BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Penelitian ini menggunakan beberapa metodologi atau langkah-langkah yang digunakan untuk mengumpulkan data hingga melakukan pembuatan laporan akhir. Berikut merupakan metodologi yang digunakan dalam penelitian ini.

3.1.1 Studi Literatur

Tahapan pertama yang dilakukan dalam melakukan penelitian ini adalah dengan melakukan studi literatur. Studi literatur dilakukan untuk memperoleh informasi yang relevan mengenai topik diabetes, *Logistic Regression*, dan beberapa penelitian sejenis. Melalui studi literatur yang dilakukan, diperoleh gambaran yang lebih menyeluruh untuk pengembangan penelitian dari penelitian sebelumnya dan informasi yang didapatkan juga lebih lengkap untuk membantu selama proses penelitian ini.

3.1.2 Pengumpulan Data

Tahapan pengumpulan data pada penelitian ini memakai tiga *dataset*, *dataset* pertama diambil dari *website* Kaggle yang berjudul ”Pima Indians Diabetes Database”. *Dataset* ini terdiri dari informasi medis pasien perempuan keturunan Indian Pima yang setidaknya berusia 21 tahun yang terdiri dari 768 data . *dataset* kedua diambil dari IEEEDataport yang berjudul ”TYPE 2 DIABETES DATASET” yang berasal dari rumah sakit Frankfurt di Jerman yang terdiri dari 2000 data. Pada *dataset* ketiga, *dataset* satu dan *dataset* dua akan digabungkan sehingga mempunyai total sejumlah 2768 data. Ketiga *dataset* ini mempunyai sembilan fitur terdiri dari delapan variabel prediktor dan satu variabel target. Berikut merupakan fitur-fitur yang digunakan dalam penelitian ini yang ada pada Tabel 3.1.

Tabel 3.1. Keterangan fitur pada *dataset*

Fitur	Tipe Data
Pregnancies	Integer
Glucose	Integer
Blood Pressure	Integer
Skin Thickness	Integer
Insulin	Integer
BMI	Float
Diabetes Pedigree Function	Float
Age	Integer
Outcome	Integer

Berikut adalah penjelasan lebih lanjut mengenai masing-masing fitur dalam *dataset* ini.

- *Pregnancies*: Jumlah kehamilan yang dialami oleh pasien. Kehamilan dapat mempengaruhi risiko diabetes gestasional dan tipe 2 pada wanita .
- *Glucose*: Hasil tes toleransi glukosa dalam dua jam setelah mengkonsumsi cairan glukosa. Glukosa darah yang tinggi adalah indikator langsung dari risiko diabetes.
- *Blood Pressure*: Tekanan darah diastolik diukur dalam mm/Hg. Tekanan darah tinggi dapat menyebabkan resistensi insulin, yang meningkatkan risiko diabetes.
- *Skin Thickness*: Ketebalan kulit diukur dalam mm, biasanya diukur pada lipatan *triceps*. Ini merupakan indikator dari lemak dan dapat berhubungan dengan tingkat obesitas dan resistensi insulin .
- Insulin: Kadar insulin dalam darah setelah dua jam mengkonsumsi makanan. Kadar insulin yang tinggi atau rendah dapat menunjukkan masalah dalam produksi atau penggunaan insulin oleh tubuh .
- *Body Mass Index* (BMI): Indeks massa tubuh dihitung berdasarkan berat dan tinggi badan. BMI tinggi menunjukkan obesitas, yang merupakan faktor risiko utama untuk diabetes tipe 2 .
- *Diabetes Pedigree Function*: Indikator ini mencerminkan riwayat diabetes dalam keluarga yang menunjukkan kemungkinan risiko diabetes berdasarkan faktor genetik.

- *Age*: Umur pasien dan memiliki risiko diabetes meningkat seiring bertambahnya usia.
- *Outcome*: Kategori biner (0 atau 1) yang menunjukkan apakah pasien menderita diabetes (1) atau tidak (0).

3.1.3 Perancangan Sistem

Pada tahap ini, sistem dirancang dengan membuat *flowchart* yang berguna untuk mengetahui alur kerja dari sistem yang akan digunakan untuk membuat klasifikasi pasien perempuan yang terkena penyakit diabetes.

3.1.4 Implementasi

Pada tahap ini rancangan sistem yang telah dibuat akan diterapkan. Sistem ini akan dibuat dengan bahasa pemrograman Python yang dijalankan dengan Jupyter Notebook yang akan digunakan untuk melakukan *preprocess* data, *training* model, validasi, dan evaluasi hasil.

3.1.5 Validasi

Pada tahap ini, sistem yang telah dibuat akan diuji dan dioptimalkan untuk meningkatkan performa model. Pengoptimalan model ini dilakukan dengan cara mengatur *hyperparamter*.

3.1.6 Evaluasi

Pada tahap evaluasi, model akan diukur kinerja nya dalam melakukan klasifikasi terhadap variabel *outcome* (variabel target), yang menentukan apakah seorang pasien menderita diabetes atau tidak. Selain itu, akan dilakukan beberapa pengukuran kinerja model terhadap prediksi seperti akurasi, *recall, precission,* dan f1-*score*. Tahap evaluasi ini dapat memberikan pemahaman mengenai keefektifan pada algoritma dalam melakukan klasifikasi dengan data yang sudah tersedia.

3.1.7 Pembuatan Laporan

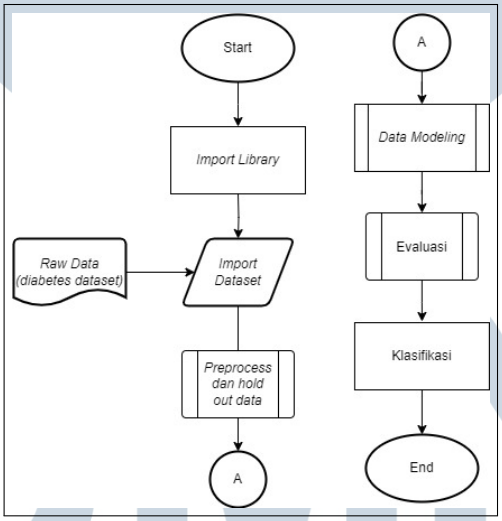
Pada tahapan ini laporan dari penelitian yang dilakukan akan dibuat dan dikembangkan berdasarkan proposal yang telah dirancang. Pada pembuatan laporan

ini hasil dari penelitian yang dilakukan akan dijelaskan secara rinci, beserta dengan beberapa dokumentasi, seperti potongan kode yang akan dilampirkan pada penelitian yang telah dilakukan, serta kesimpulan dari hasil analisis dan pembahasan yang telah dilakukan selama penelitian.

3.2 Perancangan Sistem

Sistem dibuat dengan merancang *flowchart* untuk melakukan implementasi algoritma *Logistic Regression* untuk melakukan klasifikasi pasien perempuan yang terkena penyakit diabetes. *Flowchart* dibagi menjadi *flowchart* utama dan *flowchart detail* dari salah satu bagian rancangan sistem.

3.2.1 Flowchart Utama



Gambar 3.1. *Flowchart* utama

Gambar 3.1 merupakan *flowchart* utama dari sistem klasifikasi pasien perempuan yang terkena diabetes dengan menggunakan algoritma *Logistic Regression*. Tahap pertama dalam membuat sistem ini dengan melakukan *import library* yang berguna untuk pengolahan data dan menjalankan kinerja dari model yang akan dibuat. Setelah *library* di*import* selanjutnya *raw* data dalam bentuk csv dimasukkan atau di*import* ke dalam sistem dan akan dibaca oleh sistem. Selanjutnya dilakukan *preprocess* dan *hold out* data (*split dataset*) untuk mengolah data agar lebih mudah dipahami dan membagi data menjadi dua data menjadi *training* data dan *testing* data dengan pembagian 80% untuk data *training* dan

20% untuk data *test*. Pada tahap pemodelan, model dilatih dengan menggunakan data *train*. Pada tahap evaluasi, model yang telah dilatih akan diukur kinerja nya dengan menggunakan data *test*. Jika hasil evaluasi sudah baik, akan dilanjutkan dengan tahap klasifikasi pasien perempuan yang terkena diabetes atau tidak terkena diabetes.

3.2.2 Import Library

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, f1_score,recall_score, precision_score,
   roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import roc_curve, auc
```

Kode 3.1: *Import library*

Pada Kode 3.1 merupakan daftar *library* yang di*import* ke dalam sistem yang digunakan untuk menjalankan sistem *machine learning*.

3.2.3 Import dan Membaca Raw Data

```
diabetes = pd.read_csv('Data/diabetes.csv')
diabetes.head()
```

Kode 3.2: *Read dataset*

Pada Kode 3.2, *dataset* yang memiliki format csv dibaca oleh *machine learning*. *Dataset* ini yang akan digunakan untuk melakukan pelatihan model.

Selain membaca *dataset* dilakukan juga analisis *dataset* dengan menampilkan lima baris pertama dari *dataset*. Pada penelitian ini digunakan dua dataset

	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

Gambar 3.2. Lima baris pertama *dataset* satu

Pada Gambar 3.2 terlihat *raw* data yang telah di masukkan memiliki sembilan kolom yaitu, *Pregnancies*, *Glucose*, *BloodPressure*, *SkinThickness*, Insulin, BMI, *DiabetesPedigreeFunction*, *Age*, dan *Outcome*.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

Gambar 3.3. Lima baris pertama *dataset* dua

Pada Gambar 3.3 terlihat *raw* data yang telah di masukkan memiliki sembilan kolom yaitu, *Pregnancies*, *Glucose*, *BloodPressure*, *SkinThickness*, Insulin, BMI, *DiabetesPedigreeFunction*, *Age*, dan *Outcome*.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

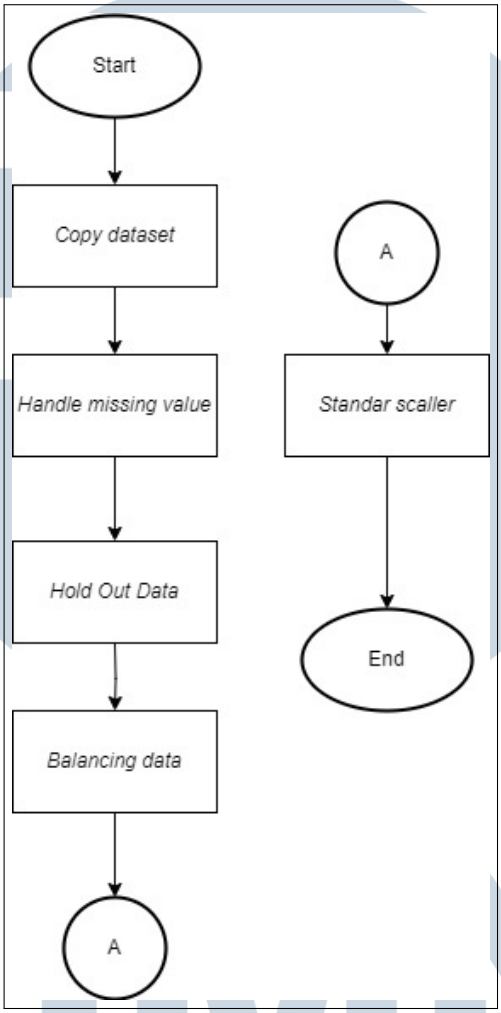
```
diabetes.shape
```

```
(2768, 9)
```

Gambar 3.4. Lima baris pertama *dataset* tiga

Pada Gambar 3.4 terlihat *raw* data yang telah di masukkan memiliki sembilan kolom yaitu, *Pregnancies*, *Glucose*, *BloodPressure*, *SkinThickness*, Insulin, BMI, *DiabetesPedigreeFunction*, *Age*, dan *Outcome* dengan jumlah sebanyak 2768 data.

3.2.4 Preprocess dan Hold Out Data



Gambar 3.5. *Flowchart preprocess* dan *hold out* data

Pada Gambar 3.5 merupakan *flowchart* dari tahapan *preprocess* dan *hold out* data. Data yang telah didapatkan, kemudian akan dilakukan data *preprocessing* yang memproses data agar dapat lebih mudah dipahami. Pada tahap pertama dalam *preprocess* data, data yang telah di*import* akan di*copy* atau membuat salinan data terlebih dahulu agar perubahan yang terjadi tidak memengaruhi data aslinya. Pada tahap kedua, data yang dipakai akan dicek terlebih *missing value*nya, *missing value* pada data akan diganti dengan *median*. Selanjutnya membagi data menjadi data *train* dan data *test* sebelum melakukan penanganan keseimbangan data yang berguna untuk menghindari adanya informasi dari data yang nantinya akan digunakan untuk *test* masuk atau tercampur ke data *training*. Pada tahap

selanjutnya, dilakukan penanganan keseimbangan terhadap data. Pada tahap terakhir, melakukan standardisasi dengan menggunakan *Standar Scaller* data agar antar variabel pada data memiliki rentang nilai (*scale*) yang sama.

Untuk proses *preprocessing* dapat dilihat dari potongan kode berikut.

```
train_data = diabetes.copy()
train_data[['Glucose','BloodPressure','SkinThickness','Insulin','
    BMI','DiabetesPedigreeFunction','Age']] = train_data[['Glucose
    ','BloodPressure','SkinThickness','Insulin','BMI','
    DiabetesPedigreeFunction','Age']].replace(0,np.NaN)
print(train_data.isnull().sum())
```

Kode 3.3: *Copy* data dan cek *missing value*

Pada Kode 3.3, *dataset* di*copy* agar pada saat melakukan pelatihan dan pengujian terhadap data, *dataset* yang asli tidak mengalami perubahan. Setelah *dataset* di*copy*, melakukan pemilihan kolom atau fitur untuk mengganti nilai nol pada data menjadi *NaN*. Kolom *Pregnancies* dikecualikan atau tidak dipilih karena secara umum, ada kemungkinan pasien tidak pernah melahirkan. Setelah nilai nol diganti menjadi *NaN*, data akan di*print* untuk melihat kolom yang memiliki *missing value*.

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

Gambar 3.6. *Missing value* pada *dataset* satu

Pada Gambar 3.6 terlihat bahwa kolom *Glucose* memiliki 5 *missing value*, kolom *BloodPressure* memiliki 35 *missing value*, kolom *SkinThickness* memiliki 227 *missing value*, kolom Insulin memiliki 374 *missing value*, dan kolom BMI memiliki 11 *missing value*. *Missing value* pada data ini akan ditangani pada tahap selanjutnya yaitu *handle missing value*.

```
Pregnancies                 0
Glucose                    13
BloodPressure              90
SkinThickness             573
Insulin                   956
BMI                        28
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Gambar 3.7. *Missing value* pada *dataset* dua

Pada Gambar 3.7 terlihat bahwa kolom *Glucose* memiliki 13 *missing value*, kolom *BloodPressure* memiliki 90 *missing value*, kolom *SkinThickness* memiliki 573 *missing value*, kolom Insulin memiliki 956 *missing value*, dan kolom BMI memiliki 28 *missing value*. *Missing value* pada data ini akan ditangani pada tahap selanjutnya yaitu *handle missing value*.

```
Pregnancies                 0
Glucose                    18
BloodPressure             125
SkinThickness             800
Insulin                  1330
BMI                        39
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Gambar 3.8. *Missing value* pada *dataset* tiga

Pada Gambar 3.8 terlihat bahwa kolom *Glucose* memiliki 18 *missing value*, kolom *BloodPressure* memiliki 125 *missing value*, kolom *SkinThickness* memiliki 800 *missing value*, kolom Insulin memiliki 1330 *missing value*, dan kolom BMI memiliki 39 *missing value*. *Missing value* pada data ini akan ditangani pada tahap selanjutnya yaitu *handle missing value*.

```
1 Glucose = train_data['Glucose'].median()
2 train_data['Glucose'] = train_data['Glucose'].fillna(Glucose)
3
4 BloodPressure = train_data['BloodPressure'].median()
5 train_data['BloodPressure'] = train_data['BloodPressure'].fillna(
      BloodPressure)
6
7 SkinThickness = train_data['SkinThickness'].median()
8 train_data['SkinThickness'] = train_data['SkinThickness'].fillna(
      SkinThickness)
```

```
9
10 Insulin = train_data['Insulin'].median()
11 train_data['Insulin'] = train_data['Insulin'].fillna(Insulin)
12
13 BMI = train_data['BMI'].median()
14 train_data['BMI'] = train_data['BMI'].fillna(BMI)
15
16 print(train_data.isnull().sum())
```

Kode 3.4: *Handle missing value*

Pada Kode 3.4, dilakukan *handle missing value* pada data dengan cara menggantikan nilai yang hilang pada kolom *Glucose, BloodPressure, SkinThickness,* Insulin, dan BMI dengan nilai tengah (*median*) dari masing-masing kolom tersebut. Setelah dilakukan *handle missing value*nya maka data akan di*print* kembali untuk melihat hasil dari *handle missing value*.

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

Gambar 3.9. Hasil dari proses *handle missing value*

Pada Gambar 3.9 menunjukkan hasil setelah melakukan *handle missing value* terlihat bahwa kolom-kolom yang mempunyai *missing value* tadi, sudah tidak ada lagi *value* yang kosong.

```
1 data_cek = train_data['Outcome'].value_counts()
2 print("Outcome:")
3 print(data_cek)
4
5 sns.countplot(x='Outcome', data=diabetes)
6 plt.title('Outcome')
7 plt.show()
```

Kode 3.5: Cek keseimbangan data

Pada Kode 3.5, dilakukan analisis data untuk memeriksa nilai dalam kolom *Outcome*, untuk mengetahui keseimbangan data. Data akan divisualisasi dengan

bentuk histogram agar dapat dilihat dengan mudah proporsi setiap *class* dalam kolom *Outcome*.

```
Outcome:
0    500
1    268
Name: Outcome, dtype: int64
```

Gambar 3.10. Hasil dari pengecekan jumlah *dataset* satu

Pada Gambar 3.10 terlihat bahwa data antar *class* tidak seimbang, dengan data *class* nol sebanyak 500 dan data pada *class* satu sebanyak 268, oleh karena itu perlu dilakukan penanganan terhadap keseimbangan data.

```
Outcome:
0    1316
1     684
Name: Outcome, dtype: int64
```

Gambar 3.11. Hasil dari pengecekan jumlah *dataset* dua

Pada Gambar 3.11 terlihat bahwa data antar *class* tidak seimbang, dengan data *class* nol sebanyak 1316 dan data pada *class* satu sebanyak 684, oleh karena itu perlu dilakukan penanganan terhadap keseimbangan data.

```
Outcome:
0    1816
1     952
Name: Outcome, dtype: int64
```

Gambar 3.12. Hasil dari pengecekan jumlah *dataset* tiga

Pada Gambar 3.12 terlihat bahwa data antar *class* tidak seimbang, dengan data *class* nol sebanyak 1816 dan data pada *class* satu sebanyak 952, oleh karena itu perlu dilakukan penanganan terhadap keseimbangan data.

```
x = train_data.drop('Outcome', axis=1)
y = train_data['Outcome']

```

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

Kode 3.6: *Split dataset*

Pada Kode 3.6, x didefinisikan sebagai data *train* yang memakai semua fitur pada data sebagai variabel prediktor kecuali kolom *Outcome*, dan y didefinisikan sebagai data *test* yang memakai fitur dari kolom *Outcome* sebagai variabel target. Data dibagi atau *hold out* menjadi data *train* dan data *test* dengan *library* dengan ukuran 80:20. Lalu sebuah angka acak dengan nilai 42 yang digunakan agar ketika sistem dijalankan data yang di*split* tidak akan berubah-ubah. Proses pembagian data menjadi data *train* dan data *test* sebelum melakukan penanganan keseimbangan data berguna untuk menghindari adanya informasi dari data yang nantinya akan digunakan untuk *test* masuk atau tercampur ke data *training*.

```
smote = SMOTE(random_state=42)
X_train_balance, y_train_balance = smote.fit_resample(X_train,
    y_train)

data_cek_smote = y_train_balance.value_counts()
print("Sesudah smote:")
print(data_cek_smote)

sns.countplot(x=y_train_balance)
plt.title('Outcome')
plt.show()
```

Kode 3.7: Implementasi SMOTE

Pada Kode 3.7 dilakukan penanganan terhadap keseimbangan data. Setelah *split* data maka *library* SMOTE akan diimplementasikan untuk menangani data yang tidak seimbang. SMOTE tidak boleh diterapkan pada data *test*, karena data *test* dipakai untuk memberikan evaluasi yang valid atas kinerja model. Setelah data sudah ditangani keseimbangannya, selanjutnya data divisualisasikan lagi untuk melihat hasil dari implementasi SMOTE.

```
Sesudah smote:
0    401
1    401
Name: Outcome, dtype: int64
```

Gambar 3.13. Hasil dari pengecekan jumlah *dataset* satu setelah SMOTE

Pada Gambar 3.13 terlihat bahwa data antar *class* telah seimbang setelah menggunakan SMOTE, dengan data *class* nol sebanyak 401 dan data pada *class* satu sebanyak 401.

```
Sesudah smote:
1    1063
0    1063
Name: Outcome, dtype: int64
```

Gambar 3.14. Hasil dari pengecekan jumlah *dataset* dua setelah SMOTE

Pada Gambar 3.14 terlihat bahwa data antar *class* telah seimbang setelah menggunakan SMOTE, dengan data *class* nol sebanyak 1063 dan data pada *class* satu sebanyak 1063.

```
Sesudah smote:
1    1452
0    1452
Name: Outcome, dtype: int64
```

Gambar 3.15. Hasil dari pengecekan jumlah *dataset* tiga setelah SMOTE

Pada Gambar 3.15 terlihat bahwa data antar *class* telah seimbang setelah menggunakan SMOTE, dengan data *class* nol sebanyak 1452 dan data pada *class* satu sebanyak 1452.

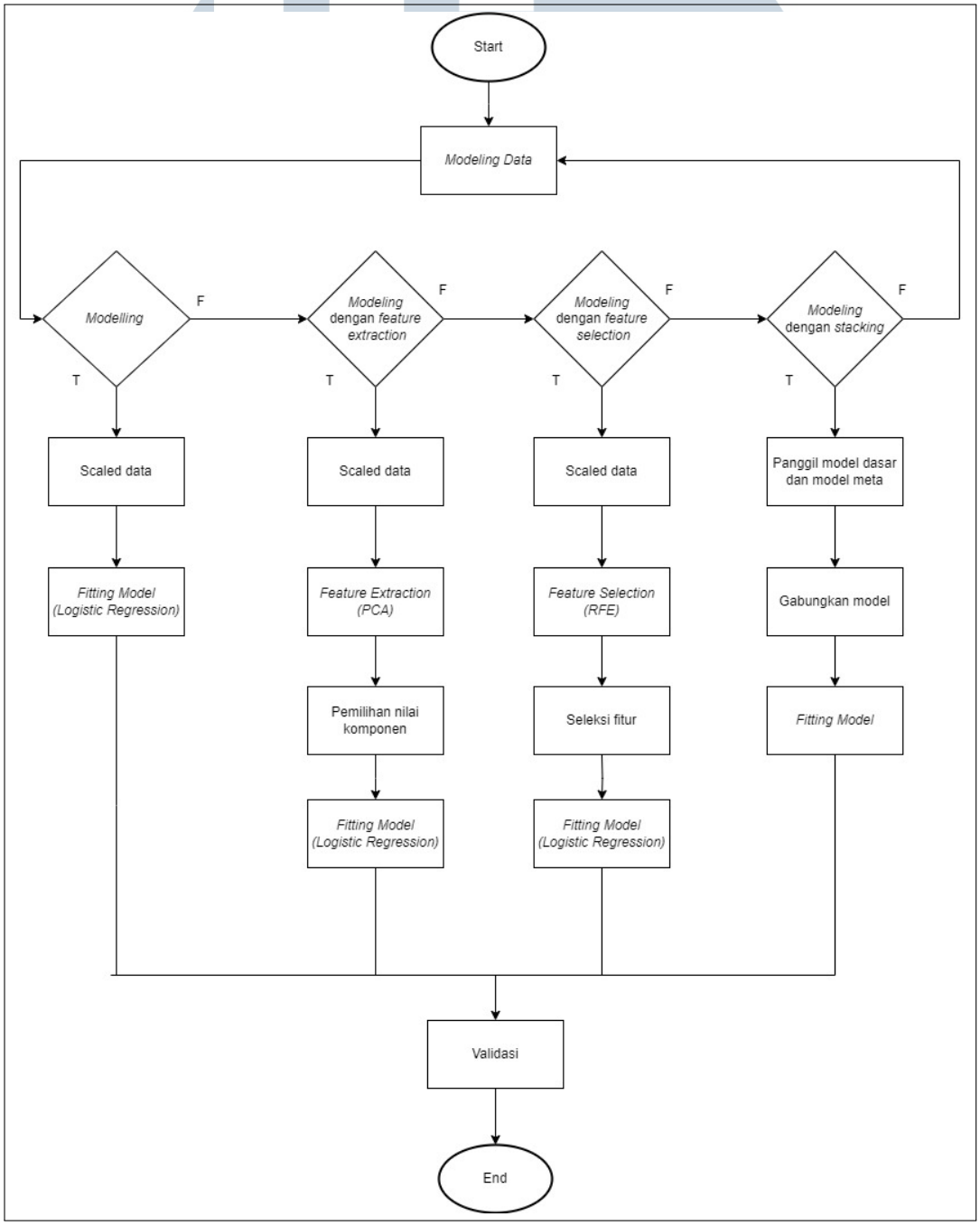
```
#Memakai dataset sebelum SMOTE
scaler_sebelum_smote = StandardScaler()
X_train_scaler_sebelum_smote = scaler_sebelum_smote.fit_transform(
    X_train)
X_test_scaler_sebelum_smote = scaler_sebelum_smote.transform(
    X_test)

#Memakai dataset sesudah SMOTE
scaler_smote = StandardScaler()
X_train_scaler = scaler_smote.fit_transform(X_train_balance)
X_test_scaler = scaler_smote.transform(X_test)
```

Kode 3.8: Standardisasi data

Pada Kode 3.8, dilakukan *scaling* atau standardisasi data. Tujuan dari standardisasi ini adalah agar semua variabel dalam dataset berada dalam rentang yang sama. Proses standardisasi ini dibagi menjadi dua tahap, yaitu

dengan menggunakan dataset sebelum implementasi SMOTE dan dataset setelah implementasi SMOTE.

3.2.5 Data Modeling



Gambar 3.16. *Flowchart* data modeling

Gambar 3.16 merupakan *flowchart* yang menunjukkan proses pemodelan data. Pada tahap pemodelan data, terdapat beberapa skenario pelatihan yang dilakukan, yaitu pemodelan hanya menggunakan *Logistic Regression*, pemodelan dengan metode *feature extraction*, pemodelan dengan metode *feature selection*, pemodelan dengan *stacking*.

Pada pelatihan tanpa *feature extraction*, tanpa *feature selection*, dan tanpa *stacking* data akan langsung diterapkan pada algoritma *Logistic Regression*.

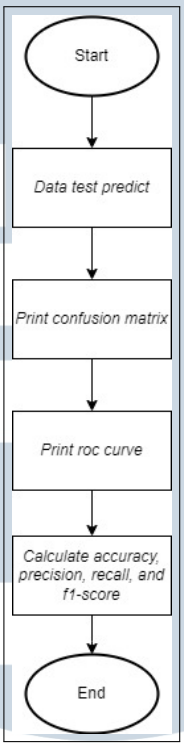
Pada pelatihan dengan *feature extraction*, digunakan teknik *feature extraction* dengan metode *Principal Component Analysis* (PCA). PCA akan mengurangi dimensi fitur dengan membuat fitur baru yang mengurangi dimensi data tanpa kehilangan informasi penting.

Pada pelatihan dengan *feature selection*, *Recursive Feature Elimination* (RFE) digunakan untuk memilih fitur yang paling relevan dengan target prediksi. RFE bekerja dengan menghapus fitur secara berulang-ulang dan membangun model pada fitur yang tersisa untuk menilai kinerja model. Fitur yang memberikan kontribusi paling sedikit terhadap akurasi model dihilangkan dalam setiap iterasi. Penggunaan RFE dalam model *Logistic Regression* membantu mengidentifikasi dan mempertahankan hanya fitur-fitur yang signifikan.

Pada pelatihan dengan metode *stacking*, dilakukan dengan menggabungkan beberapa algoritma yang digunakan untuk meningkatkan kinerja model secara keseluruhan.

Pada tahap validasi, model dapat dioptimalkan lagi dengan melakukan pengaturan *hyperparameter* untuk meningkatkan performa model dengan metode GridSearchCV.

3.2.6 Evaluasi



Gambar 3.17. *Flowchart* evaluasi

Gambar 3.17 merupakan *flowchart* dari proses evaluasi. Pada tahap evaluasi, kinerja model yang telah dilatih akan diukur menggunakan data *test*. Pertama, *confusion matrix* akan dicetak untuk melihat visualisasi dari TN (True Negative), TP (True Positive), FP (False Positive), dan FN (False Negative). Lalu mencetak kurva ROC untuk mengevaluasi kinerja model. Pengukuran kinerja dilanjutkan dengan menghitung akurasi, *recall*, *precision*, dan f1-*score*.

3.3 Spesifikasi Sistem

Spesifikasi sistem yang digunakan adalah laptop dengan spesifikasi Windows 11 dengan spesifikasi processor Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz, RAM sebesar 16GB, dan menggunakan Intel(R) UHD Graphics dan NVIDIA GeForce GTX 1650 sebagai kartu grafis.