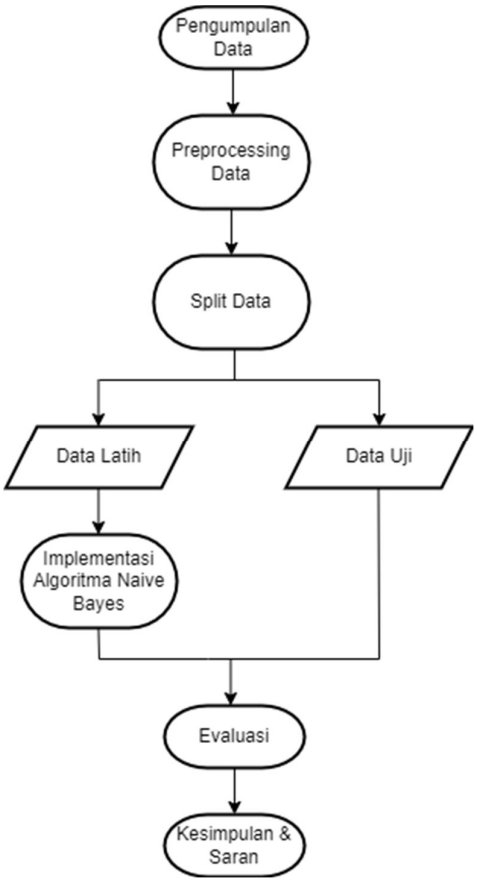
BAB II
METODE PENELITIAN

2.1 Objek Penelitian

Objek penelitian dalam studi ini adalah balita di Kota Samarinda yang mendapatkan layanan kesehatan di posyandu dan puskesmas. Penelitian ini difokuskan pada pengklasifikasian status gizi balita dengan menggunakan algoritma *Gaussian Naïve Bayes* berdasarkan atribut-atribut tertentu seperti berat badan, tinggi badan, umur, dan indeks antropometri lainnya.Prosedur Penelitian.

2.2 Tahapan Pelaksanaan Penelitian

Penelitian ini terdiri dari beberapa tahapan yang dirancang untuk mencapai tujuan penelitian. Pelaksanaan penelitian dimulai dari tahap identifikasi masalah, pengumpulan data, analisis data, hingga tahap evaluasi. Alur penelitian dapat dilihat pada Gambar 2.1 berikut ini:



Gambar 2. 1 Alur Penelitian

2.3.1 Pengumpulan Data

Data pada penelitian merupakan data sekunder yang diperoleh dari Dinas Kesehatan Kota Samarinda dengan rentang waktu dari tanggal 3 agustus 2022 hingga 31 juli 2023. Data yang digunakan yaitu data kuantitatif berupa data status gizi balita stunting. Pada proses pengumpulan

data ini diperoleh sejumlah 15.593 dengan 17 atribut dan 1 kelas target, data yang didapat tergolong *multi-classification* yang memiliki enam kelas didalamnya, yaitu gizi baik, gizi buruk, gizi kurang, gizi lebih, obesitas, dan risiki gizi lebih. Data diperoleh dari riwayat pemeriksaan status gizi dan stunting yang dilakukan di 26 puskesmas, terdiri dari 10 kecamatan di Kota Samarinda. Untuk dapat melihat keseluruhan atribut bisa dilihat dalam Tabel 2.1 dibawah ini:

Tabel 2. 1 Data status gizi balita

No	Atribut	Tipe Data	Keterangan
1	Nama	*String*	Nama Balita
2	JK	*String*	Jenis Kelamin
3	Tgl Lahir	*String*	Tanggal Lahir
4	Provinsi	*String*	Provinsi
5	Kab/Kota	*String*	Kabupaten / Kota
6	Kec	*String*	Kecamatan
7	Puskesmas	*String*	Lokasi Puskesmas
8	Posyandu	*String*	Lokasi Posyandu
9	Usia saat diukur	*Integer*	Usia balita saat dilakukan pemeriksaan
10	Berat	*Integer*	Berat Badan
11	Tinggi	*Integer*	Tinggi Badan
12	BB/U	*Integer*	Berat Badan menurut Umur
13	ZS BB/U	*Integer*	Z Score Berat Badan menurut Umur
14	TB/U	*Integer*	Tinggi Badan menurut Umur (Stunting)
15	ZS TB/U	*Integer*	Z Score Tinggi Badan menurut Umur
16	BB/TB	*Integer*	Berat Badan menurut Tinggi Badan (Status Gizi)
17	ZS BB/TB	*Integer*	Z Score Berat Badan menurut Tinggi Badan
18	Naik Berat Badan	*Kategorikal*	Tinggi Badan Kenaikan berat badan dibandingkan pemeriksaan sebelumnya

2.3.2 Preprocessing Data

Data preprocessing adalah langkah awal dan krusial dalam analisis data serta pengembangan model machine learning yang mencakup pembersihan data *(data cleaning)*, transformasi data *(data transformation)*, dan seleksi data *(normalitation data)*. Proses ini dimulai dengan *data cleaning*, di mana data yang hilang, duplikat, atau tidak konsisten diidentifikasi dan diperbaiki untuk memastikan kualitas data yang optimal. Selanjutnya, data *transformation* mengubah data ke dalam format yang sesuai untuk analisis, termasuk normalisasi, standarisasi, dan encoding variabel kategori. Terakhir, *data selection* melibatkan pemilihan subset data yang paling relevan untuk analisis lebih lanjut, menggunakan teknik seperti *feature selection* dan *instance selection*. Dengan melakukan data preprocessing, kita memastikan bahwa data yang digunakan adalah bersih, konsisten, dan relevan, sehingga meningkatkan akurasi dan efisiensi model yang dikembangkan.

a) **Data *Cleaning***

Proses pertama yang dilakukan yaitu penghapusan atau perbaikan nilai-nilai hilang, tidak valid, serta deteksi dan penanganan outlier untuk memastikan integritas dan kualitas dataset yang digunakan dalam analisis. Proses tersebut ditunjukkan pada Gambar 2.2

```
missing_values = data.isna().sum()

# Menampilkan jumlah missing values untuk setiap kolom
print("Jumlah missing values untuk setiap kolom:")
print(missing_values)
data.dropna(inplace=True)
data.drop_duplicates(inplace=True)
```

Gambar 2. 2 Data Cleaning

Pada Gambar diatas menampilkan proses data *cleaning* menggunakan metode '.isna)' untuk mengidentifikasi *missing* values dalam dataset. Setiap elemen dalam dataset yang bernilai '*NaN'* atau '*none*' akan dianggap sebagai *missing values*. Kemudian metode '*.sum()*' digunakan untuk menghitung jumlah missing dalam setiap kolom. Selanjutnya pada metode penghapusan menggunakan '*.dropna()*' dengan parameter '*inplace=True*', yang menghapus semua baris yang mengandung *missing values* dan pada metode '*.datadrop_duplicates()*' menghapus semua baris duplikat pada dataframe (Fan et al., 2021).

b) **Data *Transformation***

Data *transformation* digunakan untuk mengubah data kedalam tipe yang sesuai dalam data mining. Pada penelitian ini data yang digunakan adalah data dalam tipe numerik, sehingga data yang masih dalam tipe kategorial harus diubah terlebih dahulu (Alexandropoulos et al., 2019). Seperti pada penelitian ini, atribut jenis kelamin akan diubah kedalam bentuk biner yaitu 0 dan 1, dimana 0 menunjukan jenis kelamin perempuan dan 1 menunjukan jenis kelamin laki-laki. Begitu juga atribut lainnya 'BB//U', 'TB/U', 'BB/TB', Naik Berat Badan'. seperti pada Gambar 2.3 dibawah ini

```python
from sklearn.preprocessing import LabelEncoder

# Inisialisasi LabelEncoder
le = LabelEncoder()

# Tampilkan nama kolom DataFrame
print("Nama kolom DataFrame:", data.columns.tolist())

# Kolom yang akan di-encode dengan LabelEncoder
required_columns = ['JK', 'BB/U', 'TB/U', 'BB/TB', 'Naik Berat Badan']
categorical_columns = ['Kec', 'Pukesmas', 'Desa/Kel', 'Posyandu']

# Periksa dan ubah kolom yang diperlukan menjadi indeks numerik
for col in required_columns + categorical_columns:
    if col not in data.columns:
        print(f"Kolom '{col}' tidak ditemukan dalam DataFrame.")
    else:
        # Ubah kolom menjadi indeks numerik jika ada
        data[col] = le.fit_transform(data[col].astype(str))

# Cek hasil encoding
print(data[required_columns + categorical_columns].head())
```

Gambar 2. 3 Data *Transformation*

Kode di atas adalah untuk mengubah kolom-kolom kategori dalam *DataFrame* menjadi indeks numerik menggunakan *LabelEncoder* dari sklearn.*preprocessing*. Pertama, *LabelEncoder* diinisialisasi dan nama kolom DataFrame ditampilkan. Kemudian, daftar kolom yang akan di-encode ditentukan dalam *required_columns* dan *categorical_columns*. Selanjutnya, kode memeriksa apakah kolom yang diperlukan ada dalam *DataFrame* dan mengubahnya menjadi indeks numerik jika ada. Jika kolom tidak ditemukan, pesan akan ditampilkan. Akhirnya, hasil encoding ditampilkan untuk memastikan perubahan yang dilakukan. *Transformasi* ini diperlukan untuk mempersiapkan dataset dalam format yang lebih cocok untuk analisis statistik atau pengaplikasian model *machine learning*, memastikan interpretasi data yang lebih mudah dan akurat (Nesca et al., 2022). Setelah dilakukan proses data transformasi hasil di tampilkan pada lampiran ...

```python
# Buat Korelasi matriks
correlation_matrix = data.corr()
correlation_matrix2 = data.corr(method='spearman')
correlation_matrix3 = data.corr(method='kendall')

plt.figure(figsize=(10, 8))  # Atur lebar dan tinggi sesuai keinginan
sns.heatmap(correlation_matrix, annot=True, cmap='cividis', fmt=".3f")
plt.show()
```

Gambar 2. 4 Correlation Matrix

Gambar di atas adalah cuplikan kode Python yang menghitung dan memvisualisasikan matriks korelasi dari sebuah dataset menggunakan pustaka pandas, seaborn, dan matplotlib. Tiga jenis matriks korelasi dihitung: Pearson (data.corr()), Spearman (data.corr(method='spearman')), dan Kendall (data.corr(method='kendall')). Hasil korelasi Pearson kemudian divisualisasikan dalam bentuk heatmap menggunakan seaborn dengan skema warna "cividis", yang diatur dengan ukuran gambar 10x8 inci dan anotasi nilai korelasi ditampilkan dengan format tiga desimal.

2.3.3 Pembagian Data

Pada Gambar 2.4 kode yang diberikan menggunakan fungsi *train_test_split* dari *library scikit-learn* dalam Python, yang sangat berguna dalam membagi dataset menjadi dua subset: data latih dan data uji. Dalam contoh tersebut, *variabel x* menyimpan dataset fitur, sedangkan variabel y berisi label yang sesuai dengan setiap fitur. Dengan menentukan *test_size* = 0.2, kode tersebut membagi dataset sehingga 20% dari data akan digunakan untuk pengujian *(x_test dan y_test)*, sementara 80% sisanya akan digunakan untuk pelatihan *(x_train dan y_train)*. Pengaturan *random_state* = 42 memastikan bahwa pembagian dataset bersifat acak tetapi dapat direproduksi dengan hasil yang konsisten. Pembagian dataset ini penting dalam pengembangan model *machine learning* karena memungkinkan evaluasi objektif terhadap kinerja model pada data yang belum pernah dilihat sebelumnya (Ads et al., 2021).

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 42)
```

Gambar 2. 5 Pembagian Data

2.3.4 Pemodelan

Dalam melakukan perbandingan dan evaluasi performa dengan *Gaussian Naive Bayes* dalam melakukan klasifikasi pada dataset yang berbeda karakteristiknya. Data dibagi menjadi data pelatihan dan data pengujian, kemudian menguji model *Gaussian Naive Bayes* untuk jenis dataset tertentu (Jeevaraj et al., 2023). seperti yang terlihat pada Gambar 2.5 dibawah ini

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB, ComplementNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 42)

# Membuat dan melatih model Gaussian Naive Bayes
gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_pred_gnb = gnb.predict(x_test)
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)

# Menampilkan akurasi dari model gaussian
print("Akurasi Gaussian Naive Bayes: ", accuracy_gnb)
```

Gambar 2. 6 Implementasi *Naïve Bayes*

Kode di atas menunjukkan proses pembelajaran mesin menggunakan algoritma Gaussian Naive Bayes untuk klasifikasi. Pustaka yang diimpor mencakup fungsi untuk membagi dataset menjadi set pelatihan dan pengujian (train_test_split), berbagai jenis algoritma Naive Bayes, dan metrik akurasi. Dataset dibagi menjadi data pelatihan dan pengujian dengan proporsi 80:20. Model Gaussian Naive Bayes dibuat dan dilatih menggunakan data pelatihan (gnb.fit(x_train, y_train)), kemudian digunakan untuk memprediksi label pada data pengujian (y_pred_gnb = gnb.predict(x_test)). Akurasi model dihitung dengan membandingkan prediksi dengan label sebenarnya pada data pengujian (accuracy_gnb = accuracy_score(y_test, y_pred_gnb)), dan hasilnya ditampilkan (print("Akurasi Gaussian Naive Bayes: ", accuracy_gnb)).

2.3.5 Evaluasi Model

Pada evaluasi model, dilakukan pengujian rasio untuk menilai kinerja klasifikasi model dengan membandingkan jumlah prediksi benar terhadap total prediksi untuk setiap kategori gizi, seperti Gizi Baik, Gizi Buruk, Gizi Kurang, Gizi Lebih, Obesitas, dan Risiko Gizi Lebih. Rasio ini membantu mengidentifikasi tingkat akurasi dan kesalahan model dalam mengklasifikasikan setiap kategori, sehingga dapat digunakan untuk melakukan perbaikan dan peningkatan akurasi model secara keseluruhan.

Evaluasi klasifikasi *Naive Bayes* menggunakan metode akurasi mengukur seberapa dekat prediksi model dengan nilai sebenarnya. Akurasi dihitung dengan membagi jumlah prediksi yang benar *(True Positive dan True Negative)* dengan jumlah total prediksi (semua *True Positive, True Negative, False Positive,* dan *False Negative*), kemudian dikalikan dengan 100% untuk mendapatkan nilai persentase. Metrik ini memberikan Gambaran keseluruhan tentang seberapa baik model *Naive Bayes* dapat mengklasifikasikan data dengan tepat, mencakup kemampuan untuk mengidentifikasi baik kelas positif maupun kelas negatif dengan akurat.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \text{ x } 100\% \qquad (1)$$

Keterangan :

TP = *True Positive* merupakan jumlah nilai positif yang diklasifikasikan sebagai positif
TN = *False Positive* merupakan jumlah nilai negatif yang diklasifikasikan sebagai positif
FP = *False Negative* merupakan jumlah nilai positif yang diklasifikasikan sebagai negatif.
FN = *True Negative* merupakan jumlah nilai negatif yang diklasifikasikan sebagai negatif