BAB IV
HASIL DAN PEMBAHASAN

4.1. Hasil Pengumpulan *Dataset*

Jumlah *dataset* yang berhasil dikumpulkan sebanyak 768 data, dengan jumlah pasien positif (*outcome* 1) sebanyak 268 dan jumlah negatif *(outcome* 0) sebanyak 500 data. Dataset tersebut berasal dari data publik Pima Indians Diabetes Database, dimana didalamnya terdiri dari 8 variabel yaitu jumlah kehamilan, kadar *glukosa*, tekanan darah, ketebalan lipatan kulit *trisep*, kadar *insulin*, *index masa tubuh (IMT)*, riwayat *diabetes melitus* dalam keluarga, serta usia.

Tabel 4. 1 Dataset Indian Pima

No	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
…	…	…	…	…	…	…	…	…	…
768	1	93	70	31	0	30.4	0.315	23	0

Keterangan :

Pregnancies = Jumlah kehamilan
Glucose = Kadar gula darah
Blood pressure = Tekanan darah
Skin thickness = Ketebalan lipatan kulit *trisep*
Insulin = *Insulin*
BMI = Index Massa Tubuh (IMT)
Diabetes pedigree function = Riwayat *diabetes melitus* dalam keluarga
Age = Usia
Outcome = Hasil prediksi pasien

Selain menggunakan *dataset* diatas, juga dilakukan wawancara dengan seorang dokter ahli, wawancara dilakukan dengan Dokter Nindri di Puskesmas Paiton. Hasil dari wawancara tersebut dapat dilihat pada **Tabel 4.2.**

Tabel 4. 2 Hasil Wawancara Dokter

No	Pertanyaan	Jawaban
1	Apakah penyakit *diabetes melitus* merupakan salah satu penyakit dengan jumlah pasien tertinggi?	Ya, jumlah paseien penderita *diabetes melitus* dapat dikatakan cukup tinggi, penyakit ini masuk pada 10 penyakit terbanyak di Puskesmas Paiton.
2	Bagaimana proses identifikasi penyakit *diabetes melitus* saat ini?	Sampai saat ini proses identifikasi terhadap pasien masih mengandalkan hasil pemeriksaan serta hasil tes dari laboratorium
3	Apakah saja faktor faktor yang menyebabkan seseorang menderita *diabetes melitus*?	Banyak faktor yang bisa menyebabkan seseorang menderita *diabetes melitus* seperti hipertensi atau tekanan darah tinggi, kadar gula berlebih, obesitas yang ditandai dengan index massa tubuh ≥25kg/m2, riwayat penyakit *diabetes melitus* dalam keluarga, usia ≥ 45 tahun, jumlah kehamilan, ketebalan lipatan kulit *trisep*, jumlah kadar *insulin* dalam tubuh, kurangnya aktivitas fisik serta pola hidup dan diet tidak sehat juga dapat menjadi penyebab seseorang menderita penyakit *diabetes melitus*.

4.2. Hasil Tahap *Pre-processing*

Penanganan pertama dilakukan dengan menggunakan *excel*, nilai nol dalam variabel jumlah kehamilan dibiarkan sedangan nilai lain dikosongkan terlebih dahulu, hal itu dilakukan guna mempermudah proses penanganan tahap dua dan tiga dengan menggunakan *jupyter notebook*.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	NaN	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	NaN	NaN	30.1	0.349	47	1
767	1	93.0	70.0	31.0	NaN	30.4	0.315	23	0

Gambar 4. 1 Hasil *Pre-processing* Tahap Pertama

Setelah dilakukan penanganan tahap pertama maka jumlah nilai hilang pada variabel jumlah kehamilan akan menjadi 0.

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

Gambar 4. 2 *Missing Values* Setelah Melewati Tahap Pertama

Penanganan tahap kedua dilakukan pada variabel kadar gula/*glukosa*, tekanan darah dan Index Massa Tubuh (IMT), nilai kosong pada variabel ini akan diganti dengan mean atau rata rata.

Segmen Program 4. 1 Mencari *Mean Missing Values*

```
#RATA RATA
import math
dataset['Glucose'].fillna(math.floor(dataset['Glucose'].
mean()),inplace=True)
dataset['BMI'].fillna(math.floor(dataset['BMI'].mean()),
inplace=True)
dataset['BloodPressure'].fillna(math.floor(dataset['BloodPr
essure'].mean()),inplace=True)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	NaN	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	NaN	NaN	30.1	0.349	47	1
767	1	93.0	70.0	31.0	NaN	30.4	0.315	23	0

768 rows × 9 columns

Gambar 4. 3 Hasil *Pre-processing* Tahap Kedua

Setelah dilakukan penanganan tahap kedua maka jumlah nilai hilang pada variabel kadar gula/*glukosa*, tekanan darah dan Index Massa Tubuh (IMT) akan menjadi 0.

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

Gambar 4. 4 *Missing Values* Setelah Melewati Tahap Kedua

Penanganan tahap ketiga dilakukan pada variabel ketebalan lipatan kulit trisep dan insulin. Nilai kosong pada variabel ini akan dihilangkan karena jumlahnya terlalu banyak, sehingga dikhawatirkan akan merubah keadaan sebenarnya jika tetap digunakan.

Segmen Program 4. 2 Menghapus *Missing Values*

```
13  #Menghapus
14  dataset = dataset.dropna()
15  Dataset
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1
...	...	...	...	...	...	...	...	...	...
753	0	181.0	88.0	44.0	510.0	43.3	0.222	26	1
755	1	128.0	88.0	39.0	110.0	36.5	1.057	37	1
760	2	88.0	58.0	26.0	16.0	28.4	0.766	22	0
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0

394 rows × 9 columns

Gambar 4. 5 Hasil *Pre-processing* Tahap Ketiga

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

Gambar 4. 6 *Missing Values* Setelah Melewati Tahap Ketiga

Setelah selesai melewati ketiga tahap penanganan tersebut dapat kita lihat pada **Gambar 4.6** bahwa sudah tidak terdapat missing values, sehingga dapat dilakukan proses pembagian dataset menjadi data latih (training) dan data uji (testing). Data uji yang digunakan berjumlah 12,6% dari total dataset dan data latih berjumlah 87,4% dari total dataset.

Segmen Program 4. 3 Pembagian Dataset

```
17  x = dataset.iloc[:, 0:10].values
18  y = dataset.iloc[:, -1].values
19  from sklearn.model_selection import train_test_split
20  x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size = 0.126, random_state = 0)
```

Setelah dilakukan pembagian dataset berdasarkan aturan diatas maka didapatkan data bersih berupa 344 data latih dan 50 data uji yang siap diolah untuk proses selanjutnya.

Segmen Program 4. 4 Memanggil Data Latih

```
train = pd.DataFrame(x_train)
Train
```

	0	1	2	3	4	5	6	7	8
0	12.0	140.0	82.0	43.0	325.0	39.2	0.528	58.0	1.0
1	11.0	143.0	94.0	33.0	146.0	36.6	0.254	51.0	1.0
2	0.0	100.0	88.0	60.0	110.0	46.8	0.962	31.0	0.0
3	1.0	81.0	72.0	18.0	40.0	26.6	0.283	24.0	0.0
4	2.0	94.0	76.0	18.0	66.0	31.6	0.649	23.0	0.0
...	...	...	...	...	...	...	...	...	...
339	4.0	90.0	88.0	47.0	54.0	37.7	0.362	29.0	0.0
340	1.0	87.0	60.0	37.0	75.0	37.2	0.509	22.0	0.0
341	4.0	91.0	70.0	32.0	88.0	33.1	0.446	22.0	0.0
342	4.0	144.0	58.0	28.0	140.0	29.5	0.287	37.0	0.0
343	1.0	95.0	74.0	21.0	73.0	25.9	0.673	36.0	0.0

344 rows × 9 columns

Gambar 4. 7 Data Latih

Segmen Program 4. 5 Memanggil Data Uji

```
test = pd.DataFrame(x_test)
Test
```

	0	1	2	3	4	5	6	7	8
0	6.0	134.0	80.0	37.0	370.0	46.2	0.238	46.0	1.0
1	1.0	88.0	78.0	29.0	76.0	32.0	0.365	29.0	0.0
2	2.0	108.0	52.0	26.0	63.0	32.5	0.318	22.0	0.0
3	0.0	106.0	70.0	37.0	148.0	39.4	0.605	22.0	0.0
4	0.0	102.0	86.0	17.0	105.0	29.3	0.695	27.0	0.0
5	2.0	121.0	70.0	32.0	95.0	39.1	0.886	23.0	0.0
6	0.0	95.0	80.0	45.0	92.0	36.5	0.330	26.0	0.0
7	11.0	136.0	84.0	35.0	130.0	28.3	0.260	42.0	1.0
8	0.0	118.0	64.0	23.0	89.0	32.0	1.731	21.0	0.0
9	3.0	80.0	82.0	31.0	70.0	34.2	1.292	27.0	1.0
10	4.0	99.0	76.0	15.0	51.0	23.2	0.223	21.0	0.0

Gambar 4. 8 Data Uji

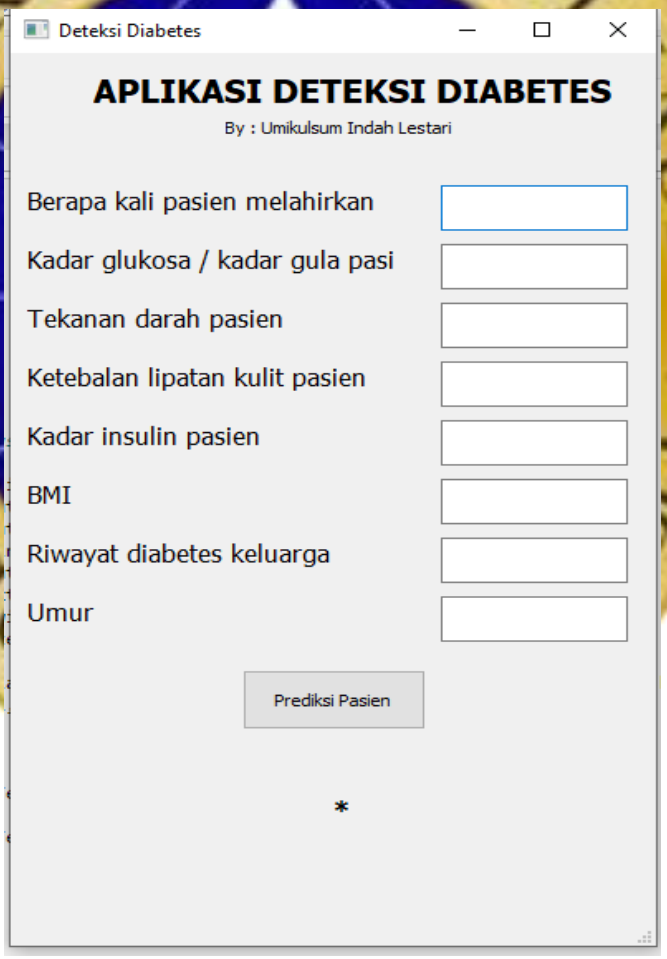
Kedua data tersebut kemudian disimpan dengan format .csv atau *Comma Separated Values*. Proses penyimpanan tersebut dapat dilihat pada **Segmen Program 4. 6** yang terletak dibawah ini :

Segmen Program 4. 6 Menyimpan Data Uji dan Data Latih

```
26  train.to_csv('/Users/lenovo/Downloads/dataset156/skripsi/
    data_latih.csv')
27  test.to_csv('/Users/lenovo/Downloads/dataset156/skripsi/
    data_uji.csv')
```

4.3.Hasil Pembuatan Design Sistem

Design sistem pendukung keputusan dibuat dengan menggunakan *QtDesigner*, design tampilan tersebut dibuat dengan tujuan untuk memudahkan pengguna sistem dalam pengoperasikan sistem pendukung keputusan yang dibuat. Hasil pembuatan design sistem tersebut kemudian disimpan dengan nama *diabetes.ui*. Berikut adalah hasil tampilan sistem pendukung keputusan yang telah dibuat :



Gambar 4. 9 Tampilan Sistem

4.4. Hasil Penerapan Metode KNN

Data bersih berupa 50 data uji dan 344 data latih yang telah didapat kemudian akan dieksekusi dengan menggunakan metode KNN. Penerapan serta penggunaan metode KNN dilakukan pada dua kali percobaan, pertama pada sistem yang dibuat (dengan menggunakan tampilan sistem pada **Gambar 4.9**) sistem dibuat dengan menggunakan bahasa pemrograman *python* dengan menggunakan aplikasi *jupyter notebook* dan yang kedua pada perhitungan dengan menggunakan *excel,* perhitungan dengan *excel* dilakukan untuk melihat hasil perhitungan jarak *euclidean* yang dihasilkan, jarak *euclidean* tersebut kemudian akan diurutkan dari yang terkecil hingga yang terbesar dan dikumpulkan label class Y, sehingga dapat dilakukan prediksi berdasarkan label class Y yang paling mayoritas.

4.4.1. Penerapan Metode KNN Pada Sistem

Langkah-langkah yang dilakukan untuk menggunakan metode KNN pada sistem :

1. Mencari Nilai k Terbaik

Langkah pertama yang perlu dilakukan dalam membuat sistem pendukung keputusan adalah mencari nikai k terbaik dengan tingkat akurasi tertingi. Langkah langkah mencarian nilai k dapat dilihat dibawah ini :

1. Melakukan import library akurasi

Segmen Program 4. 7 Import Library Akurasi

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
```

Import pandas digunakan untuk managemen data csv, import numpy digunakan untuk managemen array, import library (neighbors) untuk klasifikasi metode KNN, import library (metrics) untuk menghitung tingkat akurasi yang dihasilkan.

2. Memasukkan data latih dan data uji

Data latih dan data uji yang sebelumnya telah disimpan dengan format .csv, kemudian dimasukkan dalam sistem.

Segmen Program 4. 8 Memasukkan Data Latih dan Data Uji

```
df = pd.read_csv('/Users/lenovo/Downloads/dataset156/
skripsi/data_latih.csv')
X_train = df.values
X_train = np.delete(X_train,8,axis=1)
y_train = df['Outcome'].values

df = pd.read_csv('/Users/lenovo/Downloads/dataset156/
skripsi/data_uji.csv')
X_test = df.values
X_test = np.delete(X_test,8,axis=1)
y_test = df['Outcome'].values
```

3. Mencari nilai k paling optimal

Segmen Program 4. 9 Mencari Nilai k Paling Optimal

```
knn_clf = KNeighborsClassifier(n_neighbors=9)
knn_clf.fit(X_train,y_train)
y_pred = knn_clf.predict(X_test)
round(accuracy_score(y_test,y_pred),3)
```

Nilai k yang biasa digunakan dalam metode KNN adalah nilai ganjil, hal itu dimaksudkan untuk menghindari jumlah class Y yang sama (antara class positif dan class negatif) sehingga mempermudah dalam proses penarikan kesimpulan. Pada tahap ini dilakukan pengujian tehadap angka angka ganjil dalam rentan 1-30, sehingga menghasilkan tingkat akurasi sebagai berikut :

Gambar 4. 10 Hasil Akurasi Nilai k

```
knn_clf = KNeighborsClassifier(n_neighbors=13)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=15)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=17)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=19)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=21)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.86
```

```
knn_clf = KNeighborsClassifier(n_neighbors=23)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=25)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=27)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.84
```

```
knn_clf = KNeighborsClassifier(n_neighbors=29)
knn_clf.fit(X_train,y_train)

y_pred = knn_clf.predict(X_test)

round(accuracy_score(y_test,y_pred),3)
```

```
0.82
```

Gambar 4. 10 Hasil Akurasi Nilai k (Lanjutan)

Untuk lebih ringkasnya dapat dilihat pada table dibawah ini :

Tabel 4. 3 Hasil Akurasi Nilai k

K	Akurasi
1	76%
3	72%
5	78%
7	78%
9	86%

K	Akurasi
11	86%
13	84%
15	84%
17	84%
19	84%

K	Akurasi
21	86%
23	84%
25	84%
27	84%
29	82%

Berdasarkan uji hasil akurasi tersebut, maka didapatkan nilai k terbaik adalah 9,11 dan 21 dengan tingkat akurasi mencapai 86%. Nilai k yang akan digunakan dalam penelitian ini adalah nilai k terkecil dengan tingkat akurasi tertinggi, sehingga nilai k yang akan digunakan adalah k=9.

4. Menyimpan model akurasi paling optimal (nilai k paling optimal)

Setelah kita berhasil menemukan nilai k paling optimal tahap

selanjutnya yang perlu kita lakukan adalah menyimpan model, sehingga akan mudah digunakan untuk tahapan tahapan selanjutnya. Dalam penelitian ini proses penyimpanan model akurasi dilakukukan dengan menggunakan pickle.

Segmen Program 4. 10 Meyimpan Model

```
56  import pickle
57  with open ('knn_pickle_fix', 'wb') as r:
58       pickle.dump(knn_clf,r)
59  with open ('knn_pickle_fix', 'rb') as r:
60       knnp = pickle.load(r)
```

Dapat dilihat pada gambar diatas model disimpan dengan nama **knn_pickle_fix**. Saat proses penyimpanan berhasil maka akan ada file baru dengan nama **knn_pickle_fix** dalam folder kerja kita.

Gambar 4. 11 Hasil Penyimpanan Model

2. Pembuatan Sistem

Pada tahap ini hasil pembuatan design serta model akurasi paling optimal yang telah dibuat sebelumnya akan disatukan sehingga nantinya akan menghasilkan suatu sistem pendukung keputusan yang siap digunakan kapan saja. Langkah langkat pembuatan sistem dapat dilihat dibawah ini :

1. Import library dan memanggil model.

   Tahapan pertama yang perlu dilakukan adalah mengimport library yang dibutuhkan, library yang dibutuhkan serta memanggil model yang telah disimpan sebelumnya.

Segmen Program 4. 11 *Import Library* dan Pemanggilan Model

```
64  from PyQt5.QtWidgets import *
65  from PyQt5 import uic
66  import pickle
67  import numpy as np
68  import sklearn
69
70  with open('knn_pickle_fix','rb') as r:
71       model = pickle.load(r)
```

2. Melakukan prediksi dan menyimpan hasil prediksi pada satu variabel

Segmen Program 4. 12 Prediksi Penyakit *Diabetes Melitus* Pada Sistem

```
73  def diabetes():
74       banyakMelahirkan = float(window.input_melahirkan.
         text())
75       kadarGlukosa = float(window.input_glukosa.text())
76       tekananDarah = float(window.input_darah.text())
77       ketebalanLipatanKulit = float(window.input_kulit.
         text())
78       kadarInsulin = float(window.input_insulin.text())
79       bmi = float(window.input_bmi.text())
80       derajatRiwayatDiabetes = float(window.input_riwayat.
         text())
81       umur = float(window.input_umur.text())
82       dataPasien = np.array((banyakMelahirkan,kadarGlukosa,
         tekananDarah,ketebalanLipatanKulit,kadarInsulin,
         bmi,derajatRiwayatDiabetes, umur))
83       data = np.reshape(dataPasien, (1, -1))
84       isDiabetes = model.predict(data)
85
86       if(isDiabetes==1):
87          window.label_kesimpulan.setText('POSITIF DIABETES')
88       elif(isDiabetes==0):
89          window.label_kesimpulan.setText('NEGATIF DIABETES')
```

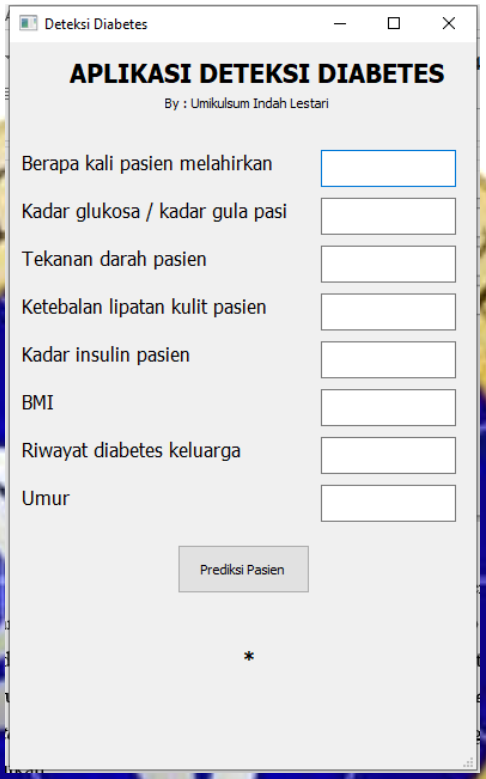
Pada tahap ini dilakukan prediksi data baru terhadap data latih yang ada, kemudia hasil dari prediksi tersebut disimpan dalam satu variable yaitu ***isDiabates***. Apabila hasil prediksi dari ***isDiabetes*** menghasilkan angka 1 maka sistem akan menampilkannya sebagai **positif *diabetes***, sedangkan jika hasil prediksi menghasilkan angka 0, maka sistem akan menampilkannya sebagai **negatif *diabetes***.

3. Menjalankan sistem

   Apabila seluruh tahapan diatas telah dilakukan dengan baik, maka tahap terakhir adalah menjalankan sistem yang telah dibuat.

Segmen Program 4. 13 Menjalankan Sistem

```
app = QApplication([])
window = QMainWindow()
uic.loadUi('diabetes.ui', window)
window.setWindowTitle('Deteksi Diabetes')
window.show()
window.button_prediksi.clicked.connect(diabetes)
app.exec_()
```



Gambar 4. 12 Hasil Tampilan Awal Sistem

Berdasarkan **Gambar 4. 12** diatas dapat dilihat bahwa terdapat 8 variabel inputan yang perlu diisi sebelum melakukan prediksi terhadap pasien, tampilan awal dari hasil prediksi hanya berupa tanda bintang (*) saja, tanda bintang (*) tersebut nantinya akan berubah sesuai menjadi **positif *diabetes*** atau **negatif *diabetes*** sesuai dengan prediksi sistem terhadap pasien yang datanya sedang diinputkan.Untuk itu dilakukan pengujian awal dengan mengambil dua data uji yaitu satu data positif dan satu data negatif.

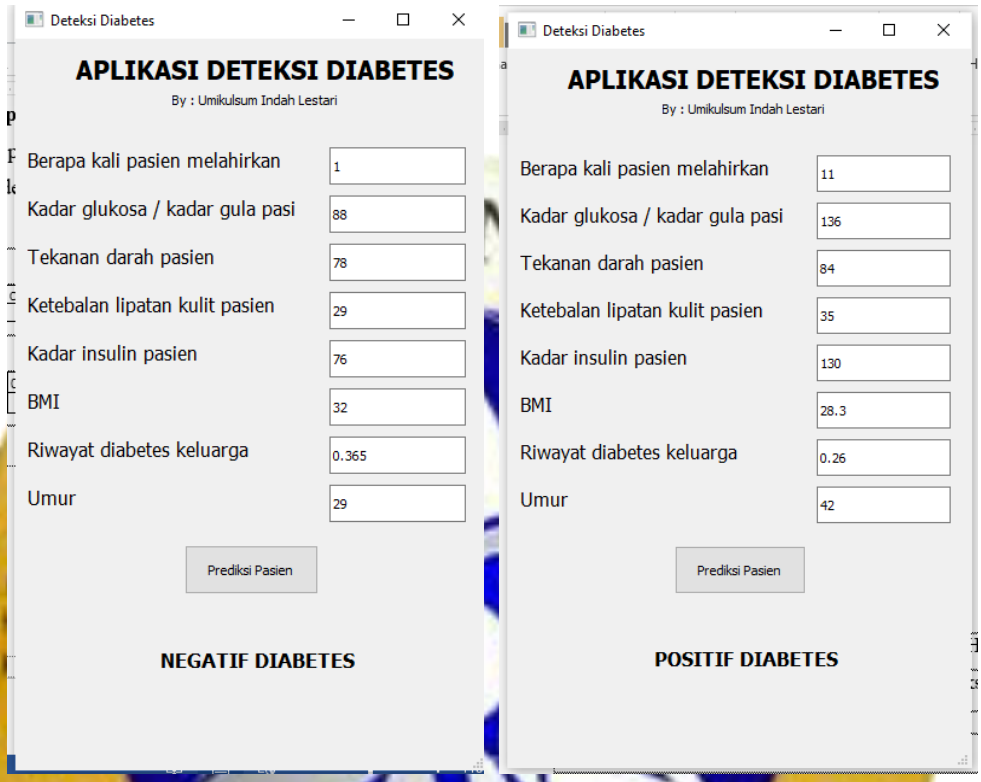
Data uji negatif :

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
1	88	78	29	76	32	0.365	29	0

Data uji positif :

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
11	136	84	35	130	28.3	0.26	42	1

Prediksi yang dihasilkan sistem terhadap dua data uji tersebut dapat dilihat dibawah ini :



(a) Hasil prediksi negatif (b) Hasil prediksi positif

Gambar 4. 13 Hasil Prediksi Sistem

4.4.2. Perhitungan Metode KNN

Berdasarkan prediksi yang telah dilakukan pada **Gambar 4. 13** dapat dilihat bahwa sistem hanya menampilkan hasil akhir atau hasil prediksinya saja tanpa proses perhitungan algoritma KNN. Untuk itu, dalam penelitian ini juga dilakukan proses perhitungan manual, untuk membuktikan bahwa prediksi yang dihasilkan oleh sistem benar adanya. Langkah langkah perhitungan metode KNN dapat dilihat pada uji perhitungan dibawah ini :

1. Perhitungan terhadap prediksi negatif

   Data uji negatif :

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
1	88	78	29	76	32	0.365	29	0

a. Menentukan nilai k.

   Berdasarkan pada tingkat akurasi sebelumnya, didapatkan nilai k paling optimal adalah 9 dengan tingkat akurasi sebesar 86%, maka nilai k yang akan digunakan dalam perhitungan ini adalah k = 9.

b. Menghitung kuadrat jarak *euclidean* masing masing data latih terhadap data uji.

   Tabel 4. 4 Perhitungan jarak *euclidean* (uji sempel negatif)

Data ke	Hitung	Hasil
1	$((12-1)^2 + (140-88)^2 + (82-78)^2 + (43-29)^2 + (325-76)^2 + (39.2-32)^2 + (0.528-0.365)^2 + (58-29)^2)^{½}$	256.7700656
2	$((11-1)^2 + (143-88)^2 + (94-78)^2 + (33-29)^2 + (146-76)^2 + (36.6-32)^2 + (0.254-0.365)^2 + (51-29)^2)^{½}$	93.81989299
3	$((0-1)^2 + (100-88)^2 + (88-78)^2 + (60-29)^2 + (110-76)^2 + (46.8-32)^2 + (0.962-0.365)^2 + (31-29)^2)^{½}$	50.8467935
4	$((1-1)^2 + (81-88)^2 + (72-78)^2 + (18-29)^2 + (40-76)^2 + (26.6-32)^2 + (0.283-0.365)^2 + (24-29)^2)^{½}$	39.4482791
...	...+ ... + ... + ...+ ...+ ... + ... + ...	....
334	$((1-1)^2 + (95-88)^2 + (74-78)^2 + (21-29)^2 + (73-76)^2 + (25.9-32)^2 + (0.674-0.365)^2 + (36-29)^2)^{½}$	14.97681088

c. Mengurutkan objek dari mulai jarak terdekat sampai jarak terjauh

Tabel 4. 5 Urutan jarak *euclidean* (uji sempel negatif)

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome	Jarak	Urutan Ke
7	83	78	26	71	29.3	0.767	36	0	12.30656751	1
5	86	68	28	71	30.2	0.364	24	0	13.20000004	2
1	87	68	34	77	37.6	0.401	24	0	13.54109656	3
1	95	74	21	73	25.9	0.673	36	0	14.97681088	4
4	91	70	32	88	33.1	0.446	22	0	16.88835578	5
2	94	76	18	66	31.6	0.649	23	0	17.26964551	6
2	81	72	15	76	30.1	0.547	25	0	17.36787621	7
3	100	68	23	81	31.6	0.949	28	0	17.62104015	8
2	100	68	25	71	38.5	0.324	26	0	18.36441344	9
2	83	65	28	66	36.8	0.629	24	0	18.55019396	10
...	...	...	...	...	...	...	...	...	...	...
1	189	60	23	846	30.1	0.398	59	1	777.4089085	344

d. Mengumpulkan label class Y (klasifikasi *nearest neighbor*).

Berdasarkan urutan jarak *euclidean* pada **Tabel 4. 5** dapat kita lihat 9 jarak terdekat dari data uji yang ada. Berdasarkan hasil tersebut dapat kita hitung jumlah prediksi positif (*outcome* 1) sebanyak 0 data dan prediksi negatif (*outcome* 0) sebanyak 9 data.

e. Melakukan prediksi berdasarkan label class Y yang paling mayoritas.

Label class Y paling mayoritas dari hasil uji tersebut adalah 0 atau negatif, jadi dapat disimpulkan bahwa data yang baru saja diinputkan adalah **negatif *diabetes***.

2. Perhitungan terhadap prediksi positif

Data uji positif :

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
11	136	84	35	130	28.3	0.26	42	1

a. Menentukan nilai k.

Berdasarkan pada tingkat akurasi sebelumnya, didapatkan nilai k paling optimal adalah 9 dengan tingkat akurasi sebesar 86%, maka nilai k yang akan digunakan dalam perhitungan ini adalah k = 9.

b. Menghitung kuadrat jarak *euclidean* masing-masing data latih terhadap data uji.

Tabel 4. 6 Perhitungan jarak *euclidean* (uji sempel positif)

Data ke	Hitung	Hasil
1	$((12-11)^2 + (140-136)^2 + (82-84)^2 + (43-35)^2 + (325-130)^2 + (39.2-28.3)^2 + (0.528-0.26)^2 + (58-42)^2)^{½}$	196.1756402
2	$((11-11)^2 + (143-136)^2 + (94-84)^2 + (33-35)^2 + (146-130)^2 + (36.6-28.3)^2 + (0.254-0.26)^2 + (51-42)^2)^{½}$	23.64085523
3	$((0-11)^2 + (100-136)^2 + (88-84)^2 + (60-35)^2 + (110-130)^2 + (46.8-28.3)^2 + (0.962-0.26)^2 + (31-42)^2)^{½}$	54.05314796
4	$((1-11)^2 + (81-136)^2 + (72-84)^2 + (18-35)^2 + (40-130)^2 + (26.6-28.3)^2 + (0.283-0.26)^2 + (24-42)^2)^{½}$	109.4755248
...	...+ ... + ... + ...+ ...+ ...+ ... + ...	...
334	$((1-11)^2 + (95-136)^2 + (74-84)^2 + (21-35)^2 + (73-130)^2 + (25.9-28.3)^2 + (0.674-0.26)^2 + (36-42)^2)^{½}$	73.26616251

c. Mengurutkan objek dari mulai jarak terdekat sampai jarak terjauh

Tabel 4. 7 Urutan jarak *euclidean* data (uji sempel positif)

Pregnancies	Glucose	BloodPressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome	Jarak	Urutan Ke
10	129	76	28	122	35.9	0.28	39	0	17.1394399	1
9	145	80	46	130	37.9	0.637	40	1	17.84102377	2
7	150	78	29	126	35.2	0.692	54	1	22.17648809	3
11	138	74	26	144	36.1	0.557	50	1	22.49284795	4
3	124	80	33	130	33.2	0.305	26	0	22.5391221	5
11	143	94	33	146	36.6	0.254	51	1	23.64085523	6
1	149	68	29	127	29.3	0.349	42	1	23.89577203	7
13	153	88	37	140	40.6	1.174	39	0	23.96091392	8
6	134	70	23	130	35.4	0.542	29	1	24.25880302	9
4	154	72	29	126	31.3	0.338	37	0	24.5561822	10
...	...	...	...	...	...	...	...	...	...	
1	189	60	23	846	30.1	0.398	59	1	718.733093	344

d. Mengumpulkan label class Y (klasifikasi *nearest neighbor*).

Berdasarkan urutan jarak *euclidean* pada **Tabel 4. 7** dapat kita lihat 9

jarak terdekat dari data uji yang ada. Berdasarkan hasil tersebut dapat kita hitung jumlah prediksi positif (*outcome* 1) sebanyak 6 data dan prediksi negatif (*outcome* 0) sebanyak 3 data.

e. Melakukan prediksi berdasarkan label class Y yang paling mayoritas. Label class Y paling mayoritas dari hasil uji tersebut adalah 1 atau positif, jadi dapat disimpulkan bahwa data yang baru saja diinputkan adalah **positif *diabetes***.

4.5. Hasil Uji Coba

Proses uji coba dilakukan pada data uji dengan menggunakan nilai k paling optimal. Data uji yang digunakan berjumlah 50 data. 50 data tersebut kemudian akan diberikan variabel nilainya dengan menggunakan confusion matrix, label nilai yang akan digunakan ada 4 yaitu *True Positive* (TP) atau Benar Positif, *True Negative* (TN) atau Benar Negatif, *False Positive* (FP) atau Salah Positif, *False Negative* (FN) atau Salah Negatif. Variabel nilai yang didapatkan kemudian akan digunakan untuk menghitung tingkat akurasi dari sistem yang dibuat.

Segmen Program 4. 14 Confusion Matriks

```
cm = confusion_matrix(y_test, y_pred)
print('===Hasil Confusion Matriks===')
cm = pd.DataFrame(cm)
print(cm)
```

```
===Hasil Confusion Matriks===
    0  1
0  34  1
1   6  9
```

Gambar 4. 14 Hasil Confusion Matriks

Berdasarkan pengujian diatas maka didapat :

True Positive (TP) atau Benar Positif = 34 data

True Negative (TN) atau Benar Negatif = 9 data

False Positive (FP) atau Salah Positif = 6 data

False Negative (FN) atau Salah Negatif = 1 data

Sehingga kemudian dapat dihitung tingkat akurasi dari sistem yang dibuat dengan rumus akurasi : $accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{34+9}{34+9+6+1} = \frac{43}{50} = 0.86$