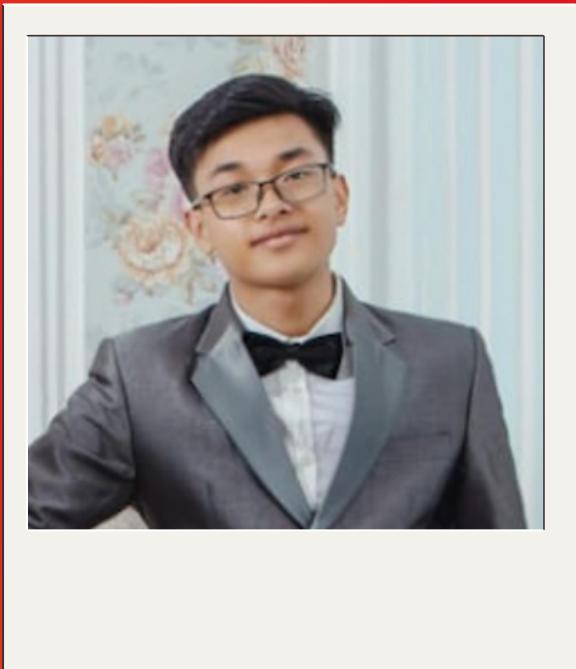




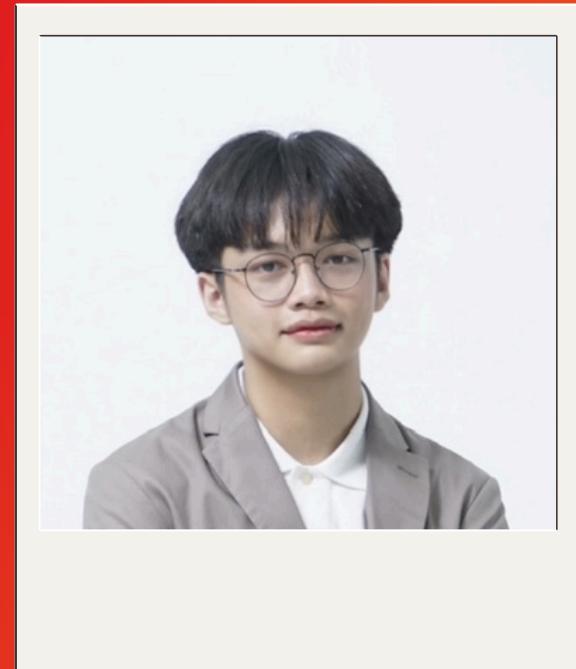
Klasifikasi dan Prediksi Potensi Diabetes Pada Pasien di Indonesia

Kelompok 7

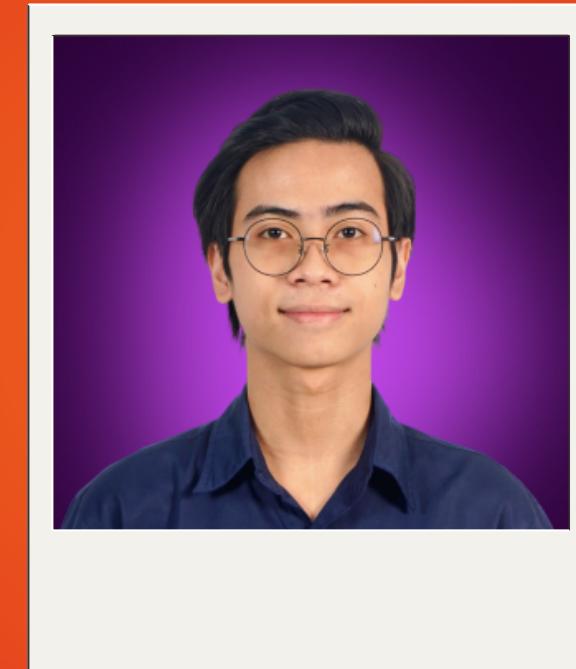
Anggota Kelompok:



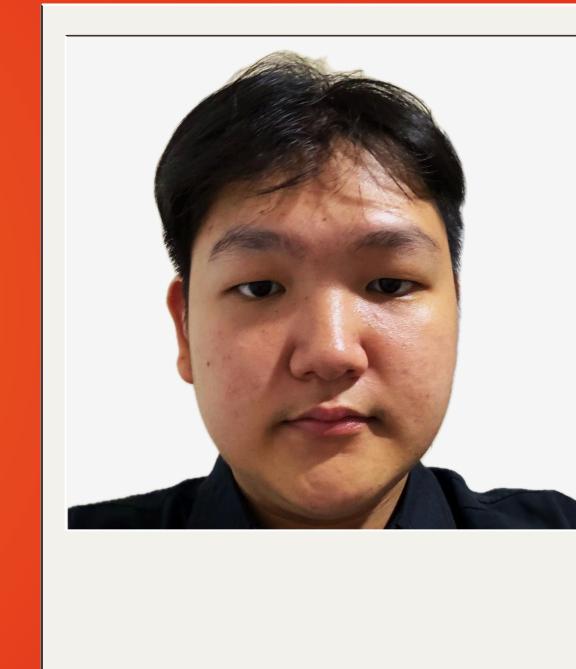
2602190772
Nathanael Sanliago Suhardjo



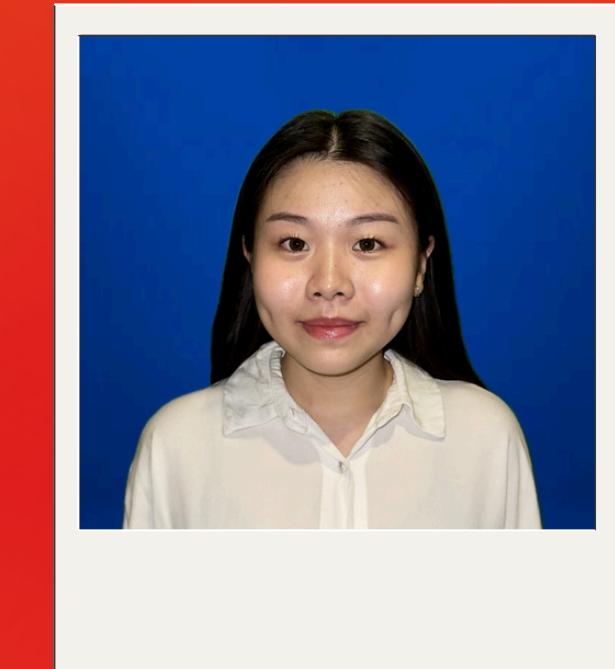
2602135603
Muhammad Alvin



2602158815
Pradipa Javier Fatah



2602159982
Nixen Jenio



2602141051
Ngu Su Ying

Latar Belakang

Diabetes mellitus adalah suatu **kondisi** yang ditandai dengan **tingginya kadar gula (glukosa) dalam darah** secara terus-menerus. Ada beberapa jenis diabetes. Dua yang paling umum disebut diabetes tipe 1 dan diabetes tipe 2.

Data **Kementerian Kesehatan** menunjukkan angka **pengidap diabetes di Indonesia** saat ini telah mencapai **19,5 juta jiwa**. Jumlah tersebut **diprediksi akan melonjak mencapai 28,5 juta penduduk pada 2045**.





1

Jumlah pasien
yang terus
meningkat
setiap tahunnya

Masalah

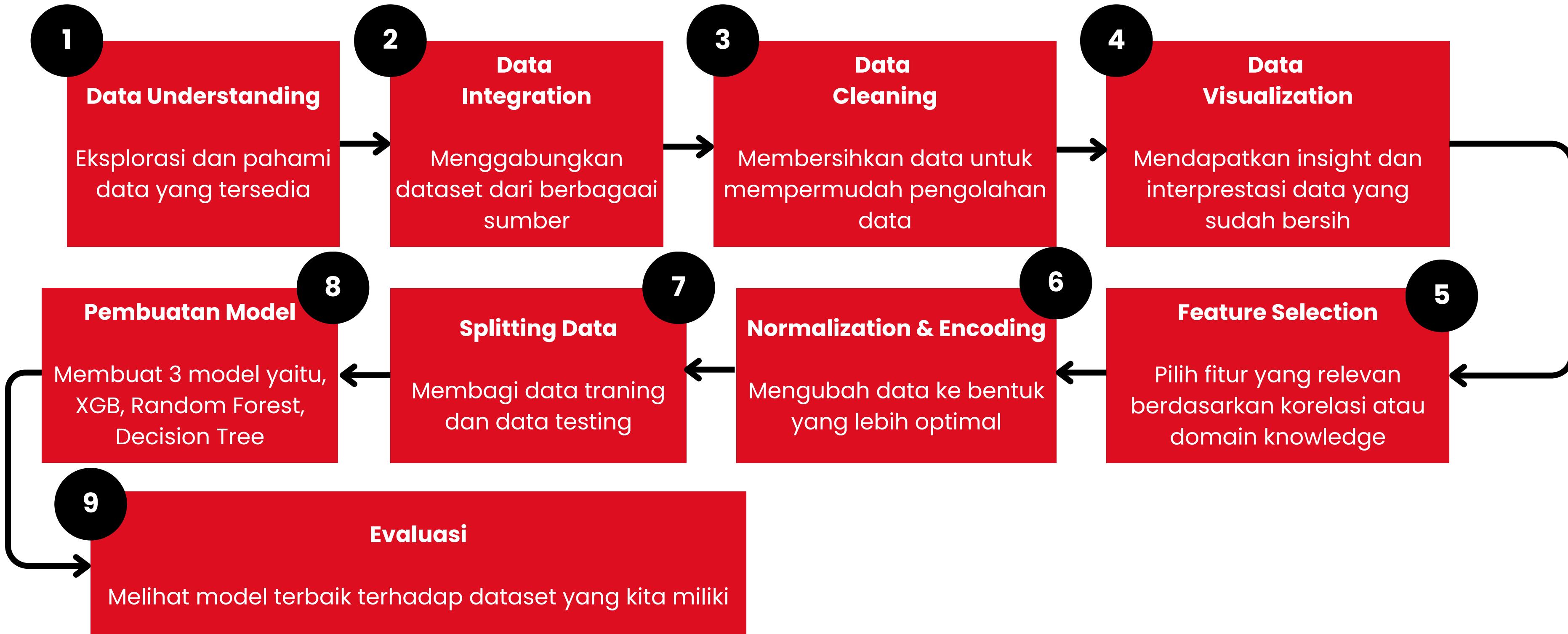
2

Komplikasi pada jantung
dan ginjal menjadi
penyebab utama
kematian pasien
diabetes di dunia

3

Pasien tidak
mengetahui
bahwa dia
mengidap
diabetes

Langkah - langkah Pembuatan Model



Isi Dataset

```

> <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1879 entries, 0 to 1878
Data columns (total 50 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PatientID        1879 non-null   int64  
 1   Age              1879 non-null   int64  
 2   Gender            1879 non-null   int64  
 3   Ethnicity         1879 non-null   int64  
 4   SocioeconomicStatus 1879 non-null   int64  
 5   EducationLevel   1879 non-null   int64  
 6   BMI               1879 non-null   float64 
 7   Smoking            1879 non-null   int64  
 8   AlcoholConsumption 1879 non-null   float64 
 9   PhysicalActivity  1879 non-null   float64 
 10  DietQuality       1879 non-null   float64 
 11  SleepQuality      1879 non-null   float64 
 12  FamilyHistoryDiabetes 1879 non-null   int64  
 13  GestationalDiabetes 1879 non-null   int64  
 14  PolycysticOvarySyndrome 1879 non-null   int64  
 15  PreviousPreDiabetes 1879 non-null   int64  
 16  Hypertension       1879 non-null   int64  
 17  SystolicBP         1879 non-null   int64  
 18  DiastolicBP        1879 non-null   int64  
 19  FastingBloodSugar  1879 non-null   float64 
 20  HbA1c              1879 non-null   float64 
 21  SerumCreatinine   1879 non-null   float64 
 22  BUNLevels          1879 non-null   float64 
 23  CholesterolTotal   1879 non-null   float64 
 24  CholesterolLDL    1879 non-null   float64 
 25  CholesterolHDL    1879 non-null   float64 
 26  CholesterolTriglycerides 1879 non-null   float64 
 27  AntihypertensiveMedications 1879 non-null   int64  
 28  Statins             1879 non-null   int64  
 29  AntidiabeticMedications 1879 non-null   int64  
 30  FrequentUrination  1879 non-null   int64  
 31  ExcessiveThirst    1879 non-null   int64  
 32  UnexplainedWeightLoss 1879 non-null   int64  
 33  FatigueLevels      1879 non-null   float64 
 34  BlurredVision      1879 non-null   int64  
 35  SlowHealingSores   1879 non-null   int64  
 36  TinglingHandsFeet  1879 non-null   int64  
 37  QualityOfLifeScore 1879 non-null   float64 
 38  HeavyMetalsExposure 1879 non-null   int64  
 39  OccupationalExposureChemical 1879 non-null   int64  
 40  WaterQuality        1879 non-null   int64  
 41  MedicalCheckupsFrequency 1879 non-null   float64 
 42  MedicationAdherence 1879 non-null   float64 
 43  HealthLiteracy      1879 non-null   float64 
 44  Diagnosis            1879 non-null   int64  
 45  DoctorInCharge      1879 non-null   object  
 46  LifestyleScore       1879 non-null   float64 
 47  DiabetesRiskScore   1879 non-null   float64 
 48  TreatmentRecommendation 1879 non-null   int64  
 49  MedicationPriorityScore 1879 non-null   float64 
dtypes: float64(21), int64(28), object(1)
memory usage: 734.1+ KB

```

#	Column
0	PatientID
1	Age
2	Gender
3	Ethnicity
4	SocioeconomicStatus
5	EducationLevel
6	BMI
7	Smoking
8	AlcoholConsumption
9	PhysicalActivity
10	DietQuality
11	SleepQuality
12	FamilyHistoryDiabetes
13	GestationalDiabetes
14	PolycysticOvarySyndrome
15	PreviousPreDiabetes
16	Hypertension
17	SystolicBP
18	DiastolicBP
19	FastingBloodSugar
20	HbA1c
21	SerumCreatinine
22	BUNLevels
23	CholesterolTotal
24	CholesterolLDL
25	CholesterolHDL
26	CholesterolTriglycerides
27	AntihypertensiveMedications
28	Statins
29	AntidiabeticMedications
30	FrequentUrination
31	ExcessiveThirst
32	UnexplainedWeightLoss
33	FatigueLevels
34	BlurredVision
35	SlowHealingSores
36	TinglingHandsFeet
37	QualityOfLifeScore
38	HeavyMetalsExposure
39	OccupationalExposureChemical
40	WaterQuality
41	MedicalCheckupsFrequency
42	MedicationAdherence
43	HealthLiteracy
44	Diagnosis
45	DoctorInCharge
46	LifestyleScore
47	DiabetesRiskScore
48	TreatmentRecommendation
49	MedicationPriorityScore

1879 Data dan 50 Atribut

*hasil penggabungan data 1 dan data 2

Penjelasan Lengkap
Setiap Atribut

Membersihkan Data

```
✓ 0s
  data = data.drop(['PatientID', 'DoctorInCharge'], axis=1)
  data
```

*menghapus data pada kolom '**PasientID**' dan '**DoctorInCharge**'

Total Data Duplicated:	0
Table Missing Value	0
Age	0
Gender	0
Ethnicity	0
SocioeconomicStatus	0
EducationLevel	0
BMI	0
Smoking	0
AlcoholConsumption	0
PhysicalActivity	0
DietQuality	0
SleepQuality	0
FamilyHistoryDiabetes	0
GestationalDiabetes	0
PolycysticOvarySyndrome	0
PreviousPreDiabetes	0
Hypertension	0
SystolicBP	0
DiastolicBP	0
FastingBloodSugar	0
HbA1c	0
SerumCreatinine	0
BUNLevels	0
CholesterolTotal	0
CholesterolLDL	0
CholesterolHDL	0
CholesterolTriglycerides	0
AntihypertensiveMedications	0
Statins	0
AntidiabeticMedications	0

*tidak ada data yang **duplicat** dan **missing value**

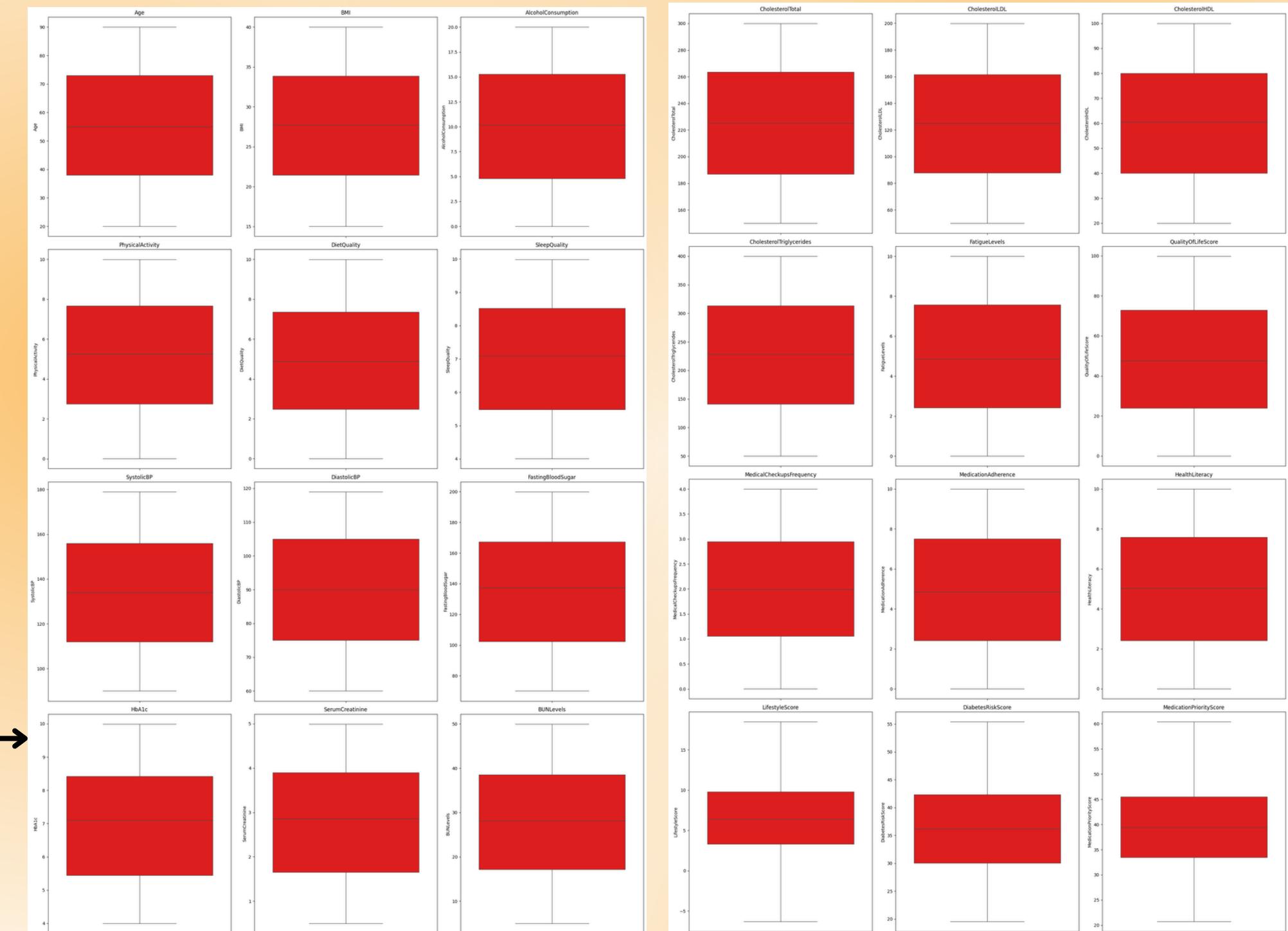
```
✓ 21s
  import warnings
  warnings.filterwarnings("ignore")

  plt.figure(figsize=(20, 60))

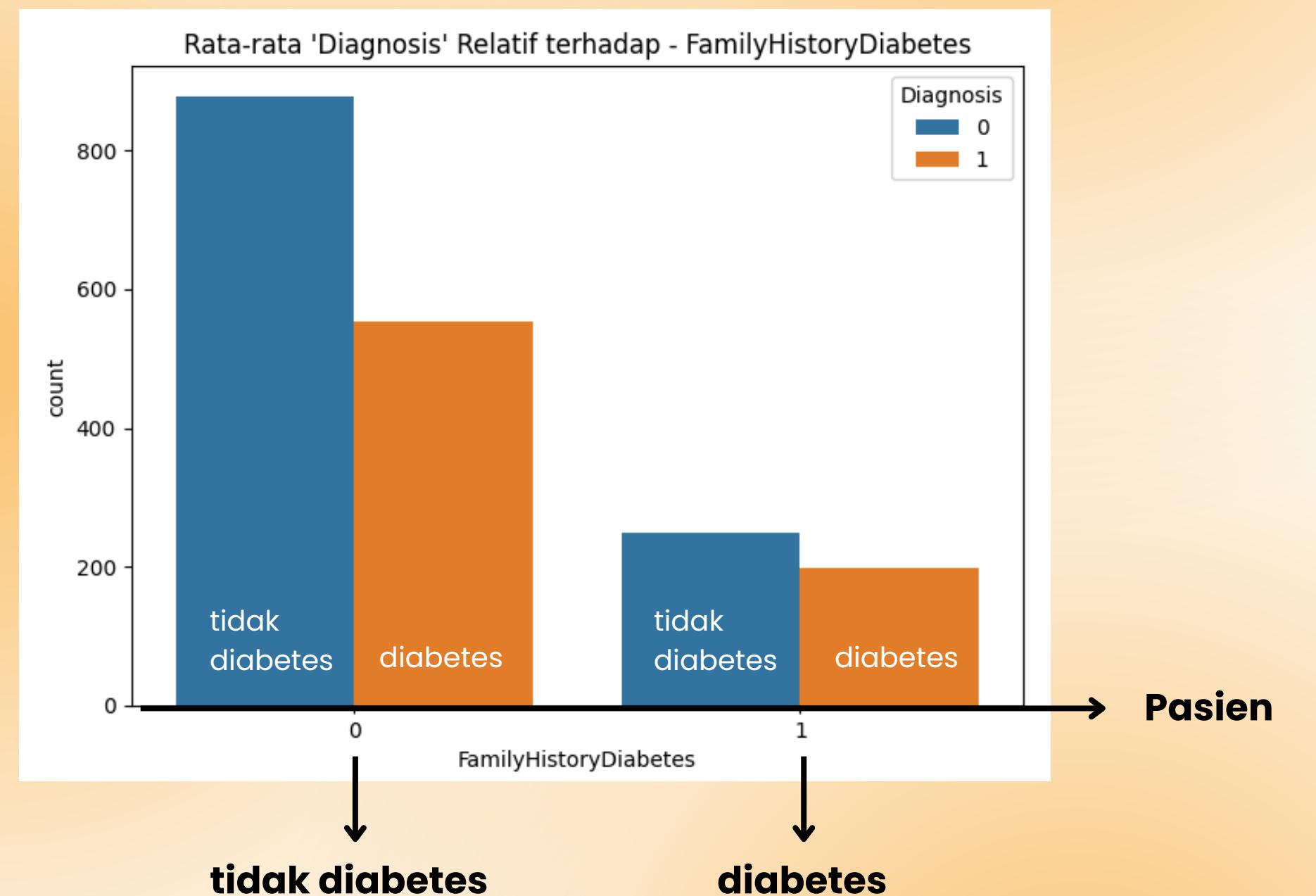
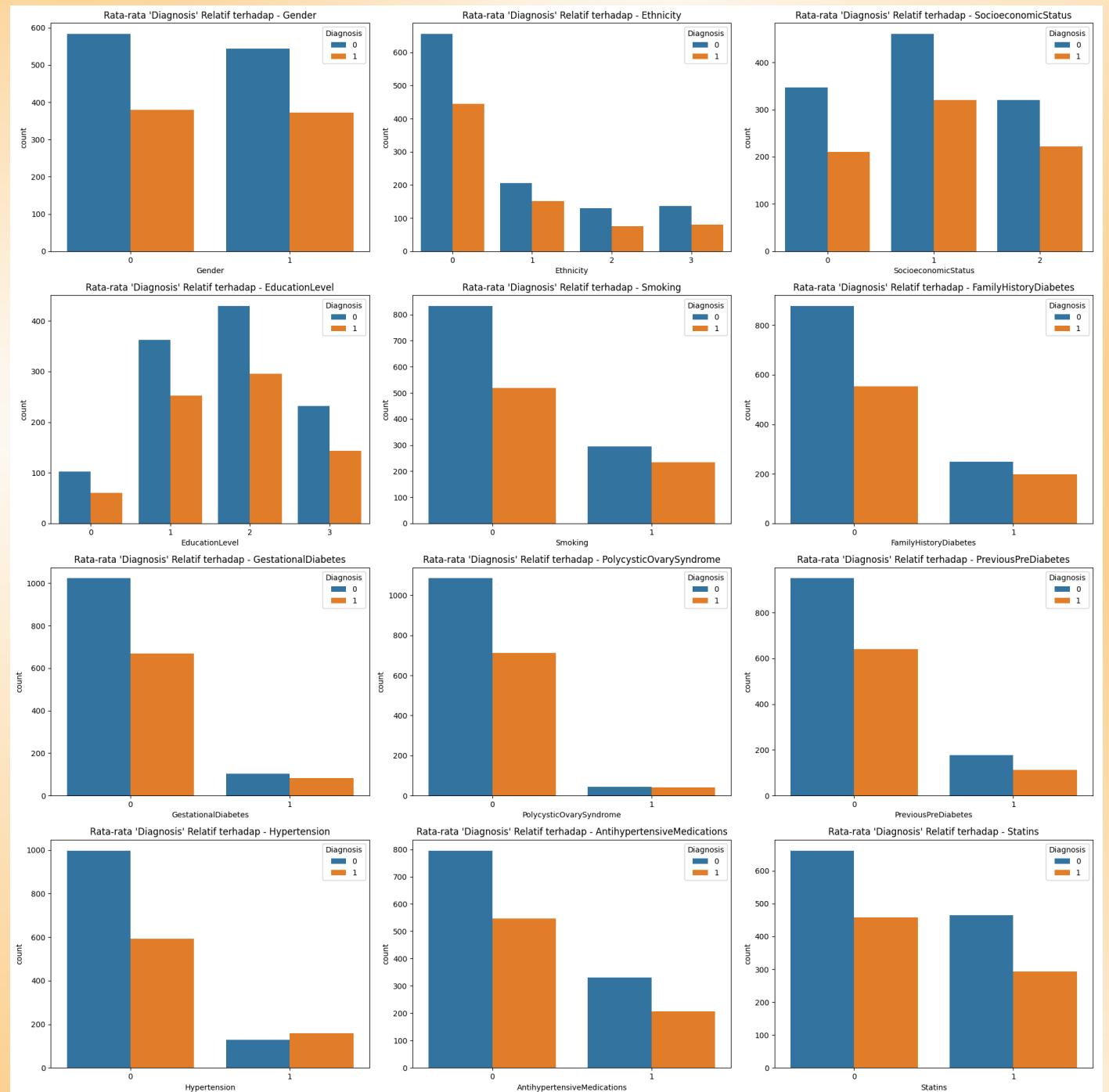
  # Grid 8 baris x 3 kolom
  for i in range(24):
    plt.subplot(8, 3, i + 1)
    sns.boxplot(y=num_dist.iloc[:, i], color='red') # Membuat boxplot untuk setiap kolom
    plt.title(num_dist.columns[i]) # Menambahkan judul berdasarkan nama kolom
    plt.tight_layout()

  plt.show()
```

*tidak ada data **outlier**



Data Visualization



HeatMap



```
# Misalkan 'target_feature' adalah nama kolom target fitur
target_feature = 'Diagnosis'

# Filter korelasi lemah terhadap target fitur
weak_corr_with_target = corr_table[target_feature][(corr_table[target_feature] > -0.01) & (corr_table[target_feature] < 0.01)]

# Tampilkan hasil
weak_corr_with_target
```

	Diagnosis
EducationLevel	-0.00
AlcoholConsumption	-0.01
PhysicalActivity	-0.01
SleepQuality	-0.00
CholesterolLDL	-0.00
SlowHealingSores	0.01
OccupationalExposureChemicals	-0.01
MedicalCheckupsFrequency	-0.01

dtype: float64

*menghapus tabel yang korelasi absolut < 0.001

Diagnosis	
EducationLevel	-0.00
AlcoholConsumption	-0.01
PhysicalActivity	-0.01
SleepQuality	-0.00
CholesterolLDL	-0.00
SlowHealingSores	0.01
OccupationalExposureChemicals	-0.01
MedicalCheckupsFrequency	-0.01

dtype: float64

```
[4]: data.drop(['EducationLevel',
              'AlcoholConsumption',
              'PhysicalActivity',
              'SleepQuality',
              'SlowHealingSores',
              'OccupationalExposureChemicals',
              'MedicalCheckupsFrequency'],
             inplace=True, axis=1)
data.head()
```

Normalization & Encoding

	Age	Gender	Ethnicity	SocioeconomicStatus	BMI	Smoking
0	44	0	1	2	32.99	1
1	51	1	0	1	39.92	0
2	89	1	0	1	19.78	0
3	21	1	1	1	32.38	1
4	27	1	0	1	16.81	0

5 rows x 41 columns

One-Hot Encoding

MedicationPriorityScore	Ethnicity_0	Ethnicity_1	Ethnicity_2	Ethnicity_3
49.27	False	True	False	False
52.24	True	False	False	False
33.87	True	False	False	False
46.89	False	True	False	False
25.42	True	False	False	False

```

from sklearn.preprocessing import MinMaxScaler
numerical_features = [
    'Age',
    'BMI',
    'DietQuality',
    'SystolicBP',
    'DiastolicBP',
    'FastingBloodSugar',
    'HbA1c',
    'SerumCreatinine',
    'BUNLevels',
    'CholesterolTotal',
    'CholesterolLDL',
    'CholesterolHDL',
    'CholesterolTriglycerides',
    'FatigueLevels',
    'QualityOfLifeScore',
    'MedicationAdherence',
    'HealthLiteracy',
    'LifestyleScore',
    'DiabetesRiskScore',
    'MedicationPriorityScore'
]

scaler = MinMaxScaler()
scaler.fit(X_train[numerical_features])
X_train[numerical_features] = scaler.transform(X_train.loc[:, numerical_features])
X_test[numerical_features] = scaler.transform(X_test.loc[:, numerical_features])

X_train[numerical_features].head()

```

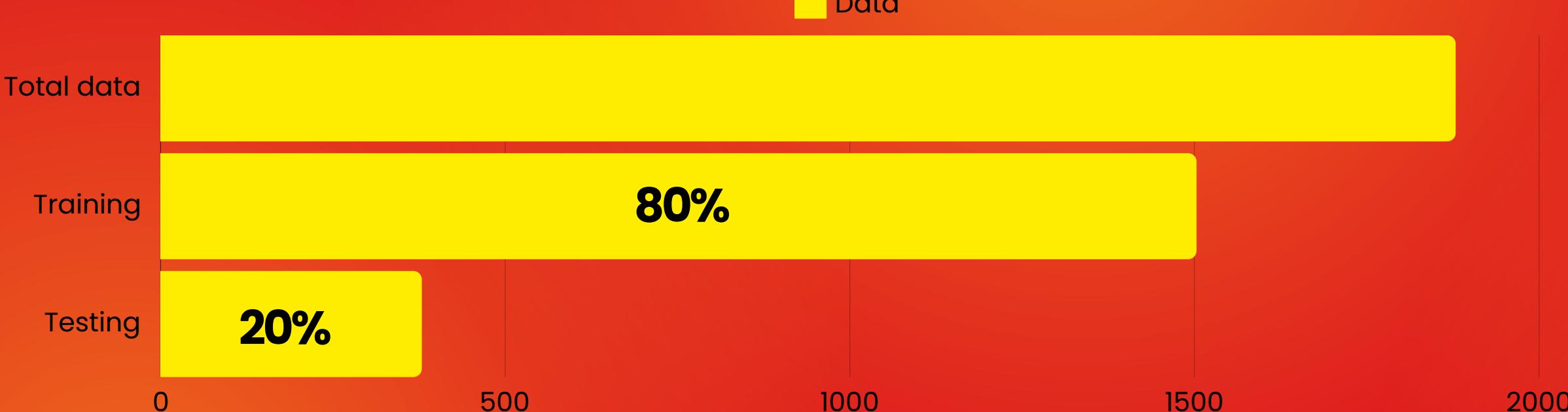
Normalization

	Age	BMI	DietQuality	SystolicBP	DiastolicBP	FastingBP
count	1503.00	1503.00	1503.00	1503.00	1503.00	1503.00
mean	0.50	0.51	0.49	0.49	0.51	0.51
std	0.29	0.29	0.29	0.29	0.30	0.30
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	0.25	0.25	0.25	0.25	0.25	0.25
50%	0.51	0.52	0.50	0.49	0.53	0.53
75%	0.76	0.76	0.74	0.73	0.77	0.77
max	1.00	1.00	1.00	1.00	1.00	1.00

*Mengubah data ke bentuk yang lebih optimal

Test & Training

```
[329] from sklearn.model_selection import train_test_split  
  
X = data.drop(["Diagnosis"],axis =1)  
y = data["Diagnosis"]  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)  
  
[330] print(f'Total # of sample in whole dataset: {len(X)}')  
print(f'Total # of sample in train dataset: {len(X_train)}')  
print(f'Total # of sample in test dataset: {len(X_test)}')  
  
→ Total # of sample in whole dataset: 1879  
Total # of sample in train dataset: 1503  
Total # of sample in test dataset: 376
```



PEMBUATAN MODEL

XGBoost Classifier

```
[ ] 1 # Main pipeline for fitting.  
2 model_XGB = XGBClassifier(use_label_encoder=False, eval_metric='logloss')  
3 model_XGB.fit(X_train, y_train)  
4 print("Training is success!")  
5 y_pred = model_XGB.predict_proba(X_test)  
6 predicted = model_XGB.predict(X_test)  
7 #print AUC, KS score, and classification report  
8 ks, auc = evaluate_ks_and_roc_auc(y_test, y_pred)  
9 matrix = classification_report(y_test, predicted)  
10 print('Classification report XGBoost Classifier : \n',matrix)  
11 cm = confusion_matrix(y_test, predicted)  
12 target_names = ["Indicates No","Indicates Yes"]  
13 plot_confusion_matrix(cm, target_names, title='Confusion matrix', cmap=None,normalize=False)  
14  
15 models.loc['ROC AUC','XGBClassifier'] = auc  
16 models.loc['KS','XGBClassifier'] = ks
```

```
ROC AUC: 0.9551  
KS: 0.8549 (p-value: 1.738e-67)  
Classification report XGBoost Classifier :  
precision recall f1-score support  
  
          0       0.92      0.97      0.95     233  
          1       0.95      0.87      0.91     143  
  
accuracy                           0.93     376  
macro avg       0.93      0.92      0.93     376  
weighted avg     0.93      0.93      0.93     376
```

PEMBUATAN MODEL

Random Forest Classifier

```
1 # Main pipeline for fitting.  
2 model_rf = RandomForestClassifier(max_depth= 20)  
3 model_rf.fit(X_train, y_train)  
4 print("Training is success!")  
5 y_pred = model_rf.predict_proba(X_test)  
6 predicted = model_rf.predict(X_test)  
7 #print AUC, KS score, and classification report  
8 ks, auc = evaluate_ks_and_roc_auc(y_test, y_pred)  
9 matrix = classification_report(y_test, predicted)  
10 print('Classification report RandomForestClassifier : \n',matrix)  
11 cm = confusion_matrix(y_test, predicted)  
12 target_names = ["Indicates No","Indicates Yes"]  
13 plot_confusion_matrix(cm, target_names, title='Confusion matrix', cmap=None,normalize=False)  
14  
15 models.loc['ROC AUC', 'RandomForestClassifier'] = auc  
16 models.loc['KS', 'RandomForestClassifier'] = ks
```

```
ROC AUC: 0.9524  
KS: 0.8490 (p-value: 2.390e-66)  
Classification report RandomForestClassifier :  
precision recall f1-score support  
  
          0       0.89      0.97      0.93      233  
          1       0.94      0.80      0.87      143  
  
accuracy                           0.91      376  
macro avg       0.92      0.89      0.90      376  
weighted avg     0.91      0.91      0.91      376
```

PEMBUATAN MODEL

Decision Forest Classifier

```
1 dt_model = DecisionTreeClassifier(random_state=20)
2
3 dt_model.fit(x_train, y_train)

4 best_dt_model = grid_search.best_estimator_
5 print("Best Parameters:", grid_search.best_params_)

6 y_pred_best = best_dt_model.predict(x_test)
7
8 print("\nAccuracy Score with Best Model:", accuracy_score(y_test, y_pred_best))
9 print("\nClassification Report:\n", classification_report(y_test, y_pred_best))
10 print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))

11
12
13
14
15
16 grid_search.fit(x_train, y_train)
```

DecisionTreeClassifier (i ?)

DecisionTreeClassifier(random_state=20)

param_grid = {
 'criterion': ['gini', 'entropy'],
 'splitter': ['best', 'random'],
 'max_depth': [None, 10, 20, 30, 40],
 'min_samples_split': [2, 5, 10],
 'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(estimator=DecisionTreeClassifier(random_state=20),
 param_grid=param_grid,
 cv=5,
 scoring='accuracy',
 verbose=1,
 n_jobs=-1)

grid_search.fit(x_train, y_train)

1 best_dt_model = grid_search.best_estimator_
2 print("Best Parameters:", grid_search.best_params_)
3
4 y_pred_best = best_dt_model.predict(x_test)
5
6 print("\nAccuracy Score with Best Model:", accuracy_score(y_test, y_pred_best))
7 print("\nClassification Report:\n", classification_report(y_test, y_pred_best))
8 print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))

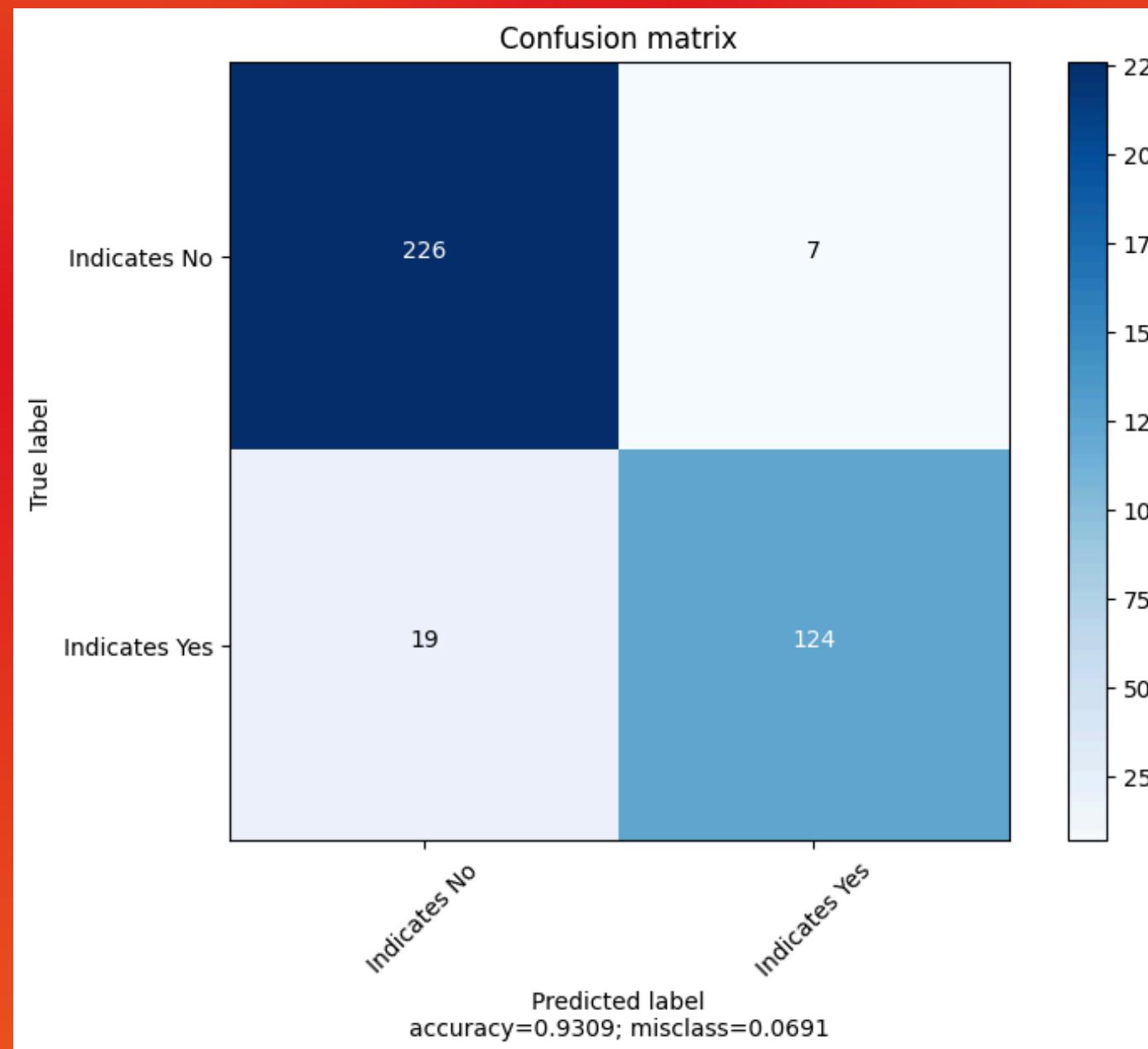
Accuracy Score with Best Model: 0.8856382978723404

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	220
1	0.86	0.87	0.86	156
accuracy			0.89	376
macro avg	0.88	0.88	0.88	376
weighted avg	0.89	0.89	0.89	376

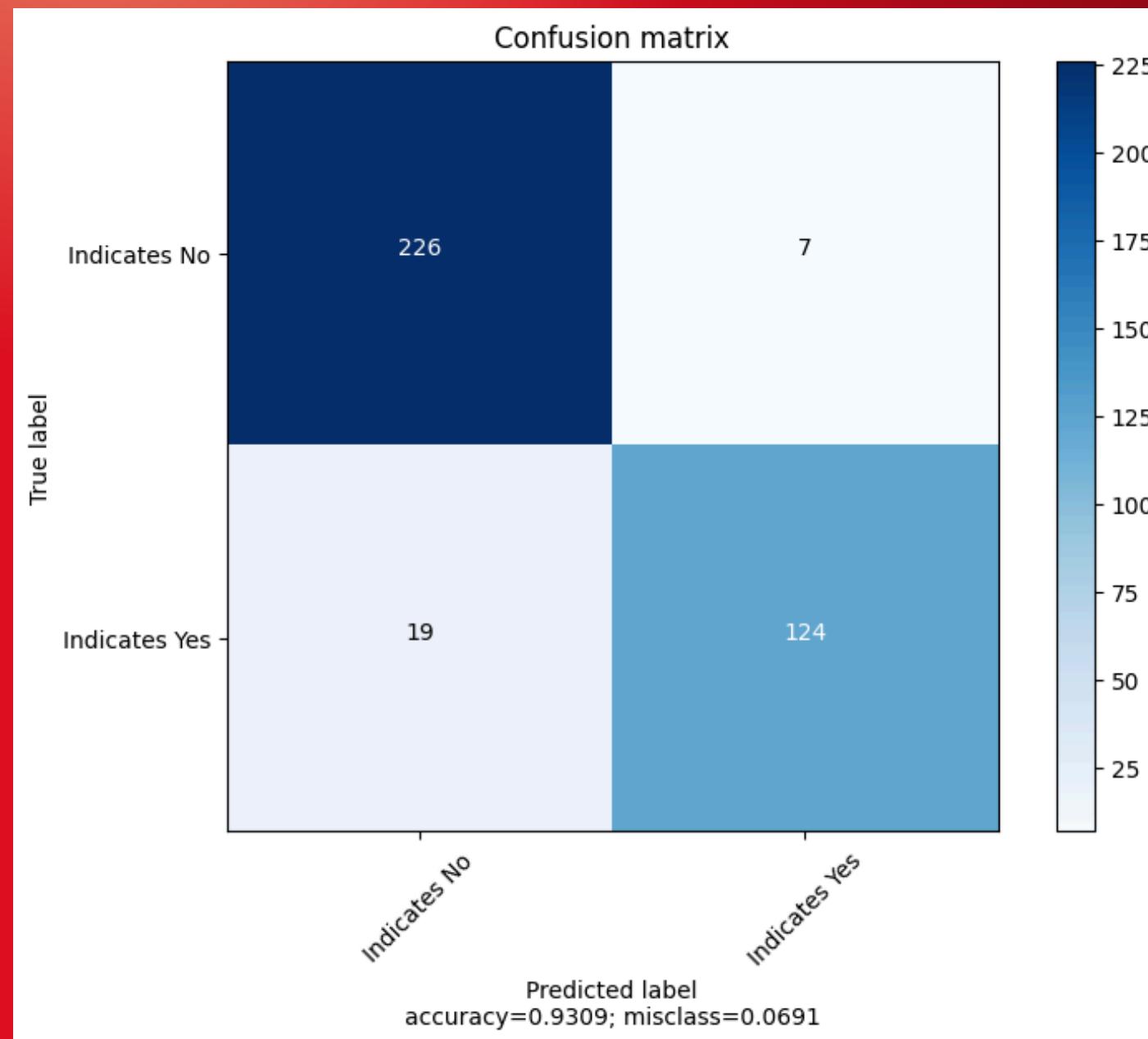
Pembahasan

XGBoost Classifier



Hasil analisis model XGBoost Classifier menunjukkan performa yang sangat baik dengan nilai **ROC AUC sebesar 0.9551**, yang menunjukkan kemampuan model membedakan kelas positif dan negatif secara efektif. Nilai **KS score yang tinggi (0.8549)** dan **p-value yang kecil** menegaskan bahwa model dapat memisahkan distribusi kedua kelas dengan jelas. Berdasarkan output, model menunjukkan **precision 0.92 dan recall 0.97** untuk kelas **negatif (0)**, serta precision **0.95 dan recall 0.87** untuk kelas **positif (1)**. Meskipun recall untuk kelas positif sedikit lebih rendah, **akurasi keseluruhan mencapai 93%**, dengan **F1-score yang tinggi (0.95** untuk kelas negatif dan 0.91 untuk kelas positif), mengindikasikan keseimbangan yang baik antara precision dan recall.

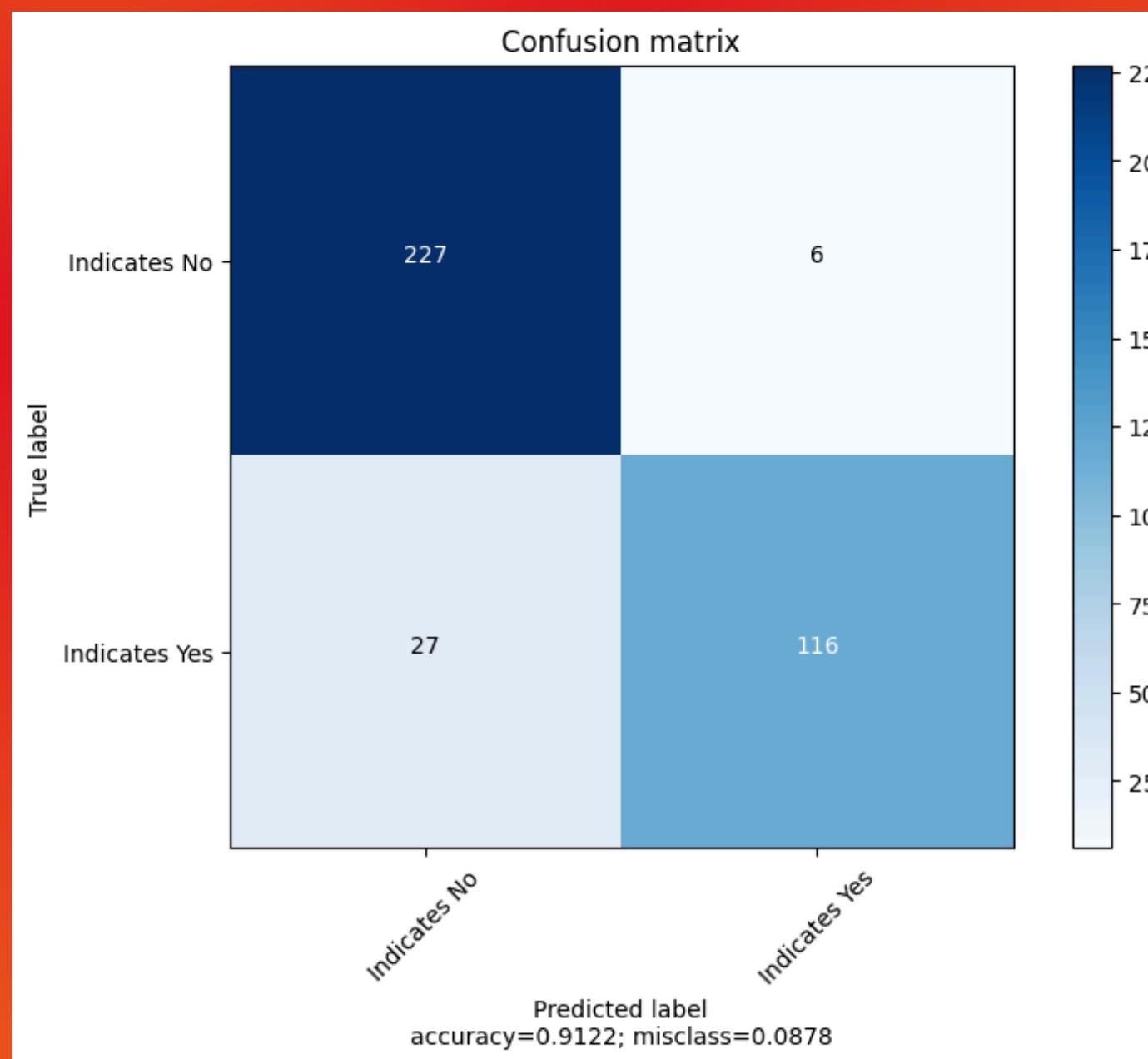
Evaluasi XGBoost Classifier



- **ROC AUC:** Model XGBoost Classifier memiliki ROC AUC sebesar 0.9551, menunjukkan kemampuan yang sangat baik dalam membedakan kelas positif dan negatif.
- **KS Score & p-value:** Nilai KS score 0.8549 dengan p-value kecil menunjukkan model dapat memisahkan distribusi kelas dengan jelas.
- **Kelas Negatif (0):** Precision 0.92 dan recall 0.97, menandakan model sangat andal dalam mendeteksi data negatif dengan kesalahan rendah.
- **Kelas Positif (1):** Precision tinggi 0.95, tetapi recall sedikit lebih rendah di 0.87, menunjukkan beberapa kasus positif masih terlewat.
- **Akurasi:** Model mencapai akurasi 93%, menunjukkan efektivitas yang tinggi dalam mengenali pola data secara umum.
- **F1-Score:** F1-score yang tinggi pada kedua kelas menunjukkan keseimbangan yang baik antara precision dan recall.
- **Perbaikan:** Perlu meningkatkan recall untuk kelas positif, bisa dilakukan dengan menyesuaikan threshold prediksi atau menggunakan teknik oversampling untuk menangani ketidakseimbangan data.

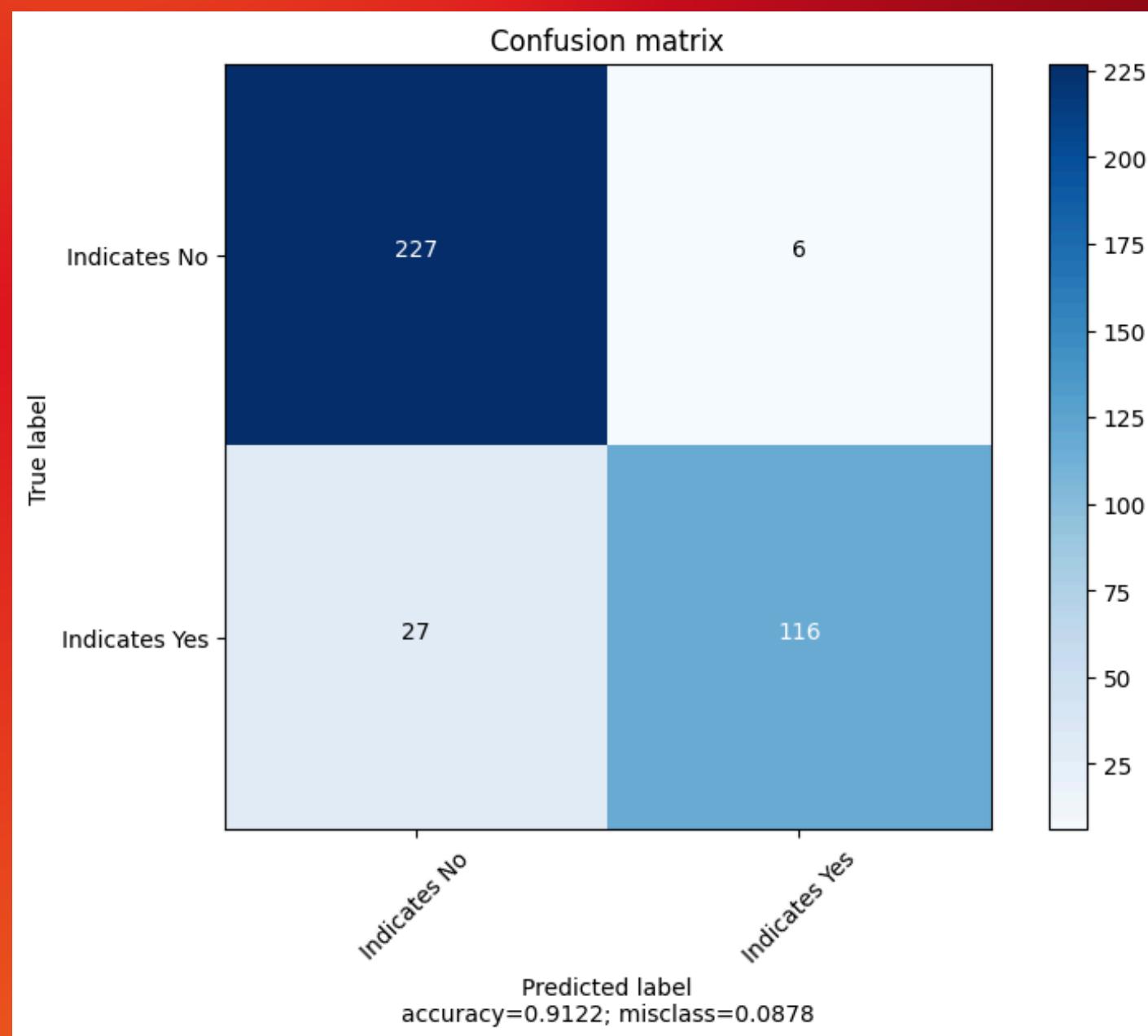
Pembahasan

Random Forest Classifier



Model Random Forest Classifier menunjukkan performa yang baik dengan **AUC sebesar 0.9444**, yang menunjukkan kemampuan diskriminasi yang sangat baik antara kelas positif dan negatif. **Precision untuk kelas positif tinggi, namun recall sedikit lebih rendah**, menunjukkan model cenderung menghindari kesalahan positif tetapi terkadang melewatkannya beberapa kasus positif. Untuk kelas negatif, model menunjukkan keseimbangan yang baik antara precision dan recall, dengan **akurasi keseluruhan mencapai 90%**. Nilai **F1-Score yang tinggi** menandakan keseimbangan yang baik antara akurasi dan deteksi. Perbaikan seperti penyesuaian threshold atau teknik oversampling pada kelas positif dapat meningkatkan deteksi kasus positif. Penyesuaian threshold atau teknik oversampling pada kelas minoritas.

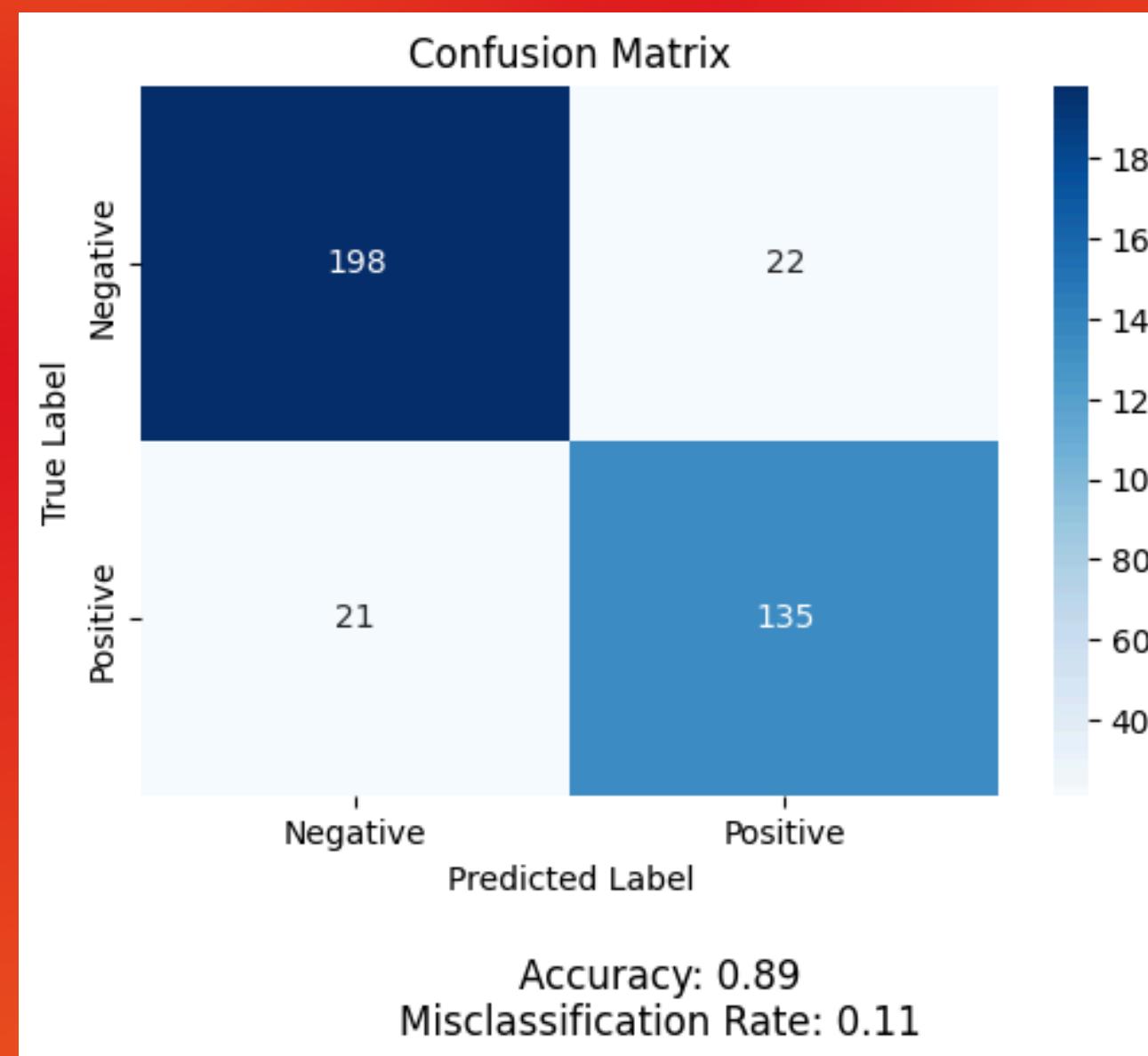
Evaluasi Random Forest Classifier



- **AUC:** Model Random Forest Classifier memiliki AUC 0.9444, menandakan kemampuan yang sangat baik dalam membedakan kelas positif dan negatif.
- **KS Score & p-value:** KS score 0.8441 dengan p-value kecil menunjukkan distribusi antara kedua kelas terpisah dengan jelas, menunjukkan kualitas diskriminasi yang sangat baik.
- **Kelas Positif (1):** Precision 0.95, menunjukkan hampir semua prediksi positif benar, namun recall 0.79 menunjukkan beberapa kasus positif terlewat.
- **Kelas Negatif (0):** Precision 0.88, recall 0.97, dan F1-score 0.93 menunjukkan model sangat baik dalam mendeteksi kelas negatif.
- **Akurasi:** Model mencapai akurasi 90%, sangat handal dalam prediksi kedua kelas.
- **Perbaikan:** Rendahnya recall untuk kelas positif menunjukkan potensi perbaikan dengan menurunkan threshold atau menggunakan oversampling untuk meningkatkan deteksi kasus positif.

Pembahasan

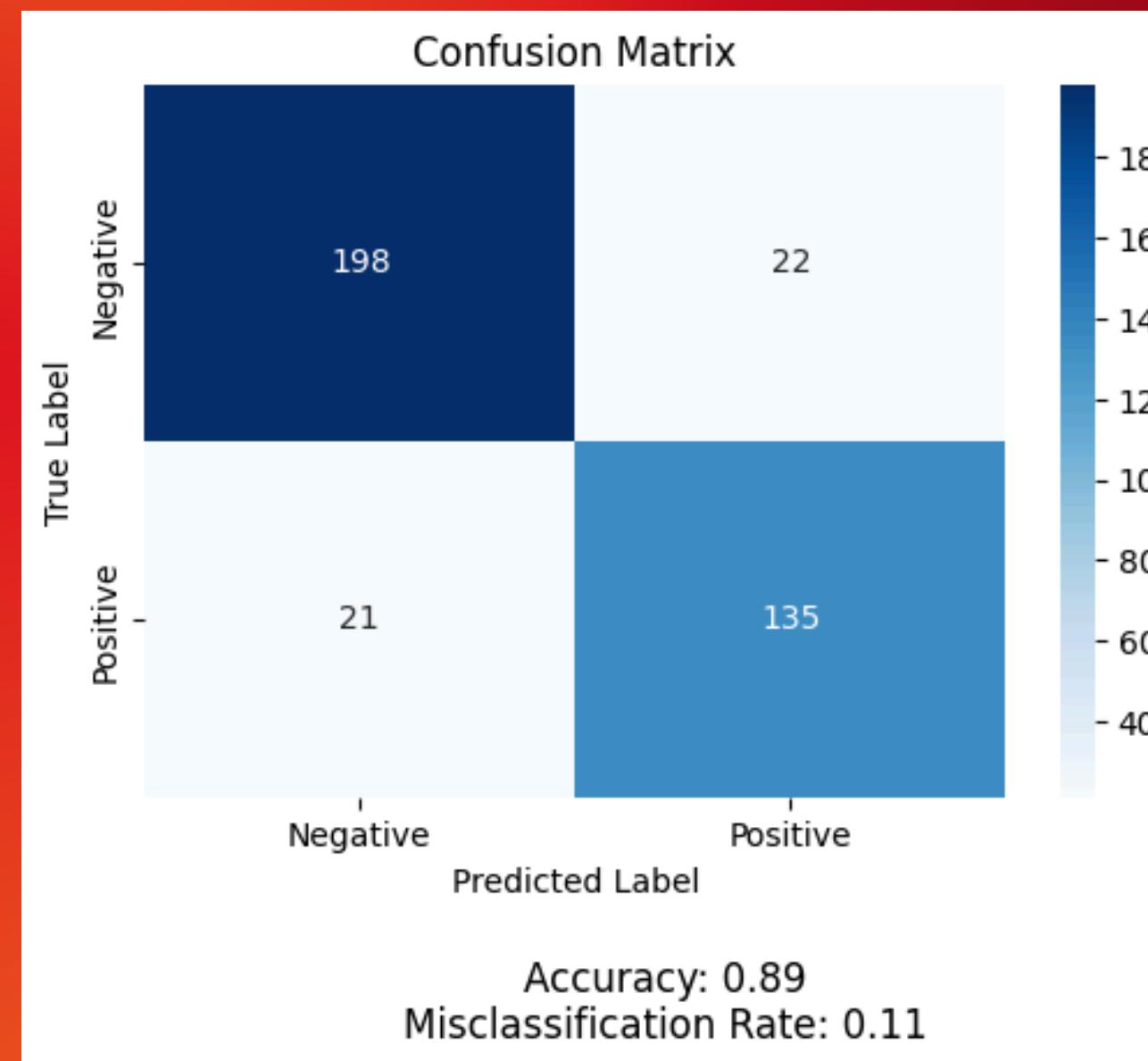
Decision Tree Classifier



Model DecisionTreeClassifier menunjukkan hasil yang cukup baik dengan **accuracy sebesar 0.89**. **Precision** dan **recall** untuk **kelas 0 (negatif)** cukup tinggi, yaitu **0.90** dan **0.90**, sementara untuk **kelas 1 (positif)**, **precision adalah 0.86** dan **recall 0.87**. Ini menunjukkan bahwa model dapat mengenali kelas 1 dengan baik, namun ada sedikit kekurangan dalam hal precision pada kelas 1, yang berarti model terkadang salah mengklasifikasikan beberapa kasus positif sebagai negatif. Selain itu, **f1-score** untuk kelas 1 adalah **0.86**, yang masih cukup baik namun dapat ditingkatkan lebih jauh.

Evaluasi

Decision Tree Classifier



- **Akurasi:** Meskipun akurasi mencapai 0.87, masih ada ruang untuk perbaikan dalam keseimbangan precision dan recall.
- **Perbaikan Kinerja:** Hyperparameter tuning, pruning, atau feature engineering dapat meningkatkan kinerja model lebih lanjut.
- **Ketidakseimbangan Kelas:** Teknik seperti SMOTE atau metode ensemble (misalnya Random Forest) bisa digunakan untuk meningkatkan kemampuan model mengenali kelas minoritas.
- **Kesimpulan:** Model sudah cukup baik, namun beberapa perbaikan masih diperlukan, terutama untuk meningkatkan prediksi pada kelas positif.

Kesimpulan

Model **XGBoost** menunjukkan kinerja sangat baik dengan hasil **AUC 0.551** dan **KS score 0.8549**, serta precision dan recall-nya yang tinggi untuk kelas negatif. Namun, recall untuk kelas positif bisa ditingkatkan lagi seperti melalui **penyesuaian threshold** atau **teknik oversampling**. Akurasi dari model ini adalah **93%**.

Untuk model **Random Forest**, **performa yang ditunjukkan juga baik dengan hasil AUC 0.9444 dan KS score 0.8441**, serta precision untuk kelas positif yang juga tinggi. Akan tetapi, recall untuk kelas positif lebih rendah. Akurasi keseluruhan dari model ini adalah **91%**.

Decision Tree Classifier memberikan hasil yang cukup baik dengan akurasi **89%**. Precision dan recall untuk kedua kelas cukup seimbang, namun performanya masih sedikit di bawah XGBoost dan Random Forest. Model ini dapat ditingkatkan dengan melakukan tuning hyperparameter, pruning, atau menggabungkannya dalam metode ensemble seperti Random Forest untuk hasil yang lebih optimal.

Secara keseluruhan, **XGBoost Classifier** adalah model terbaik untuk kasus ini. Namun, untuk meningkatkan deteksi kelas positif, penyesuaian threshold, balancing data, dan teknik lainnya masih diperlukan, agar performa pada kelas negatif tetap optimal.

Terima kasih 

