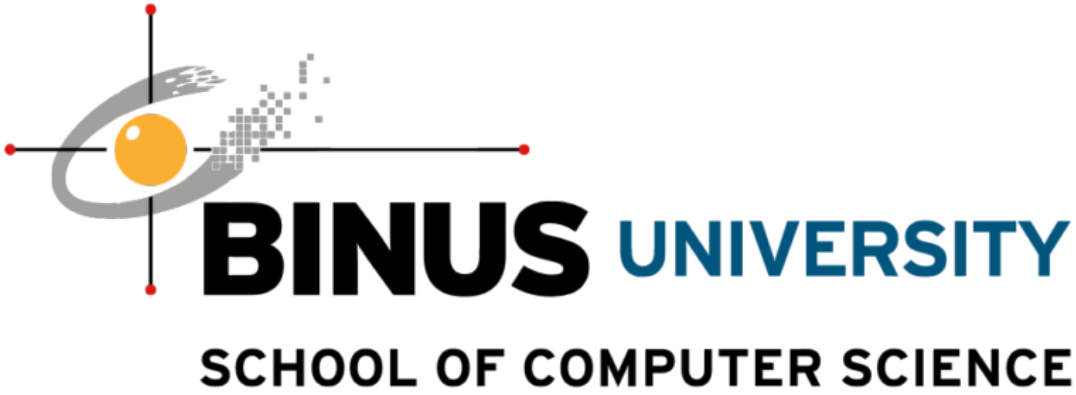
Analisis Prediktif Penyakit Tiroid Menggunakan Teknik *Stacking Ensemble* (*CatBoost*, *GradientBoost*, *XGBoost*) dengan Optimasi Optuna



Big Data Processing - COMP6579001

Oleh:

2702241894 - Derren Exellius Kurniawan

2702227630 - Edward Nathanael Setia Budi

2702256845 - Jecelyn Grizha

2702254474 - Karina Alexandra Sukamto

2702257173 - Tiffany Michelle Sugiono

2702223172 - Vincentia Angelica Boone

BAB I
LATAR BELAKANG

Penyakit tiroid merupakan gangguan pada kelenjar tiroid yang terletak di bagian depan leher yang memiliki peran penting dalam mengatur berbagai fungsi tubuh melalui produksi hormon. Kelenjar tiroid menghasilkan hormon seperti T3 (triiodotironin) dan T4 (tiroksin) yang mengatur metabolisme tubuh, suhu, denyut jantung, serta sistem saraf. Gangguan pada fungsi tiroid ini dapat menyebabkan berbagai kesehatan, mulai dari hipertiroidisme (kelebihan hormon) hingga hipotiroidisme (kekurangan hormon), serta kondisi lain seperti gondok, tiroiditis, dan kanker tiroid.

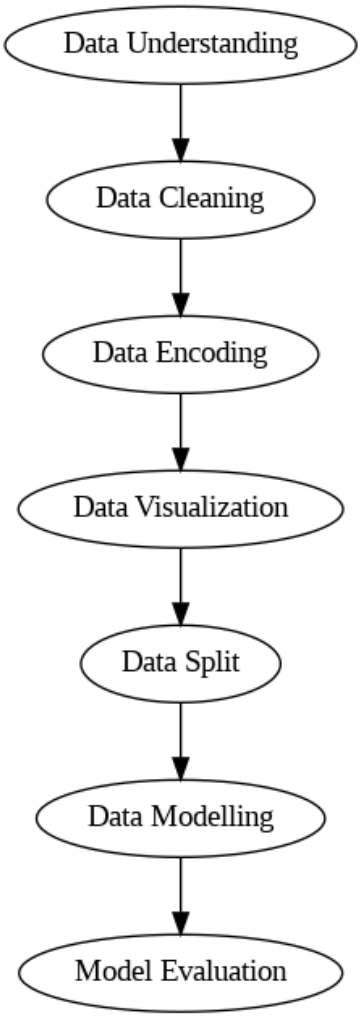
Gejala penyakit tiroid kerap kali tidak dikenali pada tahap awal karena bersifat umur, seperti kelelahan, perubahan berat badan, gangguan tidur, hingga depresi. Jika tidak ditangani secara tepat, gangguan tiroid dapat berkembang menjadi kondisi kronis yang berdampak pada kualitas hidup penderitanya. Oleh karena itu, diagnosis dini terhadap penyakit tiroid sangat penting untuk mencegah komplikasi lebih lanjut.

Di Indonesia maupun di seluruh dunia, jumlah penderita gangguan tiroid mengalami peningkatan setiap tahunnya. Deteksi penyakit yang cepat dan akurat menjadi tantangan tersendiri bagi tenaga medis, terutama ketika harus menganalisis berbagai data medis pasien yang kompleks. Dalam konteks ini, diperlukan pendekatan yang mampu membantu proses klasifikasi kondisi tiroid secara efektif berdasarkan data medis yang tersedia.

Dengan melakukan analisis terhadap data medis yang mencakup berbagai indikator seperti kadar hormon, hasil pemeriksaan laboratorium, serta riwayat medis lainnya, diharapkan dapat dilakukan prediksi dan klasifikasi jenis gangguan tiroid secara lebih akurat. Langkah ini sangat penting untuk mendukung proses pengambilan keputusan medis dan meningkatkan efektivitas pengobatan yang diberikan kepada pasien.

BAB II
METODOLOGI DAN ALUR PENGERJAAN

2.1. Alur Penelitian



2.2. Data Understanding

2.2.1. Deskripsi Dataset

Dataset yang kami gunakan diperoleh dari platform Kaggle (https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-disease-data), yang merupakan kumpulan data klinis pasien dengan adanya potensi gangguan tiroid. Data ini dikumpulkan melalui rekam medis elektronik dan hasil laboratorium dari institusi kesehatan.

Format: CSV dengan 30 kolom (fitur klinis dan target)

2.2.2. Jenis Fitur dan Penjelasannya

a. Demografi Pasien

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

b. 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

c. 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

d. 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

e. Informasi tambahan

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

2.2.3. Analisis Awal

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

Telah ditemukan bahwa terdapat missing values pada TBG dan sex, selain itu target memiliki 31 kode diagnosis yang berbeda.

2.2.4. 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'

df['target'] = df['target'].apply(categorize_target)
```

Target didistribusi dengan kategori Healthy menggunakan kode [‘-’], Hyperthyroid menggunakan kode [‘A’, ‘B’, ‘C’, ‘D’, ‘O’, ‘P’, ‘Q’, ‘S’, ‘T’], dan Hypothyroid menggunakan kode [‘E’, ‘F’, ‘G’, ‘H’].

2.2.5. Masalah Data

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

     np.int64(1487)
```

Terdapat 1487 entri yang tidak masuk ke dalam kategori yang telah ditentukan sebelumnya, jadi beberapa nilai tidak masuk ke dalam kriteria hyperthyroid_conditions, hypothyroid_conditions, atau normal_conditions

2.2.6. Hasil Target

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

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

Diketahui bahwa untuk kolom ‘target’ terdapat 4 jenis data yaitu Healthy, Hyperthyroid, Hypothyroid, dan None yang berarti data masih mengandung NaN sehingga tidak terdeteksi termasuk jenis thyroid yang mana.

2.3. Data Cleaning

2.3.1. Penanganan Missing Values pada Kolom Kategorik

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

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

     np.int64(0)
```

Mengisi nilai kosong di dalam kolom sex dengan modus atau nilai yang paling sering muncul, yaitu ‘F’. Hal ini dikarenakan kolom gender sebaiknya tidak memiliki nilai null.

2.3.2. Penanganan Missing Values pada Kolom Numerik

```
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])
```

Nilai kosong pada kolom numerik (TSH, T3, TT4, T4U, FTI, dan TBG) diisi dengan 0 dengan asumsi nilai 0 berarti tidak ada pengukuran.

2.3.3. 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')
```

Kolom TSH_measured, T3_measured, TT4_measured, T4U_measured, FTI_measured, TBG_measured dihapus karena redundan dengan nilai numeriknya. Sedangkan patient_id dihapus karena tidak relevan untuk dianalisis dan memiliki nilai standar deviasi yang tinggi sehingga tidak optimal jika dimasukkan ke dalam analisis.

2.3.4. Filtering Outlier Usia

```
[ ] df = df[df['age'] <= 100]
    df['age'].unique()

    array([29, 41, 36, 32, 60, 77, 28, 54, 42, 51, 37, 16, 43, 63, 40, 56, 85,
           67, 61, 46, 44, 82, 64, 70, 33, 53, 52, 59, 49, 35, 48, 27, 69, 76,
           73, 68, 66, 30, 58, 21, 38, 45, 62, 25, 86, 83, 75, 72, 55, 14, 15,
           39, 20, 80, 90, 23, 13, 78, 24, 71, 81, 92, 57, 74,  9, 47, 17, 11,
           50, 26, 34,  8, 79, 31, 65, 84, 12, 10, 88, 18,  1, 22,  2, 97,  6,
           19, 89, 87, 91, 94,  7,  4,  3, 93,  5, 95])
```

Pasien dengan usia >100 tahun dihapus karena outlier tidak realistis.

2.3.5. Identifikasi Data Tidak Konsisten

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

Melakukan identifikasi dengan tujuan mendeteksi inkonsistensi logis (pasien laki-laki dengan status hamil). Apabila ditemukan, baris ini akan dihapus dari data.

2.3.6. 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)
```

Bertujuan untuk mengidentifikasi masalah pada kolom kategorik (‘object’), dan juga untuk memastikan tidak ada kolom kategorik yang terlewat saat proses pembersihan.

2.4. Data Encoding

2.4.1. 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'
```

Dilakukan untuk mengubah data kategorik (non-numerik) menjadi numerik (0, 1) agar dapat diproses oleh model machine learning yang akan digunakan menggunakan ‘LabelEncoder’

2.4.2. 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()}')
```

Melakukan standarisasi nilai boolean dengan mengubah nilai ‘t’/’f’ menjadi 0/1 untuk kolom boolean. Selain itu, dilakukan verifikasi untuk

menampilkan nilai unik setelah encoding untuk memastikan transformasi yang dilakukan berhasil.

2.5. Data Visualization

2.5.1. 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()
```

Melakukan visualisasi distribusi variabel target menggunakan grafik untuk memahami sebaran kelas target (Healthy, Hyperthyroid, Hypothyroid). Grafik ini juga dapat digunakan untuk mengidentifikasi ketidakseimbangan kelas yang akan berdampak pada pemilihan metode modeling nantinya.

2.5.2. 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()
```

Melakukan visualisasi menggunakan grafik untuk menunjukkan persebaran diagnosis berdasarkan jenis kelamin pasien. Selain itu, adanya visualisasi ini dapat memberikan kami wawasan mengenai apakah ada pola perbedaan diagnosis antara laki-laki dan perempuan, yang mungkin nantinya bisa menjadi relevan ke analisis.

2.6. Data Split

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

#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)

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

Untuk memisahkan data menjadi data training dan testing, kami melakukan metode train-test split dari library sklearn.model_selection. Kami membagi data ini menjadi 20% data yang digunakan sebagai data test, dan sisanya adalah 80% untuk data training. Selain itu, random_state=42 digunakan agar pembagian data konsisten saat akan dijalankan ulang. Stratify=y digunakan untuk memastikan bahwa proporsi kelas di ‘target’ tetap terjaga antara data train dan test. Tujuan dataset ini dibagi adalah untuk melatih model pada data training, dan mengevaluasi performa model pada data yang belum pernah dilihat atau test set.

2.7. Data Modelling

2.7.1. 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)
```

Sebelum melakukan proses Data Modelling, label target akan dikodekan menjadi angka (Hyperthyroid: 0, Hypothyroid: 1, dan Healthy: 2). Selain itu, label juga akan dibinarisasi untuk kebutuhan perhitungan average precision. Dan sample weights dari setiap model akan dihitung untuk menangani ketidakseimbangan setiap kelas.

2.7.2. 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)
```

Pada tahap ini, kami menggunakan tiga model boosting yang populer untuk dilatih secara terpisah untuk mempelajari pola dari data.

- CatBoostClassifier: cocok digunakan untuk data kategorikal, dan dilatih menggunakan ‘auto_class_weights=’Balanced’’ untuk menangani ketidakseimbangan kelas
- GradientBoostingClassifier: algoritma boosting yang stabil, dan akan diberikan sample weight berdasarkan distribusi kelas agar tidak ada bias terhadap kelas yang menunjukkan mayoritas.

- XGBClassifier (XGBoost): model boosting yang diberi label encoding dan sample weight agar hasil prediksinya lebih adil dan akurat.

Ketiga model ini kami gunakan agar dapat memprediksi probabilitas 'predict_proba', yang nantinya akan digabung menggunakan teknik stacking.

2.7.3. 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
```

Setiap model telah menghasilkan probabilitas prediksi terhadap setiap kelas, dan akan dilakukan stacking dengan cara menggabungkan output probabilitas dari ketiga model tersebut. Namun, tidak semua model memiliki kontribusi yang sama efektifnya, sehingga kami perlu menentukan bobot optimal untuk setiap model.

Untuk menentukan bobot optimal, kami menggunakan Optuna, sebuah pustaka otomatisasi optimasi hyperparameter berbasis Bayesian optimization. Dengan tujuan untuk mencari bobot terbaik untuk model CatBoost, GradientBoosting, dan juga XGBoost. Selain itu, fungsi objektifnya adalah 'average_precision_score' agar performa keseluruhan maksimal. Disini, Optuna akan mencoba berbagai kombinasi bobot hingga ia menemukan kombinasi terbaik.

2.7.4. Evaluasi Akhir

```
#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')}")
```

Setelah mendapatkan bobot optimal dari ketiga model, kami menggabungkan probabilitas tersebut dan menghasilkan prediksi akhir 'combined_pred' untuk dievaluasi. Evaluasi ini akan menggunakan Confusion Matrix untuk melihat prediksi yang benar dan salah dari setiap kelas, Accuracy untuk melihat seberapa banyak prediksi model yang benar secara keseluruhan, Micro Average Precision yang merupakan gabungan presisi dan recall secara keseluruhan untuk semua kelas, dan juga Average Precision per Kelas untuk mengevaluasi performa model dalam membedakan setiap label target (Hyperthyroid, Hypothyroid, dan Healthy).

2.8. Model Evaluation

Evaluasi model dilakukan untuk memahami sejauh mana model hasil stacking dapat membedakan ketiga kelas diagnosis medis (Hyperthyroid, Hypothyroid, dan Healthy).

2.8.1. 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()
```

Confusion matrix menunjukkan performa klasifikasi model berdasarkan prediksi benar dan salah untuk setiap kelas. Matriks ini juga membantu kami untuk mengidentifikasi apakah ada kelas tertentu yang sering tertukar.

2.8.2. Precision-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()
```

Precision-Recall Curve ini menggambarkan keseimbangan antara precision dan recall untuk setiap kelas. Grafik yang akan dihasilkan ini akan sangat bermanfaat terutama pada dataset yang tidak seimbang, karena dapat menunjukkan seberapa baik model dapat mendeteksi kasus positif tanpa terlalu banyak false positives.

BAB III
EVALUASI DAN DETAIL DARI ALUR PEKERJAAN

Pada bab sebelumnya sudah diuraikan beberapa tahapan mulai dari pemahaman data, pembersihan data pembersihan data, encoding, hingga pemodelan data untuk analisis prediktif penyakit tiroid. Tiga model boosting individual, yaitu CatBoost, GradientBoost, dan XGBoost, telah dilatih. Probabilitas dari ketiga model ini kemudian digabungkan menggunakan teknik stacking, dengan bobot optimal yang dicari menggunakan Optuna. Model stacked yang dihasilkan menunjukkan performa klasifikasi yang tinggi, dengan akurasi mencapai 98.50% pada data uji untuk kategori ‘Healthy’, ‘Hyperthyroid’, atau ‘Hypothyroid’.

3.1. Analisis Kinerja Model Stacking

Model stacked yang dikembangkan menunjukkan performa yang sangat baik dengan akurasi mencapai 0.9850357839947951, yang berarti mayoritas prediksi yang dihasilkan model sesuai dengan data uji. Selain itu, Micro Average Precision metrik yang dioptimalkan dengan Optuna tercatat pada nilai tinggi, yaitu 0.9960440229614973.

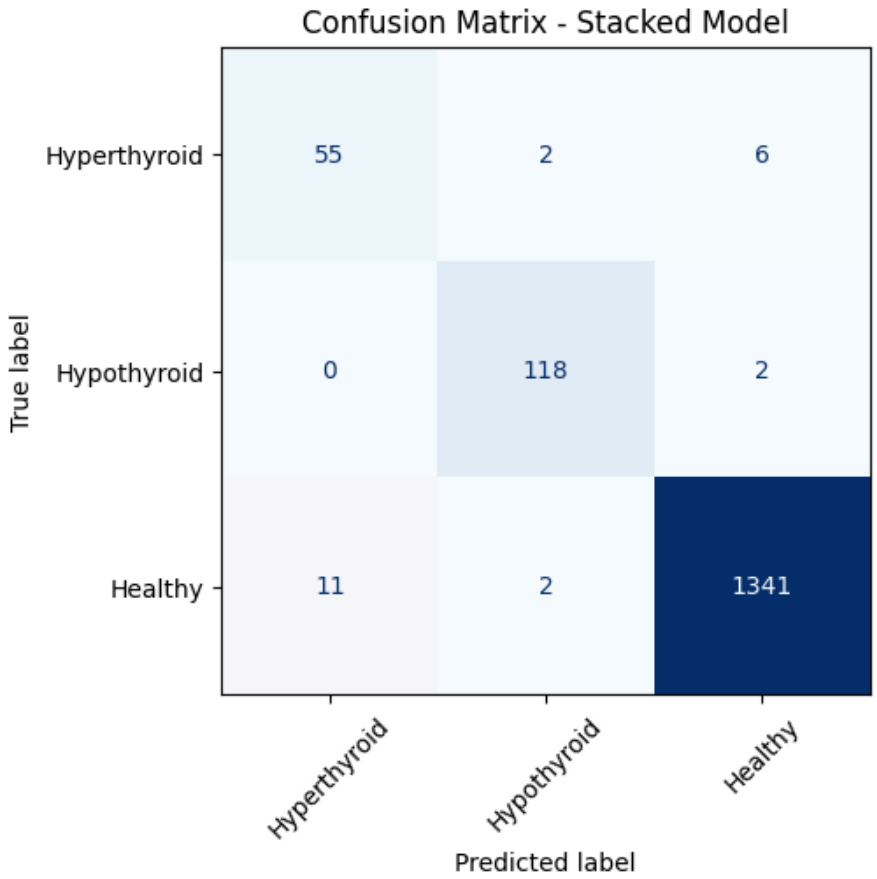
Untuk mendapatkan gambaran lebih menyeluruh, terutama dalam menghadapi kemungkinan ketidakseimbangan kelas dalam dataset, dilakukan analisis terhadap metrik Precision, Recall, dan F1-Score, baik dalam pendekatan Weighted Average maupun Macro Average. Precision mengukur sejauh mana prediksi positif benar-benar mencerminkan kondisi sebenarnya, sementara Recall menunjukkan kemampuan model dalam menangkap seluruh kasus positif yang ada. F1-Score menjadi indikator keseimbangan antara kedua metrik tersebut. Detail performa utama model stacked dirangkum dalam Tabel 3.1.

Tabel 3.1. Rangkuman Performance Metrics Model Stacked

No.	Metrics	Model Stacked (Nilai Keseluruhan)
1	Accuracy	98.50%
2	Micro Average Precision	99.60%

3	Precision (Weighted Average)	98.53%
4	Recall (Weighted Average)	98.50%
5	F1-Score (Weighted Average)	98.51%
6	Precision (Macro Average)	93.15%
7	Recall (Macro Average)	94.89%
8	F1-Score (Macro Average)	94.00%

3.1.1. Analisis Confusion Matrix



Gambar 3.1 *Confusion Matrix* Model Stacking

Confusion matrix memberikan visualisasi detail mengenai performa klasifikasi model untuk setiap kelas secara spesifik, memaparkan jumlah prediksi

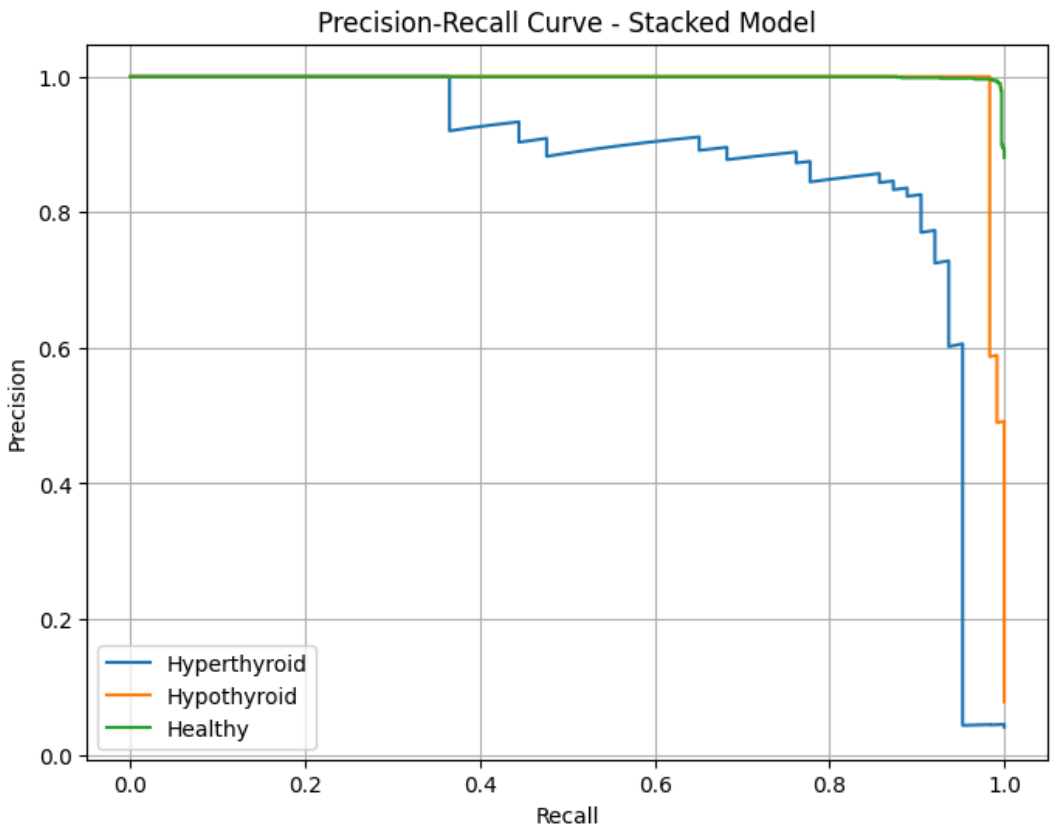
yang benar dan yang keliru. Gambar 3.1 menampilkan confusion matrix dari model stacked yang diterapkan pada data uji.

Kurva Precision-Recall (PR) sangat berguna untuk mengevaluasi performa model pada setiap kelas, terutama pada dataset dengan distribusi kelas yang tidak seimbang. Gambar 3.2 menyajikan kurva PR untuk masing-masing kelas. Performa di bawah kurva PR juga dapat dikuantifikasi dengan metrik Average Precision (AP).

Berdasarkan Gambar 3.1, interpretasi hasil prediksi adalah sebagai berikut:

- Class Hyperthyroid: Dari 63 kasus aktual, 55 kasus (87.30%) berhasil diprediksi dengan benar. Sebanyak 2 kasus keliru diklasifikasikan sebagai Hypothyroid, dan 6 kasus sebagai Healthy
- Class Hypothyroid: Dari 120 kasus aktual, 118 kasus (98.33%) berhasil diprediksi dengan benar. Hanya 2 kasus yang keliru diklasifikasikan sebagai Healthy, dan tidak ada yang keliru sebagai Hyperthyroid.
- Class Healthy: Dari 1354 kasus aktual, 1341 kasus (99.04%) berhasil diprediksi dengan benar. Sebanyak 11 kasus keliru diklasifikasikan sebagai Hyperthyroid, dan 2 kasus sebagai Hypothyroid.

3.1.2. Analisis Kurva Precision-Recall dan Average Precision per Class



Gambar 3.2 Kurva Precision-Recall Model Stacking per Class

Kurva Precision-Recall (PR) sangat berguna untuk mengevaluasi performa model pada setiap kelas, terutama pada dataset dengan distribusi kelas yang tidak seimbang. Gambar 3.2 menyajikan kurva PR untuk masing-masing kelas. Performa di bawah kurva PR juga dapat dikuantifikasi dengan metrik Average Precision (AP).

Dari Gambar 3.2 dan nilai AP yang diperoleh:

- Class ‘Healthy’ menunjukkan kurva PR yang superior, hampir sempurna mendekati sudut kanan atas, dengan nilai Average Precision sebesar 0.999355. Ini menandakan precision yang sangat tinggi secara konsisten di berbagai tingkat recall.
- Class ‘Hypothyroid’ juga menampilkan kurva PR yang sangat baik, dengan Average Precision sebesar 0.992341, menunjukkan performa yang kuat dan akurat..

- Class ‘Hyperthyroid’ memiliki kurva PR yang lebih rendah dibandingkan dua class lainnya, dengan Average Precision sebesar 0.880135. Meskipun demikian, nilai AP ini masih tergolong baik, namun bentuk kurvanya mengindikasikan adanya trade-off yang lebih nyata antara presisi dan recall. Tantangan ini seringkali berkaitan dengan jumlah sampel yang lebih sedikit pada kelas tersebut.

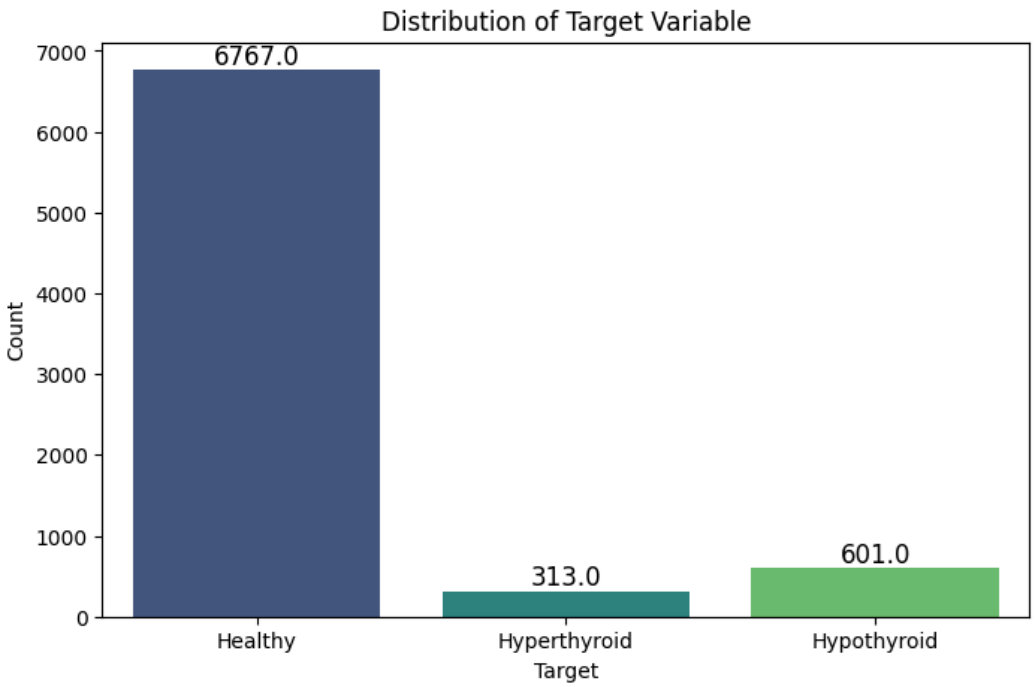
3.2. Kontribusi Optuna dalam Optimasi Bobot Stacking

Sudah diuraikan pada Bab II bahwa Optuna berperan krusial dalam menentukan bobot optimal untuk setiap model dasar (CatBoost, GradientBoosting, XGBoost) dalam arsitektur stacking. Proses optimasi ini menargetkan maksimisasi skor Micro Average Precision. Bobot optimal yang berhasil ditemukan adalah: CatBoost sebesar 0.002, GradientBoosting sebesar 0.011, dan XGBoost sebesar 0.987. Distribusi bobot ini mengindikasikan bahwa XGBoost memberikan kontribusi paling dominan dalam prediksi akhir model stacked, berdasarkan hasil optimasi. Pemahaman terhadap karakteristik data menjadi landasan penting dalam mengevaluasi hasil model.

3.3. Wawasan dari Visualisasi Data Pendukung

Pemahaman terhadap karakteristik data menjadi landasan penting dalam mengevaluasi hasil model.

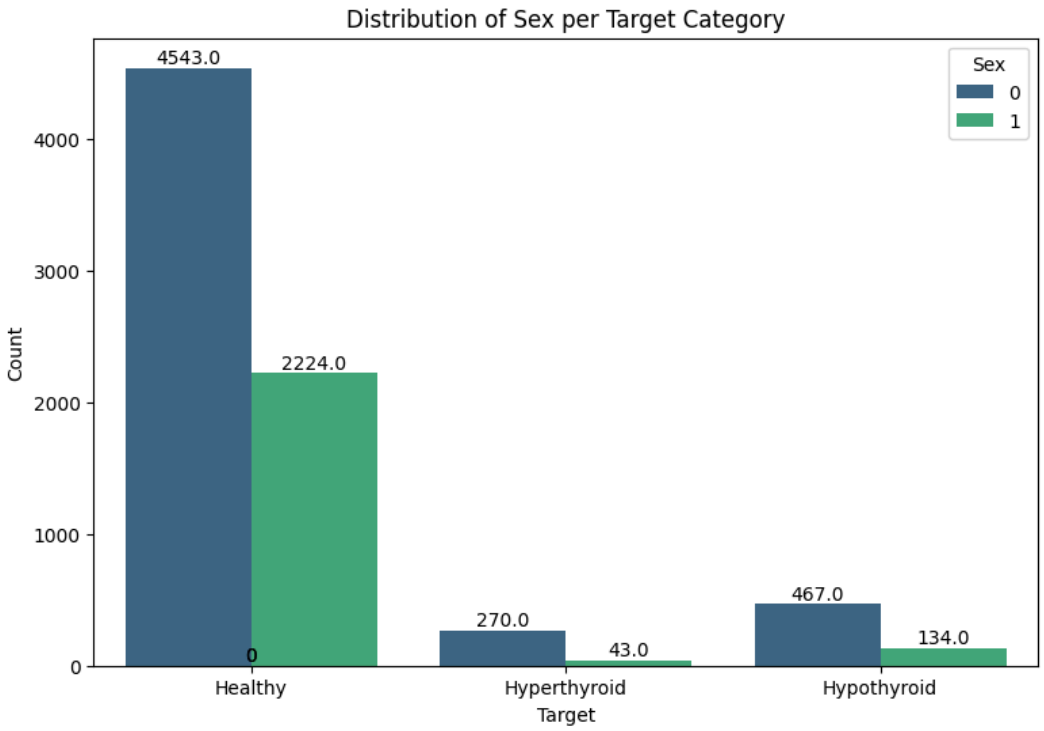
3.3.1. Distribusi Variabel Target



Gambar 3.3.1: Distribusi Variabel Target

Class ‘Healthy’ (6767 sampel) mendominasi dataset, diikuti oleh ‘Hypothyroid’ (601 sampel), dan ‘Hyperthyroid’ (313 sampel). Kondisi ini menjadi pertimbangan utama dalam pemilihan strategi pemodelan, seperti penggunaan sample_weights pada algoritma dasar, dan dalam interpretasi metrik evaluasi yang sensitif terhadap ketidakseimbangan kelas.

3.3.2. Distribusi Jenis Kelamin per Kategori Target



Gambar 3.3.2. Distribusi Jenis Kelamin per Kategori Target)

Dengan pengkodean jenis kelamin {‘F’:0, ‘M’:1}, observasi dari Gambar 3.3.2. menunjukkan proporsi jenis kelamin yang berbeda pada setiap kategori diagnosis. Informasi ini, meskipun bersifat deskriptif pada tahap ini, menjadi salah satu fitur masukan bagi model prediktif.

BAB IV
KESIMPULAN

Berdasarkan hasil analisis prediktif penyakit tiroid, penelitian kami berhasil mengembangkan model prediktif untuk klasifikasi penyakit tiroid menggunakan teknik Stacking Ensemble (CatBoost, GradientBoost, XGBoost) dengan optimasi Optuna yang menunjukan performa yang sangat baik. Model kami mencapai akurasi sebesar 98.50%, dan Micro Average Precision sebesar 99,60%, yang menunjukan kapabilitas dalam mengklasifikasikan penyakit tiroid.

Pada analisis *Confusion Matrix* menunjukan kemampuan model untuk memprediksi dengan benar dan memiliki akurasi yang sangat tinggi, khususnya pada kategori 'Healthy' 1341 dari 1354 kasus aktual berhasil diprediksi dengan benar (99,04%), dan 'Hypothyroid' 118 dari 120 kasus aktual berhasil diprediksi dengan benar (98,33%). Meskipun prediksi untuk kelas 'Hyperthyroid' sedikit lebih rendah 55 dari 63 kasus aktual berhasil diprediksi dengan benar (87,30%), performanya tetap tergolong kuat meskipun dilatih dengan data sampel yang lebih sedikit data sampel.

Peran Optuna terbukti krusial dalam memaksimalkan performa *stacking*, Optimasi bobot dengan Optuna berhasil menentukan kontribusi optimal dari masing-masing model, di mana XGBoost memberikan pengaruh dominan (0.987), diikuti oleh GradientBoost (0.011) dan CatBoost (0.002). Hal ini menunjukkan bahwa XGBoost paling efektif dalam memprediksi penyakit tiroid pada dataset ini.

Secara keseluruhan, hasil penelitian ini menunjukkan bahwa pendekatan *stacking ensemble* dengan optimasi optuna dapat menghasilkan model prediktif yang efektif untuk mendeteksi penyakit tiroid, dan memiliki potensi aplikatif untuk mendukung pengambilan keputusan di bidang medis.

REFERENSI

Afifah, M. N. (2022). *Kenali Tahap dan Stadium Sirosis Hati*. https://health.kompas.com/read/2022/07/28/190100168/kenali-tahap-dan-stadium-sirosis-hati?page=all

LINK COLAB

https://colab.research.google.com/drive/1Vf87xqpFwrq7lFvaUm9YjW9lyFXno_x4?usp=sharing