

Digital Fair Skill 33.0

CLASSIFICATION OF DIABETES DATASET

Presented by: Andhini Ramadhani Helwan | 2025

DIABETES CLASSIFICATION

Studi Kasus ini berasal dari data kaggle yang memiliki kumpulan studi kasus berasal dari Institut Nasional Diabetes, Pencernaan, dan Penyakit Ginjal. Beberapa Parameter pendukung analisis studi kasus ini, antara lain **Glukosa**: Konsentrasi glukosa plasma 2 jam dalam tes toleransi glukosa oral, **Tekanan Darah**: Tekanan darah diastolik (mm Hg), **Ketebalan Kulit**: Ketebalan lipatan kulit trisep (mm), **Insulin**: Insulin serum 2 jam (mu U/ml), **BMI**: Indeks massa tubuh (berat dalam kg/(tinggi dalam m)²), Fungsi **Pedigree Diabetes**: Fungsi silsilah diabetes, **Usia**: Usia (tahun), **Hasil**: Variabel kelas (0 atau 1).

<https://www.kaggle.com/code/asemmustafa/diabetes-classification-three-models/output>

Import Library

```
[ ] import pandas as pd
data1 = pd.read_csv('/diabetes.csv')
data1
```



	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

```
[ ] from sklearn import datasets
```

```
▶ #memuat dataset diabetes dari kaggle dan mengonversinya menjadi DataFrame
diabetes = datasets.load_digits()
```

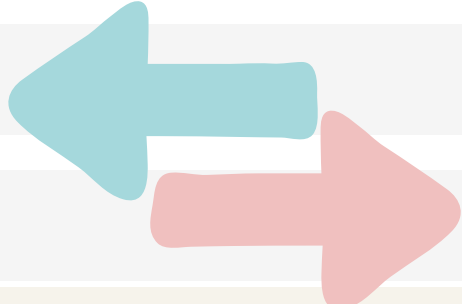
```
[ ] X = diabetes.data #inputan untuk machine learning
    y = diabetes.target #output yang diinginkan dari machine learning
```

```
[ ] # Mengonversi data fitur dan target menjadi DataFrame
df_X = pd.DataFrame(X, columns=diabetes.feature_names)
df_y = pd.Series(y, name='diabetes')
```

```
[ ] # Gabungkan fitur dan target dalam satu DataFrame
df = pd.concat([df_X, df_y], axis=1)
```

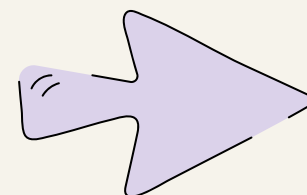
```
[ ] data =pd.read_csv('/diabetes.csv')
```

```
[ ] data.head(10)
```



	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

next



`data.info()`



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Pregnancies           768 non-null   int64
1   Glucose               768 non-null   int64
2   BloodPressure         768 non-null   int64
3   SkinThickness         768 non-null   int64
4   Insulin               768 non-null   int64
5   BMI                  768 non-null   float64
6   DiabetesPedigreeFunction 768 non-null   float64
7   Age                  768 non-null   int64
8   Outcome              768 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[]

`data.shape`



`(768, 9)`



`data.isnull().sum()`



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

`dtype: int64`

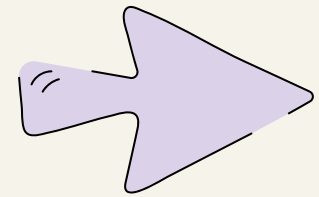
```
[ ]
```

```
data.describe()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

next



3

Split Data



```
from sklearn.model_selection import train_test_split

#membagi data menjadi train dan test
X_train, X_test, y_train, y_test=train_test_split(df_X, df_y, test_size=0.2, random_state=42)
```

4

Train the Model

```
#membuat dan melatih model Decision Tree
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```



DecisionTreeClassifier ⓘ ⓘ
DecisionTreeClassifier(random_state=42)

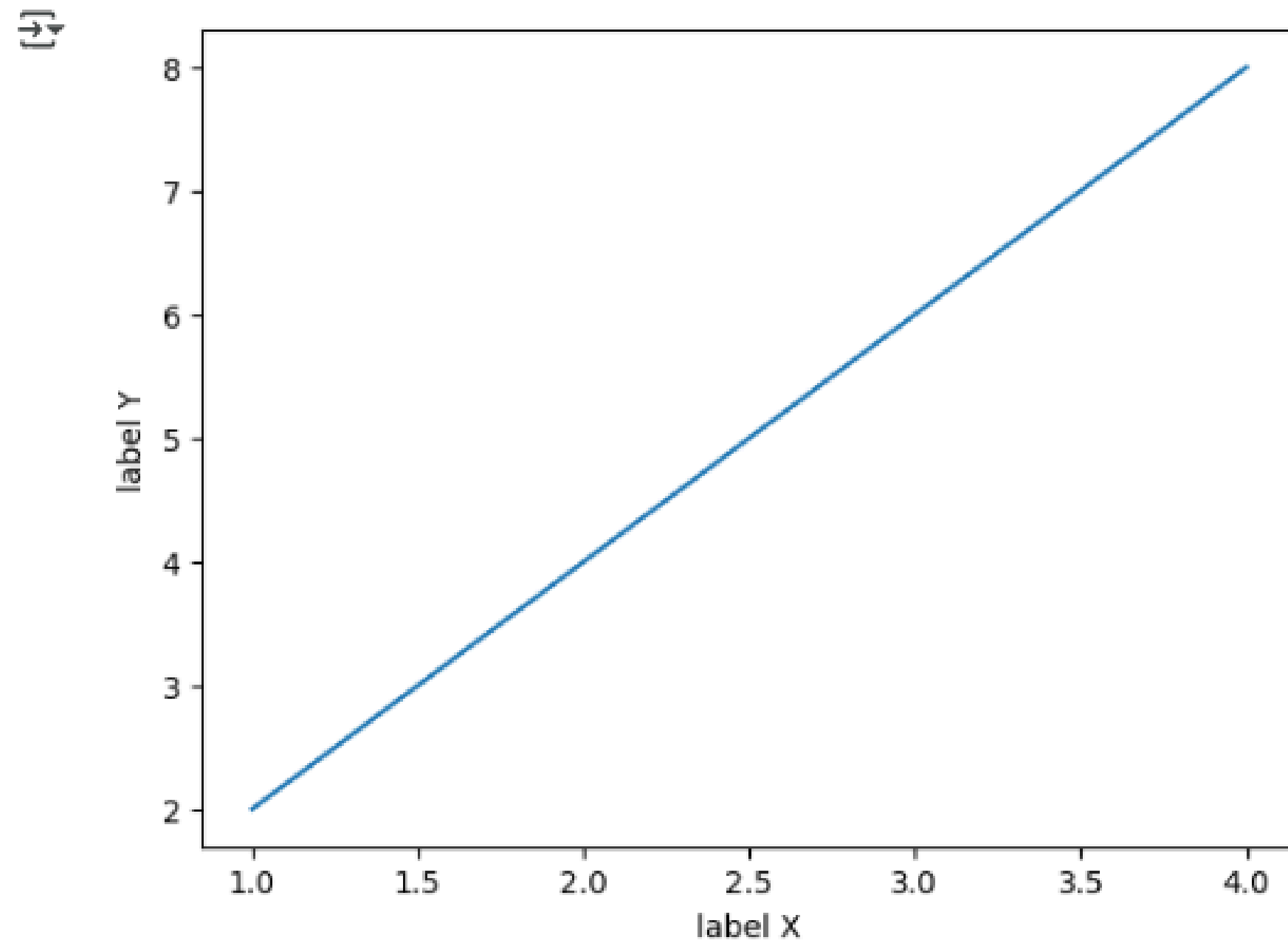
```
[ ] from sklearn.metrics import accuracy_score
```

```
▶ # 4. Memprediksi dan mengevaluasi  
y_pred = model.predict(_X_test)  
  
accuracy = accuracy_score(y_test, y_pred)  
  
print("laporan klasifikasi:")  
print(f"akurasi: {accuracy * 100:2f}%")
```

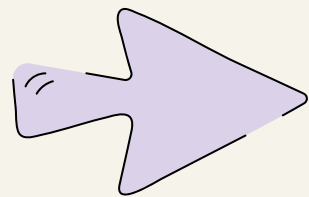
```
↪ laporan klasifikasi:  
akurasi: 84.166667%
```



```
plt.plot([1,2,3,4], [2,4,6,8])
plt.ylabel('label Y')
plt.xlabel('label X')
plt.show()
```

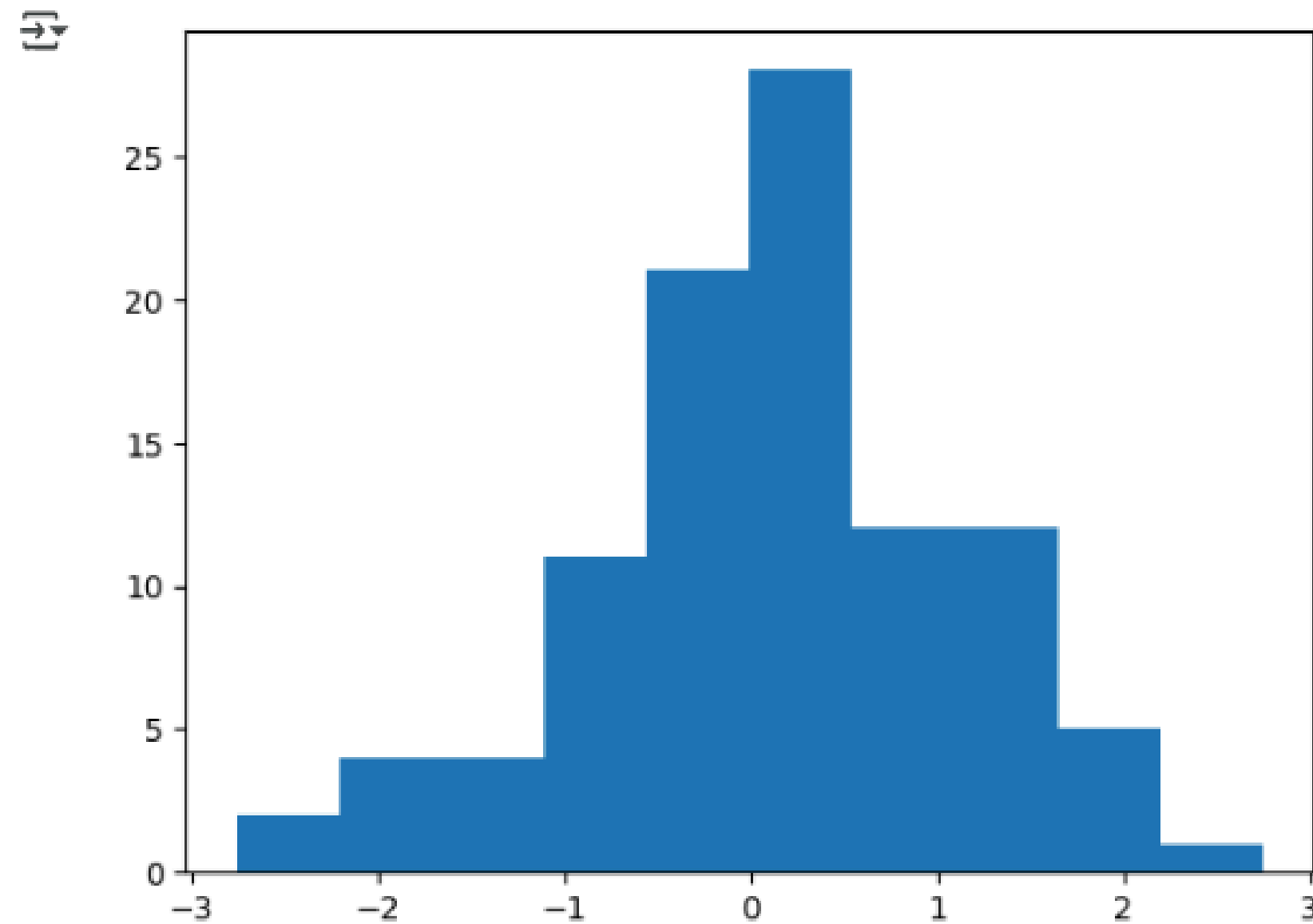


next

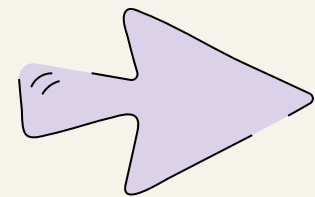


```
import numpy as np
import matplotlib.pyplot as plt
```

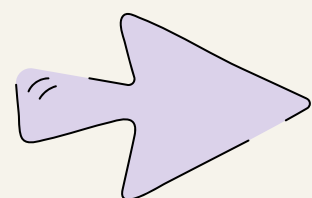
```
x = np.random.normal(size=100)
plt.hist(x, bins=10)
plt.show()
```



next



next



```
[ ] import pandas as pd
```

```
[ ] import numpy as np
```

```
rs = np.random.RandomState(0)
df = pd.DataFrame(rs.rand(10, 10))
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
# 'RdBu_r', 'BrBG_r', & PuOr_r are other good diverging colormaps
```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.347533	0.398948	0.455743	0.072914	-0.233402	-0.731222	0.477978	-0.442621	0.015185
1	0.347533	1.000000	-0.284056	0.571003	-0.285483	0.382480	-0.362842	0.642578	0.252556	0.190047
2	0.398948	-0.284056	1.000000	-0.523649	0.152937	-0.139176	-0.092895	0.016266	-0.434016	-0.383585
3	0.455743	0.571003	-0.523649	1.000000	-0.225343	-0.227577	-0.481548	0.473286	0.279258	0.446650
4	0.072914	-0.285483	0.152937	-0.225343	1.000000	-0.104438	-0.147477	-0.523283	-0.614603	-0.189916
5	-0.233402	0.382480	-0.139176	-0.227577	-0.104438	1.000000	-0.030252	0.417640	0.205851	0.095084
6	-0.731222	-0.362842	-0.092895	-0.481548	-0.147477	-0.030252	1.000000	-0.494440	0.381407	-0.353652
7	0.477978	0.642578	0.016266	0.473286	-0.523283	0.417640	-0.494440	1.000000	0.375873	0.417863
8	-0.442621	0.252556	-0.434016	0.279258	-0.614603	0.205851	0.381407	0.375873	1.000000	0.150421
9	0.015185	0.190047	-0.383585	0.446650	-0.189916	0.095084	-0.353652	0.417863	0.150421	1.000000



THANK YOU