



PERBANDINGAN METODE KLASIFIKASI UNTUK MENDETEKSI PENYAKIT DIABETES



KELOMPOK 3

1. ABDUL WAHHAB (2206028485)
2. HASTHABRATA CHRISTOPHER LIATNA (2206824741)
3. KEZIA ASTARIZA (2206048505)
4. RAISSA ANGGIA MAHARANI (2206048581)
5. SITI NUR SALAMAH (2206048833)



UNIVERSITAS
INDONESIA

ANGGOTA KELOMPOK 3



Siti Nur Salamah
2206048833



Hasthabrata Christopher L.
2206824741



Kezia Astariza
2206048505



Abdul Wahhab
2206028485



Raissa Anggia Maharani
2206048581

TABLE OF CONTENT

01 Pendahuluan

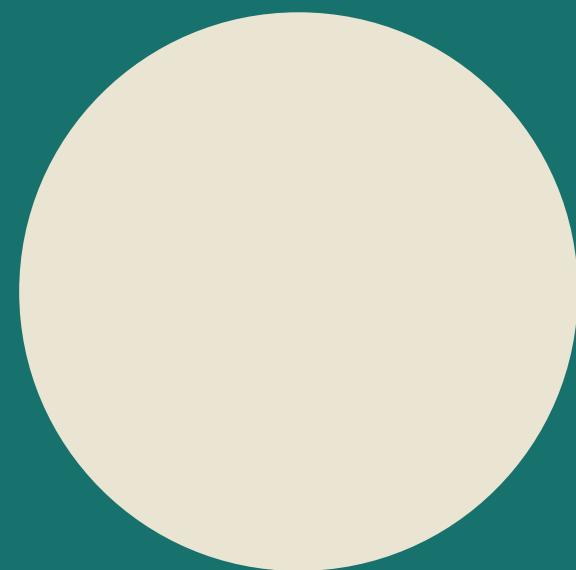
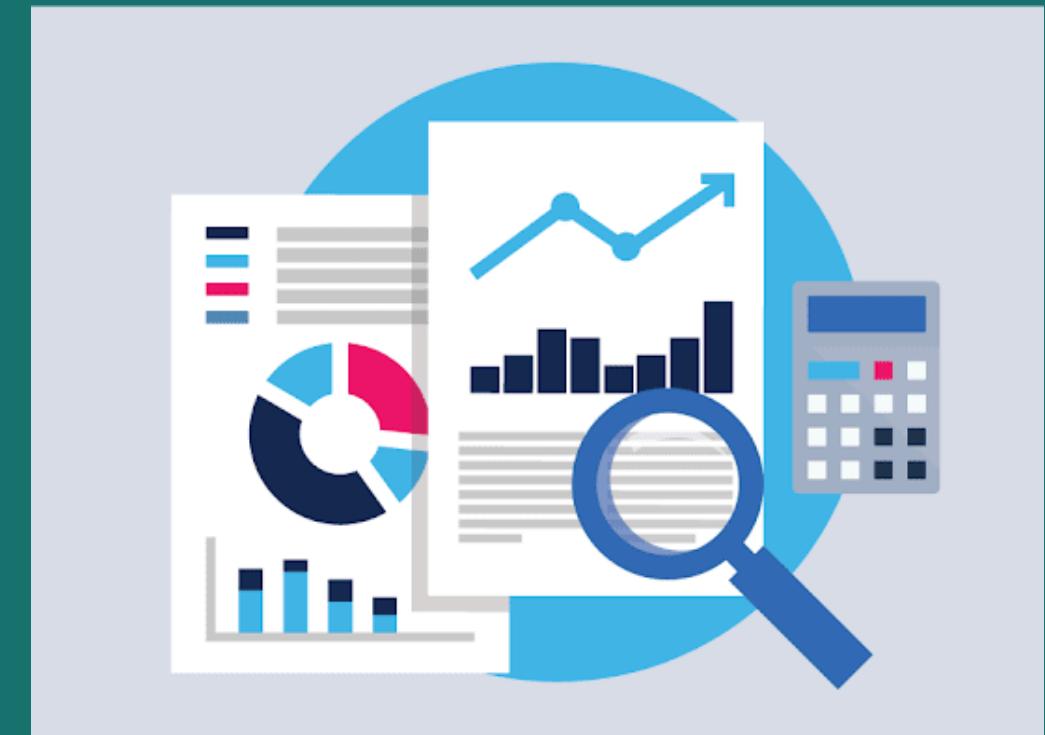
02 Data dan Metode

03 Analisis Data

04 Kesimpulan

05 Saran

06 Daftar Pustaka



PENDAHULUAN

DIABETES



Diabetes adalah salah satu masalah kesehatan paling umum di dunia. Diabetes juga dikenal sebagai "*the silent killer*" karena menurut WHO (2016), jumlah penderita diabetes meningkat dari 108 juta pada tahun 1980 menjadi sekitar 422 juta pada tahun 2014.

Diabetes terjadi ketika tubuh pengidapnya tidak lagi mampu mengambil gula (glukosa) ke dalam sel dan menggunakannya sebagai energi. Seorang yang terindikasi diabetes biasanya memiliki beberapa fitur yang menyebabkan orang tersebut terkena diabetes.

DATA DAN METODE

PENGENALAN DATA

“Diabetes Dataset”

1. SUBJEK DATA YANG DIPAKAI ADALAH :

- Wanita
- Minimal 21 Tahun
- Berketurunan Pima Indian

2. MENGASUMSIKAN BAHWA TIPE DIABETES YANG DICATAT IALAH TIPE 1 DAN TIPE 2

3. DATA DIAMBIL PADA TAHUN 1990



Orang-orang Pima Indian

PENGENALAN DATA

```
[ ] 1 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

```
[ ] 1 # (row, columns)  
2 df.shape
```

→ (768, 9)

Didapat bahwa dataset yang dipakai memiliki 768 baris dan 9 kolom.

FITUR PADA DATASET

Pregnancies

- Berapa kali seseorang pernah hamil.

Glucose

- Konsentrasi Glukosa dengan menggunakan 2 Hour test OGTT (mg/dL)
- Mengukur seberapa baik tubuh dapat memproses gula dalam jumlah besar.

BloodPressure

- Tekanan darah diastolik (mmHg)

SkinThickness

- Ketebalan lipatan kulit trisep (mm)

Insulin

- Insulin Serum 2 Jam ($\mu U/ml$)
- mengukur jumlah hormon yang memungkinkan sel menyerap glukosa (insulin)



FITUR PADA DATASET

BMI

- Indeks Massa Tubuh

DiabetesPedigreeFunction

- Fungsi yang menilai kemungkinan diabetes berdasarkan riwayat keluarga

Age

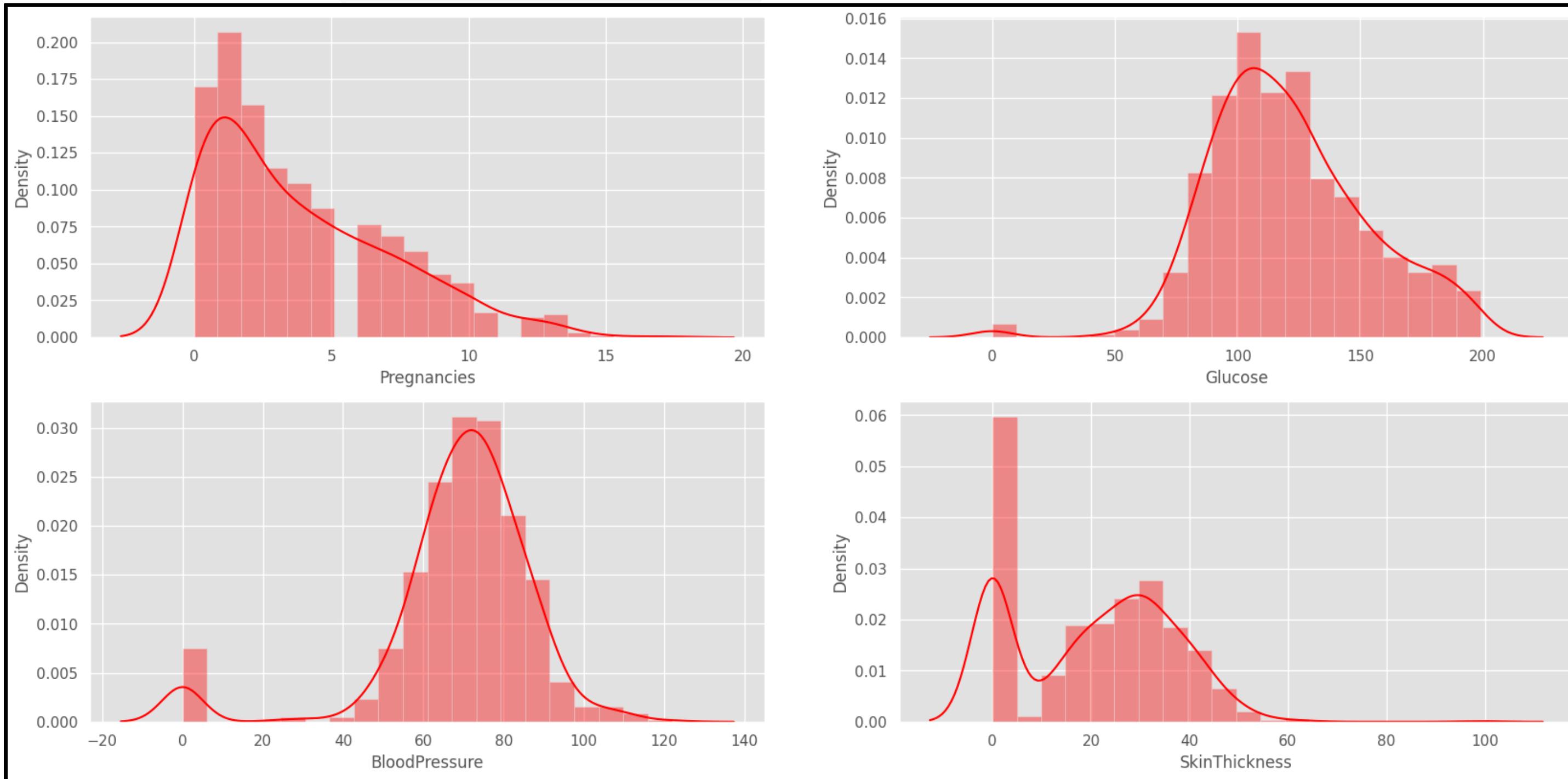
- Umur

Outcome

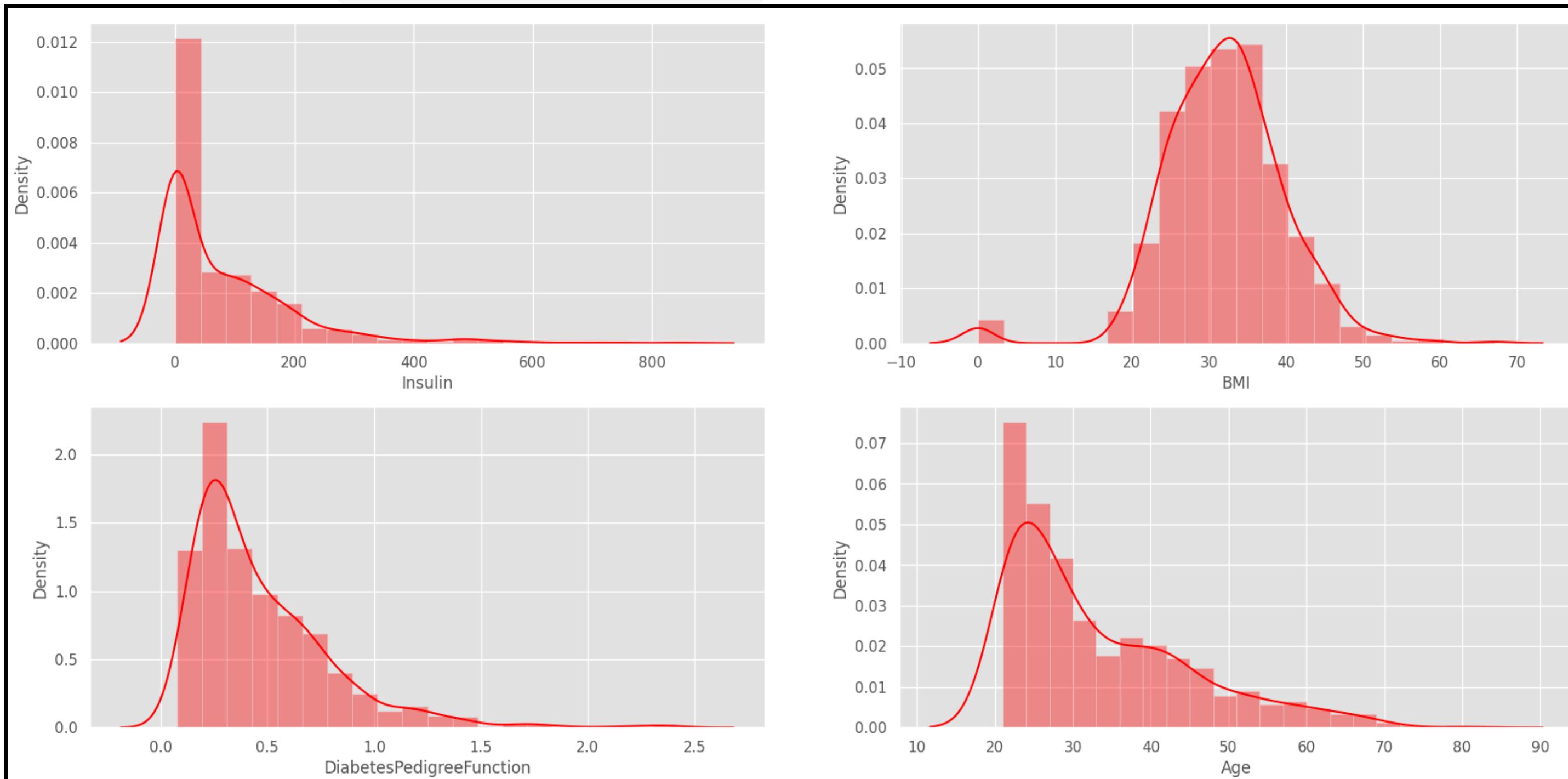
- Jika 0 maka tidak terkena diabetes dan berlaku sebaliknya untuk 1.



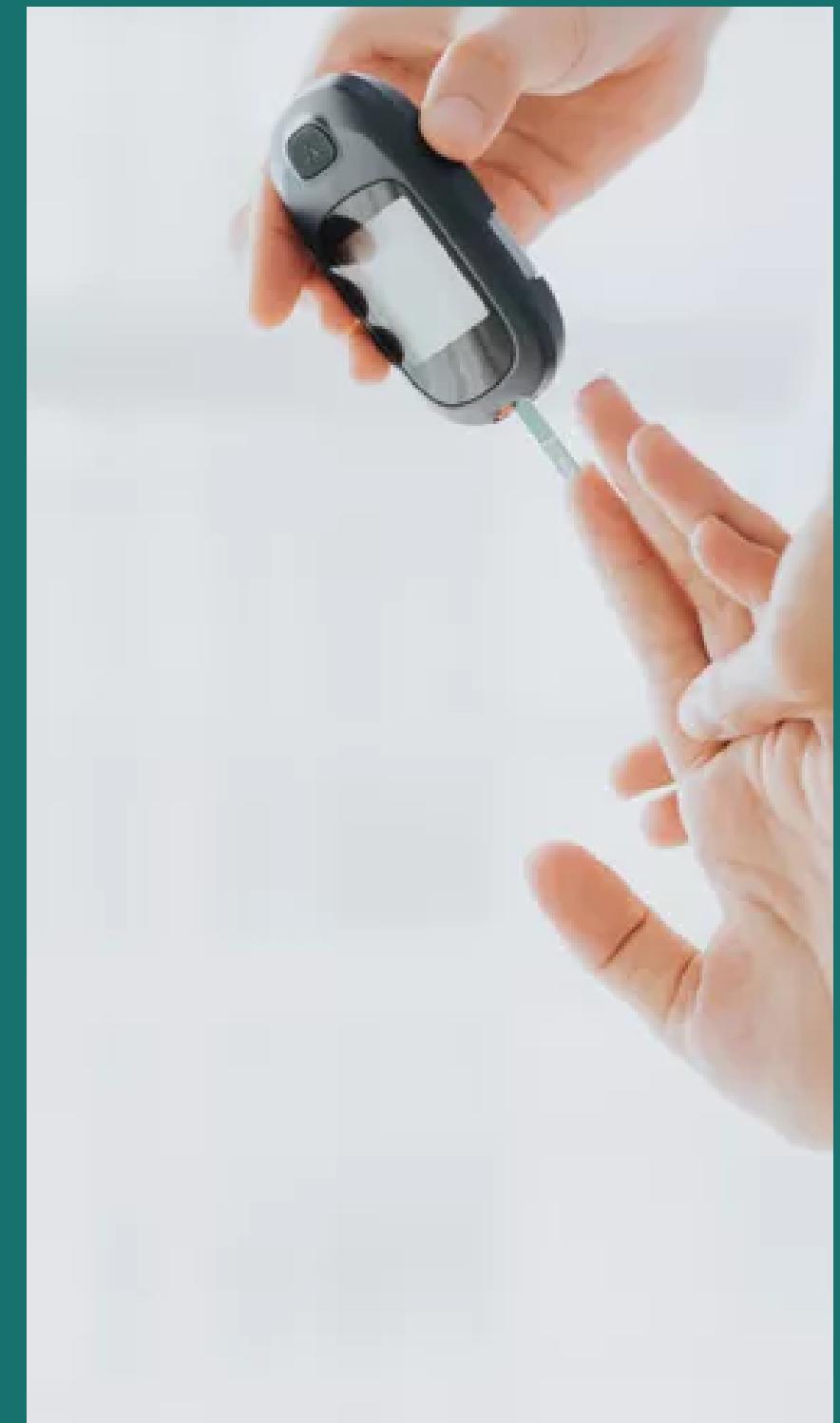
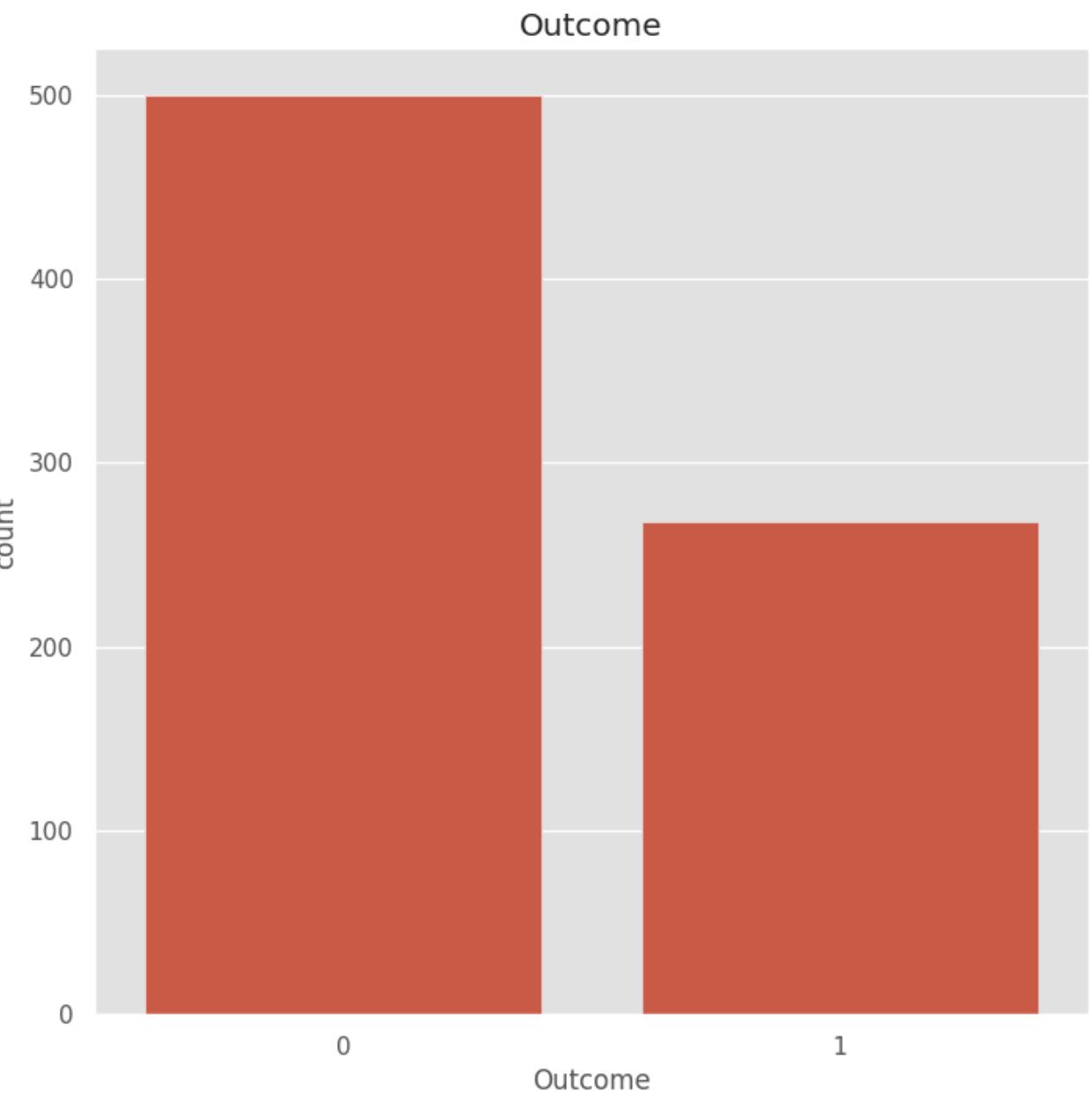
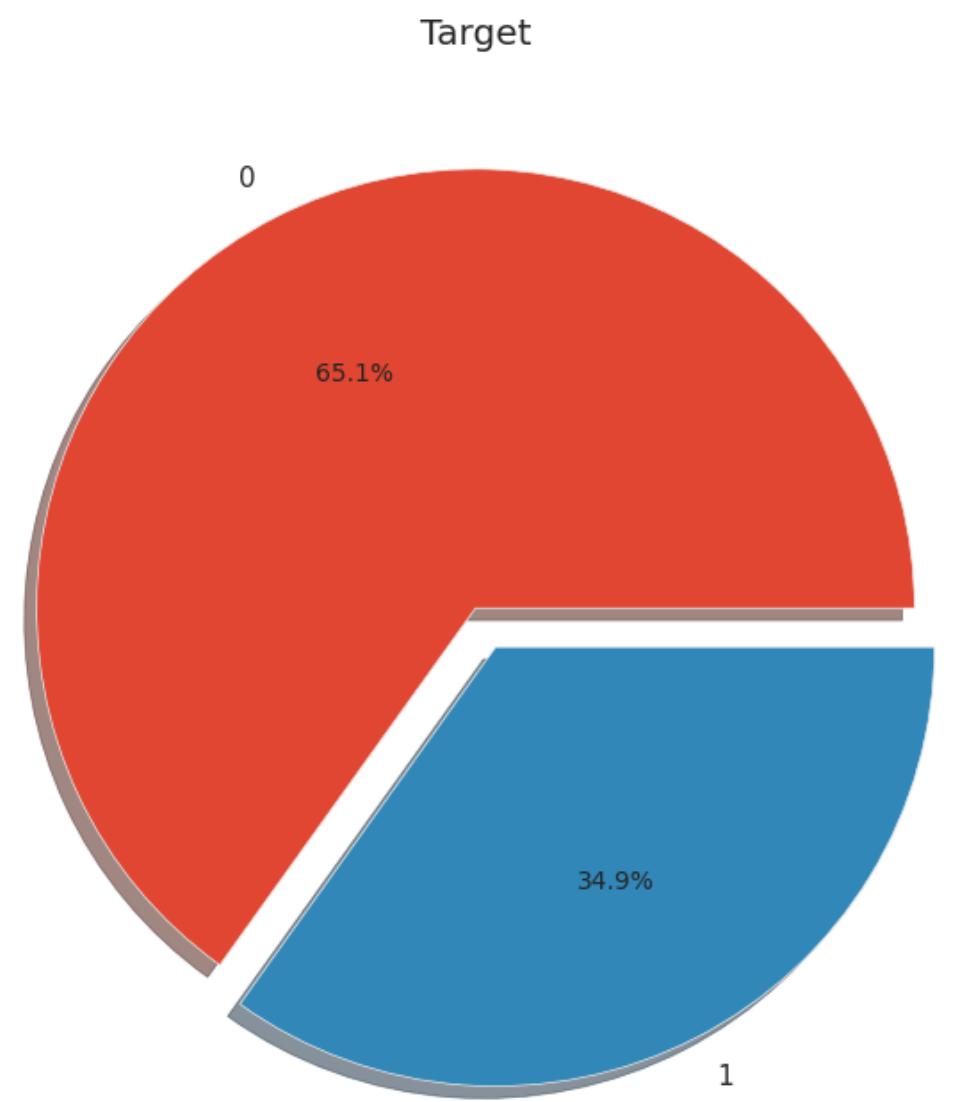
PERSEBARAN DATA TIAP FITUR



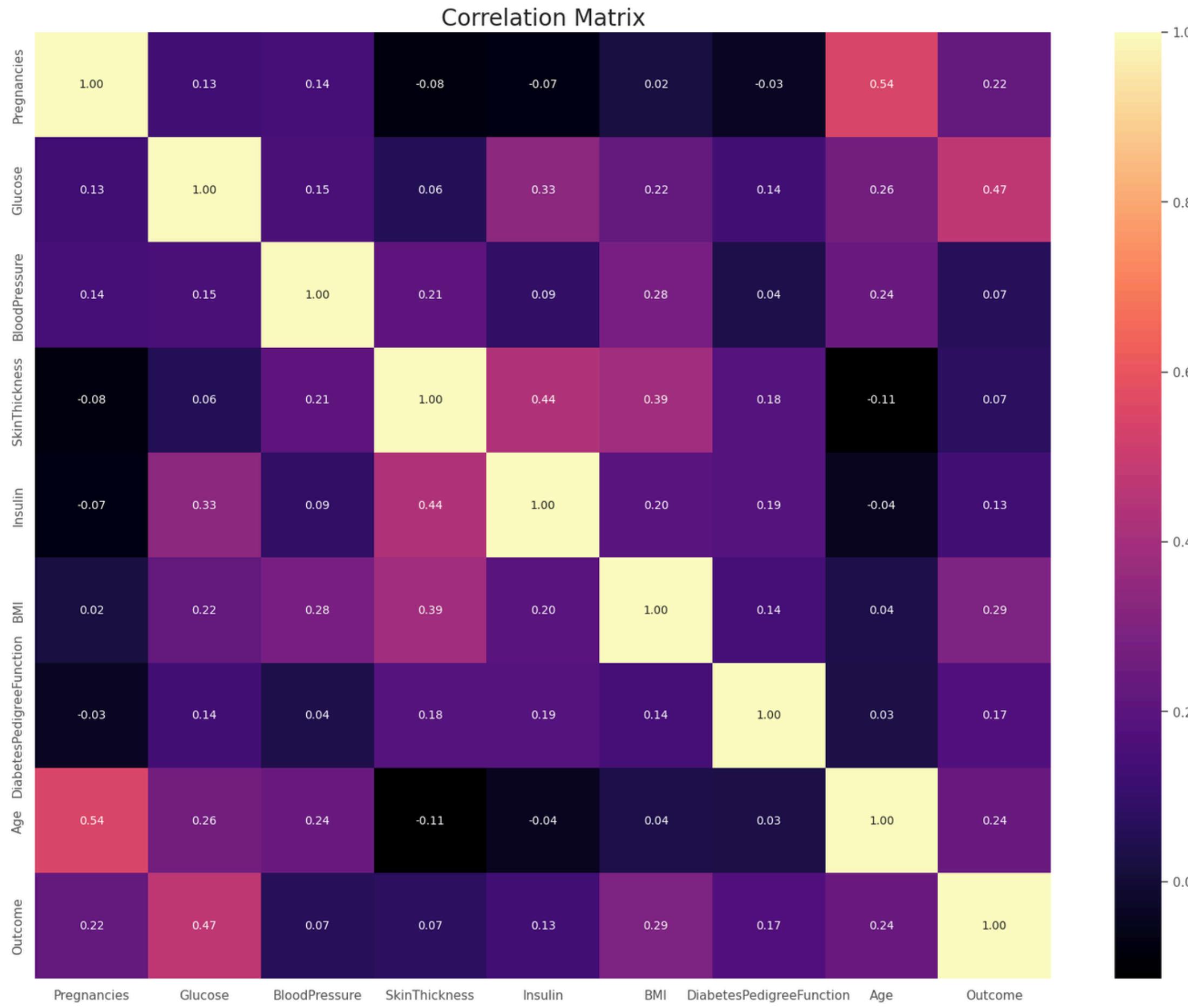
PERSEBARAN DATA TIAP FITUR



PERSENTASE OUTCOME



MATRIX KORELASI



DATASET

BERIKUT DESKRIPSI DATASET YANG DIPAKAI

```
6 df.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

PREPROCESSING DATA



```
6 df.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

Perhatikan bahwa nilai minimum dari dataset diatas ialah 0.

Namun tidak mungkin bagi Glucose, BloodPressure, SkinThickness, Insulin, dan BMI untuk memiliki nilai 0.

PREPROCESSING DATA



Maka dari itu kelompok kami memutuskan untuk :

1. Mengubah nilai abnormal (0) menjadi np.NaN

```
[ ] 1 # Ganti Data yang kosong dengan NaN  
2 df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',  
3      'BMI', 'DiabetesPedigreeFunction', 'Age']] = df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',  
4      'BMI', 'DiabetesPedigreeFunction', 'Age']].replace(0, np.NaN)
```

```
[ ] 1 # Data preprocessing Part  
2 df.isnull().sum()
```

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

2. Mengimputasi nilai NaN dengan median tiap fitur

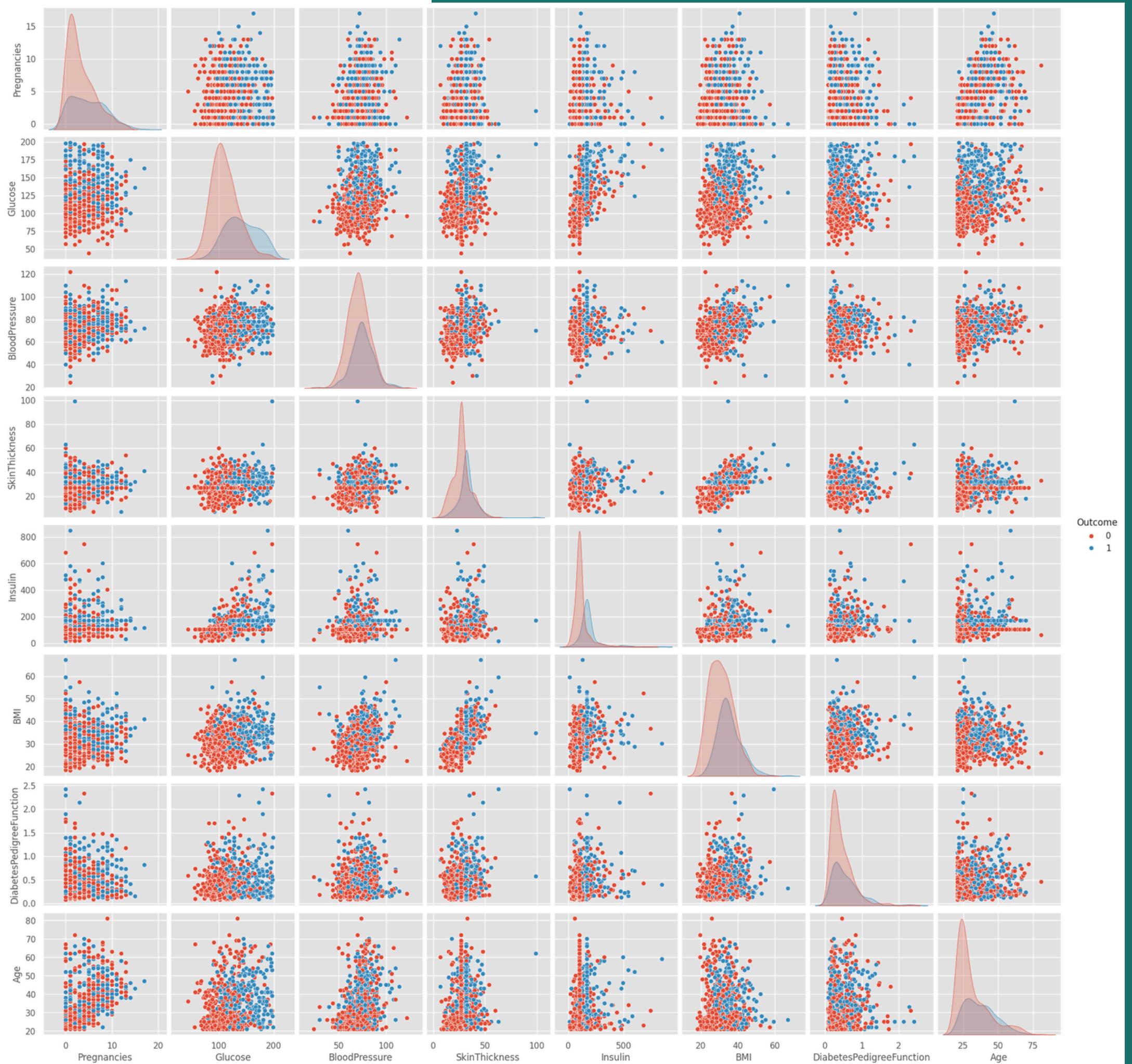
IMPUTASI DATA NULL

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6		0.627	50
1	1	85.0	66.0	29.0	102.5	26.6		0.351	31
2	8	183.0	64.0	32.0	169.5	23.3		0.672	32
3	1	89.0	66.0	23.0	94.0	28.1		0.167	21
4	0	137.0	40.0	35.0	168.0	43.1		2.288	33

```
[ ] # Data yang kosong sudah diisi dengan median  
df.isnull().sum()
```

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

PAIRPLOT



DETEKSI OUTLIER DAN PENANGANAN OUTLIER

Menghitung Q1, Q3, dan IQR dari kolom 'Insulin' di DataFrame df, lalu menetapkan batas bawah dan atas untuk outlier dengan metode IQR, dan mengganti outlier dengan batas atas.

Selanjutnya, kami menggunakan Local Outlier Factor (LOF) dari scikit-learn dengan n_neighbors=10 untuk mengidentifikasi outlier, melakukan fit_predict, menyimpan skor outlier negatif dalam df_scores, mengurutkan skor outlier negatif, dan menetapkan threshold.

Dengan threshold ini, kami membuat mask untuk memilih baris dalam df yang skornya lebih besar dari threshold dan menyaring DataFrame untuk hanya menyisakan baris non-outlier menurut LOF.

```
6 for feature in df:  
7     Q1 = df[feature].quantile(0.25)  
8     Q3 = df[feature].quantile(0.75)  
9     IQR = Q3-Q1  
10    lower = Q1-1.5*IQR  
11    upper = Q3+1.5*IQR  
12    if df[(df[feature]>upper)].any(axis=None):  
13        print(feature, "yes")  
14    else:  
15        print(feature, "no")
```

```
Pregnancies yes  
Glucose no  
BloodPressure yes  
SkinThickness yes  
Insulin yes  
BMI yes  
DiabetesPedigreeFunction yes  
Age yes  
Outcome no
```

FEATURE ENGINEERING

```
# Kategorik untuk BMI
df['NewBMI'] = NewBMI
df.loc[df["BMI"]<18.5, "NewBMI"] = NewBMI[0] # Underweight
df.loc[(df["BMI"]>18.5) & df["BMI"]<=24.9, "NewBMI"] = NewBMI[1] # Normal
df.loc[(df["BMI"]>24.9) & df["BMI"]<=29.9, "NewBMI"] = NewBMI[2] # Overweight
df.loc[(df["BMI"]>29.9) & df["BMI"]<=34.9, "NewBMI"] = NewBMI[3] # Obesity 1
df.loc[(df["BMI"]>34.9) & df["BMI"]<=39.9, "NewBMI"] = NewBMI[4] # Obesity 2
df.loc[df["BMI"]>39.9, "NewBMI"] = NewBMI[5] # Obesity 3
#mengkategorikan setiap nilai BMI dalam DataFrame df ke dalam salah satu dari enam kategori BMI

] df.head()

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome NewBMI
0 6.0 148.0 72.0 35.0 169.5 33.6 0.627 50 1 Obesity 2
1 1.0 85.0 66.0 29.0 102.5 26.6 0.351 31 0 Obesity 2
2 8.0 183.0 64.0 32.0 169.5 23.3 0.672 32 1 Obesity 2
3 1.0 89.0 66.0 23.0 94.0 28.1 0.167 21 0 Obesity 2
4 5.0 137.0 40.0 35.0 168.0 43.1 2.288 33 1 Obesity 3
```

Tujuan

- Menambahkan fitur untuk meningkatkan kinerja model

Seseorang dengan BMI tinggi cenderung memiliki lemak tubuh berlebih, yang bisa menyebabkan resistensi insulin. Resistensi insulin berarti tubuh membutuhkan lebih banyak insulin untuk menurunkan kadar glukosa darah.

Kadar glukosa yang tinggi dalam darah bisa menunjukkan bahwa tubuh tidak mampu mengendalikan gula darah secara efektif, yang merupakan ciri khas dari diabetes.

```

# Kategorik untuk BMI
df['NewBMI'] = NewBMI
df.loc[df["BMI"]<18.5, "NewBMI"] = NewBMI[0] # Underweight
df.loc[(df["BMI"]>18.5) & df["BMI"]<=24.9, "NewBMI"] = NewBMI[1] # Normal
df.loc[(df["BMI"]>24.9) & df["BMI"]<=29.9, "NewBMI"] = NewBMI[2] # Overweight
df.loc[(df["BMI"]>29.9) & df["BMI"]<=34.9, "NewBMI"] = NewBMI[3] # Obesity 1
df.loc[(df["BMI"]>34.9) & df["BMI"]<=39.9, "NewBMI"] = NewBMI[4] # Obesity 2
df.loc[df["BMI"]>39.9, "NewBMI"] = NewBMI[5] # Obesity 3
#mengkategorikan setiap nilai BMI dalam DataFrame df ke dalam salah satu dari enam kategori BMI

```

[] df.head()

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI
0	6.0	148.0	72.0	35.0	169.5	33.6		0.627	50	1 Obesity 2
1	1.0	85.0	66.0	29.0	102.5	26.6		0.351	31	0 Obesity 2
2	8.0	183.0	64.0	32.0	169.5	23.3		0.672	32	1 Obesity 2
3	1.0	89.0	66.0	23.0	94.0	28.1		0.167	21	0 Obesity 2
4	5.0	137.0	40.0	35.0	168.0	43.1		2.288	33	1 Obesity 3

- BMI
- Insulin
- Glukosa

'Normal', 'Obesity 1', 'Obesity 2',
'Obesity 3', 'Overweight'
'Underweight'
["Insulin"]>=16 and row["Insulin"]
<=166:
"Low", "Normal", "Prediabetes", "Diabetes"

NewBMI_Normal	...	NewBMI_Obesity_2	NewBMI_Obesity_3	NewBMI_Overweight	NewBMI_Underweight	NewInsulinScore_Abnormal	NewInsulinScore_Binary
0	...	0	0	0	0	1	
0	...	0	0	1	0	0	
1	...	0	0	0	0	1	
0	...	0	0	1	0	0	
0	...	0	1	0	0	1	

ENCODING

Encoding fitur kategorikal adalah proses mengubah fitur kategorikal (yang terdiri dari nilai non-numerik seperti teks atau label kategori) menjadi format numerik

One-Hot Encoding: Mengonversi kategori menjadi vektor biner. Setiap kategori unik mendapatkan kolom sendiri dengan nilai 1 jika kategori tersebut ada, dan 0 jika tidak ada.

ROBUSTSCALER

RobustScaler adalah salah satu teknik penskalaan fitur yang digunakan dalam pra-pemrosesan data. Ini digunakan untuk menormalkan fitur-fitur data agar memiliki skala yang seragam dan mengurangi dampak dari outlier

BEFORE

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148		72	35	169	33	0 50
1	1	85		66	29	102	26	0 31
2	8	183		64	32	169	23	0 32
3	1	89		66	23	94	28	0 21
4	0	137		40	35	168	43	2 33

AFTER

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.6	0.775	0.000	1.000000	1.000000	0.111111		0.0 1.235294
1	-0.4	-0.800	-0.375	0.142857	0.000000	-0.666667		0.0 0.117647
2	1.0	1.650	-0.500	0.571429	1.000000	-1.000000		0.0 0.176471
3	-0.4	-0.700	-0.375	-0.714286	-0.119403	-0.444444		0.0 -0.470588
4	-0.6	0.500	-2.000	1.000000	0.985075	1.222222		2.0 0.235294



ANALISIS DATA

DATA TRAINING DATA TESTING

```
[ ] 1 # Split Data
2 X_train, X_test, y_train , y_test = train_test_split(X,y, test_size=0.2, random_state=0)
3 X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
→ ((608, 20), (152, 20), (608,), (152,))
```

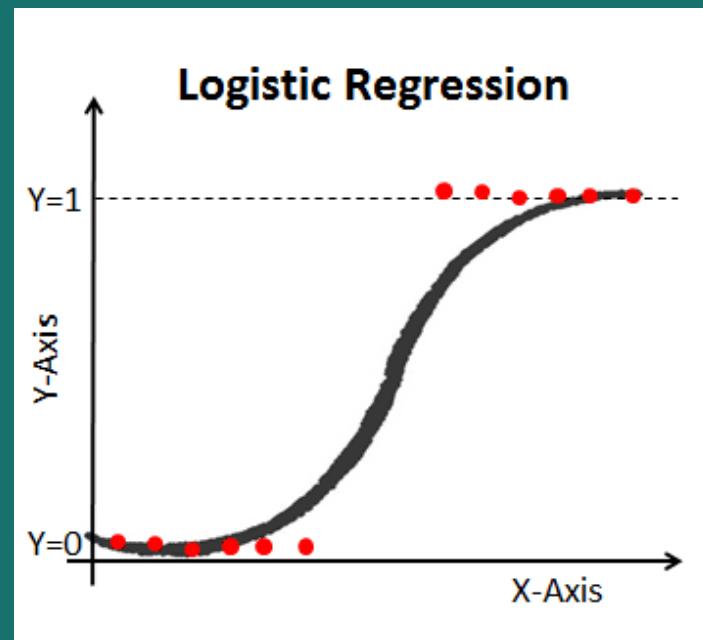
Akan dilakukan *splitting* data hasil *preprocessing* menjadi *data training* dan *data testing* dengan perbandingan 80:20 (80% untuk *training* dan 20% untuk *testing*). Diperoleh *data training* sebanyak 608 sampel dan *data testing* sebanyak 152 sampel.

Akan ditinjau 5 metode :

1. Regresi Logistik
2. Decision Tree
3. Support Vector Machine
4. KNN
5. Random Forest

REGRESI LOGISTIK

Regresi logistik adalah metode statistik yang digunakan untuk memodelkan hubungan antara satu atau lebih variabel independen dengan variabel dependen biner (dua kategori).



sumber: medium.com

- Inisialisasi model regresi logistik: `log_reg = LogisticRegression()`.
- Latih model dengan data pelatihan: `log_reg.fit(X_train, y_train)`.
- Prediksi label target untuk data uji: `y_pred = log_reg.predict(X_test)`.
- Ukur akurasi model pada data pelatihan menggunakan fungsi accuracy_score: `accuracy_score(y_train, log_reg.predict(X_train))`.

MODEL RL (1/2)

```
[ ] # Logistic Regreesion  
log_reg = LogisticRegression()  
log_reg.fit(X_train, y_train)
```

```
[ ]  
+ LogisticRegression  
LogisticRegression()
```

```
[ ] y_pred = log_reg.predict(X_test)
```

```
[ ] accuracy_score(y_train, log_reg.predict(X_train))
```

```
[ ] 0.8453947368421053
```

REGRESI LOGISTIK

Evaluasi Model Regresi Logistik :

- Precision kelas 0: 93% (prediksi kelas 0 benar-benar kelas 0).
- Precision kelas 1: 77% (prediksi kelas 1 benar-benar kelas 1).
- Recall kelas 0: 86%
- Recall kelas 1: 89%
- F1-score kelas 0: 89% (rata-rata harmonik precision dan recall untuk kelas 0).
- F1-score kelas 1: 83% (rata-rata harmonik precision dan recall untuk kelas 1).
- Akurasi keseluruhan model: **87%** (prediksi benar secara keseluruhan).

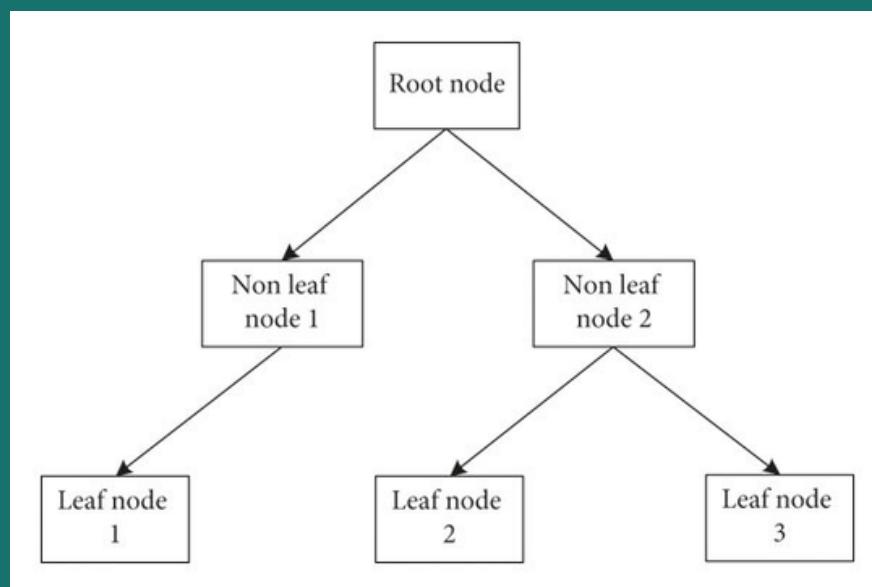
MODEL RL (2/2)

```
[ ] 1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.93	0.86	0.89	98
1	0.77	0.89	0.83	54
accuracy			0.87	152
macro avg	0.85	0.87	0.86	152
weighted avg	0.88	0.87	0.87	152

DECISION TREE

Decision tree adalah algoritma pembelajaran mesin yang menggunakan struktur pohon untuk membuat keputusan berdasarkan fitur data. Setiap node internal mewakili tes pada atribut, setiap cabang mewakili hasil tes, dan setiap daun mewakili prediksi atau klasifikasi.



sumber: researchgate.net

- Inisialisasi model DecisionTreeClassifier dan latih model.
- Tentukan parameter grid untuk hyperparameter tuning.
- Gunakan GridSearchCV untuk mencari hyperparameter terbaik dan gunakan hasil estimator terbaik tersebut.
- Cetak akurasi model pada data pelatihan.
- Hitung dan cetak akurasi model pada data uji.
- Cetak confusion matrix dan classification report.

MODEL DT (1/2)

```
1 # Decision Tree
2 DT = DecisionTreeClassifier()
3 DT.fit(X_train, y_train)
4 y_pred = DT.predict(X_test)
```

```
[ ] 1 # hyperparameter tuning of dt
2 grid_param = {
3     'criterion':['gini','entropy'],
4     'max_depth' : [3,5,7,10],
5     'splitter' : ['best','random'],
6     'min_samples_leaf':[1,2,3,5,7],
7     'min_samples_split':[1,2,3,5,7],
8     'max_features':['auto','sqrt','log2']
9 }
10 grid_search_dt = GridSearchCV(DT, grid_param, cv=50, n_jobs=-1, verbose = 1)
11 grid_search_dt.fit(X_train, y_train)
```

```
→ Fitting 50 folds for each of 1200 candidates, totalling 60000 fits
  GridSearchCV
    estimator: DecisionTreeClassifier
      DecisionTreeClassifier
```

```
[106] 1 grid_search_dt.best_params_ # Best parameter yang digunakan untuk DTC
```

```
→ {'criterion': 'gini',
'max_depth': 5,
'max_features': 'sqrt',
'min_samples_leaf': 3,
'min_samples_split': 7,
'splitter': 'best'}
```

DECISION TREE

Evaluasi Model Decision Tree :

- Precision kelas 0: 92% (prediksi kelas 0 benar-benar kelas 0).
- Precision kelas 1: 87% (prediksi kelas 1 benar-benar kelas 1).
- Recall kelas 0: 93%
- Recall kelas 1: 85%
- F1-score kelas 0: 92% (rata-rata harmonik precision dan recall untuk kelas 0).
- F1-score kelas 1: 86% (rata-rata harmonik precision dan recall untuk kelas 1).
- Akurasi keseluruhan model: **90%** (prediksi benar secara keseluruhan).

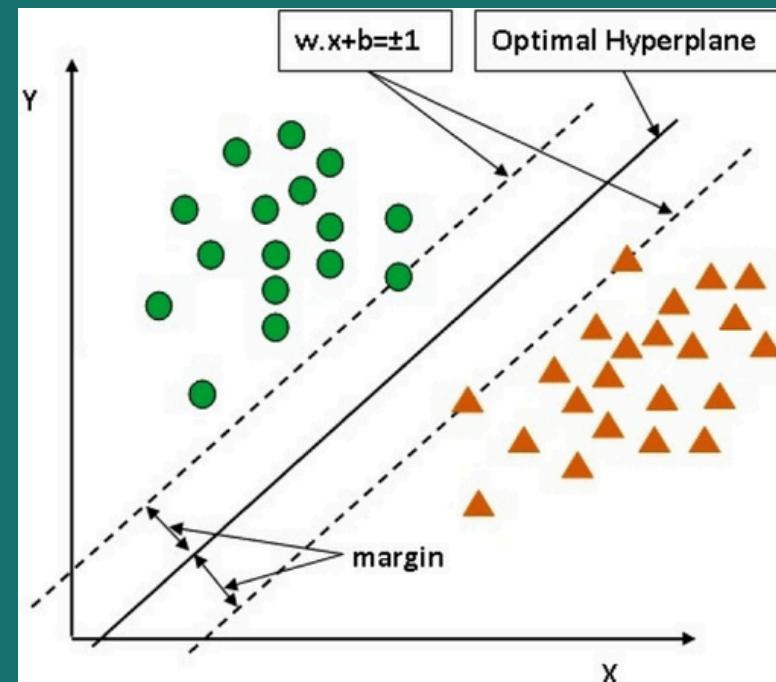
MODEL DT (2/2)

```
[108] 1 DT = grid_search_dt.best_estimator_
2 y_pred = DT.predict(X_test)
3 print(accuracy_score(y_train, DT.predict(X_train)))
4 dt_acc = accuracy_score(y_test, DT.predict(X_test))
5 print(accuracy_score(y_test, DT.predict(X_test)))
6 print(confusion_matrix(y_test, y_pred))
7 print(classification_report(y_test, y_pred))
```

```
0.9046052631578947
0.9013157894736842
[[91  7]
 [ 8 46]]
      precision    recall   f1-score   support
          0         0.92      0.93      0.92       98
          1         0.87      0.85      0.86       54
   accuracy                           0.90      152
  macro avg       0.89      0.89      0.89      152
weighted avg       0.90      0.90      0.90      152
```

SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) adalah algoritma pembelajaran mesin yang bekerja dengan mencari hyperplane optimal yang memisahkan data ke dalam kelas-kelas berbeda dengan margin maksimum.

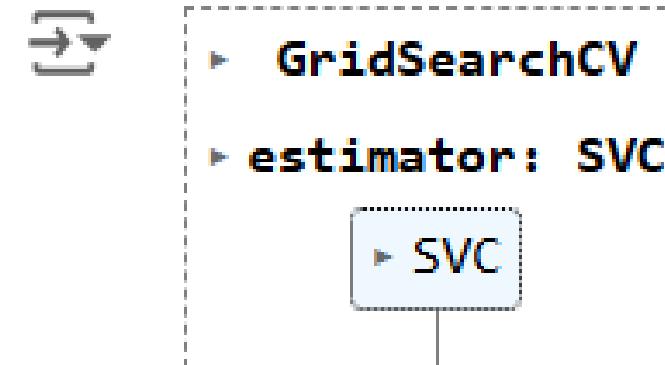


sumber: researchgate.net

- Inisialisasi model SVM dan menentukan parameter grid untuk pencarian hypermeter terbaik.
- Gunakan GridSearchCV untuk mencari kombinasi hyperparameter terbaik.
- Inisialisasi model SVM dengan parameter terbaik dan latih model.
- Cetak akurasi pada data pelatihan dan lakukan juga pada data uji.
- Cetak confusion matrix dan classification report.

MODEL SVM (1/2)

```
[ ] 1 # SVM
2 svc = SVC(probability=True)
3 parameter = {
4     "gamma": [0.0001, 0.001, 0.01, 0.1],
5     'C': [0.01, 0.05, 0.5, 0.1, 1, 10, 15, 20]
6 }
7 grid_search = GridSearchCV(svc, parameter)
8 grid_search.fit(X_train, y_train)
```



```
[ ] 1 # best_parameter
2 grid_search.best_params_
```

→ { 'C': 0.5, 'gamma': 0.1}

SUPPORT VECTOR MACHINE

Evaluasi Model SVM :

- Precision kelas 0: 94% (prediksi kelas 0 benar-benar kelas 0).
- Precision kelas 1: 81% (prediksi kelas 1 benar-benar kelas 1).
- Recall kelas 0: 89%
- Recall kelas 1: 89%
- F1-score kelas 0: 91% (rata-rata harmonik precision dan recall untuk kelas 0).
- F1-score kelas 1: 85% (rata-rata harmonik precision dan recall untuk kelas 1).
- Akurasi keseluruhan model: **89%** (prediksi benar secara keseluruhan).

MODEL SVM (2/2)

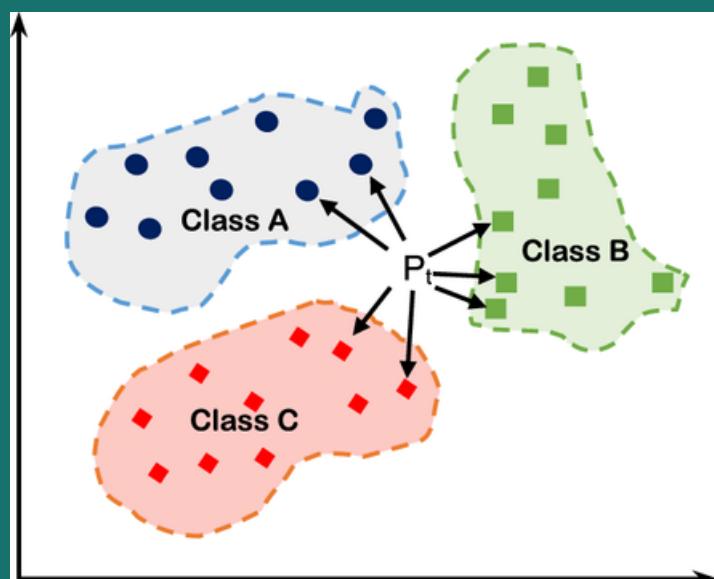
```
[103] 1 svc = SVC(C=0.5, gamma = 0.01, probability=True)
2 svc.fit(X_train, y_train)
3 y_pred = svc.predict(X_test)
4 print(accuracy_score(y_train, svc.predict(X_train)))
5 svc_acc = accuracy_score(y_test, svc.predict(X_test))
6 print(accuracy_score(y_test, svc.predict(X_test)))
7 print(confusion_matrix(y_test, y_pred))
8 print(classification_report(y_test, y_pred))
```

```
↳ 0.8305921052631579
0.8881578947368421
[[87 11]
 [ 6 48]]
```

	precision	recall	f1-score	support
0	0.94	0.89	0.91	98
1	0.81	0.89	0.85	54
accuracy			0.89	152
macro avg	0.87	0.89	0.88	152
weighted avg	0.89	0.89	0.89	152

K-NEAREST NEIGHBORS

K-Nearest Neighbors (KNN) adalah algoritma non-parametrik yang digunakan dalam pembelajaran mesin untuk mengklasifikasikan atau memprediksi nilai berdasarkan kedekatan (similarity) dengan data pelatihan di mana kedekatan dihitung menggunakan metrik jarak seperti Euclidean.



sumber: researchgate.net

- Inisialisasi model KNN dengan 'knn = KNeighborsClassifier()'
- Latih model dengan data pelatihan dengan 'knn.fit(X_train, y_train)'
- Prediksi label target untuk data uji dengan 'y_pred = knn.predict(X_test)'
- Ukur akurasi model pada data pelatihan menggunakan fungsi 'print(classification_report(y_test, y_pred))'

MODEL KNN (1/2)

```
[ ] # KNN  
knn = KNeighborsClassifier()  
knn.fit(X_train, y_train)  
y_pred = knn.predict(X_test)  
print(accuracy_score(y_train, knn.predict(X_train)))  
knn_acc = accuracy_score(y_test, knn.predict(X_test))  
print(accuracy_score(y_test, knn.predict(X_test)))  
print(confusion_matrix(y_test, y_pred))  
print(classification_report(y_test, y_pred))
```

```
→ 0.8766447368421053  
0.875  
[[89  9]  
 [10 44]]  
precision    recall   f1-score   support  
          0       0.90      0.91      0.90      98  
          1       0.83      0.81      0.82      54
```

	precision	recall	f1-score	support
accuracy				0.88
macro avg	0.86	0.86	0.86	152
weighted avg	0.87	0.88	0.87	152

K-NEAREST NEIGHBORS

Evaluasi Model KNN :

- Precision kelas 0: 90% (prediksi kelas 0 benar-benar kelas 0).
- Precision kelas 1: 83% (prediksi kelas 1 benar-benar kelas 1).
- Recall kelas 0: 91%
- Recall kelas 1: 81%
- F1-score kelas 0: 90% (rata-rata harmonik precision dan recall untuk kelas 0).
- F1-score kelas 1: 82% (rata-rata harmonik precision dan recall untuk kelas 1).
- Akurasi keseluruhan model: **88%** (prediksi benar secara keseluruhan).

MODEL KNN (2/2)

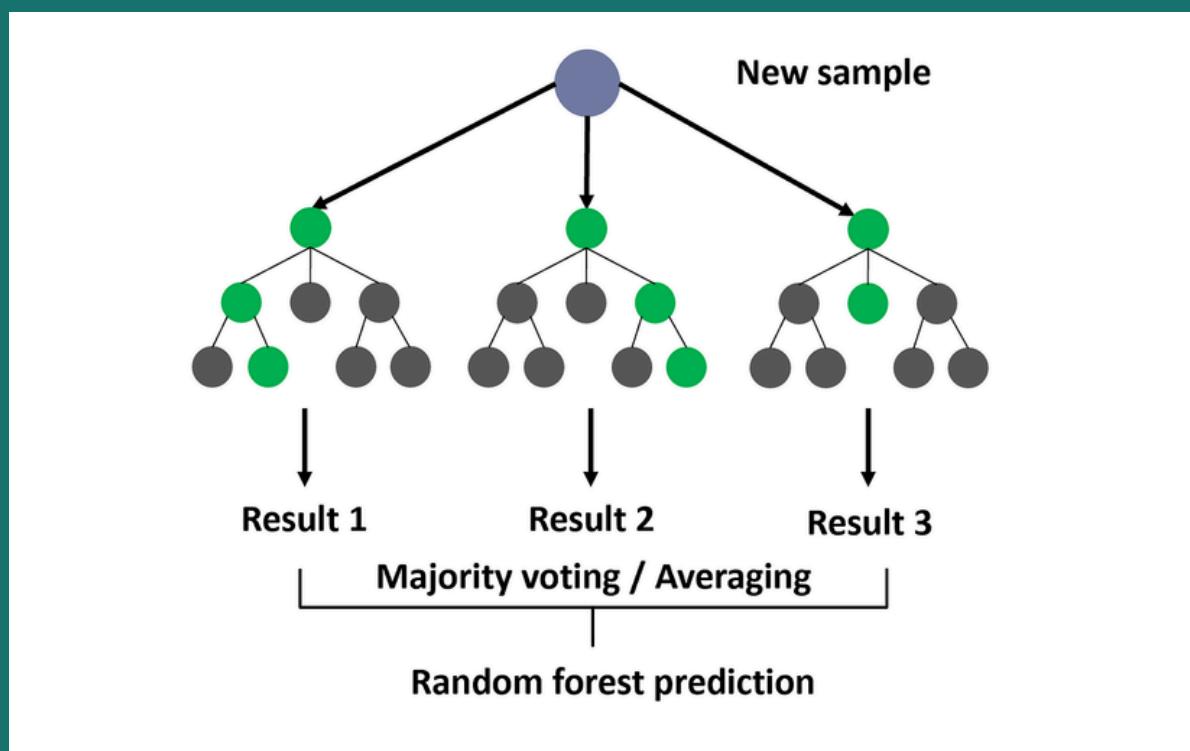
```
[ ] # KNN  
knn = KNeighborsClassifier()  
knn.fit(X_train, y_train)  
y_pred = knn.predict(X_test)  
print(accuracy_score(y_train, knn.predict(X_train)))  
knn_acc = accuracy_score(y_test, knn.predict(X_test))  
print(accuracy_score(y_test, knn.predict(X_test)))  
print(confusion_matrix(y_test, y_pred))  
print(classification_report(y_test, y_pred))
```

```
→ 0.8766447368421053  
0.875  
[[89  9]  
 [10 44]]
```

	precision	recall	f1-score	support
0	0.90	0.91	0.90	98
1	0.83	0.81	0.82	54
accuracy			0.88	152
macro avg	0.86	0.86	0.86	152
weighted avg	0.87	0.88	0.87	152

RANDOM FOREST

Random Forest adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi, yang terdiri dari banyak pohon keputusan (decision trees) yang dibangun pada subset acak dari data pelatihan. Hasil akhir diperoleh dengan menggabungkan prediksi dari semua pohon untuk meningkatkan akurasi dan mengurangi overfitting.



sumber: medium.com

- Inisialisasi model RF dengan 'RandomForestClassifier()'
- Latih model dengan data pelatihan dengan `rand_clf.fit(X_train, y_train)`
- Prediksi label target untuk data uji dengan `'y_pred = rand_clf.predict(X_test)'`
- Ukur akurasi model pada data pelatihan menggunakan fungsi `'print(classification_report(y_test, y_pred))'`

MODEL RANDOM FOREST(1/2)

```
# RandomForestClassifier
rand_clf = RandomForestClassifier(criterion = 'entropy',
                                   max_depth = 15,
                                   max_features = 0.75,
                                   min_samples_leaf = 2,
                                   min_samples_split = 3,
                                   n_estimators = 130)

rand_clf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=15, max_features=0.75,
                      min_samples_leaf=2, min_samples_split=3,
                      n_estimators=130)
```

RANDOM FOREST

Evaluasi Model Random Forest :

- Precision kelas 0: 93% (prediksi kelas 0 benar-benar kelas 0).
- Precision kelas 1: 87% (prediksi kelas 1 benar-benar kelas 1).
- Recall kelas 0: 93%
- Recall kelas 1: 87%
- F1-score kelas 0: 93% (rata-rata harmonik precision dan recall untuk kelas 0).
- F1-score kelas 1: 87% (rata-rata harmonik precision dan recall untuk kelas 1).
- Akurasi keseluruhan model: **91%** (prediksi benar secara keseluruhan).

MODEL RANDOM FOREST(2/2)

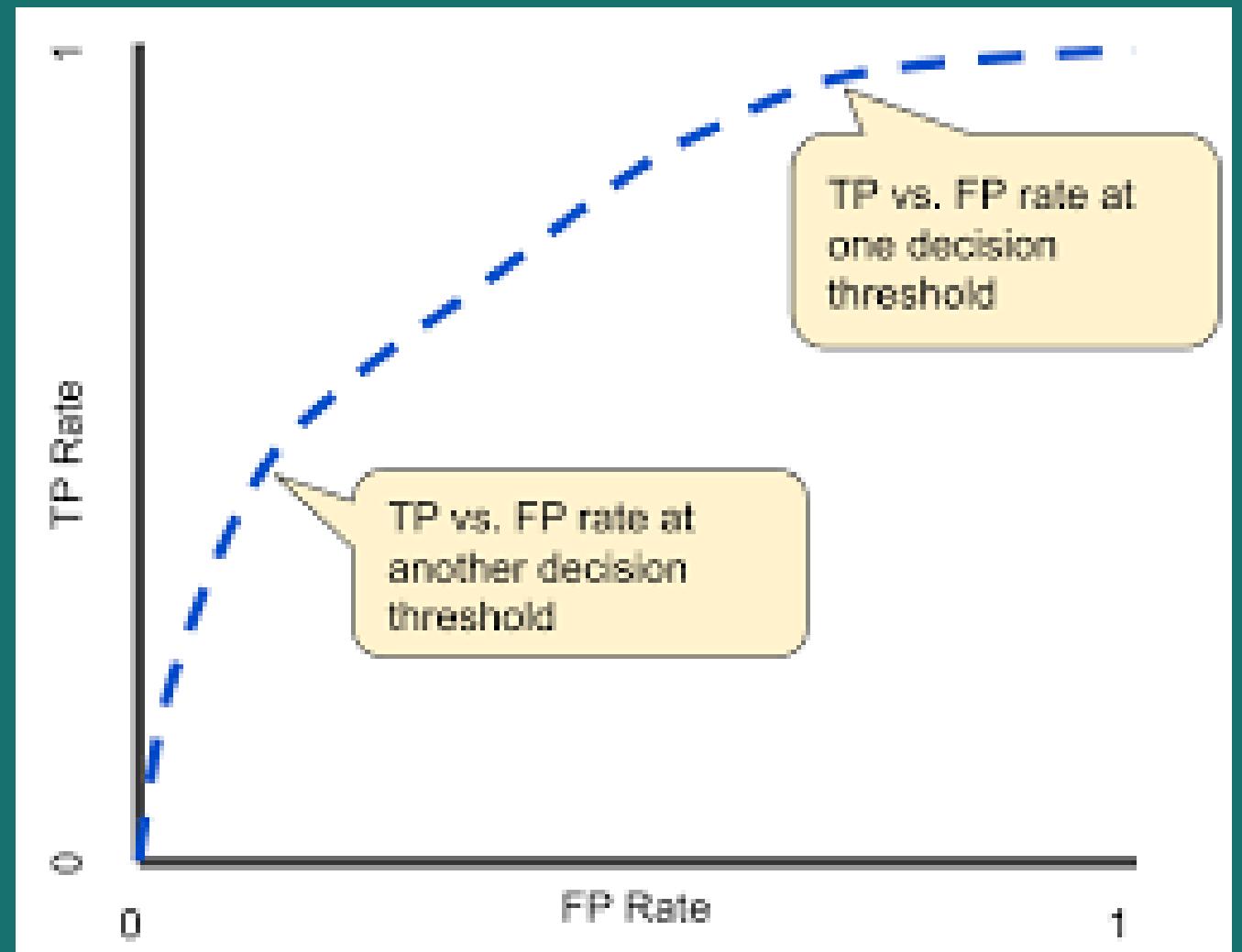
```
[ ] y_pred = rand_clf.predict(X_test)
print(accuracy_score(y_train, rand_clf.predict(X_train)))
rand_acc = accuracy_score(y_test, rand_clf.predict(X_test))
print(accuracy_score(y_test, rand_clf.predict(X_test)))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.9868421052631579
0.9078947368421053
[[91  7]
 [ 7 47]]
```

	precision	recall	f1-score	support
0	0.93	0.93	0.93	98
1	0.87	0.87	0.87	54
accuracy			0.91	152
macro avg	0.90	0.90	0.90	152
weighted avg	0.91	0.91	0.91	152

KURVA ROC

Kurva ROC (kurva karakteristik operasi penerima) adalah grafik yang menunjukkan kinerja model klasifikasi di semua ambang klasifikasi.



RUMUS ROC

$$ROC = \left(\frac{\text{currentvalue}}{\text{previousvalue}} - 1 \right) \times 100$$

Sumber : <https://developers.google.com/>

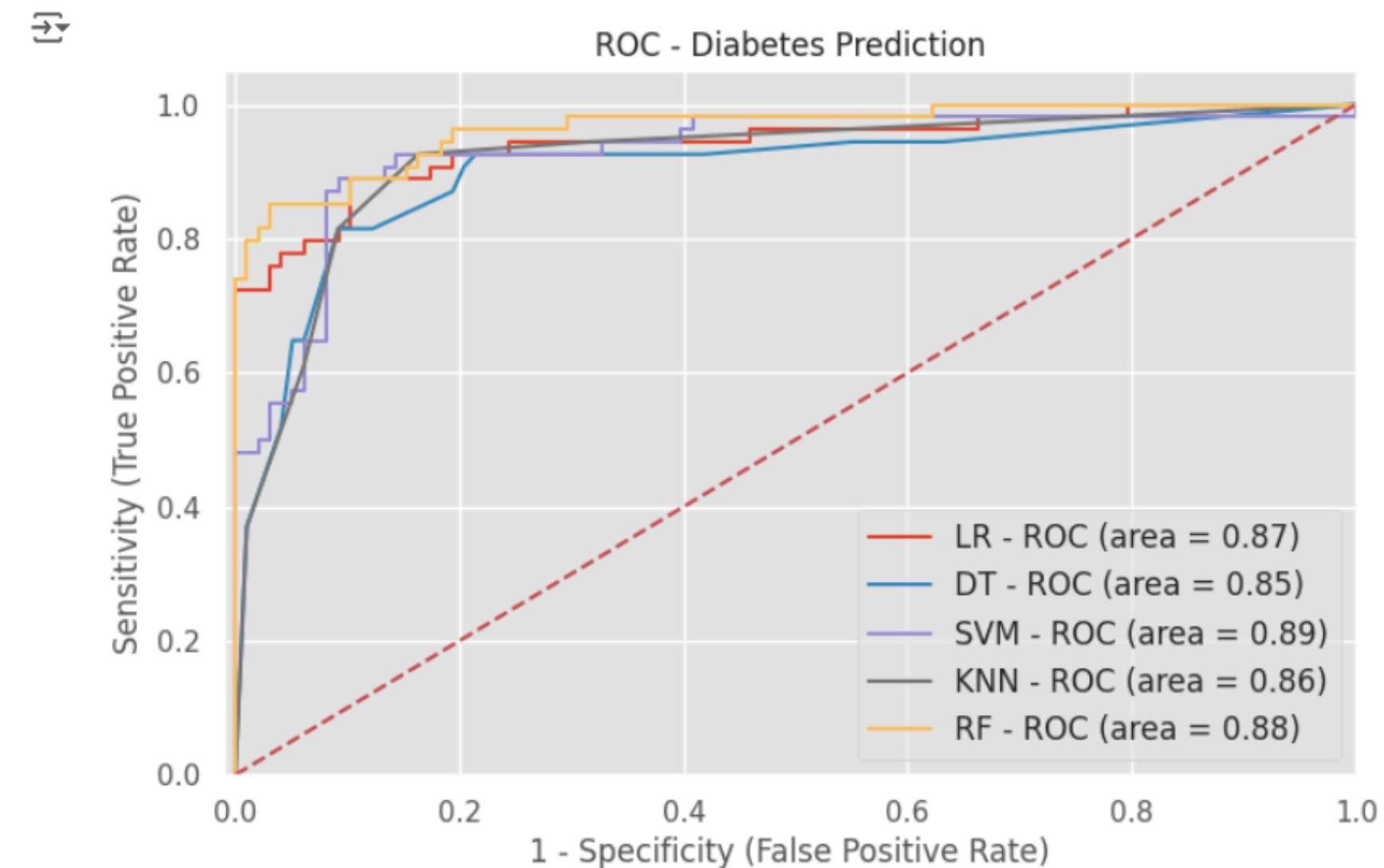
HASIL MODEL COMPARISON

```
[ ] 1 # Model Comparison
2 models = pd.DataFrame({
3     'Model': ['Logistic Regression', 'KNN', 'SVM', 'Decision Tree Classifier',
4               'Random Forest Classifier'],
5     'Score': [100*round(log_reg_acc,4), 100*round(knn_acc,4), 100*round(svc_acc,4),
6               100*round(dt_acc,4), 100*round(rand_acc,4)]})
7 models.sort_values(by = 'Score', ascending = False)
```

	Model	Score
4	Random Forest Classifier	90.79
3	Decision Tree Classifier	90.13
2	SVM	88.82
1	KNN	87.50
0	Logistic Regression	86.84

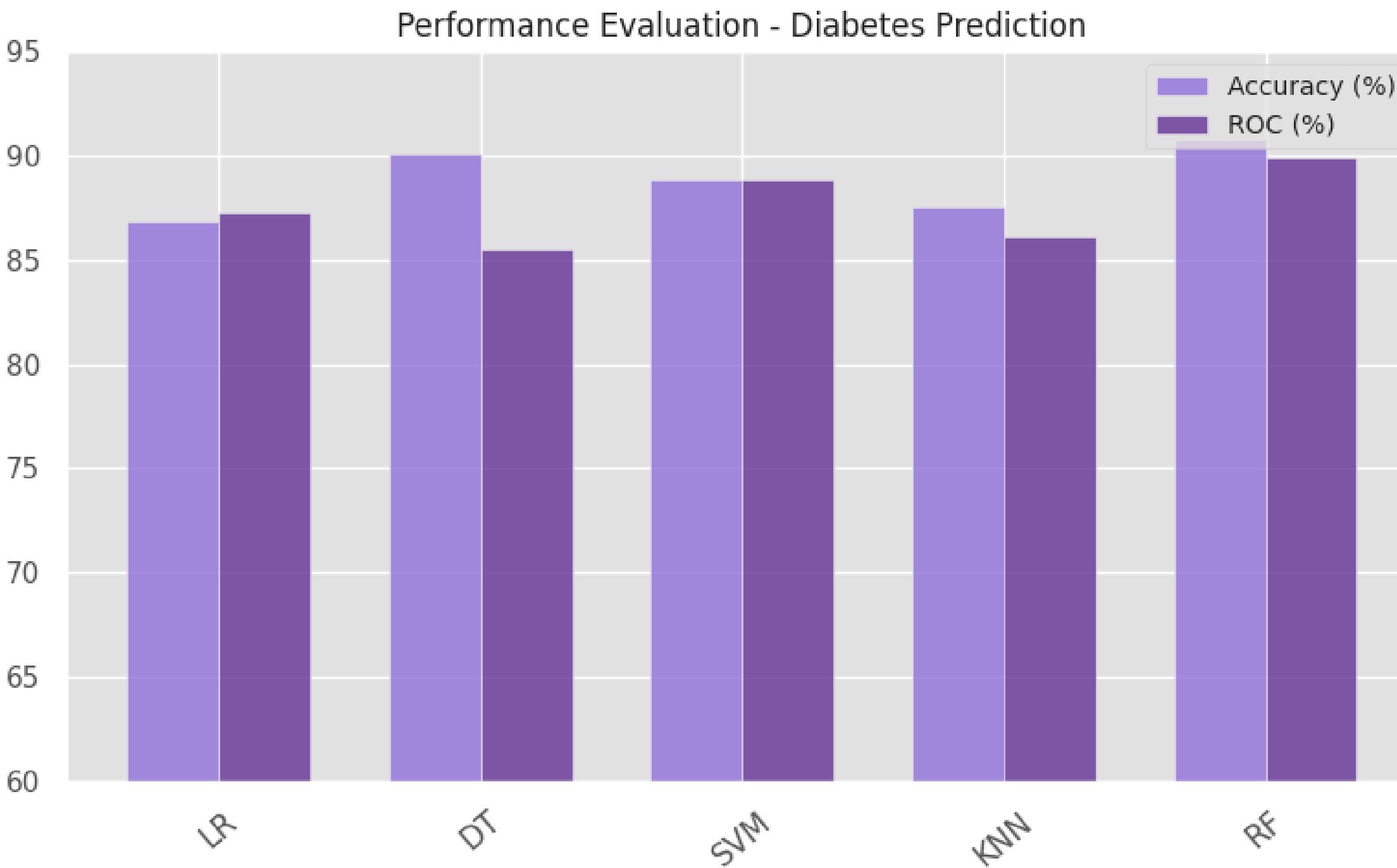
Perbandingan model menunjukkan bahwa urutan model berdasarkan akurasi tertinggi adalah Random Forest Classifier berada di puncak, diikuti oleh Decision Tree Classifier, SVM, KNN, dan akhirnya Logistic Regression.

HASIL KURVA ROC



Dalam plot kurva ROC, semakin dekat kurva ROC ke sudut kiri atas, semakin baik performa model dan semakin tinggi nilai AUC (dari 0 hingga 1), semakin baik performa model. Terlihat bahwa SVM dan RF memiliki nilai AUC yang lebih tinggi.

EVALUASI PERFORMA PREDIKSI DIABETES



Dapat dilihat bahwa Random Forest memiliki akurasi dan ROC paling tinggi dibandingkan model klasifikasi lainnya.



KESIMPULAN & SARAN

KESIMPULAN

Dapat disimpulkan urutan model yang terbaik secara urutan dalam memprediksi terkenanya diabetes adalah

1. Random Forest (RF),
2. Decision Tree (DT),
3. Support Vector Machine (SVM),
4. K-Nearest Neighbors (KNN), dan
5. Logistic Regression (LR)

Model Klasifikasi	Accuracy	ROC AUC
Logistic Regression (LR)	86,84%	87%
Decision Tree (DT)	90,13%	86%
Support Vector Machine (SVM)	88,82%	88%
K-Nearest Neighbors (KNN)	87,5%	86%
Random Forest (RF)	90,79%	89%

SARAN

1. Pilih **Random Forest** karena menunjukkan akurasi dan ROC yang tinggi.
2. Lakukan **tuning hyperparameter** dengan Grid Search atau Random Search untuk performa optimal.
3. Gunakan **validasi silang** untuk menghindari overfitting.
4. Pertimbangkan stratified k-fold cross-validation untuk estimasi performa yang robust.
5. **Analisis fitur** untuk memastikan relevansi dan hilangkan fitur yang tidak relevan atau redundan.
6. Gunakan teknik **seleksi fitur** atau dimensionality reduction (misalnya PCA) untuk meningkatkan performa model.
7. Tambahkan **data pelatihan** jika memungkinkan untuk meningkatkan generalisasi model.
8. **Implementasikan** dan monitor performa model di lingkungan produksi untuk memastikan kinerja dengan data nyata.

DAFTAR PUSTAKA

- Palimkar, P., Shaw, R. N., & Ghosh, A. (2022). Machine learning technique to prognosis diabetes disease: Random forest classifier approach. In Advanced Computing and Intelligent Technologies: Proceedings of ICACIT 2021 (pp. 219-244). Springer Singapore.
- Posonia, A. M., Vigneshwari, S., & Rani, D. J. (2020, December). Machine learning based diabetes prediction using decision tree J48. In 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 498-502). IEEE.
- Rajendra, P., & Latifi, S. (2021). Prediction of diabetes using logistic regression and ensemble techniques. Computer Methods and Programs in Biomedicine Update, 1, 100032.



THANK YOU

● FOR YOUR ATTENTION