DIABETES

CLASSIFICATION MODEL:
PREDICT WHETHER A PATIENT HAS DIABETES
BASED ON DIAGNOSTIC MEASUREMENTS

TUGAS AKHIR SAINS DATA - KELOMPOK 3

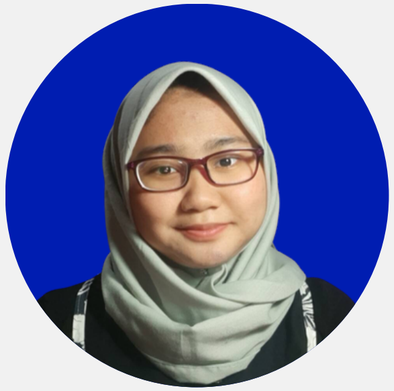
OUR TEAM

Aldiranta

2106722676

Cendikia Sigra Dewi

2106635322

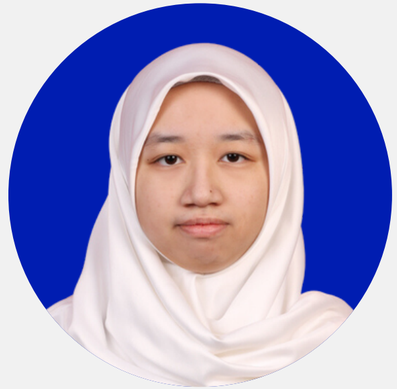


Stephen Wirana

2106722745

Zalfa Aretha Fahira Ranstya

2106707536



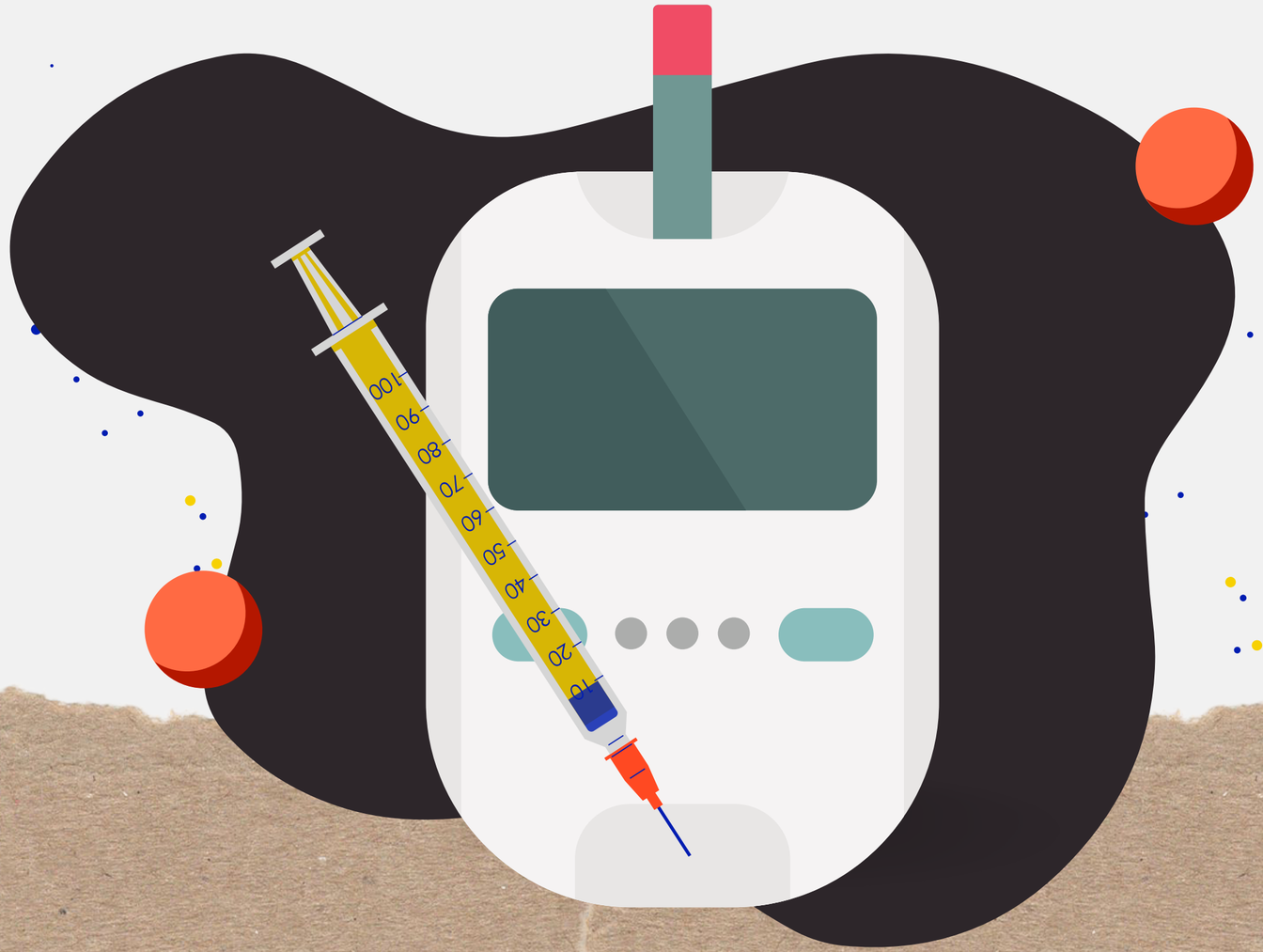
CONTENTS

MASALAH

1. MASALAH

AKAN DIBUAT BEBERAPA MODEL UNTUK MELAKUKAN PROSES KLASIFIKASI UNTUK MELIHAT APAKAH PASIEN TERKENA PENYAKIT DIABETES ATAU TIDAK BERDASARKAN RIWAYAT KESEHATANNYA

METODE DAN DATA



2. METODE DAN DATA

DATASET: Riwayat kesehatan pasien wanita berusia di atas 21 tahun milik National Institute of Diabetes and Digestive and Kidney Diseases

FITUR PADA DATASET:

- **Pregnancies:** Frekuensi kehamilan
- **Glucose:** Konsentrasi Plasma Glukosa dua jam melalui Oral Glucose Tolerance Test
- **BloodPressure:** Tekanan darah diastolik (mm Hg)
- **SkinThickness:** Ketebalan fold kulit triceps (mm)
- **Insulin:** 2-Jam serum insulin (mu U/ml)
- **BMI:** Body Mass Index (berat badan dalam kg/(tinggi badan dalam m)^2)
- **DiabetesPedigreeFunction:** Diabetes pedigree function
- **Age:** Umur (tahun)
- **Outcome:** Variabel Kelas (0 atau 1)

NIH National Institute of Diabetes and Digestive and Kidney Diseases

2. METODE DAN DATA

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

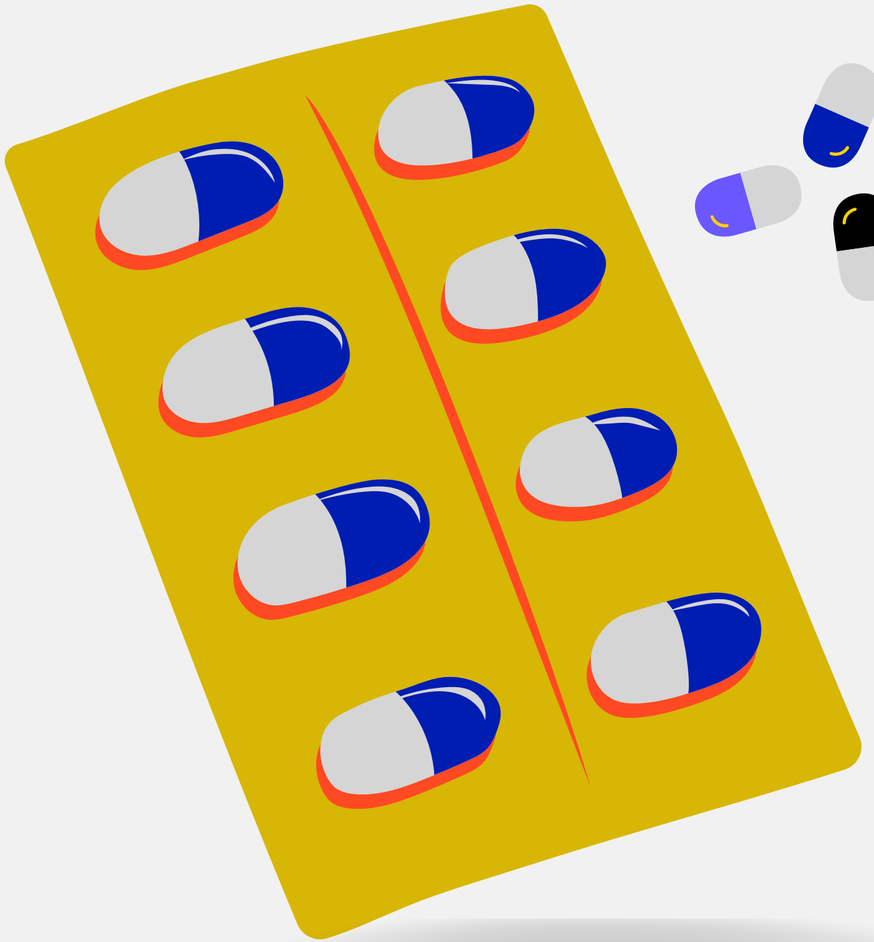
768 rows × 9 columns

2. METODE DAN DATA

Langkah langkah implementasi:

1. Data Observation
2. Handling Missing Values
3. Exploratory Data Analysis
4. Model Selection
5. Classification
6. Feature Important

- Logistic Regression
- Support Vector Machine
- Random Forest
- Deep Neural Network



IMPLEMENTASI

3. IMPLEMENTASI

I. Data Observation

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
!pip install keras-tuner
import tensorflow as tf
import keras_tuner as kt
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

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

```
df = pd.read_csv('diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.shape
```

```
(768, 9)
```

Dalam project ini, kami mempunyai data sebanyak 768 baris dengan 9 fitur

3. IMPLEMENTASI

I. Data Observation

```
1 df.isna().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
```

Dapat dilihat bahwa dataset tersebut tidak memiliki nilai NaN, tetapi terdapat ketidaklogisan data (misal: terdapat insulin yang bernilai 0), jadi harus diubah menjadi data yang kosong dahulu

```
1 df[df.loc[:,[ 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI']] == 0] = np.nan
```

3. IMPLEMENTASI

I. Data Observation

```
1 df.isna().sum()
```

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

Dapat dilihat ternyata terdapat data yang kosong pada fitur tertentu

3. IMPLEMENTASI

II. Data Cleaning

```
missing = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
for i in missing:
  if i == 'BMI':
    df[i] = df[i].fillna(df[i].mean())
  else:
    df[i] = df[i].fillna(math.floor(df[i].mean()))
```

```
df_org.isna().sum()
```

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

Dilakukan metode imputasi menggunakan mean untuk mengatasi missing value

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)

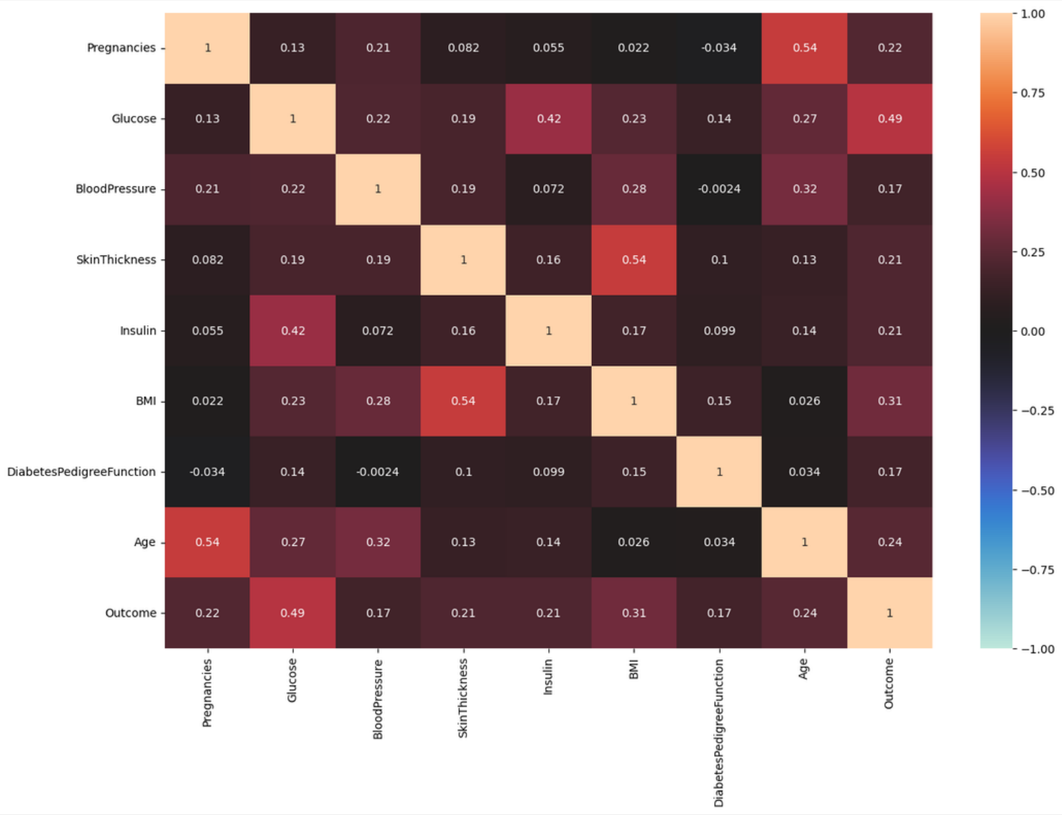
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.682292	72.386719	29.108073	155.28125	32.457464	0.471876	33.240885	0.348958
std	3.369578	30.435999	12.096642	8.791221	85.02155	6.875151	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.00000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	25.000000	121.50000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	29.000000	155.00000	32.400000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	155.00000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.00000	67.100000	2.420000	81.000000	1.000000

Statistik Deskriptif dari dataset

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)

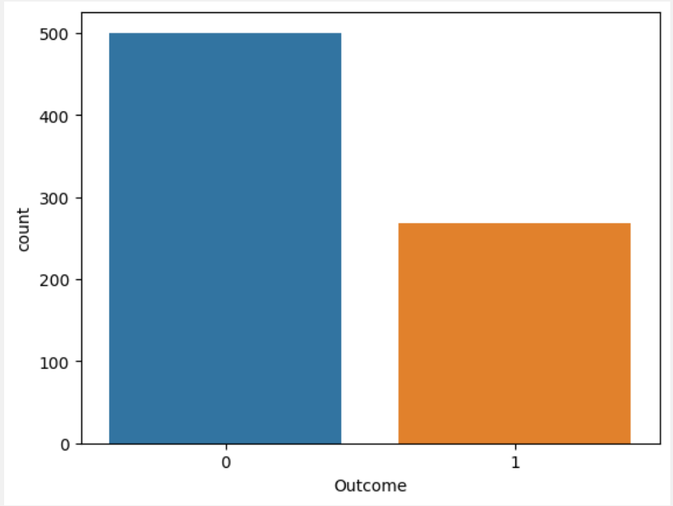
```
1 corr = df.corr()
2 plt.subplots(figsize=(15,10))
3 sns.heatmap(corr, vmin=-1, center=0, vmax=1, annot=True)
4 plt.show()
```



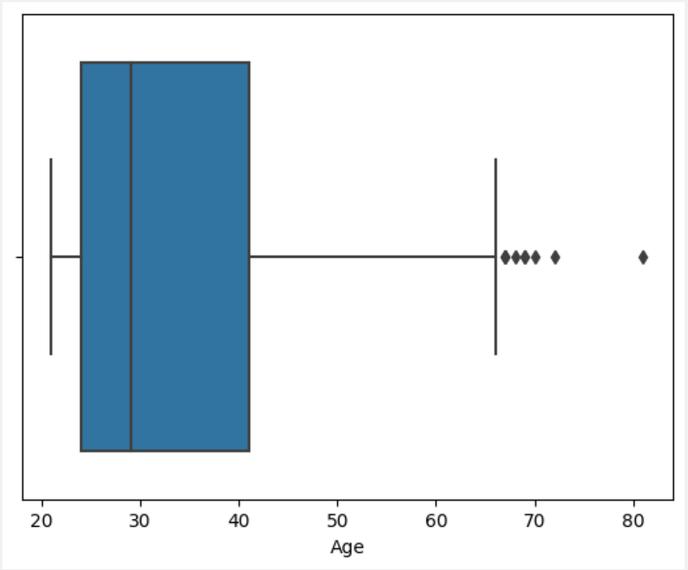
Korelasi antar Fitur

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)



Perhatikan pada fitur "Outcome" memiliki ketimpangan sehingga perlu dilakukan **Resampling**



Perhatikan fitur pada dataset, khususnya fitur "Age", memiliki outlier sehingga data outlier tersebut perlu dihilangkan

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)

```python
# Define a function to remove outliers based on the IQR method
def remove_outliers(df, features, threshold=1.5):
    filtered_df = df.copy()
    for feature in features:
        Q1 = df[feature].quantile(0.25)
        Q3 = df[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - threshold * IQR
        upper_bound = Q3 + threshold * IQR
        filtered_df = filtered_df[(filtered_df[feature] >= lower_bound) & (filtered_df[feature] <= upper_bound)]
    return filtered_df

# Specify the features/columns to remove outliers from
features = df.columns.drop(['Outcome','Insulin'])

# Remove outliers using the defined function
filtered_df = remove_outliers(df, features)

# Display the filtered DataFrame
filtered_df
```

Handling Outliers

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	5	116.0	74.0	29.0	155.0	25.6	0.201	30	0
...	...	...	...	...	...	...	...	...	...
622	9	89.0	62.0	29.0	155.0	22.5	0.142	33	0
623	2	122.0	70.0	27.0	155.0	36.8	0.340	27	0
624	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
625	1	126.0	60.0	29.0	155.0	30.1	0.349	47	1
626	1	93.0	70.0	31.0	155.0	30.4	0.315	23	0

627 rows × 9 columns

Handling Outliers

3. IMPLEMENTASI

III. Exploratory Data and Analysis (EDA)

```
y = df['Outcome']
X = df.drop('Outcome',axis=1).copy()
```

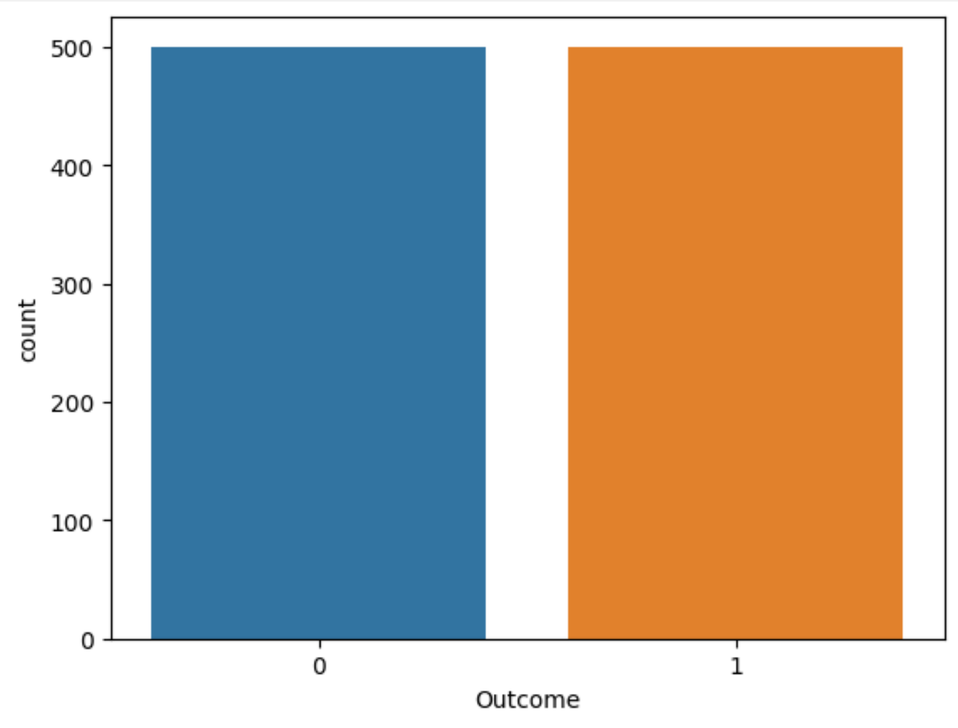
```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_resampled, y_resampled = ros.fit_resample(X, y)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	5	116.0	74.0	29.0	155.0	25.6	0.201	30	0
...	...	...	...	...	...	...	...	...	...
821	10	111.0	70.0	27.0	155.0	27.5	0.141	40	1
822	6	134.0	80.0	37.0	370.0	46.2	0.238	46	1
823	5	109.0	62.0	41.0	129.0	35.8	0.514	25	1
824	7	147.0	76.0	29.0	155.0	39.4	0.257	43	1
825	13	152.0	90.0	33.0	29.0	26.8	0.731	43	1

826 rows × 9 columns

Ukuran dataset bertambah menjadi 826 baris dan 9 kolom

Setelah dilakukan resampling dengan metode oversampling didapatkan bahwa nilainya sudah sama



3. IMPLEMENTASI

IV. Model Selection

```
y = df['Outcome']
X = df.drop('Outcome',axis=1).copy()
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20, random_state=42)
```

Dilakukan train-test split dengan perbandingan 80:20

3. IMPLEMENTASI

IV. Model Selection: Logistic Regression

```
lr = LogisticRegression()
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
lr.fit(X_train,y_train)
```

```
from sklearn.model_selection import (GridSearchCV, RandomizedSearchCV)

params = {'penalty':['l1','l2', 'elasticnet'],
          'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
           "C": [100, 10, 1.0, 0.1, 0.01]}

grid_search_model = GridSearchCV(lr, params, cv=kfold)
grid_search_model.fit(X_train, y_train)
```

LogisticRegression

LogisticRegression(C=10, solver='liblinear')

```
{'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
```

3. IMPLEMENTASI

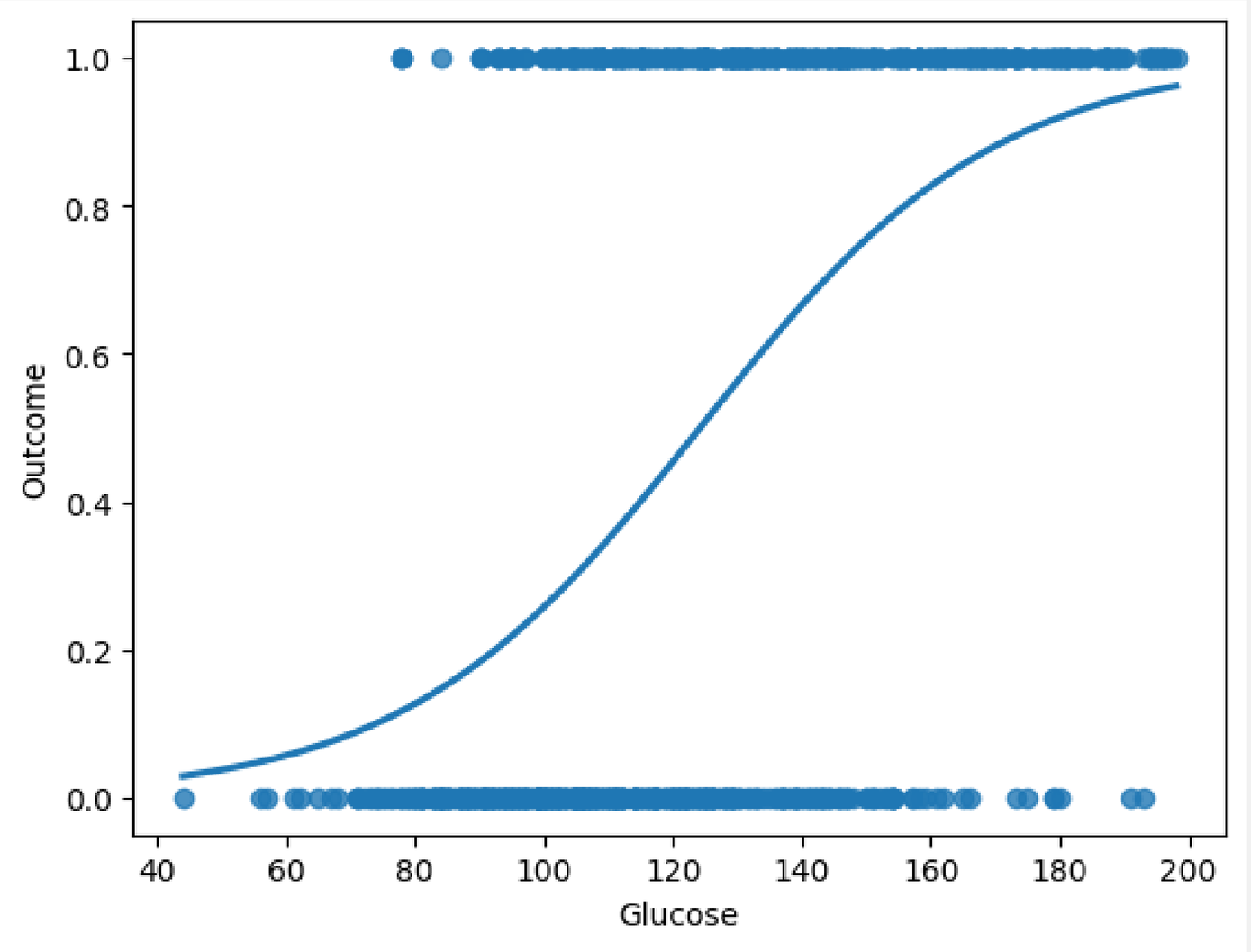
IV. Model Selection: Logistic Regression

Classification Report for Data Test

	precision	recall	f1-score	support
0	0.71	0.72	0.72	86
1	0.70	0.69	0.69	80
accuracy			0.70	166
macro avg	0.70	0.70	0.70	166
weighted avg	0.70	0.70	0.70	166

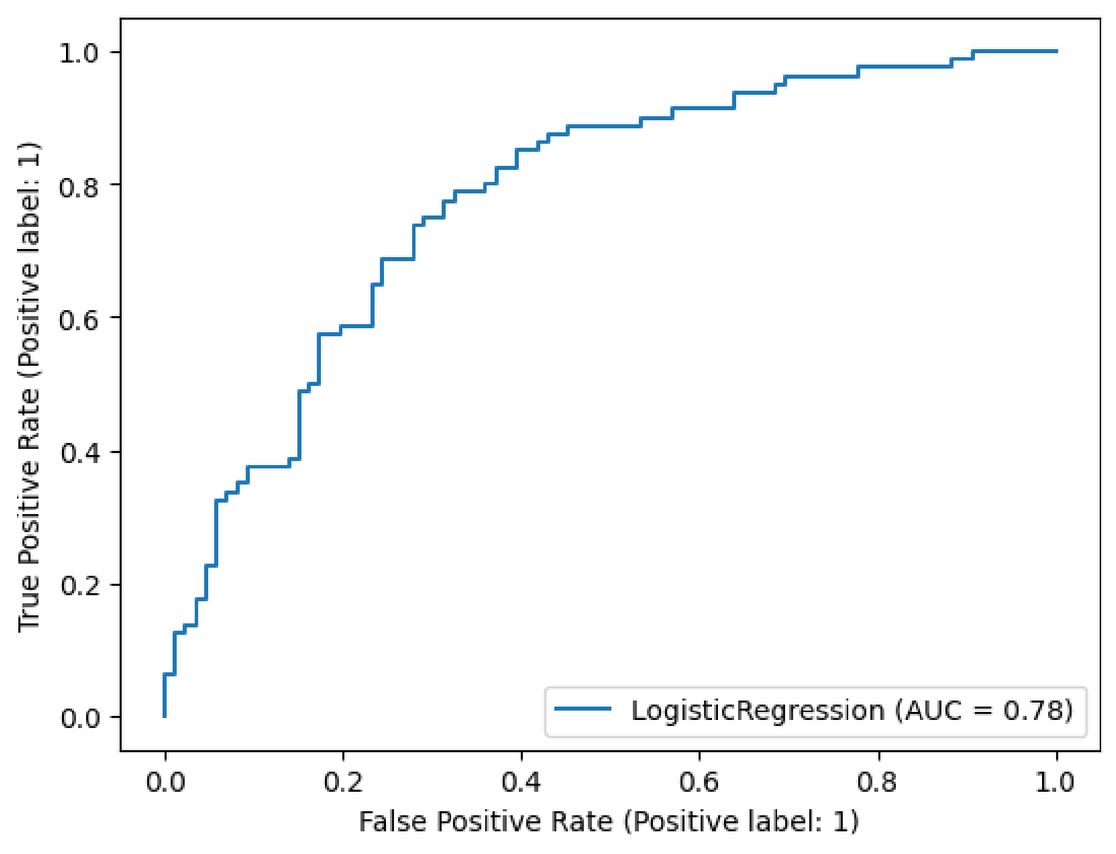
3. IMPLEMENTASI

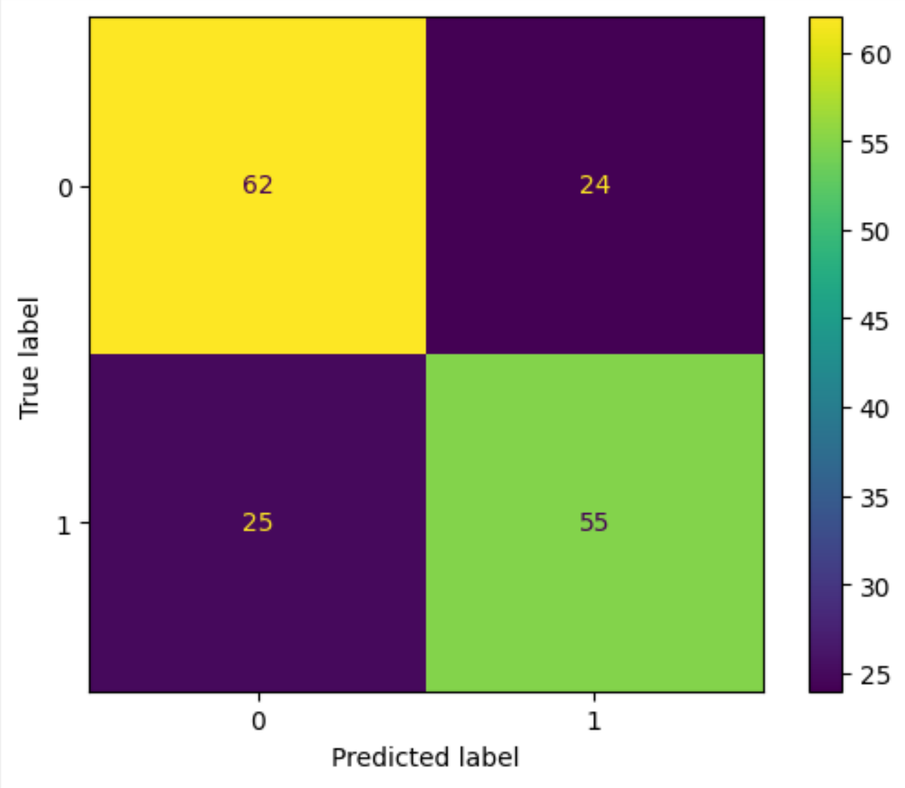
IV. Model Selection: Logistic Regression



3. IMPLEMENTASI

IV. Model Selection: Logistic Regression





3. IMPLEMENTASI

IV. Model Selection: Support Vector Machine

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
sc = StandardScaler()
cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X_train.loc[:,cols] = sc.fit_transform(X_train.loc[:,cols])
X_test.loc[:,cols] = sc.transform(X_test.loc[:,cols])
```

```
svc = SVC(probability=True)

params = {'C': [0.1, 1, 10, 100, 1000],
          'kernel':['linear', 'poly']}

grid_search = GridSearchCV(svc, params, cv=kfold, scoring='f1')
grid_search.fit(X_train,y_train)
```

SVC

```
SVC(C=1000, kernel='poly', probability=True)
```

3. IMPLEMENTASI

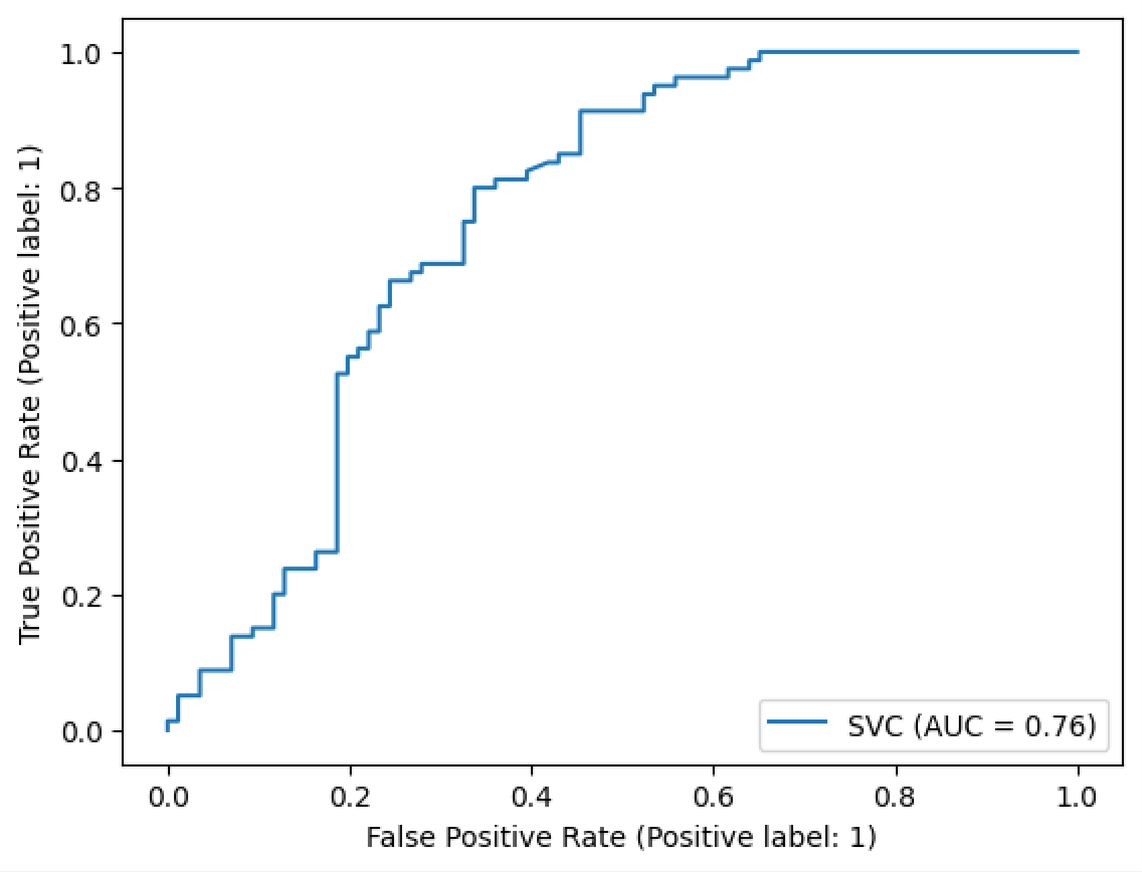
IV. Model Selection: Support Vector Machine

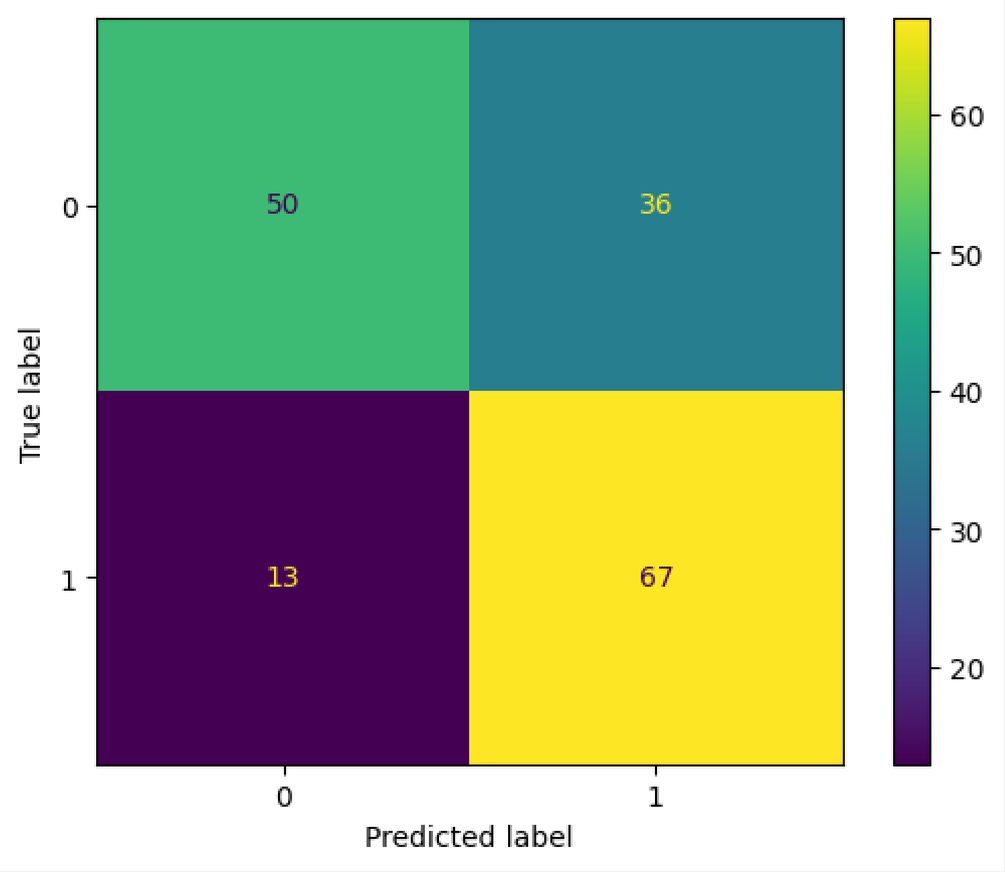
Classification Report for Data Test

	precision	recall	f1-score	support
0	0.79	0.58	0.67	86
1	0.65	0.84	0.73	80
accuracy			0.70	166
macro avg	0.72	0.71	0.70	166
weighted avg	0.72	0.70	0.70	166

3. IMPLEMENTASI

IV. Model Selection: Support Vector Machine





3. IMPLEMENTASI

IV. Model Selection: Random Forest

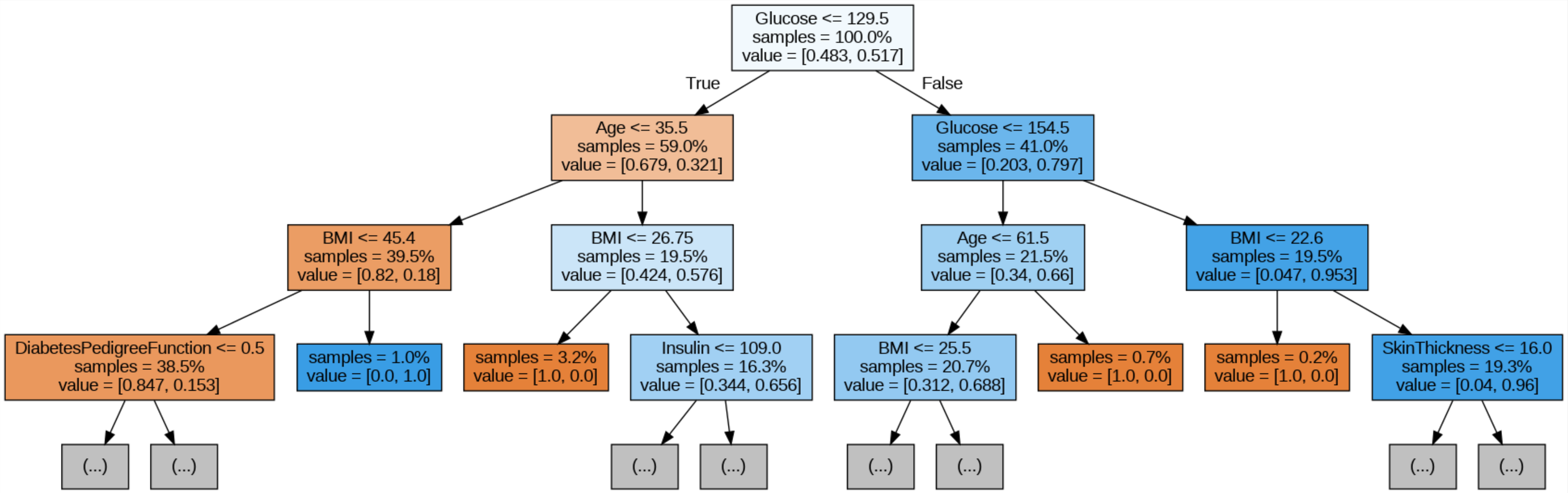
```
rf = RandomForestClassifier()
param_grid =  dict(n_estimators = [100,200,300,500], max_depth = [100,200],  max_features = [2,4,6,8] )
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           scoring='accuracy', cv=kfold, return_train_score=True, n_jobs=-1)

grid_search.fit(X_train, y_train)
```

RandomForestClassifier

RandomForestClassifier(max_depth=100, max_features=4)

3. IMPLEMENTASI

IV. Model Selection: Random Forest



3. IMPLEMENTASI

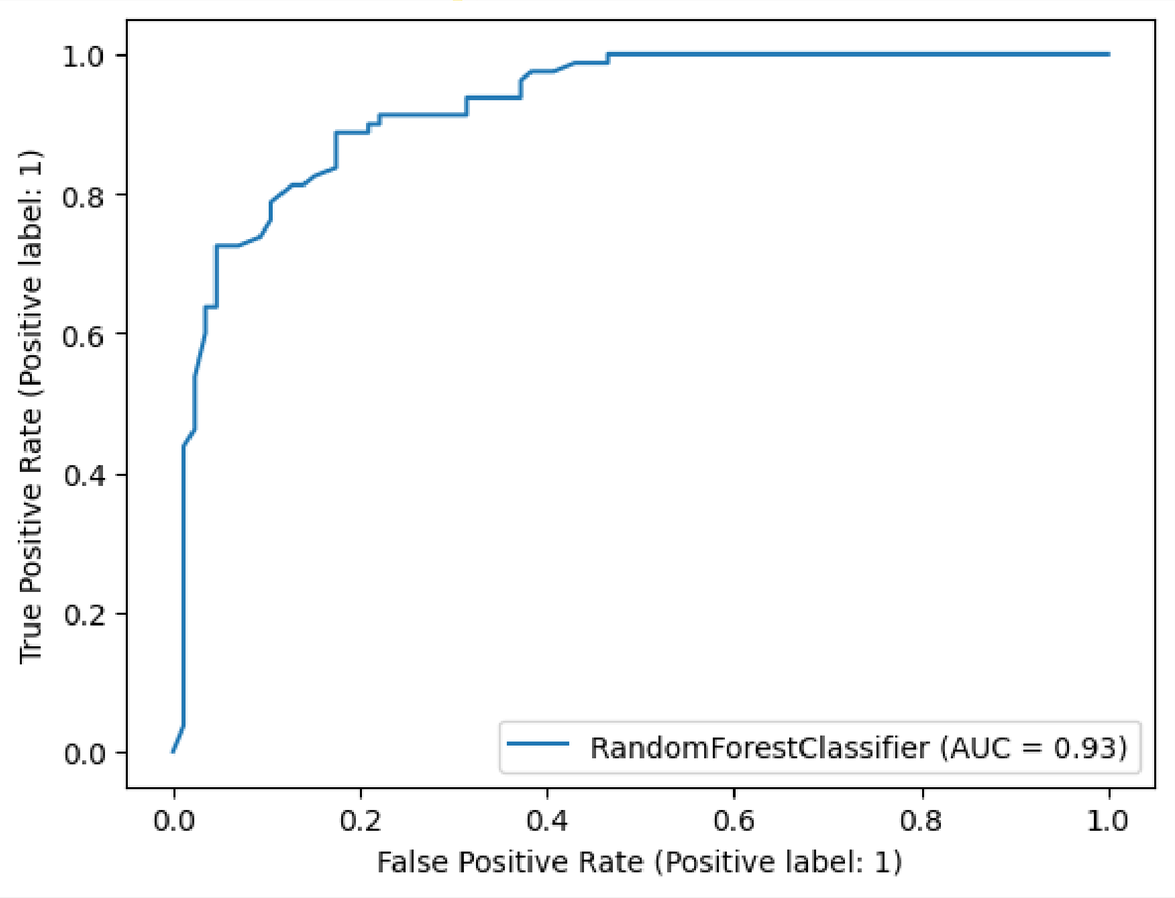
IV. Model Selection: Random Forest

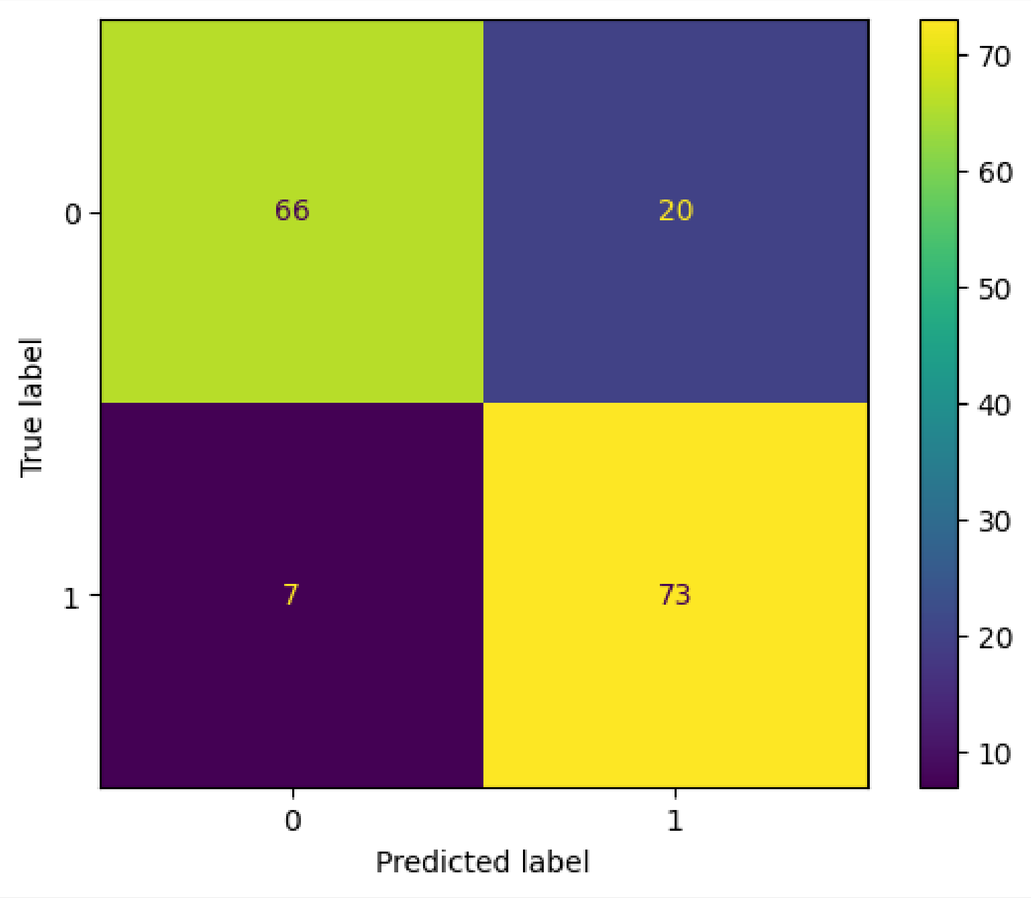
Classification Report for Data Test

	precision	recall	f1-score	support
0	0.90	0.77	0.83	86
1	0.78	0.91	0.84	80
accuracy			0.84	166
macro avg	0.84	0.84	0.84	166
weighted avg	0.85	0.84	0.84	166

3. IMPLEMENTASI

IV. Model Selection: Random Forest





3. IMPLEMENTASI

IV. Model Selection: Deep Neural Network

```
sc = StandardScaler()
cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X_train.loc[:,cols] = sc.fit_transform(X_train.loc[:,cols])
X_test.loc[:,cols] = sc.transform(X_test.loc[:,cols])
```

```
model4 = Sequential()
model4.add(Dense(8, input_shape=(8,), activation='relu'))
model4.add(Dense(8, activation='relu'))
model4.add(Dense(1, activation='sigmoid'))
model4.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

```
from keras.callbacks import EarlyStopping, ModelCheckpoint
callback_a = EarlyStopping(monitor='val_loss', mode='min', patience=20, verbose=1)
```

```
history = model4.fit(X_train, y_train, validation_data=(X_test, y_test),
                     epochs=150, batch_size=10,callbacks=[callback_a])
```

```
{'verbose': 1, 'epochs': 150, 'steps': 66}
```

3. IMPLEMENTASI

IV. Model Selection: Deep Neural Network

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 8)                 72

 dense_4 (Dense)             (None, 8)                 72

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 153
Trainable params: 153
Non-trainable params: 0
```

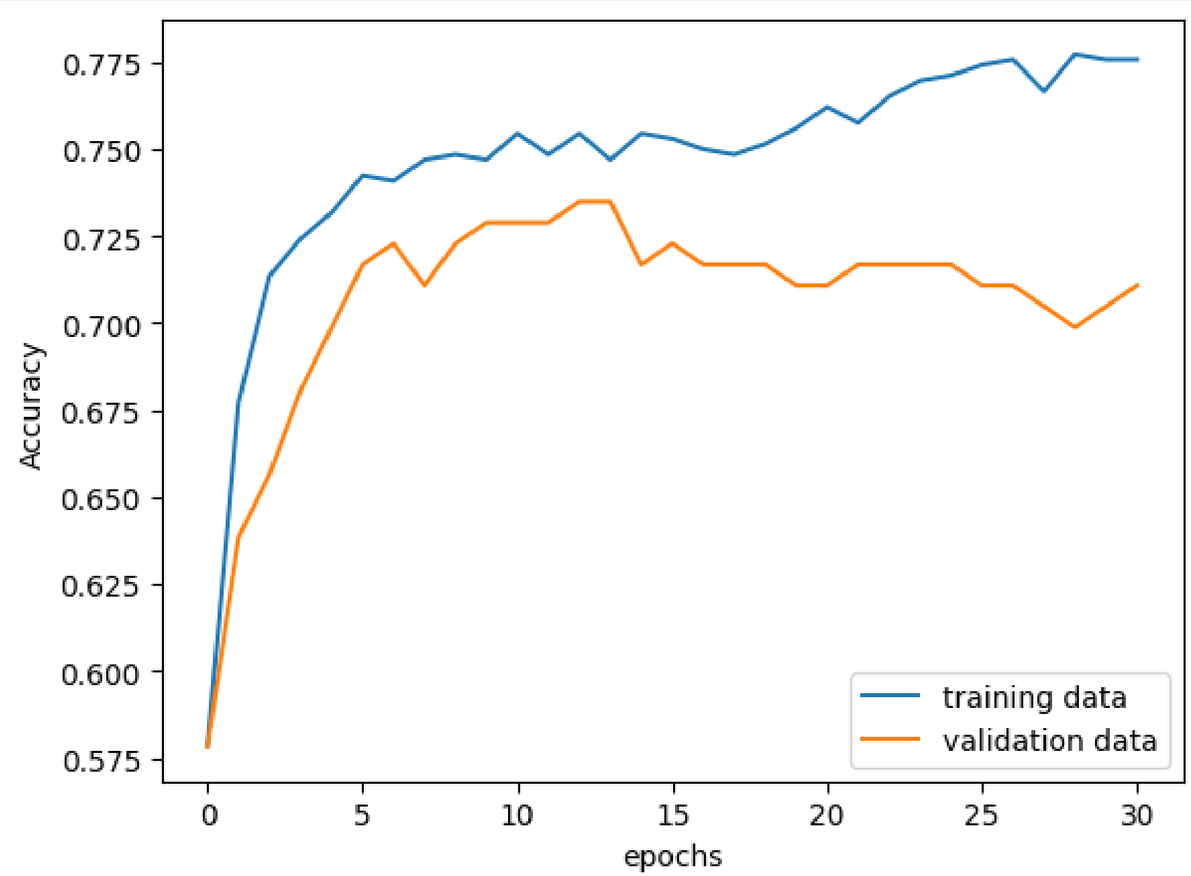
3. IMPLEMENTASI

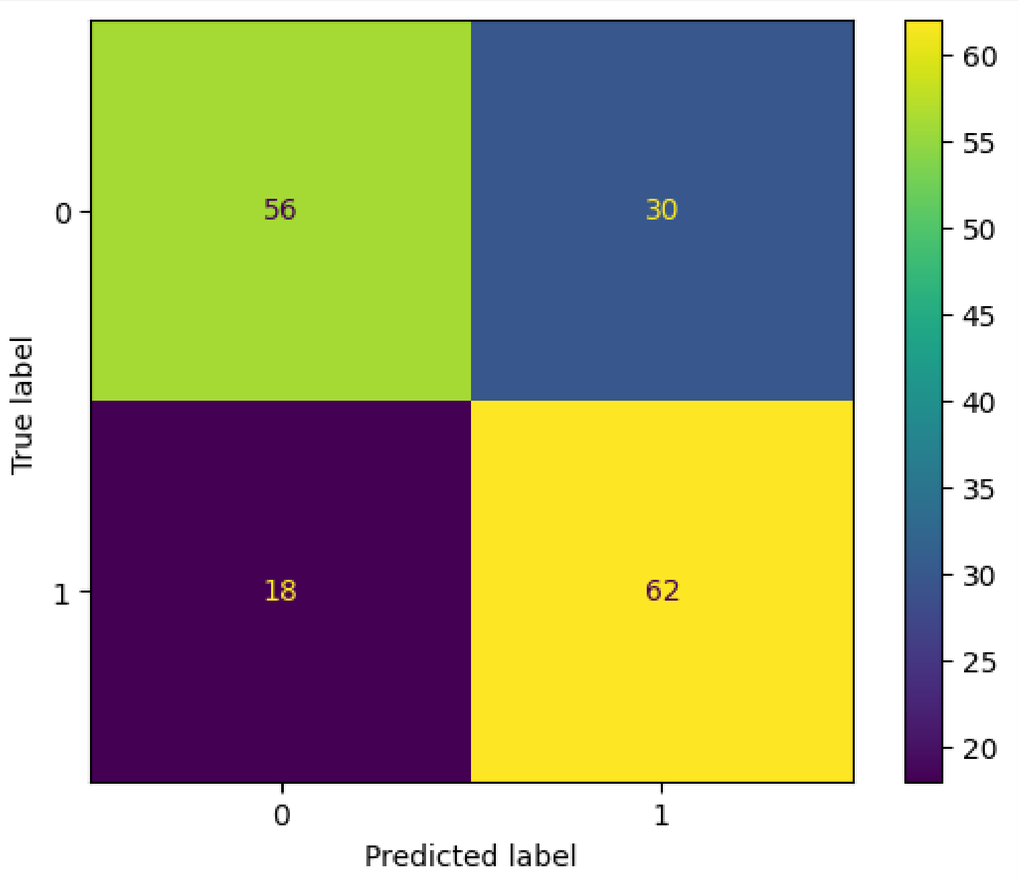
IV. Model Selection: Deep Neural Network

	precision	recall	f1-score	support
0	0.73	0.72	0.73	86
1	0.70	0.71	0.71	80
accuracy			0.72	166
macro avg	0.72	0.72	0.72	166
weighted avg	0.72	0.72	0.72	166

3. IMPLEMENTASI

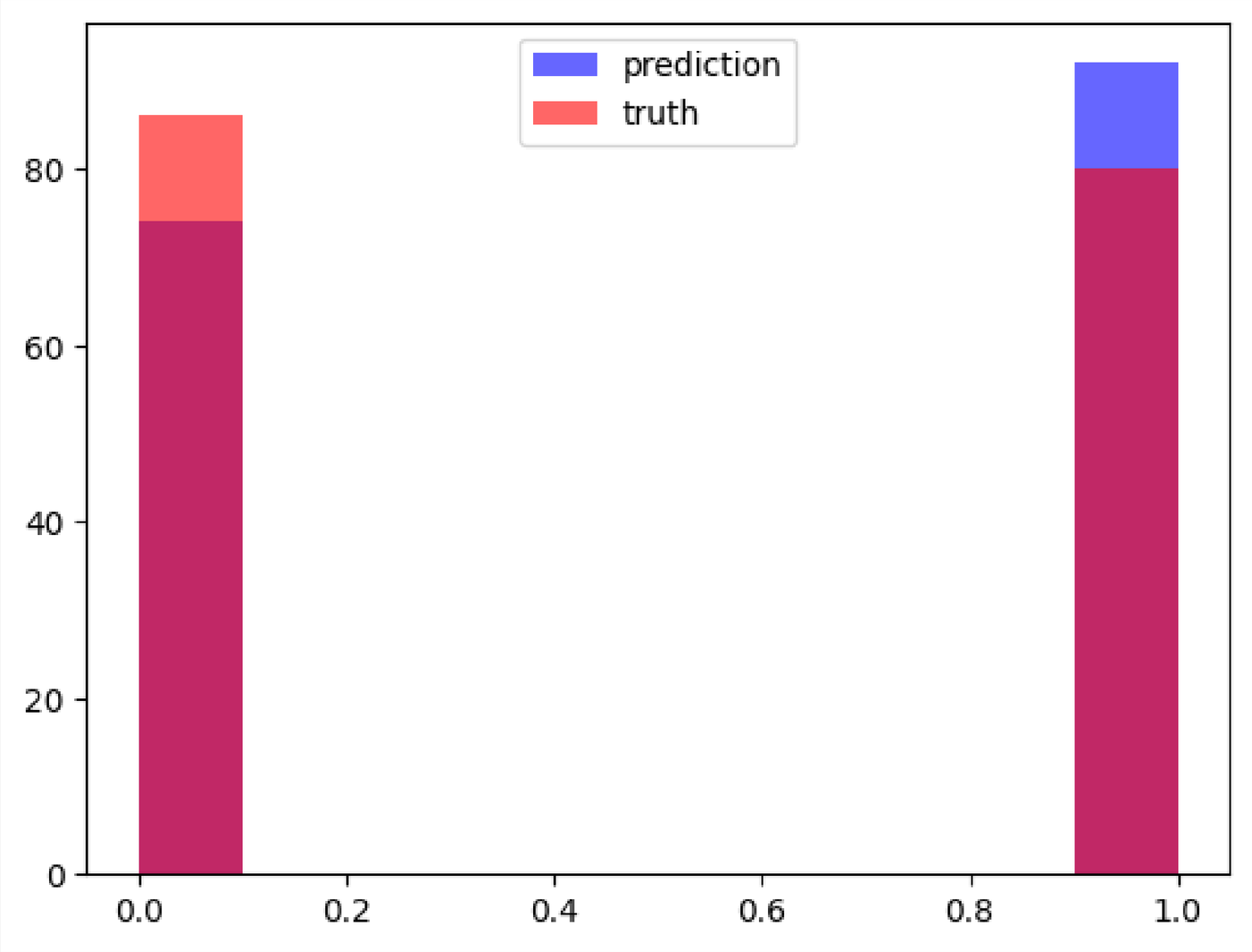
IV. Model Selection: Deep Neural Network





3. IMPLEMENTASI

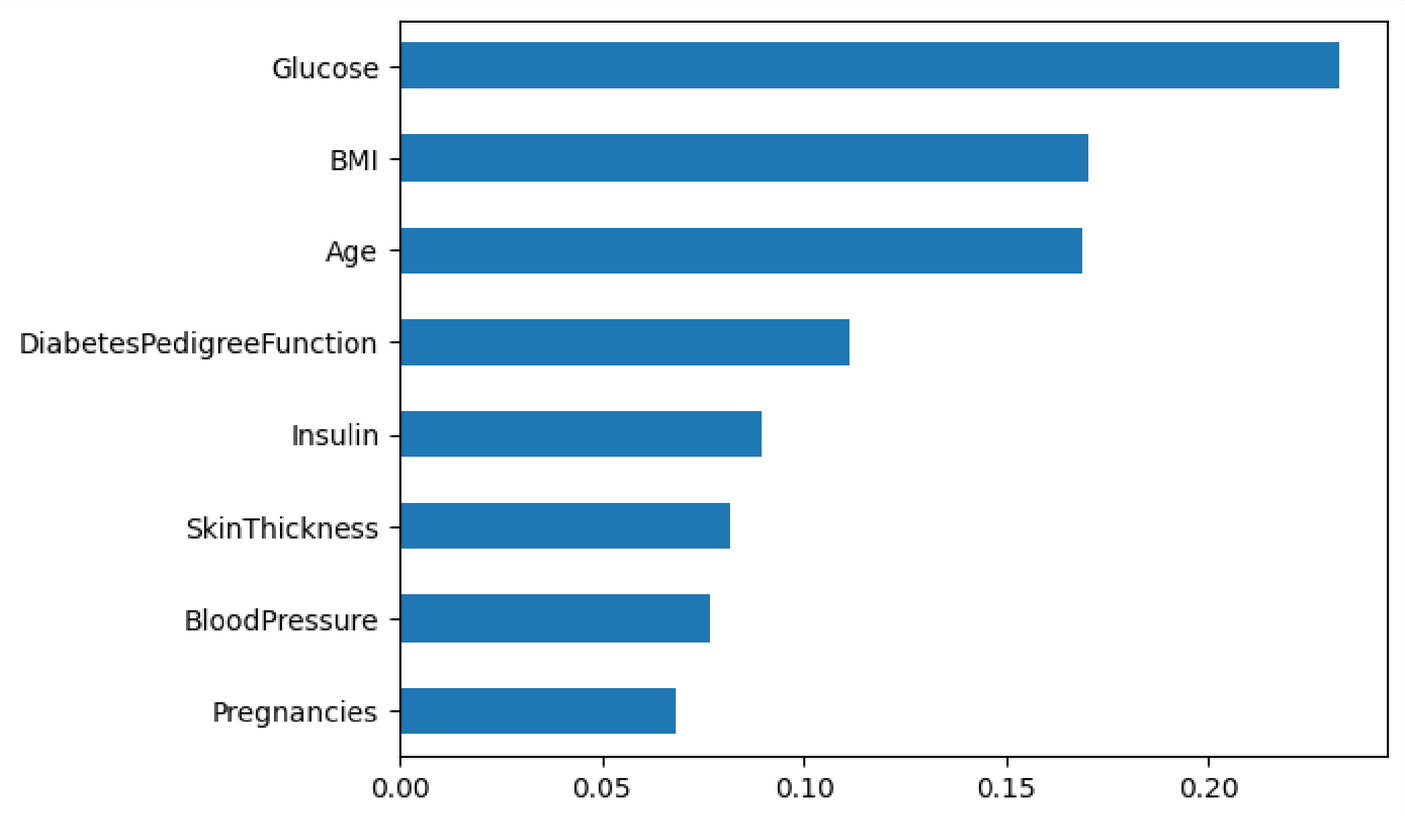
IV. Model Selection: Deep Neural Network



ANALISIS DATA

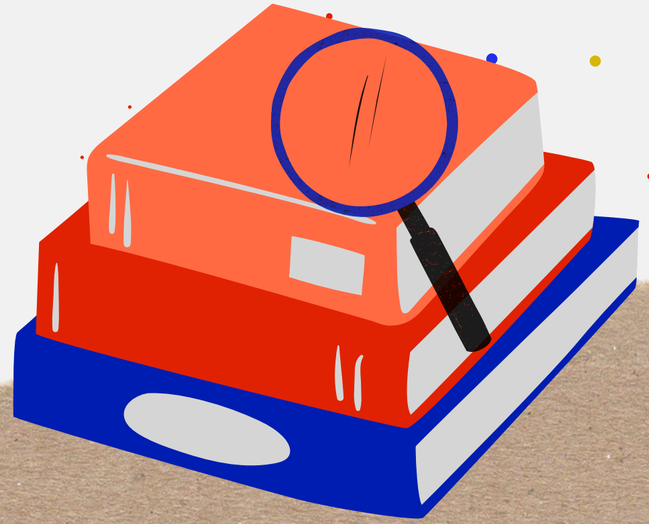
4. ANALISIS DATA

Hasil dan Pembahasan



Berikut adalah hasil yang diperoleh dengan menggunakan model Random Forest. Terlihat bahwa **glukosa** menjadi faktor penyebab utama seseorang terkena diabetes diikuti oleh **BMI** dan umur (**Age**).

KESIMPULAN DAN SARAN

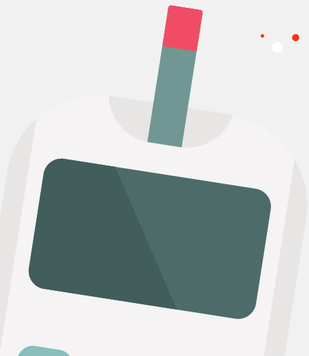


KESIMPULAN

Dari pengolahan dan analisis data disimpulkan bahwa model dengan Random Forest memprediksi data dengan baik karena memiliki akurasi yang paling tinggi dibandingkan model-model lain yang telah kami coba dan didapatkan hasil akhir bahwa glukosa merupakan penyebab utama seseorang terkena diabetes.

SARAN

Agar didapatkan hasil yang lebih optimal mungkin bisa digunakan metode lain yang mungkin saja memiliki tingkat akurasi yang lebih tinggi dari pada Random Forest. Namun tentu saja perlu diperhatikan fitur-fitur yang ada.



THANK YOU