



LAPORAN UJIAN AKHIR SEMESTER

PENERAPAN MODEL DECISION TREE MENGGUNAKAN PYTHON UNTUK MENDETEKSI ADANYA PENYAKIT DIABETES

Laporan Ini Disusun Guna Memenuhi Salah Satu Syarat Pada Mata Kuliah
Data Mining Pada Program Studi Sarjana Teknik Informatika Fakultas Ilmu
Komputer Universitas Dian Nuswantoro

Disusun Oleh:

Muhammad Rizky Setiawan (A11.2022.14793)

Noor Adekah Apriyana (A11.2022.14382)

**JURUSAN TEKNIK INFORMATIKA-S1
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO
2024**

BAB 1

DATASET

1.1 Penjelasan Dataset yang Digunakan

Dataset yang kami gunakan merupakan gabungan dari dua buah data public yang kami ambil dari website *Kaggle.com* yang berisikan data pasien terjangkit diabetes. Pada data pasien terjangkit diabetes tersebut terdapat fitur-fitur yang berisi umur pasien, kadar gula darah, body mass index (bmi), serta kepastian apakah pasien positif atau negatif terkena diabetes.

BAB 2

PERMASALAHAN

DAN TUJUAN EKSPERIMEN

2.1 Permasalahan

1. Bagaimana cara memprediksi pasien terjangkit diabetes
2. Bagaimana memperoleh akurasi yang tinggi dalam memprediksi pasien terjangkit diabetes

2.2 Tujuan Eksperimen

1. Memperoleh hasil prediksi pasien terjangkit diabetes
2. Menghasilkan akurasi yang tinggi dalam memprediksi pasien terjangkit diabetes

BAB 3

MODEL DAN ALUR

TAHAPAN EKSPERIMEN

3.1 Model dan Alur Tahapan Eksperimen

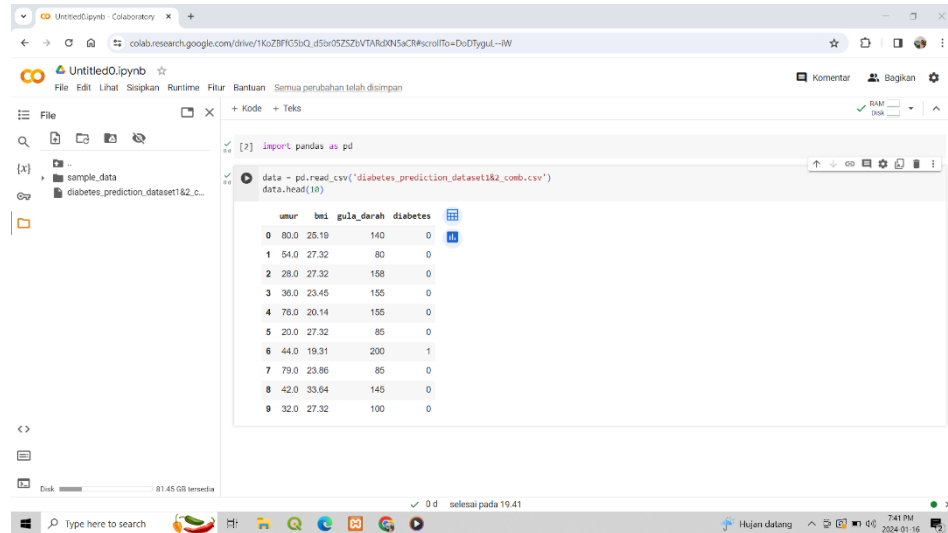
Algoritma yang kami gunakan dalam memprediksi pasien terjangkit diabetes yaitu decision tree, di mana data-data pasien yang terjangkit maupun tidak, akan diterjemahkan menjadi struktur pohon yang memiliki cabang-cabang keputusan dan daun-daun sebagai hasil prediksi atau klasifikasi.

BAB 4

PERFORMA MODEL

4.1 Uji Performa Model

a. Import library pandas, kemudian masukkan dataset dengan format csv

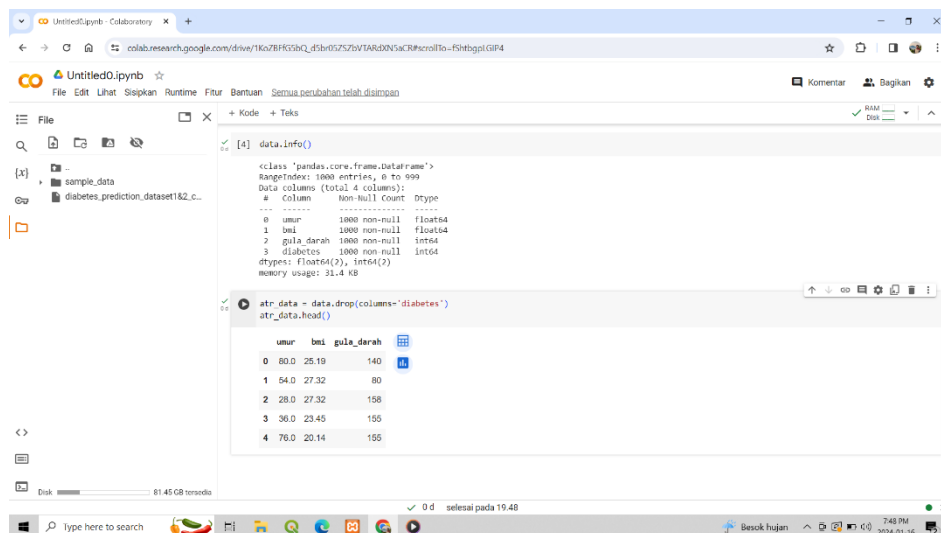


The screenshot shows a Google Colab notebook with the following code and output:

```
[2]: import pandas as pd
data = pd.read_csv('diabetes_prediction_dataset1&2_comb.csv')
data.head(10)
```

	umur	bmi	gula_darah	diabetes
0	80.0	25.19	140	0
1	54.0	27.32	80	0
2	28.0	27.32	158	0
3	36.0	23.45	155	0
4	76.0	20.14	155	0
5	20.0	27.32	85	0
6	44.0	19.31	200	1
7	79.0	23.86	85	0
8	42.0	33.64	145	0
9	32.0	27.32	100	0

b. Tentukan atribut, di sini kami melakukan drop pada kolom “diabetes”



The screenshot shows a Google Colab notebook with the following code and output:

```
[4]: data.info()
atr_data = data.drop(columns='diabetes')
atr_data.head()
```

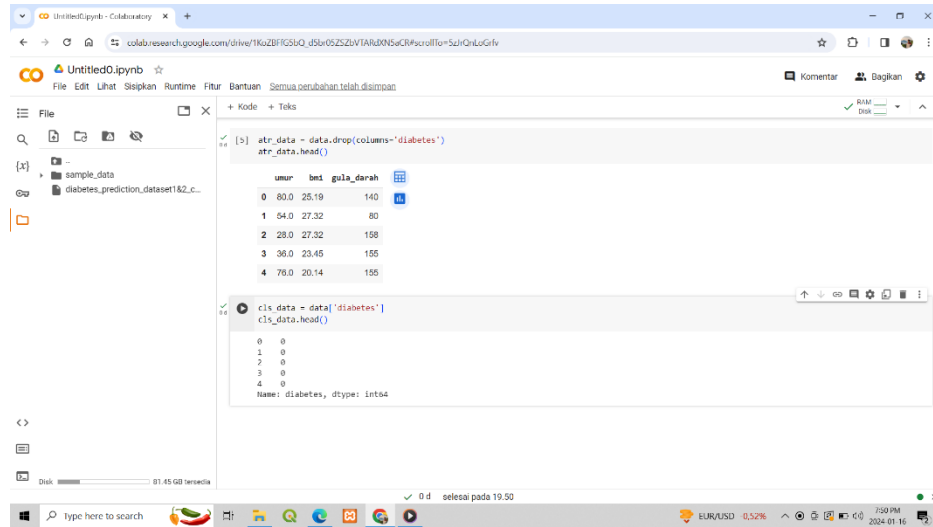
The output of `data.info()` is:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   umur       1000 non-null   float64 
 1   bmi        1000 non-null   float64 
 2   gula_darah 1000 non-null   int64   
 3   diabetes   1000 non-null   int64   
dtypes: float64(2), int64(2)
memory usage: 31.4 kb
```

The output of `atr_data.head()` is:

	umur	bmi	gula_darah
0	80.0	25.19	140
1	54.0	27.32	80
2	28.0	27.32	158
3	36.0	23.45	155
4	76.0	20.14	155

c. Buat variabel baru untuk mendefinisikan target atribut, yaitu kolom “diabetes”



The screenshot shows a Google Colab notebook with two code cells. The first cell removes the 'diabetes' column from the dataset and displays the first five rows of the remaining data. The second cell creates a new 'diabetes' target variable from the original dataset and displays its first five rows.

```
[5] atr_data = data.drop(columns='diabetes')
atr_data.head()
```

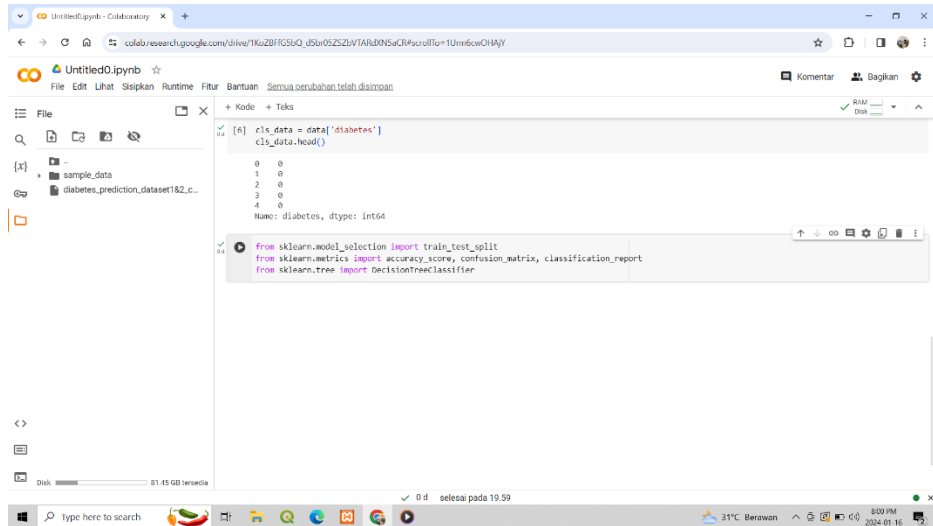
	umur	bmi	gula_darah
0	80.0	25.19	140
1	84.0	27.32	80
2	28.0	27.32	198
3	36.0	23.45	155
4	76.0	20.14	155

```
[6] cls_data = data['diabetes']
cls_data.head()
```

	diabetes
0	0
1	0
2	0
3	0
4	0

Name: diabetes, dtype: int64

d. Panggil library sklearn untuk membagi data "train_test_split", kemudian hitung akurasi skor dengan "accuracy_score", lalu gunakan algoritma "DecisionTreeClassifier".



The screenshot shows a Google Colab notebook with two code cells. The first cell imports the 'diabetes' column from the dataset and displays its first five rows. The second cell imports the necessary sklearn libraries for model selection, metrics, and decision tree classification.

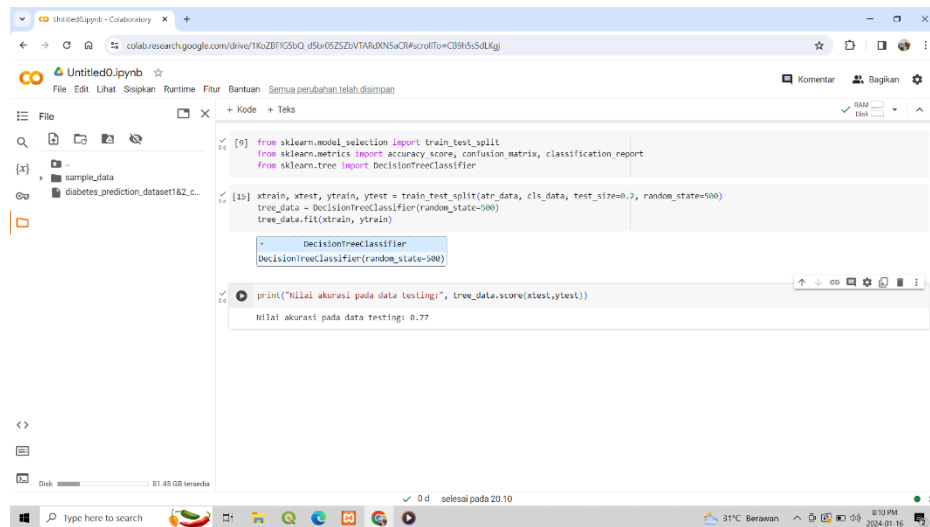
```
[6] cls_data = data['diabetes']
cls_data.head()
```

	diabetes
0	0
1	0
2	0
3	0
4	0

Name: diabetes, dtype: int64

```
[7] from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
```

- e. Lakukan test split pada dataset yang telah kami masukkan, lalu kami tentukan data testingnya sebanyak 20% dengan 500 data diambil secara random. maka didapatkan nilai akurasi pada data testing yaitu sebesar 0.77



```
[9] from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier

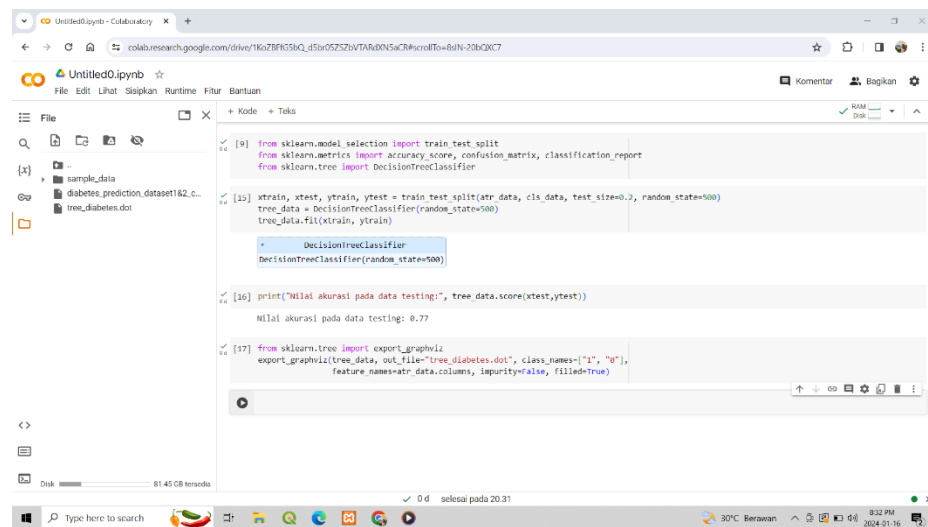
[15] xtrain, xtest, ytrain, ytest = train_test_split(atr_data, cls_data, test_size=0.2, random_state=500)
tree_data = DecisionTreeClassifier(random_state=500)
tree_data.fit(xtrain, ytrain)

DecisionTreeClassifier
DecisionTreeClassifier(random_state=500)

[16] print("Nilai akurasi pada data testing:", tree_data.score(xtest,ytest))

Nilai akurasi pada data testing: 0.77
```

- f. Lakukan import library sklearn.tree kemudian export dan tentukan nama file dalam format .dot



```
[9] from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier

[15] xtrain, xtest, ytrain, ytest = train_test_split(atr_data, cls_data, test_size=0.2, random_state=500)
tree_data = DecisionTreeClassifier(random_state=500)
tree_data.fit(xtrain, ytrain)

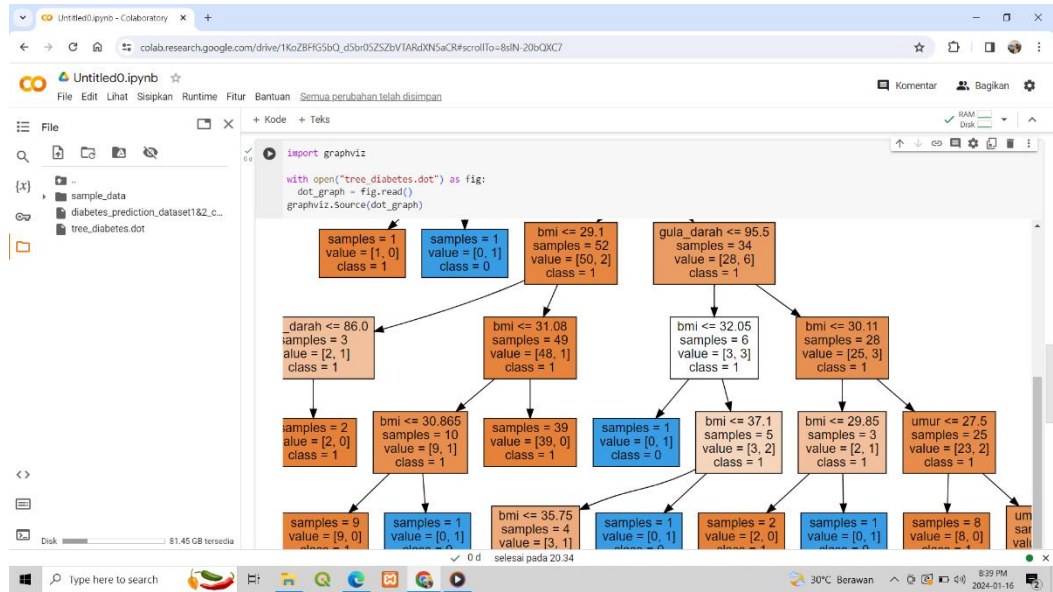
DecisionTreeClassifier
DecisionTreeClassifier(random_state=500)

[16] print("Nilai akurasi pada data testing:", tree_data.score(xtest,ytest))

Nilai akurasi pada data testing: 0.77

[17] from sklearn.tree import export_graphviz
export_graphviz(tree_data, out_file="tree_diabetes.dot", class_names=["1", "0"],
feature_names=atr_data.columns, impurity=False, filled=True)
```

g. Langkah terakhir yaitu import graphviz lalu buka (open) file "tree_diabetes.dot" yang tadi telah diexport



BAB 5

KESIMPULAN DAN REKOMENDASI

5.1 Kesimpulan

Setelah dilakukan pengujian dengan memasukkan data testing sebanyak 20% dengan percobaan pertama didapatkan akurasi sebesar 0.77 dari 500 data yang diambil secara random, dan percobaan ke dua didapatkan akurasi sebesar 0.74 dari 1000 data yang diambil secara random. Yang artinya apabila data diperbanyak sebanyak 2 kali, maka akan mengalami penurunan sebanyak 3%. Dengan demikian dapat disimpulkan bahwa semakin banyak data yang diuji maka akan semakin sulit untuk mendapatkan tingkat akurasi yang kuat.