Analisis Prediktif Penyakit Tiroid Menggunakan Teknik Stacking Ensemble dengan Optimasi Optuna

Kelompok 8 - LC01

2702241894 - Derren Exellius Kurniawan
2702227630 - Edward Nathanael Setia Budi
2702256845 - Jecelyn Grizha
2702254474 - Karina Alexandra Sukamto
2702257173 - Tiffany Michelle Sugiono
2702223172 - Vincentia Angelica Boone

LATAR BELAKANG

Pentingnya Kesehatan Tiroid

- Kelenjar tiroid mengatur metabolisme, suhu tubuh, dan detak jantung.
- Gangguan tiroid dapat menyebabkan kelelahan, berat badan tidak stabil, hingga gangguan jantung.

Masalah yang Dihadapi

- Gejala sering tidak disadari karena mirip penyakit umum.
- Jika terlambat ditangani → bisa menjadi kondisi kronis.

Kenapa Perlu Diklasifikasi Dini?

- Diagnosis dini = penanganan lebih cepat & akurat.
- Data medis (hasil lab, hormon, dll.) dapat dimanfaatkan untuk mengenali jenis gangguan tiroid.

Data Understanding

Deskripsi Dataset

Dataset yang digunakan:

https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-disease-data

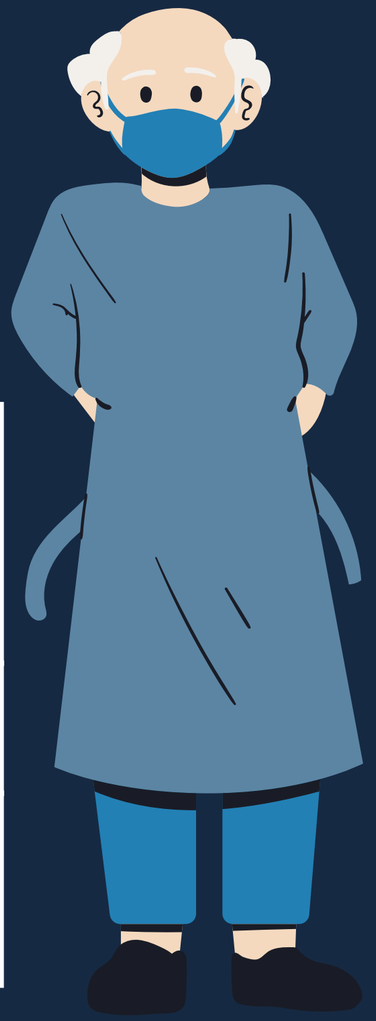
Berisi kumpulan data klinis pasien yang berpotensi mengalami gangguan tiroid, dikumpulkan melalui:

- Rekam medis elektronik (Electronic Health Records)
- Hasil laboratorium dari berbagai institusi kesehatan

Dataset ini mencakup informasi seperti hasil uji fungsi tiroid, gejala klinis, dan riwayat medis pasien yang relevan untuk analisis dan prediksi penyakit tiroid.

Data Understanding

Fitur dalam Dataset



Demografi Pasien

Fitur	Tipe Data	Deskripsi
age	Integer	Usia pasien dalam tahun
sex	String	Jenis kelamin pasien (‘M’ atau ‘F’)

Data Understanding

Fitur dalam Dataset

Riwayat Pengobatan dan Kondisi Klinis

Fitur	Tipe Data	Deskripsi
on_thyroxine	Boolean	Apakah pasien menggunakan hormon tiroksin
query_on_thyroxine	Boolean	Apakah ada dugaan pasien menggunakan tiroksin
on_antithyroid_meds	Boolean	Apakah pasien menggunakan obat anti tiroid
sick	Boolean	Apakah pasien sedang sakit
pregnant	Boolean	Apakah pasien sedang hamil
thyroid_surgery	Boolean	Riwayat operasi tiroid
I131_treatment	Boolean	Riwayat terapi Iodium-131 untuk tiroid

Data Understanding

Fitur dalam Dataset

Gejala dan Riwayat Penyakit Lain

Fitur	Tipe Data	Deskripsi
query_hypothyroid	Boolean	Dugaan pasien memiliki hipotiroidisme
query_hyperthyroid	Boolean	Dugaan pasien memiliki hipertiroidisme
lithium	Boolean	Konsumsi lithium yang dapat mempengaruhi tiroid
goitre	Boolean	Apakah ada gondok
tumor	Boolean	Riwayat tumor
hypopituitary	Boolean/Float	Gangguan hipopituitarisme
psych	Boolean	Riwayat gangguan psikiatrik

Data Understanding

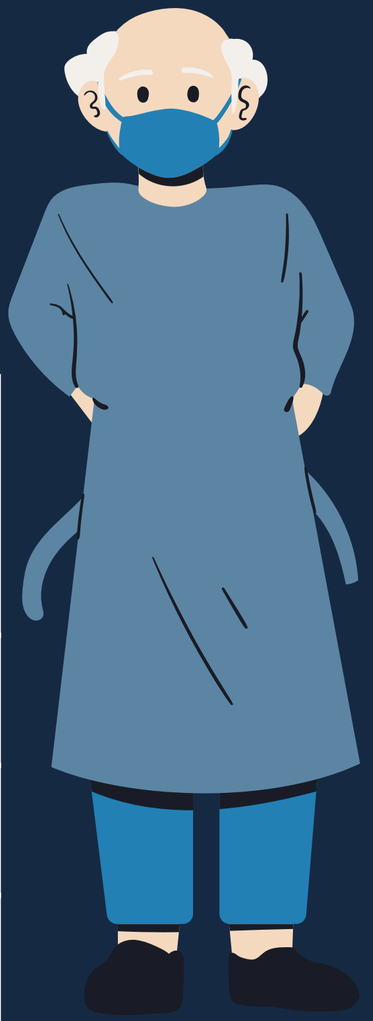
Fitur dalam Dataset

Hasil Laboratorium

Fitur	Tipe Data	Deskripsi
TSH_measured, TSH	Boolean, Float	Apakah TSH diukur dan nilainya
T3_measured, T3	Boolean, Float	Apakah T3 diukur dan nilainya
TT4_measured, TT4	Boolean, Float	Apakah Total T4 diukur dan nilainya
T4U_measured, T4U	Boolean, Float	Apakah T4 uptake diukur dan nilainya
FTI_measured, FTI	Boolean, Float	Apakah Free Thyroxine Index diukur dan nilainya
TBG_measured, TBG	Boolean, Float	Apakah TBG diukur dan nilainya

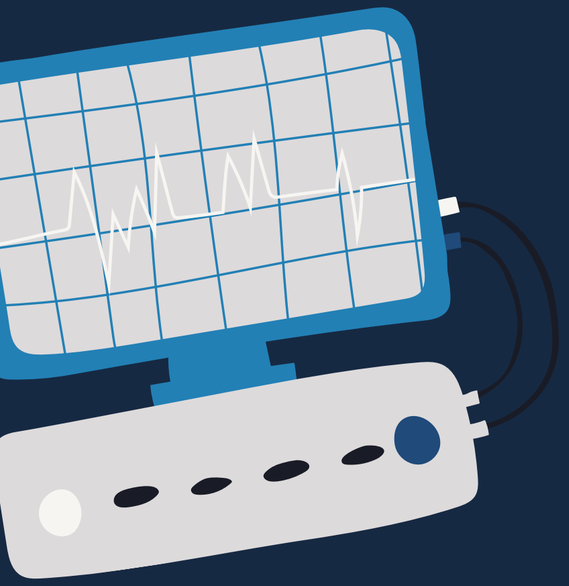
Data Understanding

Fitur dalam Dataset



Informasi Tambahan

Fitur	Tipe Data	Deskripsi
referral_source	String	Sumber rujukan pasien
target	String	Diagnosis akhir (label)
patient_id	String	ID pasien (untuk identifikasi)



Data Understanding

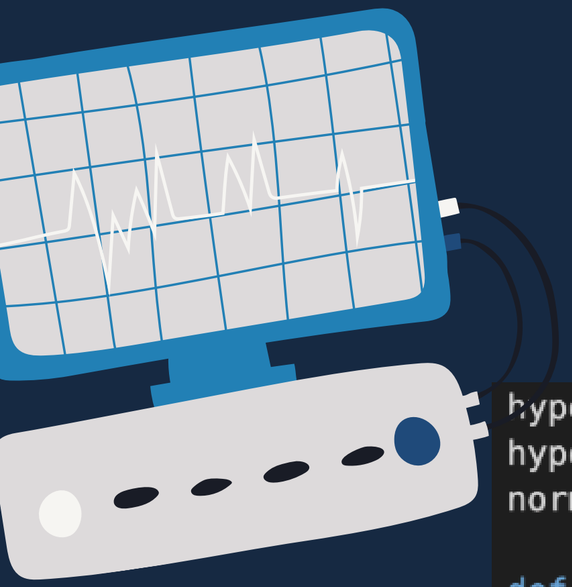
Analisis Awal

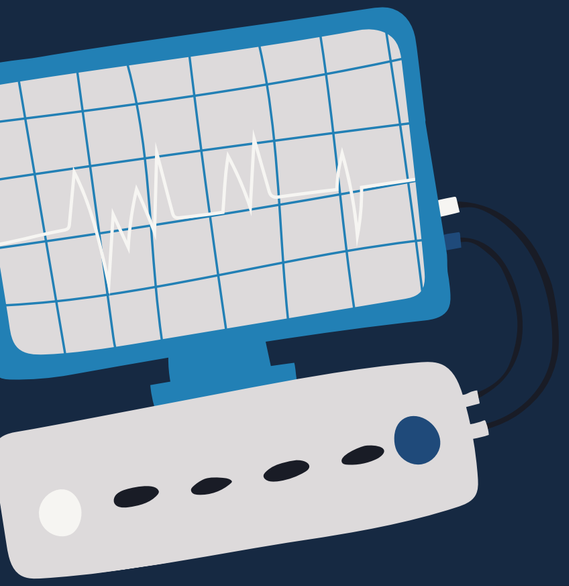
```
df.head()  #Melihat 5 baris pertama
df.describe()  #Statistik deskriptif
df.isna().sum()  #Cek missing values
df['target'].unique()  #Nilai unik target
```

Data Understanding

Analisis Variabel Target

```
hyperthyroid_conditions = ['A', 'B', 'C', 'D', 'O', 'P', 'Q', 'S', 'T']
hypothyroid_conditions = ['E', 'F', 'G', 'H']
normal_conditions = ['-']

def categorize_target(value):
    diagnoses = value.split('|')
    for diagnosis in diagnoses:
        if diagnosis in hyperthyroid_conditions:
            return 'Hyperthyroid'
    for diagnosis in diagnoses:
        if diagnosis in hypothyroid_conditions:
            return 'Hypothyroid'

    for diagnosis in diagnoses:
        if diagnosis in normal_conditions:
            return 'Healthy'

                'target'].apply(categorize_target)
```



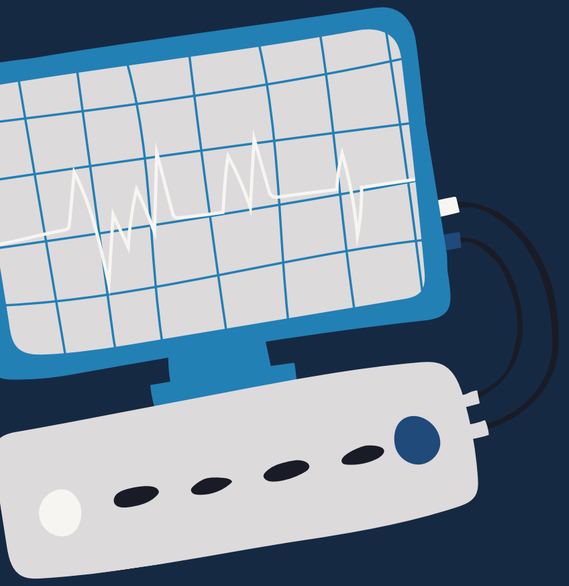


Data Understanding

Masalah Data

```
[ ]  df['target'].isnull().sum()
```

```
np.int64(1487)
```



Data Understanding

Hasil Target

```
df['target'].unique()
```

```
array(['Healthy', 'Hyperthyroid', 'Hypothyroid', None], dtype=object)
```

Data Cleaning

Penanganan Missing Values pada Kolom Kategorik

Menggunakan SimpleImputer untuk mengisi null value dengan nilai modus

```
[ ]  df['sex'] = df['sex'].fillna(df['sex'].mode()[0])
```

```
[ ]  df['sex'].isna().sum()
```

```
np.int64(0)
```

Data Cleaning

Penanganan Missing Values pada Kolom Numerik

Menggunakan SimpleImputer untuk mengisi null value dengan nilai 0

```
[ ]  columns_to_fill = ['TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']
     df[columns_to_fill] = SimpleImputer(strategy='constant', fill_value=0).fit_transform(df[columns_to_fill])
```

Data Cleaning

Penghapusan Kolom Redundan

```
[ ]  columns_to_drop = ['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured', 'TBG_measured', 'patient_id']
     df.drop(columns_to_drop, axis = 1, inplace = True)
```

```
[ ]  df.columns
```

```
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_meds', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U',
       'FTI', 'TBG', 'referral_source', 'target'],
      dtype='object')
```

Data Cleaning

Identifikasi Data Tidak Konsisten

```
[ ]  df[(df['sex']=='Male') & (df['pregnant'] == 't')]
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyro

0 rows × 23 columns

Data Cleaning

Penyimpanan Kolom Kategorik

```
obj_col = []
for i in df.columns:
    if df[i].dtype == 'object':
        print(f'Unique values in {i}: {df[i].unique()}')
        obj_col.append(i)
```

Data Encoding

Label Encoding untuk Kolom Kategorik

```
[ ]  print(df['referral_source'].unique()) #Mengecek nilai unik di kolom 'referral_source' sebelum encoding

     ['other' 'SVI' 'SVHC' 'STMW' 'SVHD' 'WEST']

[ ]  le = LabelEncoder() #Inisialisasi LabelEncoder

[ ]  df['sex'] = df['sex'].map({'F':0, 'M':1}) #Encoding kolom 'sex'

[ ]  df['referral_source'] = le.fit_transform(df['referral_source']) #Encoding kolom 'referral_source'
```

Data Encoding

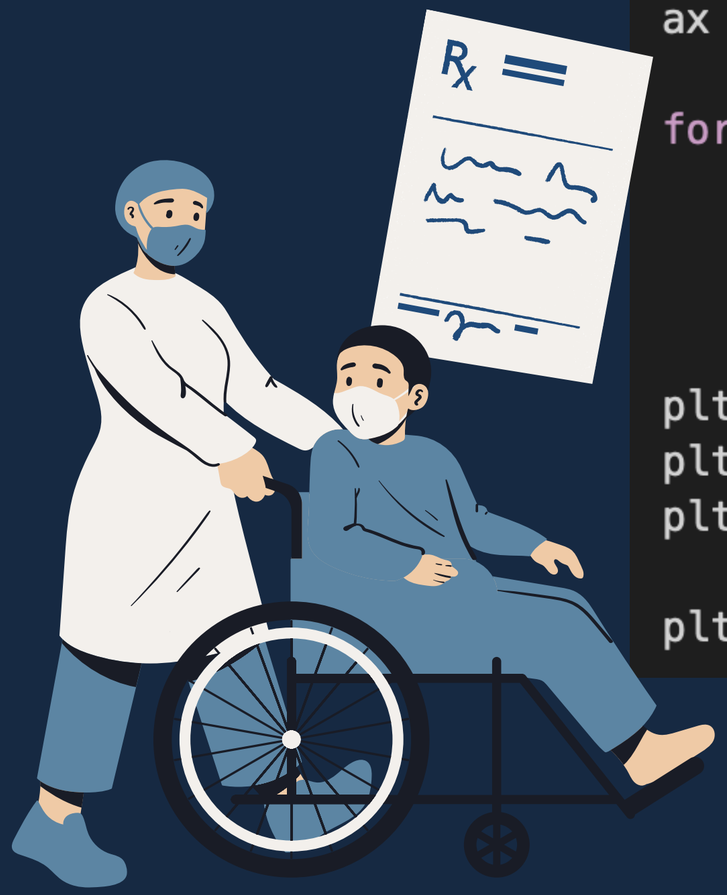
Binary Encoding untuk Kolom Boolean

```
[ ]  #Mengkonversi semua kolom object (kecuali 'target') dari 't'/'f' ke 0/1
     for i in df.columns:
         if df[i].dtype == 'object' and i != 'target':
             df[i] = df[i].map({'f':0, 't':1})
```

```
     #Verifikasi hasil encoding
     for i in obj_col:
             print(f'Unique values in {i}: {df[i].unique()}')
```

Data Visualization

Distribusi Variabel Target



```
plt.figure(figsize=(8, 5))
ax = sns.countplot(x='target', data=df, palette='viridis')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=12)

plt.title('Distribution of Target Variable')
plt.xlabel('Target')
plt.ylabel('Count')

plt.show()
```

Data Visualization

Distribusi Gender per Kategori Target

```
plt.figure(figsize=(9,6))

ax = sns.countplot(x='target', hue='sex', data=df, palette='viridis')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=10)

plt.title('Distribution of Sex per Target Category')
plt.xlabel('Target')
plt.ylabel('Count')
plt.legend(title='Sex')

plt.show()
```

Data Split

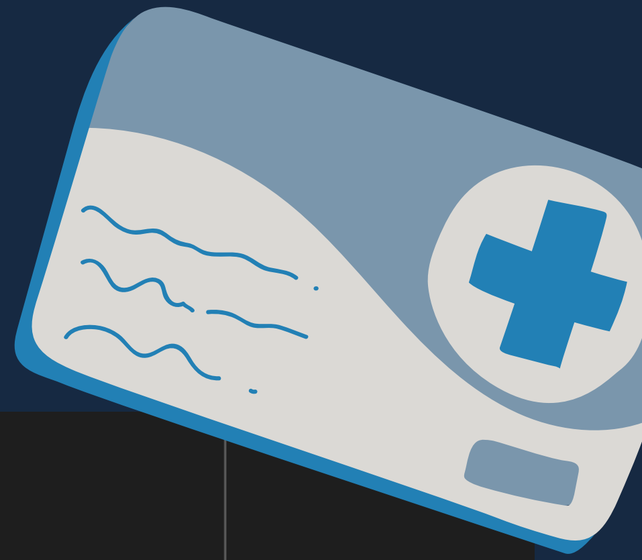
```python
#Memisahkan fitur (X) dan target (y)
x = df.drop('target', axis = 1)
y = df['target']
```

```python
#Split data dengan proporsi 20% testing dan 80% training
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)
```

```python
#Verifikasi distribusi kelas di data training
np.unique(y_train)
```

Data Modelling

Preprocessing untuk Model



```
import numpy as np
import optuna
from catboost import CatBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.metrics import accuracy_score, average_precision_score
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

target_map = {'Hyperthyroid': 0, 'Hypothyroid': 1, 'Healthy': 2}
y_train_encoded = y_train.map(target_map)
y_test_encoded = y_test.map(target_map)

y_test_binarized = label_binarize(y_test, classes=['Hyperthyroid', 'Hypothyroid', 'Healthy'])

sample_weights_cb = compute_sample_weight(class_weight='balanced', y=y_train)
y_train_encoded = np.vectorize(target_map.get)(y_train)
sample_weights_xgb = compute_sample_weight(class_weight='balanced', y=y_train_encoded)
```

Data Modelling

Pelatihan Model Individual

```
# CatBoost
cat = CatBoostClassifier(auto_class_weights='Balanced', verbose=False, random_state=42)
cat.fit(x_train, y_train)

# GradientBoosting
gb = GradientBoostingClassifier(random_state=10)
gb.fit(x_train, y_train, sample_weight=sample_weights_cb)

# XGBoost
xgb = XGBClassifier(random_state=10, use_label_encoder=False, eval_metric='mlogloss')
xgb.fit(x_train, y_train_encoded, sample_weight=sample_weights_xgb)
```

Data Modelling

Probabilistic Stacking & Hyperparameter Tuning Menggunakan Optuna

```
cat_proba = cat.predict_proba(x_test)
gb_proba = gb.predict_proba(x_test)
xgb_proba = xgb.predict_proba(x_test)

def objective(trial):
    w_cat = trial.suggest_float("w_cat", 0, 1)
    w_gb = trial.suggest_float("w_gb", 0, 1 - w_cat)
    w_xgb = 1 - w_cat - w_gb

    combined_proba = w_cat * cat_proba + w_gb * gb_proba + w_xgb * xgb_proba

    score = average_precision_score(y_test_binarized, combined_proba, average='micro')
    return score

study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=100)

#Mengambil bobot terbaik
best_weights = study.best_params
w_cat = best_weights['w_cat']
w_gb = best_weights['w_gb']
w_xgb = 1 - w_cat - w_gb
```

Data Modelling

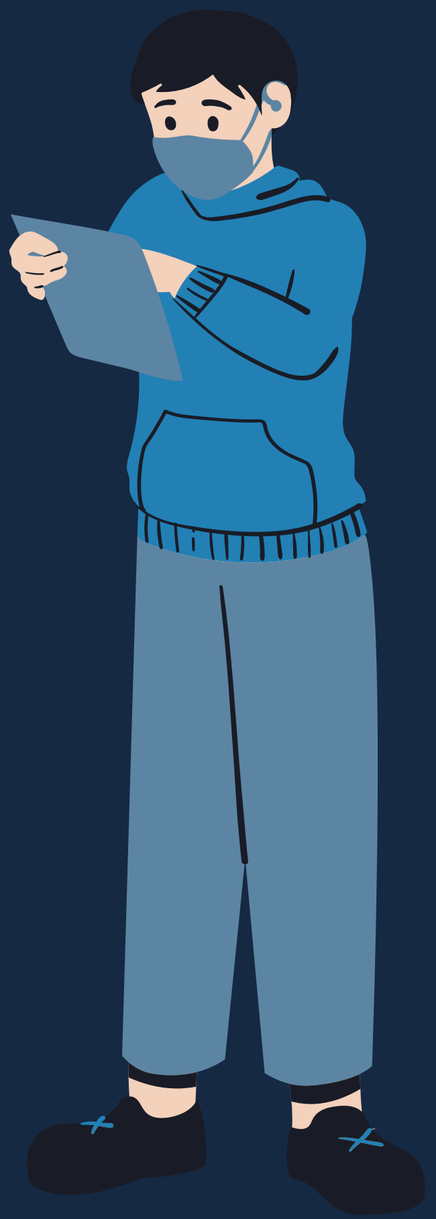
Evaluasi Akhir

```python
#Mengambil bobot terbaik
best_weights = study.best_params
w_cat = best_weights['w_cat']
w_gb = best_weights['w_gb']
w_xgb = 1 - w_cat - w_gb

combined_proba = w_cat * cat_proba + w_gb * gb_proba + w_xgb * xgb_proba
combined_pred = np.argmax(combined_proba, axis=1)

print("\nBest Weights Found by Optuna:")
print(f"CatBoost: {w_cat:.3f}, GradientBoosting: {w_gb:.3f}, XGBoost: {w_xgb:.3f}\n")
print(f"Confusion Matrix:\n{confusion_matrix(y_test_encoded, combined_pred)}")
print(f"Accuracy: {accuracy_score(y_test_encoded, combined_pred)}")
print(f"Micro Average Precision: {average_precision_score(y_test_binarized, combined_proba, average='micro')}")
```

Model Evaluation

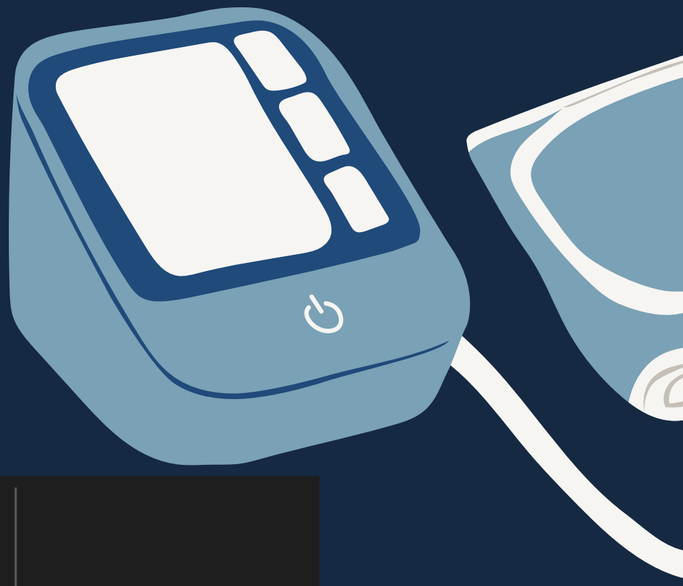
Confusion Matrix

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

disp = ConfusionMatrixDisplay.from_predictions(
    y_test_encoded, combined_pred,
    display_labels=['Hyperthyroid', 'Hypothyroid', 'Healthy'],
    cmap=plt.cm.Blues,
    colorbar=False,
    xticks_rotation=45
)
plt.title('Confusion Matrix - Stacked Model')
plt.show()
```



Model Evaluation

Precisioin-Recall Curve per Kelas



```
from sklearn.metrics import precision_recall_curve

plt.figure(figsize=(8,6))

for i, class_label in enumerate(['Hyperthyroid', 'Hypothyroid', 'Healthy']):
    precision, recall, _ = precision_recall_curve(y_test_binarized[:, i], combined_proba[:, i])
    plt.plot(recall, precision, label=f'{class_label}')

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - Stacked Model')
plt.legend()
plt.grid(True)
plt.show()
```

Evaluasi Hasil

Metrics	Model Stacked
Accuracy	98.50%
Micro Average Precision	99.60%
Precision (Weighted Average)	98.53%
Recall (Weighted Average)	98.50%
F1-Score (Weighted Average)	98.51%
Precision (Macro Average)	93.15%
Recall (Macro Average)	94.89%
F1-Score (Macro Average)	94.00%

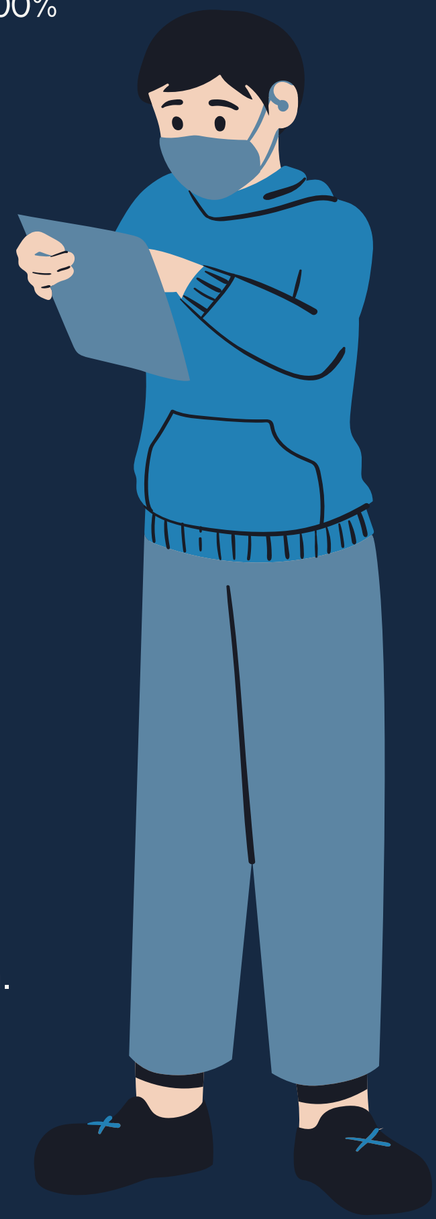
Metrics	Model Stacked
Accuracy	98.50%
Micro Average Precision	99.60%
Precision (Weighted Average)	98.53%
Recall (Weighted Average)	98.50%
F1-Score (Weighted Average)	98.51%
Precision (Macro Average)	93.15%
Recall (Macro Average)	94.89%
F1-Score (Macro Average)	94.00%

Accuracy

Formula: Prediksi Benar / Total Prediksi x 100%

Total prediksi: 1537 Kasus

Prediksi benar: 1514 Kasus
Prediksi salah: 23 Pasien



Micro Average Precision

Mengukur ketepatan prediksi dengan mempertimbangkan semua kelas secara bersamaan.

Weighted Average

Memberikan bobot lebih besar pada kelas dengan sampel lebih banyak

Performa termasuk baik karena model excellent pada kelas Healthy (kelas mayoritas

Macro Average

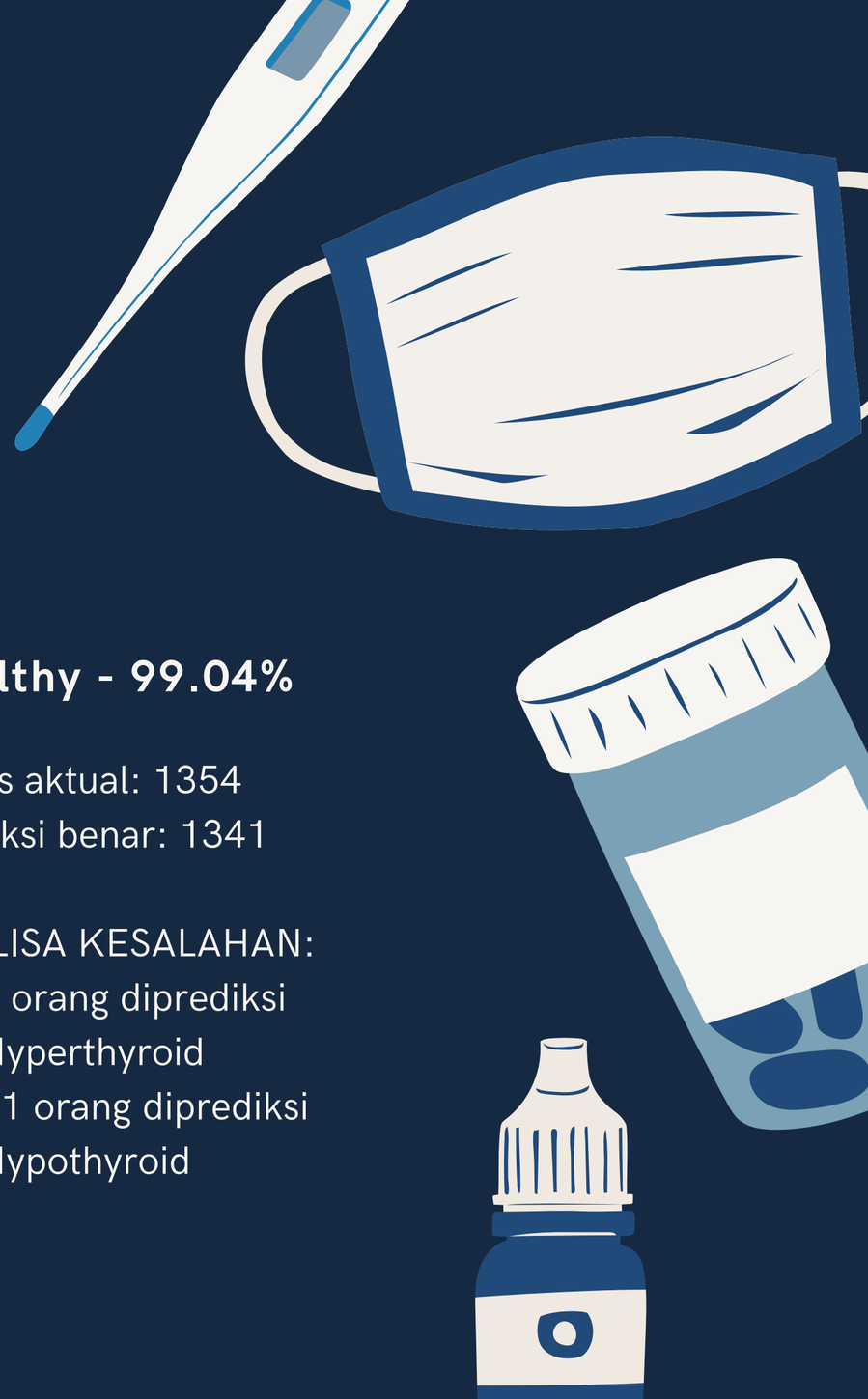
Menghitung rata - rata sederhana dari semua kelas

Sedikit lebih rendah karena challenge pada kelas Hyperthyroid (kelas minoritas)

Metrics	Model Stacked
Accuracy	98.50%
Micro Average Precision	99.60%
Precision (Weighted Average)	98.53%
Recall (Weighted Average)	98.50%
F1-Score (Weighted Average)	98.51%
Precision (Macro Average)	93.15%
Recall (Macro Average)	94.89%
F1-Score (Macro Average)	94.00%

Confusion Matrix

Memberikan visualisasi detail mengenai performa klasifikasi model untuk setiap kelas secara spesifik, memaparkan jumlah prediksi yang benar dan yang keliru.



Hyperthyroid - 87.30%

Kasus aktual: 63
Prediksi benar: 55

ANALISA KESALAHAN:

- 6 orang diprediksi Healthy
- 2 orang diprediksi Hypothyroid

Hypothyroid- 98.33%

Kasus aktual: 120
Prediksi benar: 118

ANALISA KESALAHAN:

- 2 orang diprediksi Healthy

Healthy - 99.04%

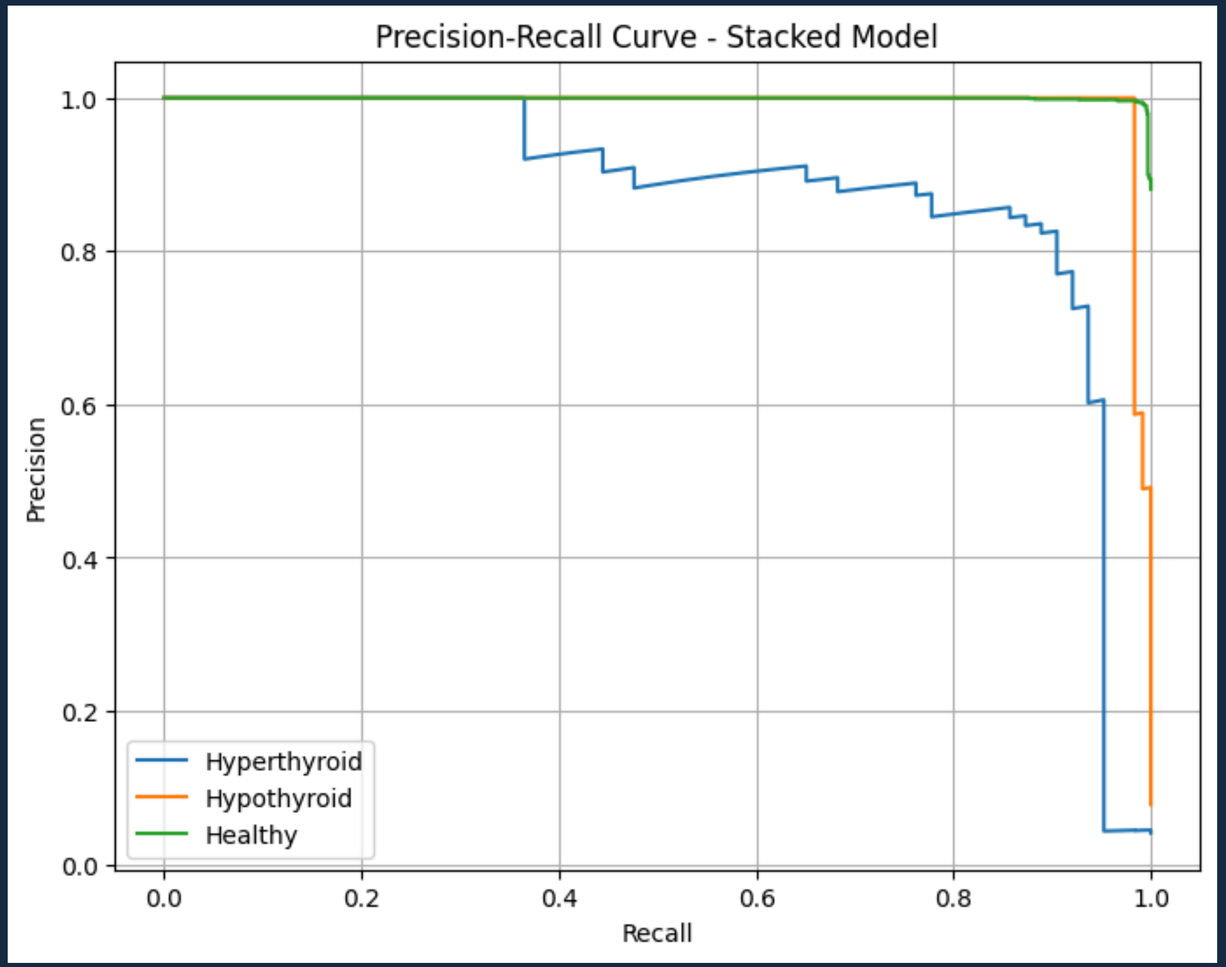
Kasus aktual: 1354
Prediksi benar: 1341

ANALISA KESALAHAN:

- 2 orang diprediksi Hyperthyroid
- 11 orang diprediksi Hypothyroid

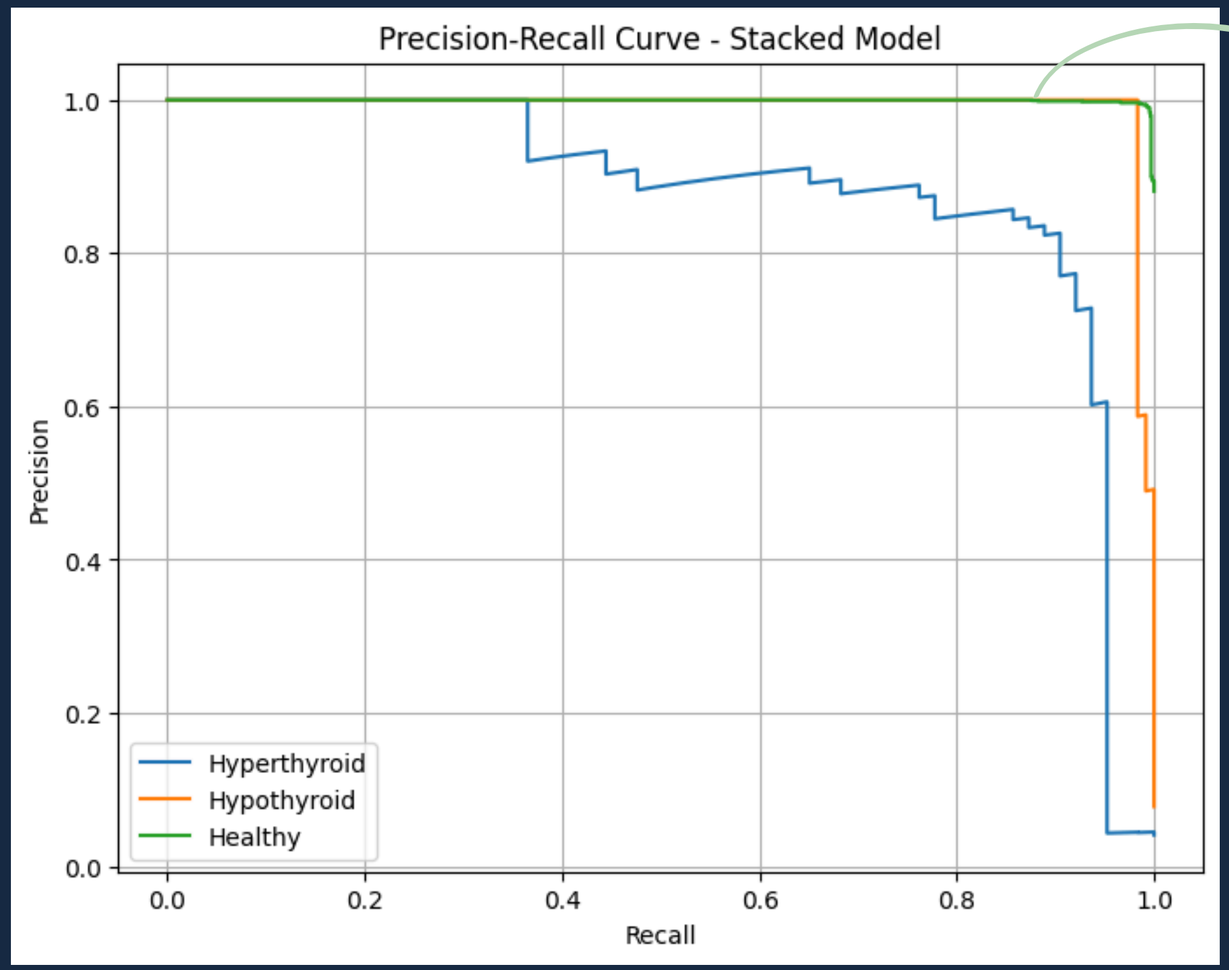
Precision-Recall Curve

Memberikan insight mendalam tentang evaluasi ketepatan dan kelengkapan model pada setiap kelas, terutama pada dataset dengan distribusi kelas yang tidak seimbang



Healthy Class

Average Precision: 0.999355



Kurva hampir mencapai sudut kanan atas → hampir sempurna

Precision sangat tinggi secara konsisten di berbagai tingkat recall

Makna:

- Model sangat confident dalam mengidentifikasi orang sehat
- Hampir tidak ada false positive

Hypothoroid Class

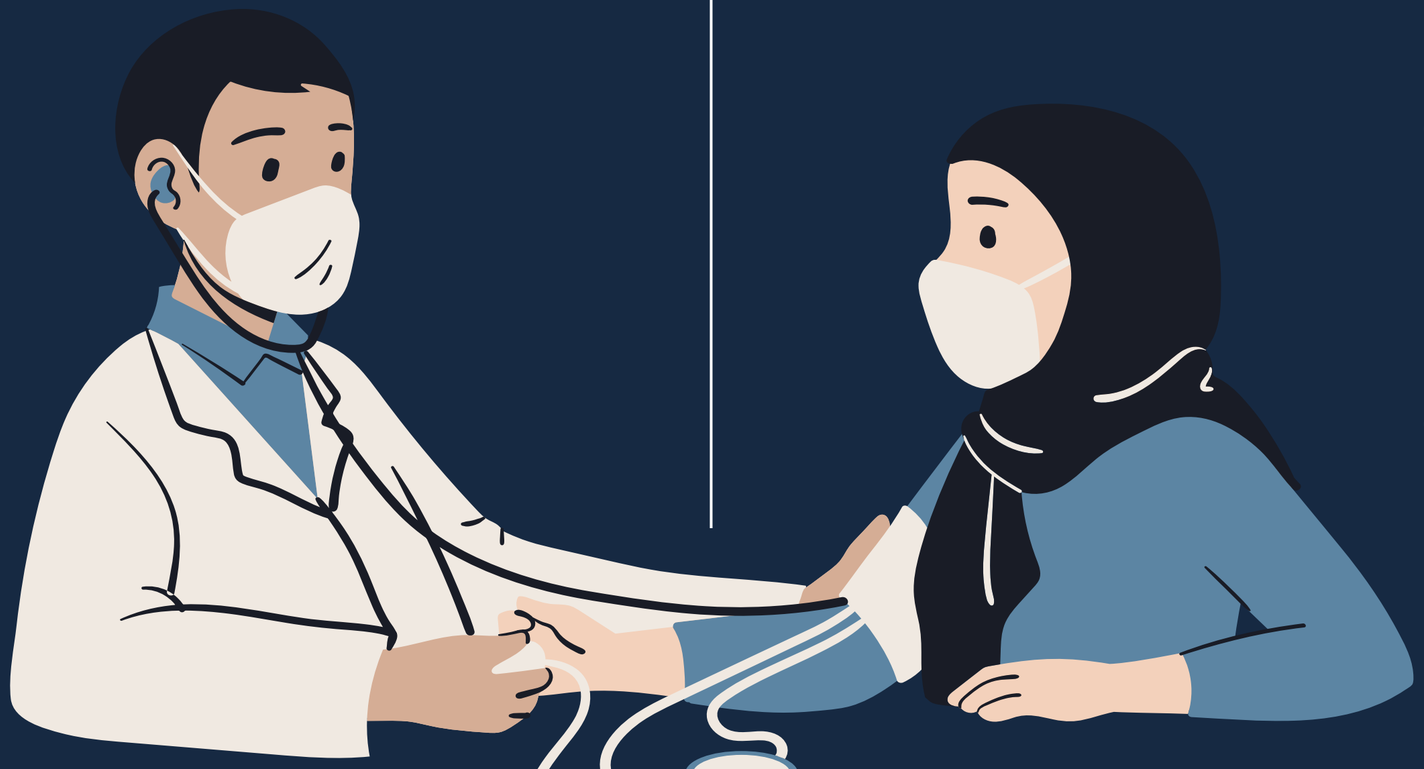
Average Precision: 0.992341

Kurva sangat baik dengan sedikit penurunan direcall tinggi

Konsistensi tinggi sepanjang spektrum threshold

Makna:

- Model reliable dalam mendeteksi Hypothyroid
- Balance yang baik antara precision dan recall



Hyperthoroid Class

Average Precision: 0.880135

Kurva mengindikasikan adanya trade-off yang lebih nyata antara presisi dan recall

Precision turun lebih cepat saat recall meningkat

Tantangan:

- Limited training samples menyebabkan generalisasi kurang optimal
- Model harus memilih antara miss beberapa kasus atau false alarm

Kontribusi Optuna

Optuna berperan krusial dalam menentukan bobot optimal untuk setiap model dasar dalam arsitektur stacking.

Proses Pencarian:

- Optuna mencoba ribuan kombinasi bobot secara sistematis
- Setiap kombinasi dievaluasi menggunakan Micro Average Precision
- Proses ini seperti ‘trial-and-error’ yang sophisticated

Hasil optimal:

- XGBoost: 98.7% - kontribusi paling dominan
- GradientBoost: 1.1%
- CatBoost: 0.2%

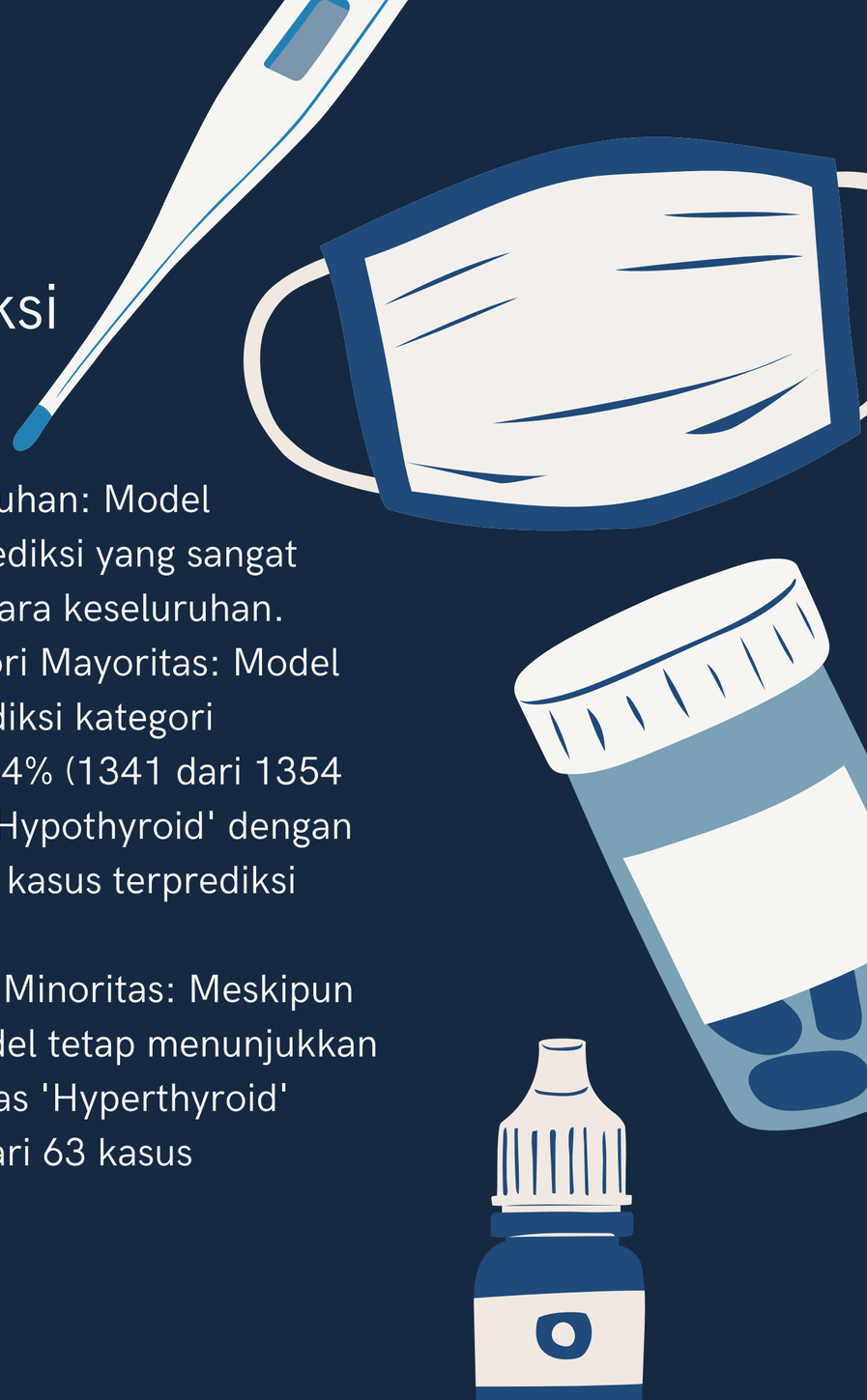
Kesimpulan

Performa Model

- Teknik Stacking Ensemble: Model dibangun menggunakan teknik Stacking Ensemble (CatBoost, GradientBoost, dan XGBoost)dan di optimasi dengan Optuna.
- Akurasi Tinggi: Model mencapai tingkat akurasi sebesar 98.50% dan Micro Average sebesar 99.60%.
- Kapabilitas Klasifikasi: Angka performa ini menegaskan kapabilitas model yang tinggi dalam mengklasifikasikan penyakit tiroid secara efektif.

Kemampuan Prediksi yang unggul

- Kemampuan Prediksi Keseluruhan: Model menunjukkan kemampuan prediksi yang sangat baik dengan akurasi tinggi secara keseluruhan.
- Performa Unggul pada Kategori Mayoritas: Model sangat efektif dalam memprediksi kategori 'Healthy' dengan akurasi 99.04% (1341 dari 1354 kasus terprediksi benar) dan 'Hypothyroid' dengan akurasi 98.33% (118 dari 120 kasus terprediksi benar).
- Performa Kuat pada Kategori Minoritas: Meskipun data sampel lebih sedikit, model tetap menunjukkan performa yang kuat untuk kelas 'Hyperthyroid' dengan akurasi 87.30% (55 dari 63 kasus terprediksi benar).



Kontribusi Optuna

Optimasi Krusial: Peran Optuna terbukti krusial dalam memaksimalkan performa model stacking dengan melakukan optimasi bobot secara efektif.

Penentuan Kontribusi Optimal: Optuna berhasil menentukan kontribusi optimal dari setiap model penyusun (base model): XGBoost (0.987), GradientBoost (0.011), dan CatBoost (0.002).

Dominasi XGBoost: Hasil optimasi menunjukkan bahwa XGBoost memiliki pengaruh yang dominan dalam model stacking, mengindikasikan efektivitas tertinggi dalam memprediksi penyakit tiroid pada dataset yang digunakan.

Keseluruhan

Efektivitas Pendekatan Stacking Ensemble: Penelitian ini berhasil menunjukkan bahwa pendekatan Stacking Ensemble dengan optimasi Optuna sangat efektif dalam membangun model prediktif untuk deteksi penyakit tiroid.

Aplikasi di Bidang Medis: Model ini memiliki potensi aplikatif yang signifikan untuk mendukung pengambilan keputusan yang lebih baik di bidang medis, khususnya dalam diagnosis penyakit tiroid.

Potensi Model yang Kuat: Model yang dihasilkan memiliki kemampuan prediktif yang kuat, terbukti dari performa tinggi yang dicapai.

Thank you!