FINAL PROJECT MATA KULIAH DATA MINING

Klasifikasi Data Mining untuk Prediksi Status Gizi Balita Menggunakan Algoritma Support Vector Machine (SVM)

(Studi kasus: Puskesmas Wonoayu)

Dosen pengampu :

Amalia Anjani Arifiyanti, S.Kom., M.Kom.

Disusun oleh Kelompok 2:

Bagus Dwi Putra Adiyono	21082010195
Muhammad Faiq Al Abiyyi	21082010203
M Ilham Praditya A.N	21082010205

PARALEL-B

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL VETERAN JAWA TIMUR

2024

LATAR BELAKANG

Status gizi balita merupakan indikator penting untuk kesehatan dan perkembangan anak di masa depan. Malnutrisi, baik gizi buruk maupun gizi lebih, pada balita dapat menyebabkan berbagai masalah kesehatan, gangguan pertumbuhan, dan penurunan kemampuan kognitif. Sebaliknya, status gizi yang baik akan mendukung tumbuh kembang optimal dan meningkatkan kualitas hidup anak. Di Indonesia, masalah gizi balita, baik gizi kurang maupun gizi lebih, masih menjadi perhatian. Data Riset Kesehatan Dasar (Riskesdas) tahun 2018 menunjukkan *prevalensi* stunting (gizi kurang kronis) pada balita di Indonesia masih sebesar 36,4%, sementara *prevalensi overweight* dan obesitas mencapai 8%. Oleh karena itu, diperlukan upaya deteksi dini dan intervensi yang tepat untuk mencegah dampak jangka panjang. Pemanfaatan teknologi, khususnya *Machine Learning*, dapat membantu dalam mendeteksi dini status gizi balita secara lebih efisien dan akurat. Melalui studi kasus ini, diharapkan dapat dikembangkan sistem prediksi status gizi balita yang akurat dan praktis. Pada studi kasus ini menggunakan SVM sebagai algoritma *Machine Learning*. Pada studi kasus ini juga menggunakan 4 skenario untuk pembanding menghasilkan model yang baik, dimana kami menggunakan data *balance* dan *imbalance* pada algoritma SVM dan Decision Tree. Dataset yang digunakan merupakan dataset yang diambil di Puskesmas Wonoayu. Sistem ini diharapkan dapat membantu tenaga kesehatan dalam melakukan deteksi dini, identifikasi balita berisiko, dan memberikan rekomendasi intervensi gizi yang tepat sasaran, sehingga berkontribusi dalam upaya peningkatan status gizi balita di Indonesia.

LANGKAH-LANGKAH PEMBUATAN SISTEM

A. Data Preparation

Install library dan read dataset

```
import pandas as pd
import numpy as np

# Load data (file CSV)
file_path = '/content/sample_data/AGUSTUS.csv'  # Ganti dengan path file Anda
data = pd.read_csv(file_path)
```

```
data.head()
```

	No	NIK	Nama	JK	Tgl Lahir	BB Lahir	TB Lahir	Nama Ortu	Prov	Kab/Kota	...	TB/U	ZS TB/U	BB/TB	ZS BB/TB	Naik Berat Badan	PMT Diterima (kg)	Jml Vit A	KPSP	KIA	Detail
0	1	3515100208190050	MAULANA	L	8/2/2019	3.0	49.0	EKA / FATKHUL	JAWA TIMUR	KAB SIDOARJO	...	Normal	-0.18	Gizi Lebih	2.50	T	-	1.0	-	-	
1	2	3515100308190049	M. ARKANA SY	L	8/3/2019	3.4	49.0	SYAICUL	JAWA TIMUR	KAB SIDOARJO	...	Normal	-0.32	Normal	-0.63	N	-	1.0	-	-	
2	3	3515104508190032	HILYA INDAH	P	8/5/2019	3.0	49.0	0	JAWA TIMUR	KAB SIDOARJO	...	Normal	-0.93	Normal	-0.49	N	-	1.0	-	-	
3	4	3515100508190002	KENZIE AR RISKY	L	8/5/2019	3.1	49.0	WACHID	JAWA TIMUR	KAB SIDOARJO	...	Normal	-1.82	Normal	-0.66	N	-	1.0	-	-	
4	5	3515104508190027	SAUSAN AMIRA	P	8/5/2019	3.3	50.0	SIGIT	JAWA TIMUR	KAB SIDOARJO	...	Normal	-1.87	Normal	-0.24	N	-	1.0	-	-	

5 rows × 35 columns

B. Melakukan EDA

- Menampilkan statistik deskriptif dari dataset berupa numerik

```
data.describe()
```

	No	BB Lahir	TB Lahir	Berat	Tinggi	LiLA	ZS BB/U	ZS TB/U	ZS BB/TB	Jml Vit A
count	3552.000000	3552.000000	3552.000000	3552.000000	3552.000000	3539.000000	3552.000000	3552.000000	3552.000000	3334.0
mean	1776.500000	3.085985	48.898789	13.131534	89.556898	13.594518	-0.278133	-0.765512	0.198894	1.0
std	1025.518405	1.515460	4.477051	3.535343	12.368406	3.294849	1.042200	0.867618	1.139499	0.0
min	1.000000	0.000000	0.000000	2.200000	46.000000	0.000000	-3.330000	-4.000000	-2.630000	1.0
25%	888.750000	3.000000	49.000000	10.400000	80.175000	14.000000	-1.030000	-1.430000	-0.620000	1.0
50%	1776.500000	3.000000	49.000000	13.000000	91.700000	14.000000	-0.260000	-0.870000	0.180000	1.0
75%	2664.250000	3.000000	50.000000	15.600000	99.700000	15.000000	0.440000	-0.230000	0.940000	1.0
max	3552.000000	35.000000	58.000000	32.400000	119.900000	22.000000	4.780000	4.750000	5.820000	1.0

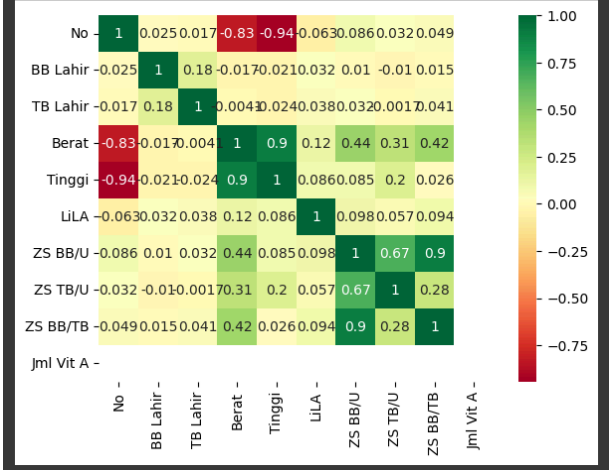
- Cek tipe data, jumlah data, jumlah atribut

```
print(data.dtypes)
data = pd.DataFrame(data)
print(data.shape)
```

```
No                  int64
NIK                object
Nama               object
JK                 object
Tgl Lahir          object
BB Lahir          float64
TB Lahir          float64
Nama Ortu          object
Prov               object
Kab/Kota           object
Kec                object
Pukesmas           object
Desa/Kel           object
Posyandu           object
RT                 object
RW                 object
Alamat             object
Usia Saat Ukur     object
Tanggal Pengukuran object
Berat             float64
Tinggi            float64
Cara Ukur          object
LiLA              float64
BB/U               object
ZS BB/U           float64
TB/U               object
ZS TB/U           float64
BB/TB              object
ZS BB/TB          float64
Naik Berat Badan   object
PMT Diterima (kg)  object
Jml Vit A         float64
KPSP               object
KIA                object
Detail             object
dtype: object
(3552, 35)
```

- Cek Heatmap Correlation

```
import seaborn as sns
import pandas as pd

def titanic_corr(data):
    numeric_data = data.select_dtypes(include=['number'])
    correlation = numeric_data.corr()
    sns.heatmap(correlation, annot=True, cbar=True, cmap='RdYlGn')

titanic_corr(data)
```



C. Pemilihan Atribut

Dari EDA diatas didapat insight untuk menentukan atribut yang sesuai sehingga dilakukan pemilihan atribut yang ingin digunakan untuk membuat model dan menyimpannya dalam variabel data, sehingga dataset baru hanya berisi kolom-kolom terpilih tersebut.

```
# Pilih kolom yang relevan
columns_to_keep = ['JK', 'Usia Saat Ukur', 'Berat', 'Tinggi', 'LiLA', 'BB/TB']
data = data[columns_to_keep]
```

D. Missing Value & Isi dengan Mean (LiLA)

Pada tahapan ini dilakukan pengecekan atribut yang memiliki missing value, dan ternyata terdapat missing value pada atribut LiLA. Pada tahap ini dilakukan penanganan missing value dengan menggunakan Mean.

```
print(data.isnull().sum())
print(data.shape)
```

```
JK              0
Usia Saat Ukur  0
Berat           0
Tinggi          0
LiLA           13
BB/TB           0
dtype: int64
(3552, 6)
```

```
data.fillna(data.select_dtypes
 (include='number').mean(), inplace=True)
print(data.isnull().sum())
print(data.shape)
```

```
JK              0
Usia Saat Ukur  0
Berat           0
Tinggi          0
LiLA            0
BB/TB           0
dtype: int64
(3552, 6)
```

E. Normalisasi Umur

Pada tahap ini, dilakukan mengubah data usia pada atribut 'Usia Saat Ukur' yang berformat "X tahun - Y bulan - Z hari" menjadi total bulan dan menyimpannya di atribut baru bernama 'Umur'.

	JK	Usia Saat Ukur	Berat	Tinggi	LiLA	BB/TB
0	L	5 Tahun - 0 Bulan - 14 Hari	23.0	109.4	15.000000	Gizi Lebih
1	L	5 Tahun - 0 Bulan - 13 Hari	17.1	108.7	14.000000	Normal
2	P	5 Tahun - 0 Bulan - 11 Hari	16.1	105.2	14.000000	Normal
3	L	5 Tahun - 0 Bulan - 11 Hari	15.0	101.7	15.000000	Normal
4	P	5 Tahun - 0 Bulan - 11 Hari	15.1	100.7	14.000000	Normal

Proses ini dilakukan dengan menggunakan *regular expression (Regex)* untuk mengekstrak angka tahun, bulan, dan hari, lalu mengkonversinya menjadi total bulan dengan asumsi 1 bulan = 30 hari.

```
import re

# 1. Konversi kolom 'Usia Saat Ukur' menjadi format bulan
def convert_to_months(age_str):
    """
    Mengubah umur dalam format 'X tahun - Y bulan - Z hari' ke format bulan.
    """
    # Inisialisasi nilai default
    years, months, days = 0, 0, 0

    # Menggunakan regex untuk menangkap angka pada string umur
    year_match = re.search(r'(\d+)\s*[Tt]ahun', str(age_str))  # Menangkap tahun
    month_match = re.search(r'(\d+)\s*[Bb]ulan', str(age_str))  # Menangkap bulan
    day_match = re.search(r'(\d+)\s*[Hh]ari', str(age_str))  # Menangkap hari

    # Menangkap hasil dari regex (jika ada)
    if year_match:
        years = int(year_match.group(1))
    if month_match:
        months = int(month_match.group(1))
    if day_match:
        days = int(day_match.group(1))

    # Konversi umur menjadi bulan (dengan asumsi 1 bulan = 30 hari)
    total_months = (years * 12) + months + (days / 30)

    # Kembalikan hasil yang dibulatkan ke 2 desimal
    return round(total_months, 2)

data['Umur'] = data['Usia Saat Ukur'].apply(convert_to_months)
data = data.drop('Usia Saat Ukur', axis=1)  # Drop kolom 'Usia Saat Ukur' asli
```

F. One-hot encoding untuk kolom 'JK' (Jenis Kelamin)

Pada tahap ini dilakukan mengubah kolom 'JK' (Jenis Kelamin) yang berisi data kategorikal (misalnya, "Laki-laki", "Perempuan") menjadi beberapa kolom numerik baru menggunakan teknik One-Hot Encoding.

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
encoded_jk = enc.fit_transform(data[['JK']])
encoded_df = pd.DataFrame(encoded_jk, columns=enc.get_feature_names_out(['JK']))
data = pd.concat([data, encoded_df], axis=1)
data = data.drop('JK', axis=1)
```

G. Mapping kolom BB/TB

Pada tahap ini dilakukan mapping pada atribut “BB/TB” agar data siap diolah oleh algoritma machine learning yang umumnya lebih mudah memproses data numerik, serta menyimpannya di kolom baru bernama 'Status'.

```
data['BB/TB'].value_counts()
```

BB/TB	count
Normal	2716
Beresiko Gizi Lebih	591
Gizi Lebih	155
Obesitas	51
Gizi Kurang	39

dtype: int64

```
def map_bbtb(status):
    mapping = {'Gizi Kurang': 0, 'Normal': 1,
               'Beresiko Gizi Lebih': 2, 'Gizi Lebih': 3, 'Obesitas': 4}
    return mapping.get(status, np.nan)

data['Status'] = data['BB/TB'].apply(map_bbtb)

# Drop kolom asli BB/TB setelah mapping
data = data.drop('BB/TB', axis=1)
```

H. Menampilkan Hasil Preprocessing

```
print("\nData setelah preprocessing:")
print(data.head(10))
```

```
Data setelah preprocessing:
   Berat  Tinggi  LiLA   Umur  JK_L  JK_P  Status
0  23.00   109.4  15.0  60.47   1.0   0.0       3
1  17.10   108.7  14.0  60.43   1.0   0.0       1
2  16.10   105.2  14.0  60.37   0.0   1.0       1
3  15.00   101.7  15.0  60.37   1.0   0.0       1
4  15.10   100.7  14.0  60.37   0.0   1.0       1
5  16.75   105.2  14.0  60.37   1.0   0.0       1
6  31.30   118.2  20.0  60.33   1.0   0.0       4
7  13.20    99.4  14.0  60.33   0.0   1.0       1
8  19.50   108.4  15.0  60.30   1.0   0.0       1
9  19.10   107.6   0.0  60.27   0.0   1.0       1
```

Setelah melakukan preprocessing data, langkah selanjutnya adalah membuat modeling dengan menggunakan 4 skenario, dimana skenarionya menggunakan data *imbalance* dengan menggunakan 2 algoritma yaitu SVM dan Decision Tree. Skenario selanjutnya yaitu menggunakan data yang sudah *balance* dengan menggunakan 2 algoritma yaitu SVM dan Decision Tree.

I. Simpan hasil dataset yang telah di pre-processing.

```
# Simpan hasil
data.to_csv('processed_AGUSTUS.csv', index=False)
print("\nData telah disimpan ke 'processed_AGUSTUS.csv'")
```

Penyimpanan data ini bertujuan agar nantinya data tersebut akan di filtering dan diolah secara manual dan randomize untuk membuat data menjadi balance.

J. Pemodelan Data Imbalance

Pada tahapan ini dilakukan pembuatan model dengan data *imbalance* menggunakan algoritma SVM dan Decision Tree. Pembagian data dilakukan dengan 70% digunakan untuk melatih model agar mempelajari pola, hubungan antar fitur, dan target dari data ini. Selanjutnya 30% data digunakan untuk memastikan bahwa model dapat menggeneralisasi dengan baik pada data baru. Berikut merupakan detail dari skenario tersebut:

1. SKENARIO 1 (Menggunakan SVM)

```python
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve, confusion_matrix
import matplotlib.pyplot as plt

# Pisahkan fitur (X) dan target (y)
X = data.drop('Status', axis=1)  # Hapus kolom 'Status' dari fitur
y = data['Status']

# Bagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inisialisasi dan latih model SVM
svm_model = SVC(kernel='linear', probability=True)  # Pastikan probability=True untuk prediksi probabilitas
svm_model.fit(X_train, y_train)

# Prediksi pada data testing
y_pred = svm_model.predict(X_test)

# Evaluasi akurasi model
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.69      0.82        13
           1       0.95      0.95      0.95       798
           2       0.77      0.78      0.77       184
           3       0.78      0.86      0.82        49
           4       1.00      0.64      0.78        22

    accuracy                           0.91      1066
   macro avg       0.90      0.78      0.83      1066
weighted avg       0.91      0.91      0.91      1066
```

```
Confusion Matrix:
[[  9   4   0   0   0]
 [  0 761  37   0   0]
 [  0  36 144   4   0]
 [  0   0   7  42   0]
 [  0   0   0   8  14]]
```

Setelah dilakukan pemodelan SVM dihasilkan accuracy sebesar **0.91**

2. SKENARIO 2 (Menggunakan Decision Tree)

```python
from sklearn.tree import DecisionTreeClassifier

# Pisahkan fitur (X) dan target (y)
X = data.drop('Status', axis=1)
y = data['Status']

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inisialisasi dan latih model Decision Tree
model = DecisionTreeClassifier(random_state=42) # You can adjust hyperparameters here
model.fit(X_train, y_train)

# Prediksi pada data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.33      0.23      0.27        13
           1       0.95      0.96      0.96       798
           2       0.81      0.73      0.77       184
           3       0.64      0.76      0.69        49
           4       0.83      0.68      0.75        22

    accuracy                           0.90      1066
   macro avg       0.71      0.67      0.69      1066
weighted avg       0.90      0.90      0.90      1066
```

```
Confusion Matrix:
[[  3  10   0   0   0]
 [  6 770  22   0   0]
 [  0  34 135  15   0]
 [  0   0   9  37   3]
 [  0   0   1   6  15]]
```

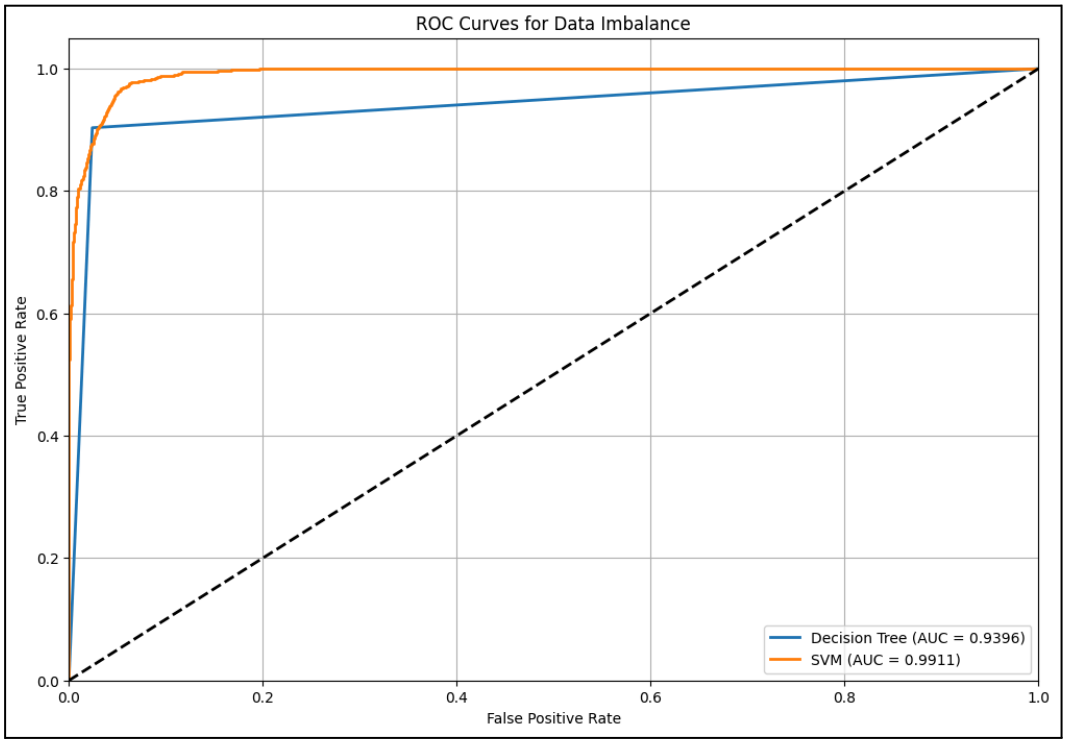
Setelah dilakukan pemodelan Decision Tree dihasilkan accuracy sebesar **0.90**

3. ROC AUC Untuk Data Imbalance

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import label_binarize

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize your models
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "SVM": SVC(probability=True),
}

# Fit the models and get predictions
predictions = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)  # Fit on the original labels
    # Store the probability predictions for the test set
    predictions[model_name] = model.predict_proba(X_test)

# Binarize the output for ROC AUC calculation
unique_classes = np.unique(y)  # Get unique classes from original y
ytest_bin = label_binarize(y_test, classes=unique_classes)  # Use unique classes
n_classes = ytest_bin.shape[1]  # Number of classes after binarization

# Set up the plot for all models
plt.figure(figsize=(12, 8))
```

```
# Calculate and plot ROC curve for each model
for model_name, preds in predictions.items():
    # Calculate ROC curve and AUC for each model
    fpr, tpr, _ = roc_curve(ytest_bin.ravel(), preds.ravel())
    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.plot(fpr, tpr, lw=2, label=f'{model_name} (AUC = {roc_auc:.4f})')

# Plotting diagonal line for random chance
plt.plot([0, 1], [0, 1], 'k--', lw=2)

# Adding titles and labels
plt.title('ROC Curves for Data Imbalance')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.legend(loc='lower right')
plt.grid()

# Show the combined plot
plt.show()
```

Selanjutnya dilakukan membandingkan performa model Decision Tree dan SVM dengan data *imbalance* menggunakan metrik ROC AUC untuk memilih model terbaik.



K. Pemodelan Data Balance

Pada tahapan ini dilakukan pembuatan model dengan data yang sudah *balance* menggunakan algoritma SVM dan Decision Tree. Data yang sudah *balance* didapatkan dari hasil preprocessing data *imbalance* yang tadi telah disimpan pada tahap sebelumnya (Tahap I), lalu dilakukan pemilihan data secara manual dan *randomize* agar mendapatkan data yang seimbang. Berikut merupakan detail dari skenario tersebut:

Data yang seimbang bisa dilihat pada gambar di bawah ini.

- Read Data dan hasilnya

```
import pandas as pd
import numpy as np

# Load data (file CSV)
file_path = '/content/sample_data/Agustus_Balance.csv'  # Ganti dengan path file Anda
data = pd.read_csv(file_path)
```

```
Data setelah preprocessing:
   Berat  Tinggi  LiLA  Umur  JK_L  JK_P  Status
0   31.5   116.7  19.0  59.8     1     0       4
1   24.9   106.5  17.0  59.2     0     1       4
2   30.4   112.3  21.0  58.7     1     0       4
3   23.7   102.8  16.0  58.3     0     1       4
4   29.8   114.6  15.0  57.2     1     0       4
5   22.1    99.7  19.0  56.4     0     1       4
6   25.3   109.3  20.0  55.8     1     0       4
7   28.6   114.2  14.0  55.3     1     0       4
8   24.4   107.1  18.0  54.8     0     1       4
9   26.2   105.8  15.0  54.1     1     0       4
```

- Read Status sebagai target class.

```
data['Status'].value_counts()
```

Status	count
2	181
1	156
3	155
0	139
4	102

dtype: int64

1. **SKENARIO 3 (Menggunakan SVM)**

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve, confusion_matrix
import matplotlib.pyplot as plt

# Pisahkan fitur (X) dan target (y)
X = data.drop('Status', axis=1)  # Hapus kolom 'Status' dari fitur
y = data['Status']

# Bagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inisialisasi dan latih model SVM
svm_model = SVC(kernel='linear', probability=True)  # Pastikan probability=True untuk prediksi probabilitas
svm_model.fit(X_train, y_train)

# Prediksi pada data testing
y_pred = svm_model.predict(X_test)

# Evaluasi akurasi model
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        37
           1       0.89      0.95      0.92        41
           2       0.96      0.84      0.90        58
           3       0.84      1.00      0.91        46
           4       1.00      0.89      0.94        38

    accuracy                           0.93       220
   macro avg       0.94      0.93      0.93       220
weighted avg       0.93      0.93      0.93       220
```

```
Confusion Matrix:
[[36  1  0  0  0]
 [ 0 39  2  0  0]
 [ 0  4 49  5  0]
 [ 0  0  0 46  0]
 [ 0  0  0  4 34]]
```

Setelah dilakukan pemodelan SVM dihasilkan accuracy sebesar **0.93**

2. **SKENARIO 4 (Menggunakan Decision Tree)**

```
from sklearn.tree import DecisionTreeClassifier

# Pisahkan fitur (X) dan target (y)
X = data.drop('Status', axis=1)
y = data['Status']

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inisialisasi dan latih model Decision Tree
model = DecisionTreeClassifier(random_state=42) # You can adjust hyperparameters here
model.fit(X_train, y_train)

# Prediksi pada data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Classification Report:

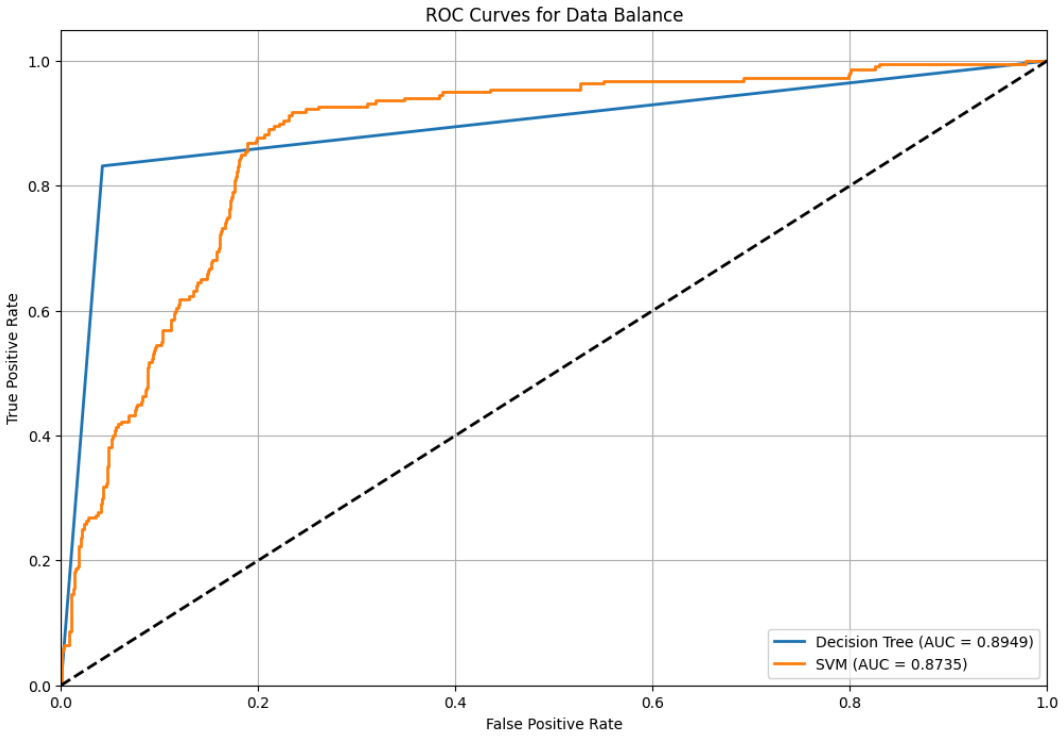
	precision	recall	f1-score	support
0	0.95	1.00	0.97	37
1	0.95	0.88	0.91	41
2	0.85	0.86	0.85	58
3	0.74	0.74	0.74	46
4	0.87	0.87	0.87	38
accuracy			0.86	220
macro avg	0.87	0.87	0.87	220
weighted avg	0.86	0.86	0.86	220

Confusion Matrix:

```
[[37  0  0  0  0]
 [ 2 36  2  1  0]
 [ 0  2 50  6  0]
 [ 0  0  7 34  5]
 [ 0  0  0  5 33]]
```

Setelah dilakukan pemodelan Decision Tree dihasilkan accuracy sebesar **0.86**

3. ROC AUC Untuk Data Balance



L. Pemilihan Model yang Terbaik dan Deployment

Berdasarkan pembuatan model yang telah dilakukan pada 4 skenario di atas terdapat hasil akurasi seperti dibawah ini:

Model	**Data Imbalance**	**Data Balance**
Support Vector Machine (SVM)	0.91	0.93
Decision Tree	0.90	0.86

Berdasarkan tabel diatas, SVM memberikan akurasi tinggi baik pada data *imbalance* (0.91) maupun *balance* (0.93). SVM bekerja lebih baik dalam kedua kondisi (*imbalance* dan *balance*), dengan peningkatan performa saat menggunakan data yang sudah *balance*. Hal ini menunjukkan bahwa SVM bekerja lebih optimal ketika distribusi data seimbang karena model dapat lebih fokus mempelajari pola dari kedua kelas tanpa didominasi oleh satu kelas tertentu. Dengan analisis ini, dapat disimpulkan bahwa SVM lebih stabil dan efektif, terutama ketika data seimbang, sedangkan Decision Tree memerlukan penyesuaian lebih lanjut pada data balanced untuk menghindari overfitting.

Dari hasil tersebut, dapat disimpulkan bahwa skenario dengan model terbaik adalah Menggunakan **SVM** dengan **data *balance*** yang menghasilkan **akurasi 0.93**. Setelah itu, dilakukan deploy model menggunakan streamlit seperti dibawah ini:

Deployment Menggunakan Streamlit

Menyimpan Model yang telah ditentukan dalam bentuk Pickle

```
# Mem-pickle-kan model

import pickle

# Nama pickle - classifier.pkl
pickle_out = open("classifier.pkl", "wb")

# Nama model - classifier
pickle.dump(svm_model,  pickle_out)
pickle_out.close()
```

Mendeploy model dengan menggunakan Streamlit

```
%%writefile app.py
import streamlit as st
import pandas as pd
import pickle

# Load the trained model
pickle_in = open('classifier.pkl', 'rb')
classifier = pickle.load(pickle_in)

# Function to preprocess input data
def preprocess_input(jk, umur, berat, tinggi, lila):
    """
    Preprocess user input to match the dataset structure with one-hot encoded gender (JK_L and JK_P).
    """
    data = pd.DataFrame({
        'Berat': [berat],
        'Tinggi': [tinggi],
        'LiLA': [lila],
        'Umur': [umur],  # Age in months
        'JK_L': [1 if jk == "L" else 0],  # JK_L is 1 if gender is Laki-Laki
        'JK_P': [1 if jk == "P" else 0],  # JK_P is 1 if gender is Perempuan
    })
    return data

# Streamlit app
st.set_page_config(
    page_title="Prediksi Status Gizi Balita",
    page_icon="👶",
    layout="centered",
    initial_sidebar_state="expanded",
)
```

```
# Custom CSS for UI Styling
st.markdown("""
    <style>
    body {
        background-color: #f9f9f9;
        font-family: "Arial", sans-serif;
    }
    .title {
        font-size: 36px;
        font-weight: bold;
        color: #4caf50;
        text-align: center;
        margin-top: -20px;
        margin-bottom: 20px;
    }
    .sidebar .sidebar-content {
        background: #fffbe6;
    }
    .main-button {
        background-color: #4caf50;
        color: white;
        font-size: 16px;
        padding: 10px 15px;
        border-radius: 8px;
        border: none;
    }
    .main-button:hover {
        background-color: #45a049;
    }
    </style>
""", unsafe_allow_html=True)

# Page Title
st.markdown('<div class="title">Prediksi Status Gizi Balita</div>', unsafe_allow_html=True)

# App Subtitle
st.subheader("Streamlit Gizi Balita Classifier ML SVM")
```

```
# Input fields with better formatting
st.markdown("### Masukkan Data Balita:")
berat = st.number_input("Berat Badan (kg):", min_value=0.0, value=0.0, step=0.1)
tinggi = st.number_input("Tinggi Badan (cm):", min_value=0.0, value=0.0, step=0.1)
lila = st.number_input("Lingkar Lengan Atas (cm):", min_value=0.0, value=0.0, step=0.1)
umur = st.number_input("Umur (0-60 bulan):", min_value=0.0, value=0.0, step=0.1)  # Numeric input for age in months
jk = st.selectbox("Jenis Kelamin:", ["L", "P"], help="Pilih jenis kelamin balita (L = Laki-Laki, P = Perempuan)")

# Prediction button with custom style
if st.button("Prediksi", help="Klik tombol ini untuk melihat hasil prediksi status gizi."):
    try:
        # Preprocess input
        input_data = preprocess_input(jk, umur, berat, tinggi, lila)

        # Ensure all required features are present and numeric
        if input_data.isnull().values.any() or (input_data < 0).any(axis=None):
            raise ValueError("Pastikan semua input diisi dengan nilai valid (positif).")

        # Predict the class (status gizi)
        prediction = classifier.predict(input_data)[0]

        # Convert prediction to human-readable label
        label_mapping = {
            0: '🔪 Gizi Kurang',
            1: '🌳 Normal',
            2: '🍔 Beresiko Gizi Lebih',
            3: '🍩 Gizi Lebih',
            4: '💩 Obesitas'
        }

        # Display result
        st.success(f"Prediksi Status Gizi: {label_mapping[prediction]}")
        st.balloons()

    except ValueError as e:
        st.error(f"Error: {e}. Periksa input Anda.")
    except Exception as e:
        st.error(f"Terjadi kesalahan tak terduga: {e}")
```

Lalu install dan jalankan streamlit

```
!pip install streamlit -q

     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 47.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 84.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 4.8 MB/s eta 0:00:00

!wget -q -O - ipv4.icanhazip.com

34.57.11.66

!streamlit run app.py & npx localtunnel --port 8501

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://34.57.11.66:8501

your url is: https://petite-parents-start.loca.lt
```

Tampilan sistem pada deploy Streamlit

Link Github : https://github.com/Bagusdpa4/FP-DM-2