled, unlabeled, or labeled data using mixed data.
- Machine learning is based on algorithms with data labeled (Supervised Learning), unlabeled (Unsupervised Learning), or using mixed data (Reinforcement Learning).
- Model training is the process of running a machine algorithm learning to be able to process the dataset in it which has been divided into training data and optimize the algorithm to find certain patterns or outputs.
- This function will produce a rule and also a data structure called a trained machine learning model.

Modeling Data

SVM(Support Vector Machine)

used to complete classification, regression, and outlier detection.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Encode kolom kategorikal menjadi numerik
encoder = LabelEncoder()
df['Gender_encoded'] = encoder.fit_transform(df['Gender'])
df['Area_encoded'] = encoder.fit_transform(df['Area'])
df['AreaType_encoded'] = encoder.fit_transform(df['AreaType'])
df['HouseType_encoded'] = encoder.fit_transform(df['HouseType'])
df['District_encoded'] = encoder.fit_transform(df['District'])

# Pisahkan fitur (X) dan label (y)
X = df[['Gender_encoded', 'Age', 'NS1', 'IgG', 'IgM', 'Area_encoded', 'AreaType_encoded', 'HouseType_encoded', 'District_encoded']]
y = df['Outcome']

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model SVM
clf = SVC()

# Latih model SVM dengan data latih
clf.fit(X_train, y_train)
```

```python
# Latih model SVM dengan data latih
clf.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = clf.predict(X_test)

# Hitung akurasi model
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi Model SVM:", accuracy)
```

```
Akurasi Model SVM: 0.605
```

Modeling Data

Decision Tree

Decision trees are powerful algorithms, meaning they can be used in complex problems.

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

# Lakukan prediksi pada data pengujian
y_pred_dtc = dtc.predict(X_test)

# Hitung akurasi model
accuracy = accuracy_score(y_test, y_pred_dtc)

print("Akurasi Model:", accuracy)
```

```
Akurasi Model: 1.0
```

Logistic Regression

One method commonly used for classification. In the case of classification, logistic regression works by calculating the class probability of a sample.

```
from sklearn import linear_model

logistic = linear_model.LogisticRegression()
logistic.fit(X_train, y_train)

# Lakukan prediksi pada data pengujian
y_pred_logistic = logistic.predict(X_test)

# Hitung akurasi model
accuracy = accuracy_score(y_test, y_pred_logistic)

print("Akurasi Model:", accuracy)
```

```
Akurasi Model: 1.0
```

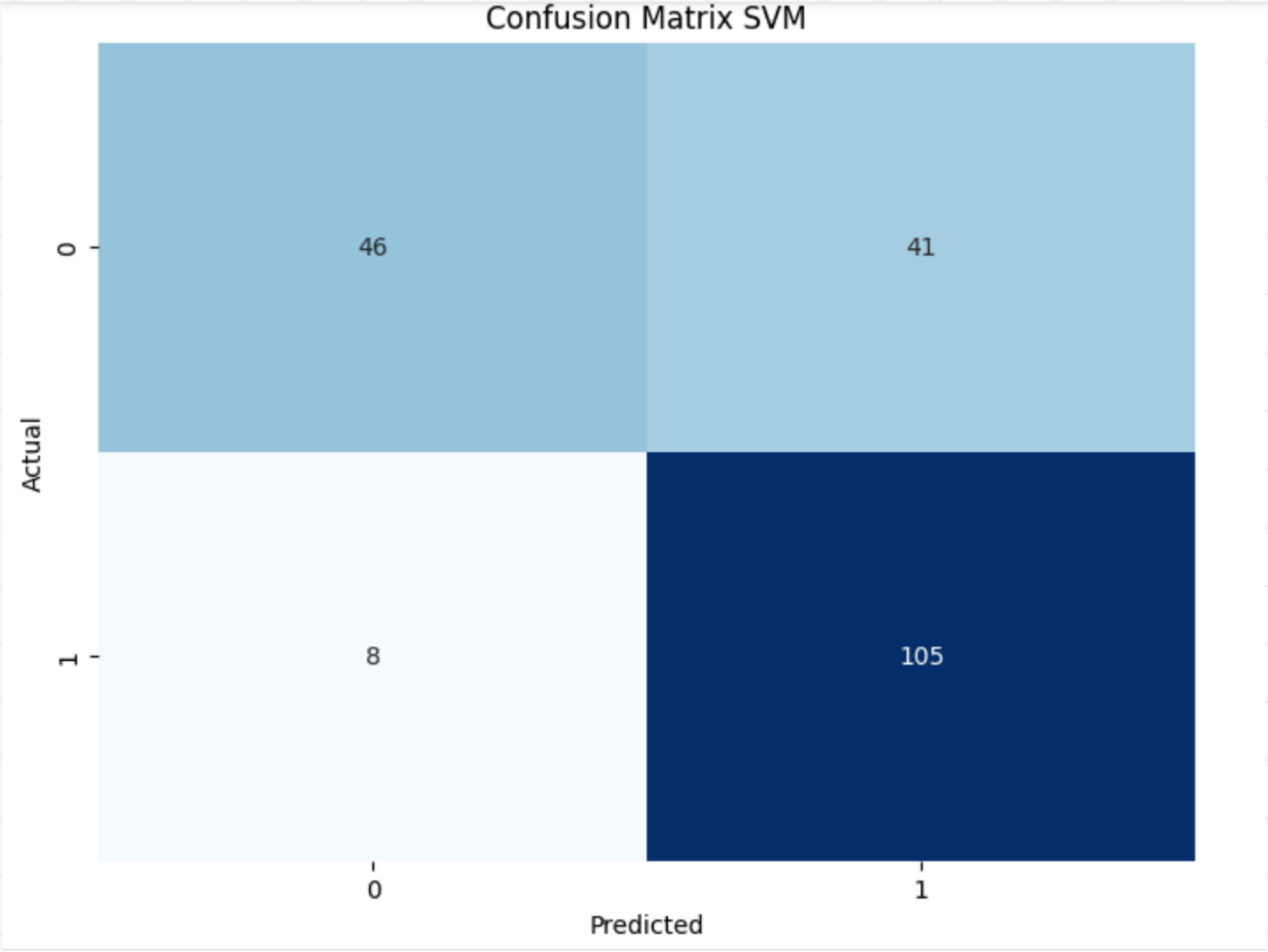
Evaluation

- In the process of developing a machine learning model, no all the resulting models can be used immediately. Data Scientists need to carry out testing and evaluation of that model.
- Testing and evaluating models is an important task for a Data Scientist to ensure model quality and reliability which was developed. With this task in mind, Data Scientists can confirm decisions and insightstaken from the data can become a solid basis inbusiness decision making or in developing effective solution

Evaluation

Confusion Matrix SVM

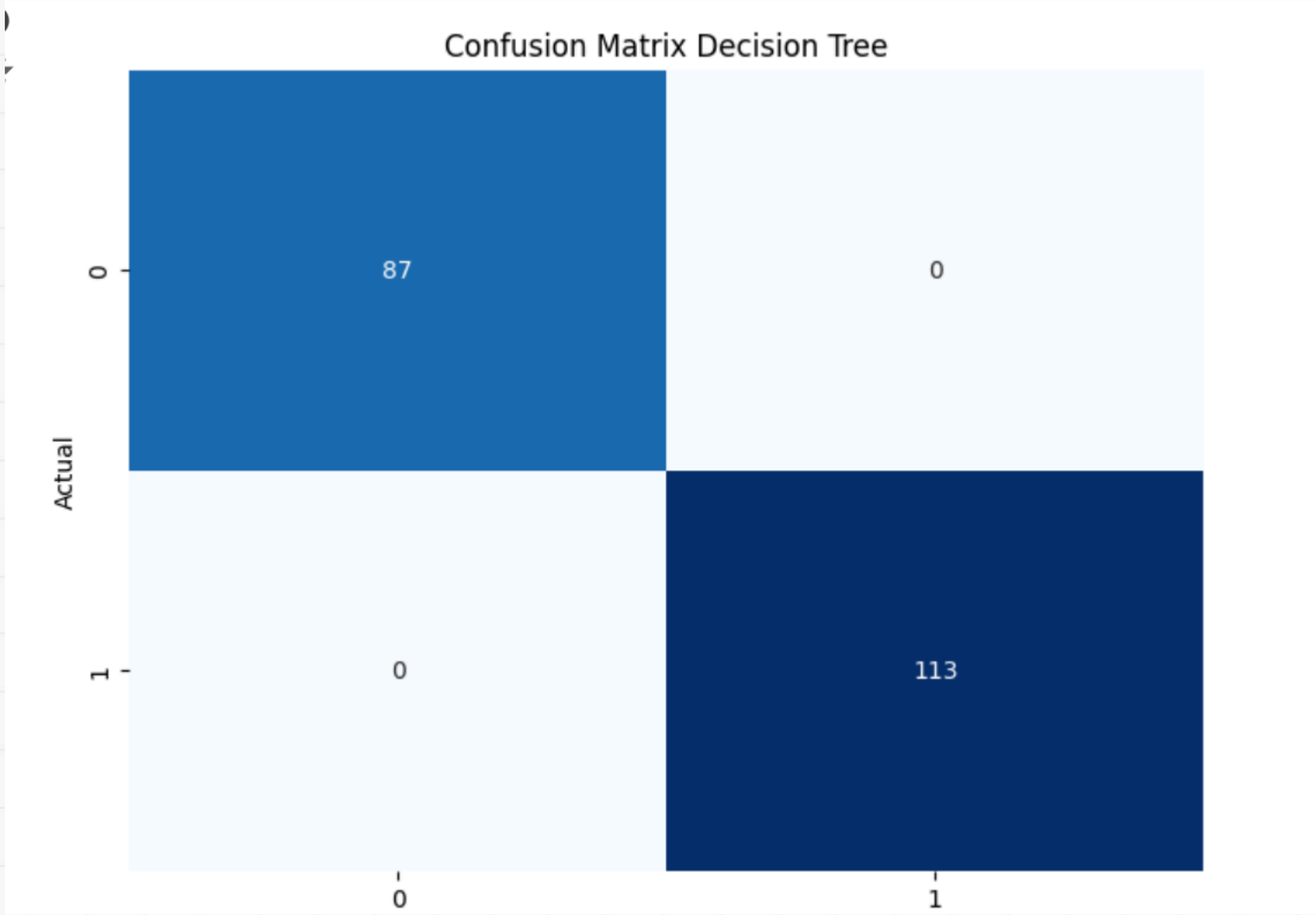
- A total of 46 data were proven correct in the prediction for label 0
- A total of 105 data were proven correct in predictions for label 1



Evaluation

Confusion Matrix Decision Tree

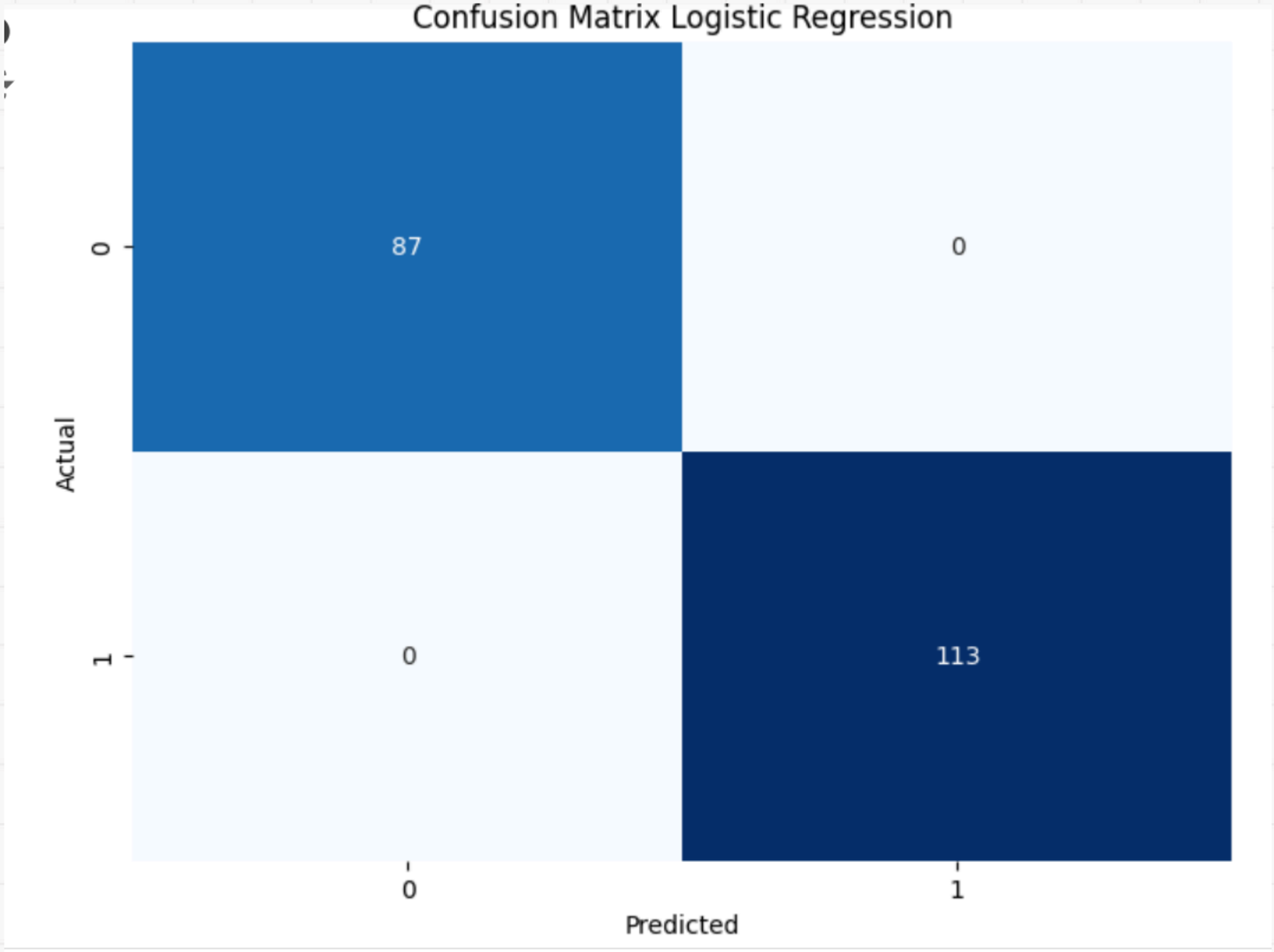
- A total of 87 data were proven correct in the prediction for label 0
- A total of 113 data were proven correct in predictions for label 1



Evaluation

Confusion Matrix Logistic Regression

- A total of 87 data were proven correct in the prediction for label 0
- A total of 113 data were proven correct in predictions for label 1



Evaluation

Predicting Dengue Fever Outcome Using Logistic Regression

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder

# Step 1: Memuat dataset dan pra-pemrosesan data
df = pd.read_csv('Dataset.csv')

# Menghapus kolom yang tidak diperlukan untuk prediksi
df = df.drop('District', axis=1)

# Memisahkan fitur (X) dan target (y)
X = df.drop('Outcome', axis=1)  # Fitur-fitur untuk melakukan prediksi
y = df['Outcome']  # Target atau label yang ingin diprediksi (0 atau 1)

# Menggunakan LabelEncoder untuk mengkodekan variabel kategorikal
label_encoders = {}
for column in ['Gender', 'Area', 'AreaType', 'HouseType']:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = le

# Step 2: Pelatihan model
model = LogisticRegression()
model.fit(X, y)
```

```python
# Step 3: Fungsi prediksi berdasarkan input
def predict_outcome(input_data):
    input_df = pd.DataFrame([input_data])

    # Menggunakan label encoder yang sama yang digunakan sebelumnya
    for col, le in label_encoders.items():
        input_df[col] = le.transform(input_df[col])

    # Prediksi Outcome
    prediction = model.predict(input_df)

    return prediction[0]

# Contoh penggunaan fungsi predict_outcome
input_data = {
    'Gender': 'Female',
    'Age': 30,
    'NS1': 1,
    'IgG': 1,
    'IgM': 0,
    'Area': 'Mirpur',
    'AreaType': 'Undeveloped',
    'HouseType': 'Building'
}

predicted_outcome = predict_outcome(input_data)
print(f"Predicted Outcome: {predicted_outcome}")
```

```
Predicted Outcome: 1
```

Conclusion

By using heatmap correlation analysis, it can be concluded that the most significant factors in determining whether someone has dengue fever are NS1 and IgG, with each having a very high correlation with the outcome, namely 0.96 and 0.97. This shows that NS1 and IgG test results can be strong predictors of this condition.

From the evaluation results of the three machine learning models used, namely SVM, Decision Tree, and Logistic Regression, it is known that Decision Tree and Logistic Regression achieved the highest accuracy, reaching 1.0. This indicates that the two models are very suitable for modeling and predicting data related to dengue fever.

Evaluation results using the confusion matrix from decision trees and logistic regression show that from 200 data evaluated from 33% of the data tested, the model succeeded in correctly predicting 87 cases labeled 0 (not infected) and 113 cases labeled 1 (infected). Using SVM succeeded in correctly predicting 46 cases labeled 0 (not infected) and 105 cases labeled 1 (infected). This shows that the decision tree and logistic regretion models have good performance.

By applying machine learning techniques such as logistic regression, we can utilize information such as gender, age, NS1, IgG, IgM test results, as well as location and type of residence to predict the possibility of someone being infected with dengue fever. This model produces output in the form of predicted outcomes which can be 1 (infected with dengue fever) or 0 (not infected).

Thus, using machine learning approaches and appropriate evaluations like these, we can better understand and predict the risk of contracting dengue fever based on significant factors from available health data

VIEW IN COLAB

THANK YOU

FOR YOUR ATTENTION

rakaarrayan27@gmail.com