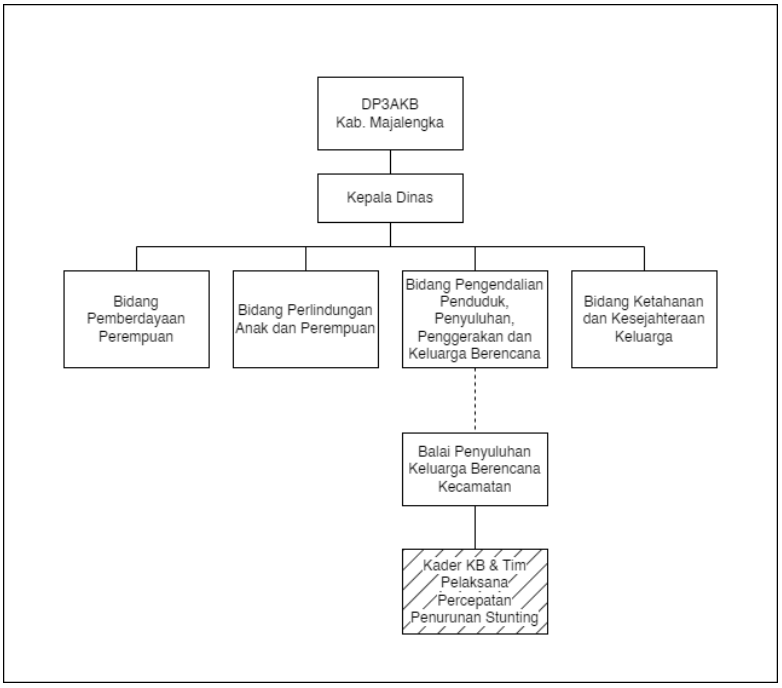
BAB IV
HASIL DAN PEMBAHASAN

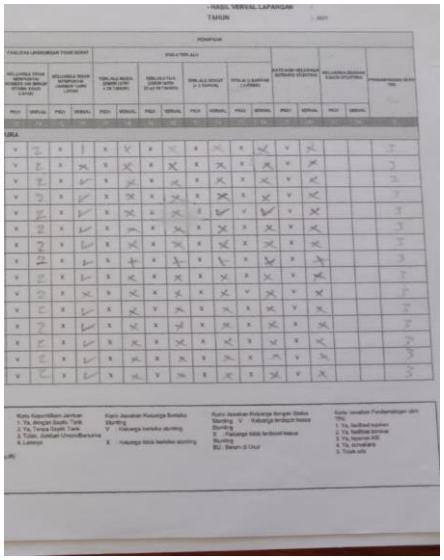
Penelitian ini dilakukan selama 5 bulan pada Dinas Pemberdayaan Perempuan Perlindungan Anak dan Keluarga Berencana (DP3AKB) Kabupaten Majalengka. Pembahasan keluarga beresiko *stunting* ini termasuk dalam rangkaian percepatan pencegahan *stunting*. Dalam penanganan kasus *stunting* ini tidak hanya dilakukan oleh DP3AKB yang dikomandoi oleh arahan pusat, yaitu BKKBN. Penanganan kasus *stunting* juga berkaitan dengan Dinas Kesehatan dan Dinas Sosial. Namun untuk tahapan pencegahan kasus *stunting* dengan pendekatan keluarga beresiko *stunting* dilakukan oleh DP3AKB yang melibatkan kader KB tiap desa dan kecamatan. Alur birokrasi dalam proses percepatan pencegahan *stunting* dapat dilihat pada Gambar 4.1.



Gambar 4. 1 *Struktur DP3AKB Kab. Majalengka*

Kader KB tiap desa dan kecamatan melakukan survey dan pendataan keluarga beresiko *stunting* setiap tahun. Proses pendataan dilakukan dengan pencatatan manual berupa kertas form isian yang sudah dicetak dalam bentuk kertas HVS seperti pada Gambar 4.2. Dalam penentuan kategori beresiko masih digunakan secara manual. Hal ini menjadi fokus utama dalam penentuan kategori beresiko diperlukan algoritma model prediksi yang memiliki akurasi tinggi agar proses penentuan kategori beresiko lebih akurat dan efektif. Tindak lanjut dari hasil prediksi keluarga beresiko *stunting* ini akan menjadi pertimbangan

dalam program pendampingan pencegahan *stunting* ataupun pendampingan keluarga berencana.



Gambar 4. 2 *Form Pendataan Keluarga Beresiko Stunting*

Dari rumusan masalah yang sudah disusun pada BAB II, dapat disimpulkan bahwa perlu dibuat suatu model algoritma prediksi untuk menentukan kategori beresiko dalam pendataan keluarga beresiko *stunting* sebagai pertimbangan program pendampingan yang akan dilakukan oleh Balai KB Kecamatan. Dalam pembahasan penelitian ini akan mengolah dataset untuk memuat model algoritma prediksi yang memiliki akurasi tinggi disertai sistem prediksi sederhana untuk implementasi model prediksi yang dibuat menggunakan proses *Knowledge Discovery In Database* (KDD).

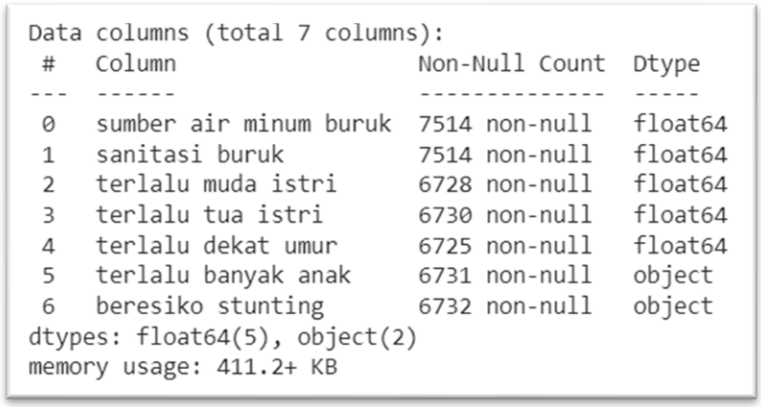
4.1 *Knowledge Discovery In Database* (KDD)

Proses pembahasan pertama penelitian ini berdasarkan hasil dari proses *Knowledge Discovery In Database* (KDD). KDD berisi rangkaian tahap pengolahan data yang beruntun untuk menghasilkan sebuah *knowledge*.

4.1.1 Data *Selection*

Data yang digunakan pada penelitian ini adalah data primer keluarga beresiko *stunting* tahun 2022 yang bersumber dari Dinas Pemberdayaan Perempuan Perlindungan Anak dan Keluarga Berencana (DP3AKB) Kabupaten Majalengka. Data yang dipakai adalah data Kecamatan Argapura. Berdasarkan pada Gambar 4.3 atribut pada data prediksi keluarga beresiko *stunting* ini berjumlah 7 jenis yaitu sumber air minum buruk, sanitasi buruk, terlalu muda istri, terlalu tua istri, terlalu dekat umur, terlalu banyak anak, dan beresiko

stunting. Tujuh atribut data yang menjadi faktor keluarga beresiko *stunting* ini berdasarkan pada kriteria atau aturan yang sudah dirumuskan oleh BKKBN, yang divisualisasikan seperti pada Gambar 1.2.



```
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   sumber air minum buruk  7514 non-null   float64
 1   sanitasi buruk          7514 non-null   float64
 2   terlalu muda istri      6728 non-null   float64
 3   terlalu tua istri       6730 non-null   float64
 4   terlalu dekat umur      6725 non-null   float64
 5   terlalu banyak anak     6731 non-null   object
 6   beresiko stunting       6732 non-null   object
dtypes: float64(5), object(2)
memory usage: 411.2+ KB
```

***Gambar 4. 3** Informasi Dataset*

Berdasarkan Gambar 4.3, pada dataset ini berisi 7 kolom dengan 2 jenis tipe data yaitu float64 dan object. Jumlah isi data pun masih belum seragam dan memiliki angka yang berbeda. Dataset ini masih berupa data mentah yang selanjutnya akan dilakukan tahap *preprocessing*.

***Tabel 4. 1** Informasi Record Dataset*

index	sumber air minum buruk	sanitasi buruk	terlalu muda istri	terlalu tua istri	terlalu dekat umur	terlalu banyak anak	beresiko *stunting*
0	0.0	0.0	0.0	0.0	0.0	0	0
1	0.0	0.0	0.0	0.0	0.0	0	0
2	0.0	0.0	0.0	0.0	0.0	0	0
3	0.0	0.0	0.0	0.0	0.0	0	0
4	0.0	0.0	0.0	0.0	0.0	0	0
…	…	…	…	…	…	…	…
7516	0.0	1.0	0.0	0.0	0.0	0	1

Tabel 4.1 ini menjelaskan isi data yang ada pada dataset. Dapat disimpulkan bahwa isi dari tabel tersebut berupa *binary values* yang hanya berisi data berupa angka 1 dan 0 namun masih belum seragam tipe datanya sehingga belum layak untuk dibuat model prediksi dan perlu dirapikan terlebih dahulu. Jumlah index pada dataset mencapai 7516 data.

4.1.2 *Preprocessing* Data

Preprocessing data dilakuan untuk merapikan data agar layak diolah untuk pembuatan model prediksi. Pada dataset ini terdapat banyak *missing value* atau data yang tidak berisi nilai. Pada kasus keluarga beresiko *stunting* ini peneliti memilih *drop value* karena tipe data berupa *binary* dan konteks nya "iya" dan "tidak" sehingga jika dilakukan imputasi akan mempengaruhi isi data karena isi dari dataset ini berupa numerik "1" dan "0" saja.

```
#MENGHITUNG MISSING VALUE
df.isnull() .sum()

sumber air minum buruk      3
sanitasi buruk              3
terlalu muda istri        789
terlalu tua istri         787
terlalu dekat umur        792
terlalu banyak anak       786
beresiko stunting         785
dtype: int64
```

Gambar 4. 4 *Missing Value*

Berdasarkan pada Gambar 4.4 *missing value* pada dataset ini memiliki jumlah yang sangat banyak. Sehingga *missing value* tersebut harus dibuang agar tidak mengganggu dalam kinerja pembuatan model prediksi.

```
#MENGHAPUS MISSING VALUE
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
```

Gambar 4. 5 *Menghapus Missing Values*

Langkah selanjutnya adalah melakukan penghapusan *missing value*. dapat dilihat pada Gambar 4.5 dengan menghapus baris yang memiliki *missing value*.

Tabel 4. 2 *Record Dataset Setelah Penghapusan Missing Values*

index	sumber air minum buruk	sanitasi buruk	terlalu muda istri	terlalu tua istri	terlalu dekat umur	terlalu banyak anak	beresiko *stunting*
0	0.0	0.0	0.0	0.0	0.0	0	0
1	0.0	0.0	0.0	0.0	0.0	0	0
2	0.0	0.0	0.0	0.0	0.0	0	0

3	0.0	0.0	0.0	0.0	0.0	0	0
4	0.0	0.0	0.0	0.0	0.0	0	0
…	…	…	…	…	…	…	…
6708	0.0	1.0	0.0	0.0	0.0	0	1

Setelah menghapus semua *missing value*, maka dataset sudah selesai dari tahap *preprocessing*. Index data pada dataset pun telah berkurang dari 7516 menjadi 6708 seperti pada Tabel 4.2. Berikut adalah tampilan data setelah *preprocessing:*

```
#MISSING VALUE SETELAH PREPROCESSING
df.isnull() .sum()

sumber air minum buruk    0
sanitasi buruk            0
terlalu muda istri        0
terlalu tua istri         0
terlalu dekat umur        0
terlalu banyak anak       0
beresiko stunting         0
dtype: int64
```

***Gambar 4. 6** Missing Values Setelah Preprocessing*

Setelah menghapus semua *missing value*, maka dataset sudah terbebas dari *missing value*. Hasil dari Gambar 4.6 memperlihatkan semua *missing value* sudah teratasi.

```
#INFORMASI DATA SETELAH PREPROCESSING
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6709 entries, 0 to 6708
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   sumber air minum buruk  6709 non-null   float64
 1   sanitasi buruk          6709 non-null   float64
 2   terlalu muda istri      6709 non-null   float64
 3   terlalu tua istri       6709 non-null   float64
 4   terlalu dekat umur      6709 non-null   float64
 5   terlalu banyak anak     6709 non-null   object
 6   beresiko stunting       6709 non-null   object
dtypes: float64(5), object(2)
memory usage: 367.0+ KB
```

***Gambar 4. 7** Informasi Dataset Setelah Preprocessing*

Gambar 4.7 menjelaskan bahwa proses preprocessing sudah selesai. *Missing value* sudah teratasi dan indeks data pun sama jumlahnya. Sehingga bisa dilanjutkan ke tahap selanjutnya.

4.1.3 *Transformation* Data

Trasnformasi data dilakukan apabila terdapat perbedaan dan ketidaksesuaian tipe data pada dataset. Pada dataset ini terdapat ketidaksesuaian tipe data float dan object yang dinilai kurang efektif untuk pembuatan model sehingga perlu diubah menjadi tipe data integer untuk pembuatan model prediksi. Hasilnya dapat dilihat pada Gambar 4.8.

```
#TRANSFORMASI DATA
df = df.astype(int)
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6709 entries, 0 to 6708
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   sumber air minum buruk  6709 non-null   int64
 1   sanitasi buruk          6709 non-null   int64
 2   terlalu muda istri      6709 non-null   int64
 3   terlalu tua istri       6709 non-null   int64
 4   terlalu dekat umur      6709 non-null   int64
 5   terlalu banyak anak     6709 non-null   int64
 6   beresiko stunting       6709 non-null   int64
dtypes: int64(7)
memory usage: 367.0 KB
```

Gambar 4. 8 *Transformasi Data*

Tabel 4. 3 *Record Dataset Setelah Transformasi Data*

index	sumber air minum buruk	sanitasi buruk	terlalu muda istri	terlalu tua istri	terlalu dekat umur	terlalu banyak anak	beresiko *stunting*
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…
6708	0	1	0	0	0	0	1

Setelah dilakukan transformasi data, tipe data menjadi seragam sesuai pada Tabel 4.3 dan jauh berbeda dengan Tabel 4.2 yang masih belum rapi.

4.1.4 Data Mining

Tahapan selanjutnya adalah data *mining*, yang akan mengolah data dan mengolah proses pembuatan model. Tahap ini terbagi menjadi beberapa bagian:

1. **Penentuan Variabel X dan Y**

   Dalam proses pembuatan model prediksi, diperlukan pemisahan antara variabel faktor dan variabel penentu. Hal ini sering

direpresentasikan dengan variabel x dan variabel y. Variabel x berisi kolom yang menjadi faktor, sedangkan variabel y berisi kolom yang menjadi penentu. Pada kasus ini, variabel x terdiri dari kolom sumber air minum buruk, sanitasi buruk, terlalu muda istri, terlalu tua istri, terlalu dekat umur, dan terlalu banyak anak. Sedangkan variabel y terdiri dari kolom beresiko *stunting*. Pada Gambar 4.9 menjelaskan tahap pembagian variabel x dan y menggunakan df.iloc.

```
#MEMBAGI VARIABEL X DAN Y
x = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Gambar 4. 9 *Membagi Variabel X dan Variabel Y*

```
print(x)

[[0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 ...
 [0 1 0 0 0 0]
 [0 1 0 0 0 0]
 [0 1 0 0 0 0]]

print (y)

[0 0 0 ... 1 1 1]
```

Gambar 4. 10 *Tampilan Isi Variabel X dan Variabel Y*

Setelah dilakukan pembagian variabel x dan y, isi data akan terbagi menjadi dua bagian seperti yang terlihat pada Gambar 4.10 dengan keterangan variabel x berisi 6 kolom yang berisi data faktor dan variabel y berisi 1 kolom yang berisi data penentu.

2. Data *Training* & Data *Testing*

Pembagian data *training* dan data *testing* selalu ada dalam proses prediksi. Data *training* digunakan untuk pembuatan model prediksi, sedangkan data *testing* digunakan untuk pengujian akurasi model prediksi.

```
#MEMBAGI DATA TRAINING & DATA TESTING
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=42

print("x_train = ", len(x_train))
print("x_test = ", len(x_test))
print("y_train = ", len(y_train))
print("y_test = ", len(y_test))

x_train =  5367
x_test =  1342
y_train =  5367
y_test =  1342
```

Gambar 4. 11 *Data Training & Data Testing*

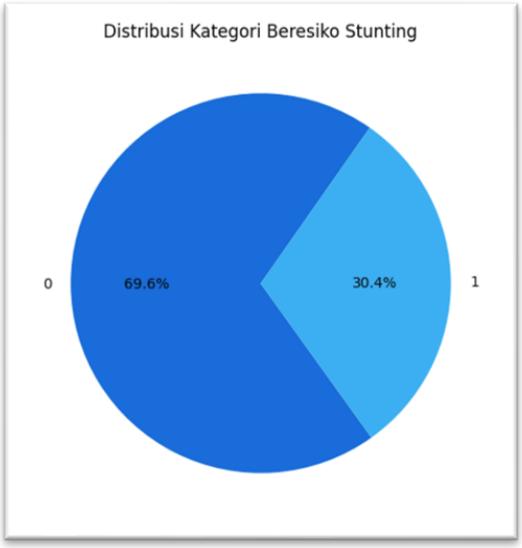
Gambar 4.11 adalah tahap pembagian data *training* & data *testing*. Hasil dari pembagian tersebut data *training* berjumlah 5367 dan data *testing* berjumlah 1342 dengan rasio pembagiannya 80:20 merujuk kepada penelitian terdahulu.

3. *Synthetic Minority Over-sampling Technique* (SMOTE)

Tahap ini adalah tahap menganalisis karakter dataset dengan melakukan pengecekan terhadap potensi *imbalance* data. Dalam pembuatan model prediksi, isi dataset harus seimbang pada variabel y atau variabel penentu. Pada Gambar 4.12, dari dataset yang ada, jumlah pada kolom beresiko *stunting* atau variabel y memiliki ketidakseimbangan sehingga perlu diseimbangkan dahulu sebelum tahap pembuatan model.

```
#ANALISIS DATA PADA KOLOM BERESIKO STUNTING
class_distribution = np.bincount(y_train)
print(class_distribution)

[3736 1631]
```

Gambar 4. 12 *Analisis Kolom Beresiko Stunting*



Gambar 4. 13 *Chart Distribusi Beresiko Stunting*

Setelah dilakukan analisis, pada data *training* yang akan digunakan memiliki kasus *imbalance* data. Visualisasi dari *imbalance* data dapat dilihat pada Gambar 4.13 sehingga berdasarkan informasi tersebut diperlukan upaya untuk menyeimbangkan data karena model prediksi tidak dapat dibuat apabila data *training* dalam kondisi *imbalance*.

Peneliti memilih metode *oversampling* SMOTE untuk mengatasi kasus *imbalance* data ini. Konsep SMOTE adalah menyeimbangkan data dengan mesintesis data baru pada kelas minoritas agar jumlah datanya dapat mengimbangi kelas mayoritas. Penjelasan tahap SMOTE terdapat pada Gambar 4.14.

```
#Penerapan Oversampling SMOTE
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=123)

x_train_resampled, y_train_resampled = sm.fit_resample(x_train, y_train)

y_train_resampled_series = pd.Series(y_train_resampled)
```

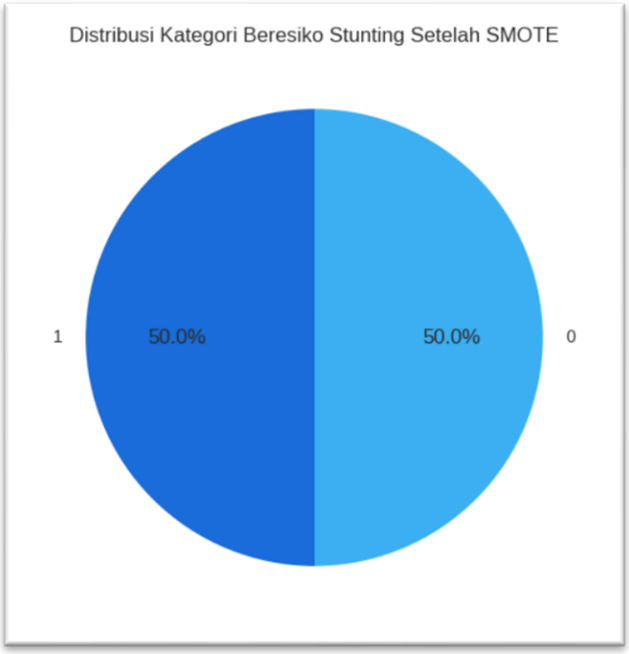
Gambar 4. 14 *Oversampling SMOTE*

Setelah dilakukan tahap *oversampling* SMOTE, data yang semula tidak seimbang menjadi seimbang dengan jumlah yang sama. Hasil data yang sudah dilakukan tahap *oversampling* SMOTE dapat dilihat pada Gambar 4.15.

```
#PERBANDINGAN DATA OVERSAMPLING
print('Before Sampling')
print(pd.Series(y_train).value_counts())
print('')
print('After Sampling')
print(y_train_resampled_series.value_counts())

Before Sampling
0    3736
1    1631
dtype: int64

After Sampling
1    3736
0    3736
dtype: int64
```

Gambar 4. 15 *Perbandingan Hasil Oversampling*

Visualisasi perbandingan jumlah data setelah dilakukan tahap *oversampling* dapat dilihat pada Gambar 4.16. Sekarang setiap kategori memiliki jumlah data yang sama.



Gambar 4. 16 *Chart Distribusi Beresiko Stunting Setelah SMOTE*

Proses *oversampling* SMOTE juga tidak menyebabkan potensi munculnya *missing value* baru sehingga data sintesis yang dibuat tidak akan mempengaruhi data yang sudah ada. Berikut pada Gambar 4.17 adalah informasi *missing value* pada data yang telah dilakukan proses *oversampling* SMOTE.

```
#MENGECEK MISSING VALUES SETELAH OVERSAMPLING
print(pd.Series(y_train).isna().sum())

0
```

***Gambar 4. 17** Missing Value Setelah Oversampling*

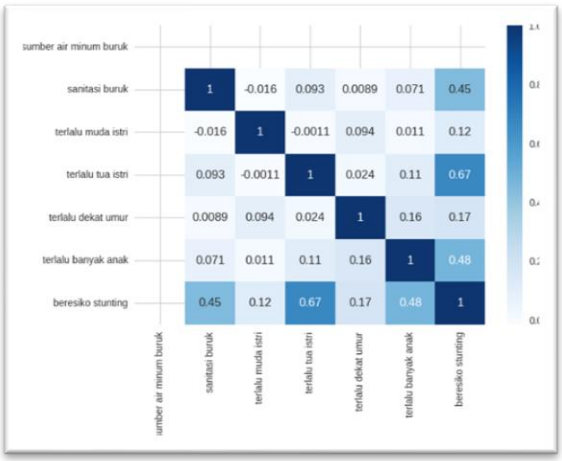
4. Pembuatan Model

Sebelum model dibuat, perlu diketahui hubungan antar variabel yang ada pada dataset. Dari hasil yang sudah ada, pada Gambar 4.18 merupakan tahap membuat *heatmap* korelasi.

```
#KORELASI DATA
sns.heatmap(df.corr(), cmap="Blues", annot=True)
```

***Gambar 4. 18** Korelasi Data*

Hasil visualisasi korelasi antar variabel terhadap variabel beresiko *stunting* dapat dilihat pada Gambar 4.19, yang menunjukkan bahwa korelasi memiliki nilai positif semua, sehingga semua variabel yang ada pada dataset dapat digunakan untuk pembuatan model prediksi.



***Gambar 4. 19** Heatmap Korelasi Data*

Menurut penelitian sebelumnya, apabila terdapat korelasi negatif, maka atribut dapat dipertimbangkan untuk dibuang atau tetap dimasukkan dalam proses pembuatan model. Setelah mengetahui dan menganalisis korelasi antar variabel dengan variabel beresiko *stunting,* maka model sudah dapat dibuat.

```
#PEMBUATAN MODEL NAIVE BAYES
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

```
GaussianNB
GaussianNB(priors=None, var_smoothing=1e-09)
```

Gambar 4. 20 *Algoritma Naïve Bayes*

Gambar 4.20 menjelaskan proses pembuatan model Naïve Bayes. Dalam pembuatan model prediksi, algoritma Naïve Bayes dibangun menggunakan data *training* dari dataset keluarga beresiko *stunting*.

4.1.5 Evaluasi Model

Setelah model dibuat menggunakan data *training*, selanjutnya data *testing* digunakan untuk evaluasi model. Dalam tahap evaluasi model ini, dilakukan evaluasi model berupa penghitungan akurasi model dan *confusion matrix*. Proses pembuatan akurasi model bertujuan untuk evaluasi model yang merujuk pada BAB II persamaan (2.2) sampai dengan persamaan (2.5).

```
#AKURASI DALAM BENTUK PERSEN
akurasi = accuracy_score(y_test, y_pred)
print("Tingkat Akurasi : %d" %(akurasi*100), "%")
```

```
Tingkat Akurasi : 98 %
```

Gambar 4. 21 *Nilai Akurasi Model*

Perhitungan akurasi model menggunakan accuracy score yang direpresentasikan menggunakan persen seperti yang terlihat pada Gambar 4.22. Accuracy score memproses penilaian akurasi model dengan memprediksi benar dan salah secara tepat dengan proses kalkulasi pada persamaan (2.2) pada penjelasan 2.8.1. Didapatkan akurasi 98% dalam penentuan benar dan salah pada model ini.

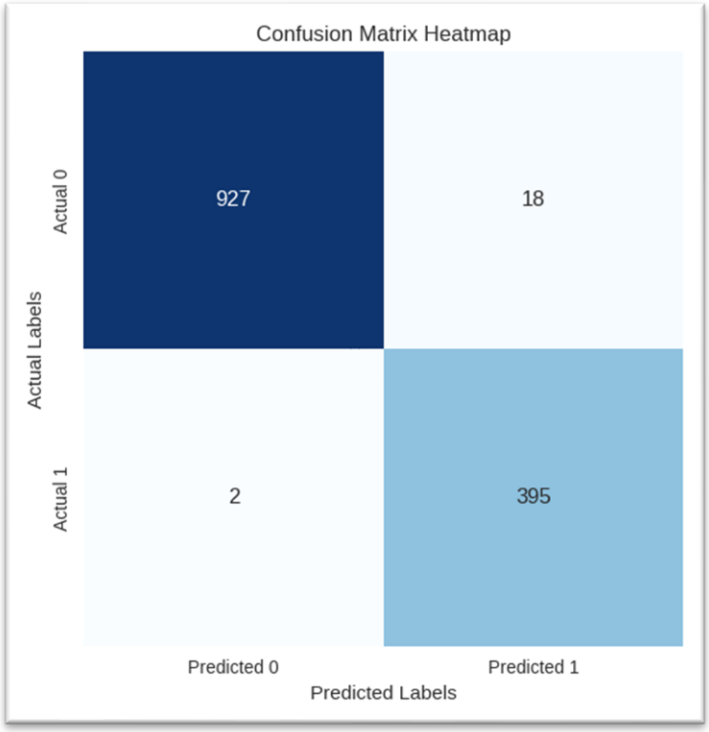
```
#AKURASI MODEL
akurasi = classification_report(y_test, y_pred)
print(akurasi)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       945
           1       0.96      0.99      0.98       397

    accuracy                           0.99      1342
   macro avg       0.98      0.99      0.98      1342
weighted avg       0.99      0.99      0.99      1342
```

Gambar 4. 22 *Akurasi dalam Bentuk Persen*

Proses perhitungan akurasi selanjutnya menggunakan classification report yang menampilkan informasi berdasarkan precision, recall, dan f1-score. Proses kalkulasinya diperoleh berdasarkan persamaan (2.3) sampai dengan (2.5) pada penjelasan 2.8.2 sampai 2.8.4.

Precision mengukur akurasi dan memberikan report dari jumlah prediksi positif yang benar dengan hasil kategori tidak beresiko 100% dan beresiko 96%. Rata-rata nilai precision sebesar 98%. Recall mengidentifikasi kasus yang berjenis positif dengan hasil kategori tidak beresiko 98% dan beresiko 99%. Rata-rata nilai recall sebesar 99%. F1-score menganalisis keseimbangan antara precision dan recall dengan hasil kategori tidak beresiko 99% dan beresiko 98%. Rata-rata nilai F1-Score sebesar 98%.



Gambar 4. 23 *Heatmap Confusion Matrix*

Selanjutnya digunakan *heatmap confusion matrix* untuk menampilkan sebaran data hasil prediksi. Hasil dari *confusion matrix* pada Gambar 4.23 memperlihatkan jumlah data yang tersebar pada hasil prediksi. Rumus *confusion matrix* pada Tabel 2.1 diterapkan pada tahap ini, dengan penjelasan sebagai berikut:

0 = Tidak beresiko

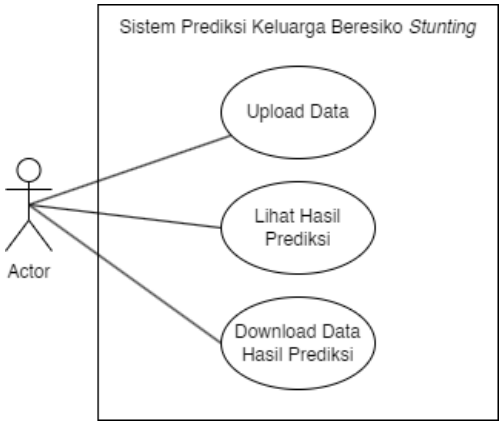
1 = Beresiko

- TP (*True Positive*), Terdapat 927 data yang dinyatakan "tidak beresiko *stunting*" dan diprediksi secara akurat.
- FN (*False Negative*), Terdapat 18 data yang dinyatakan "tidak beresiko" namun diprediksi sebaliknya.
- FP (*False Positive*), Terdapat 2 data yang sebenarnya "beresiko" tetapi diprediksi sebagai "tidak beresiko".
- TN (*True Negative*), Terdapat 395 data diprediksi "beresiko" secara akurat.

-

4.2 Implementasi

Tahap implementasi adalah penerapan model prediksi yang sudah dibuat pada sebuah sistem, sehingga dapat dihasilkan sebuah sistem prediksi keluarga beresiko *stunting.* Namun sebelum membuat model prediksi, diperlukan *use case diagram* untuk memvisualisasikan proses jalannya sistem.



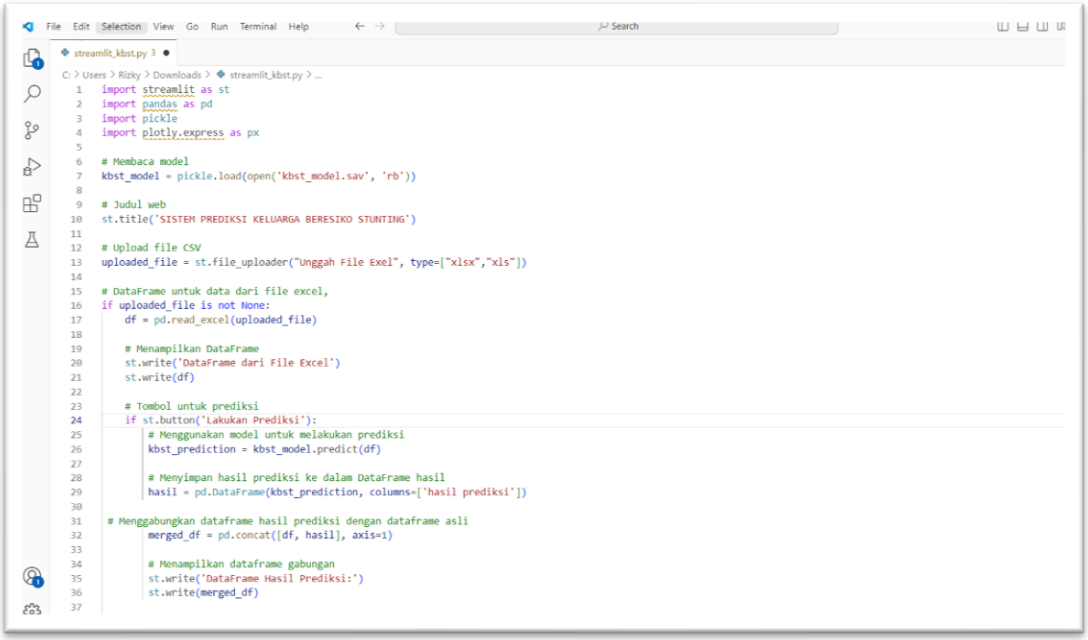
Gambar 4. 24 *Use Case Diagram*

Pada rancangan sistem prediksi sederhana ini, hanya memerlukan dua tahap untuk melakukan proses prediksi. Seperti pada Gambar 4.24 sistem sederhana ini hanya membutuhkan input angka untuk diproses dan memproses *submit* angka yang sudah dimasukkan sehingga selanjutnya akan tampil kategori dari data tersebut apakah beresiko atau tidak.

```
#MENYIMPAN MODEL
import pickle
model = 'kbst_model.sav'
pickle.dump(classifier, open(model, 'wb'))
```

Gambar 4. 25 *Menyimpan Model Prediksi*

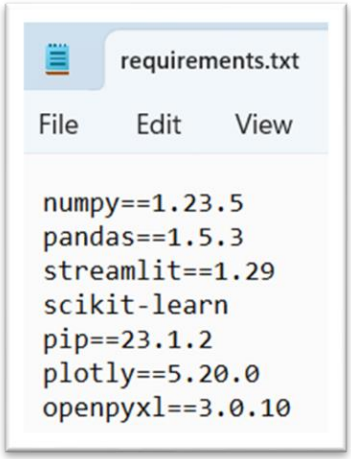
Hal pertama yang perlu dilakukan dalam pembuatan model adalah dengan menyimpan model prediksi dengan format sav agar dapat diproses untuk pembuatan sistem. Setelah model prediksi dibuat seperti pada Gambar 4.25, model tersebut akan diimplementasikan menjadi sebuah sistem prediksi sederhana berbasis web menggunakan streamlit. Diperlukan kode program untuk menjalankan streamlit, dan ringkasan kode nya adalah pada Gambar 4.26 berikut:



```python
import streamlit as st
import pandas as pd
import pickle
import plotly.express as px

# Membaca model
kbst_model = pickle.load(open('kbst_model.sav', 'rb'))

# Judul web
st.title('SISTEM PREDIKSI KELUARGA BERESIKO STUNTING')

# Upload file CSV
uploaded_file = st.file_uploader("Unggah File Exel", type=["xlsx","xls"])

# DataFrame untuk data dari file excel,
if uploaded_file is not None:
    df = pd.read_excel(uploaded_file)

    # Menampilkan DataFrame
    st.write('DataFrame dari File Excel')
    st.write(df)

    # Tombol untuk prediksi
    if st.button('Lakukan Prediksi'):
        # Menggunakan model untuk melakukan prediksi
        kbst_prediction = kbst_model.predict(df)

        # Menyimpan hasil prediksi ke dalam DataFrame hasil
        hasil = pd.DataFrame(kbst_prediction, columns=['hasil prediksi'])

# Menggabungkan dataframe hasil prediksi dengan dataframe asli
        merged_df = pd.concat([df, hasil], axis=1)

        # Menampilkan dataframe gabungan
        st.write('DataFrame Hasil Prediksi:')
        st.write(merged_df)
```

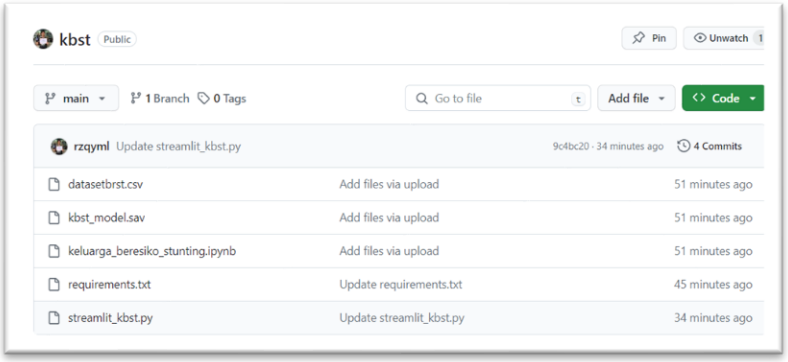
Gambar 4. 26 *Source Code Pembuatan Streamlit*

Selanjutnya *library* yang dipakai pada python dan streamlit dikumpulkan dalam sebuah file dengane extensi ".txt" agar *library* dapat terbaca pada streamlit. Jenis *library* yang dimasukkan terdapat pada Gambar 4.27. *Library* tertentu diberi keterangan versi agar dapat memudahkan sistem dalam mengidentifikasi *library*.



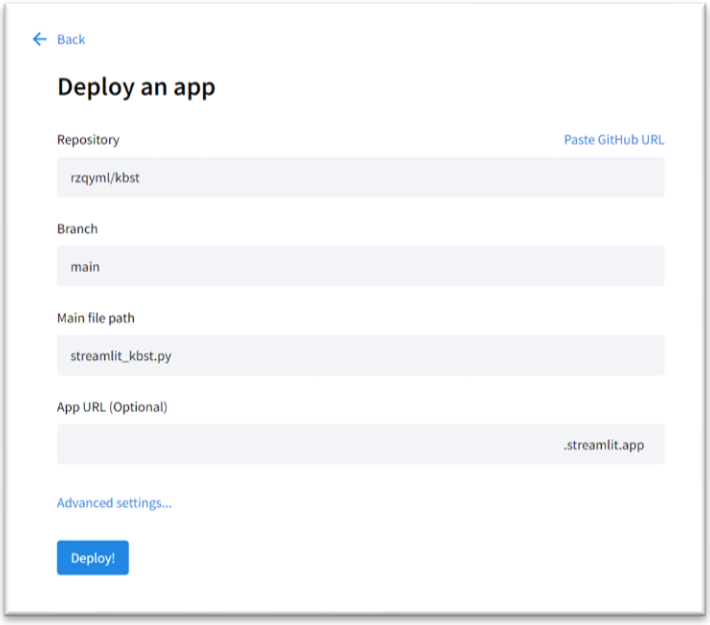
Gambar 4. 27 *Library Streamlit*

Dalam proses implementasi sistem prediksi menggunakan streamlit ini, diperlukan beberapa file yaitu dataset, model prediksi, project python, daftar library, dan source code streamlit seperti pada Gambar 4.28. File tersebut perlu diunggah pada repositori github karena streamlit akan mengakses file tersebut melalui repositori github.

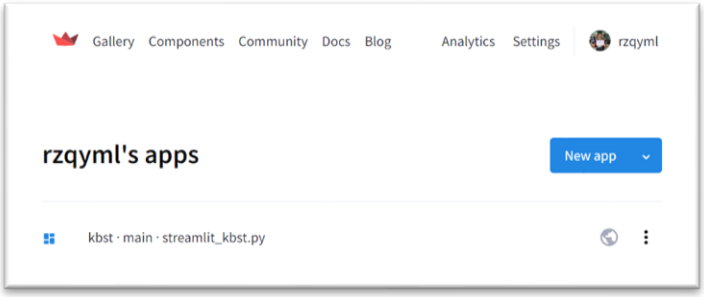


Gambar 4. 28 *Upload Repositori Github*

Setelah semua file berada pada repositori github, dapat dilakukan implementasi model menjadi sistem prediksi dengan platform streamlit. Gambar 4.29 adalah proses embuatan sistem/aplikasi prediksi sederhana akan mengakses beberapa variabel dari repositori github.

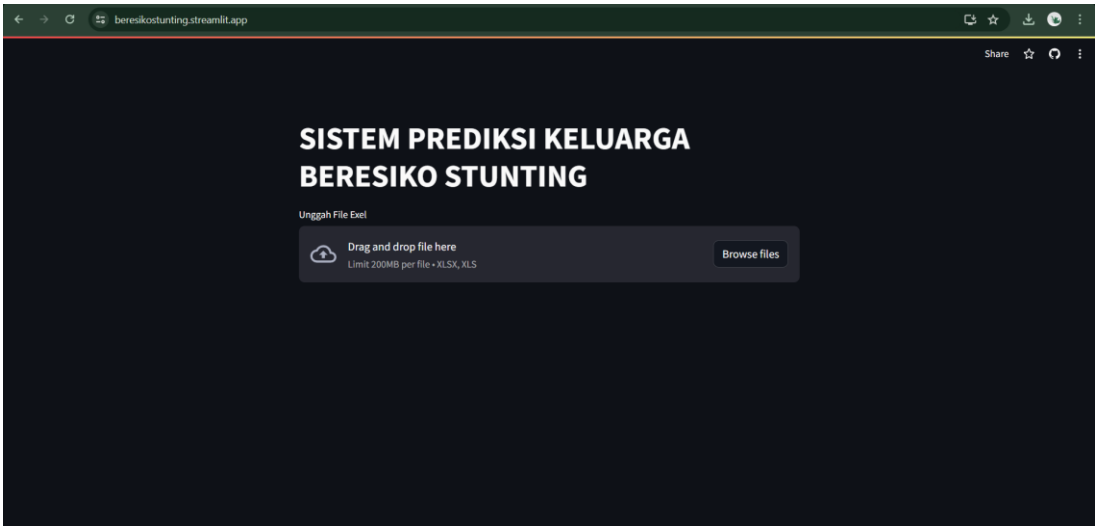


Gambar 4. 29 *Sinkronisasi Streamlit dengan Repositori Github*



Gambar 4. 30 *Pembuatan Aplikasi Prediksi pada Streamlit*

Setelah aplikasi dibuat, dan tampilan halaman sudah seperti Gambar 4.30, sistem sudah dapat digunakan untuk proses prediksi dalam penentuan kategori keluarga beresiko *stunting* dengan mengupload dataset.

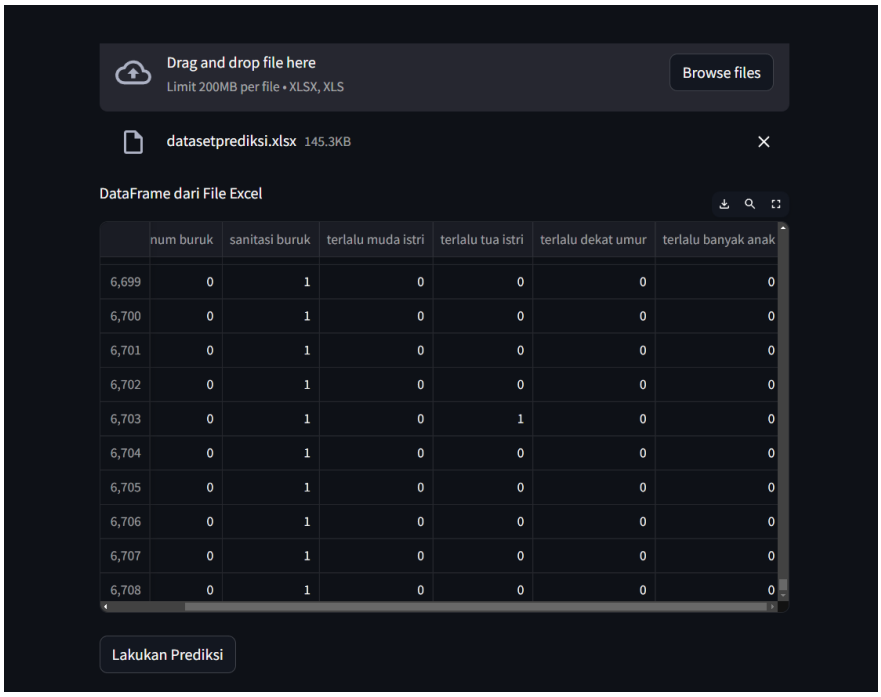


Gambar 4. 31 *Tampilan Sistem Prediksi Sederhana Menggunakan Streamlit*

Tema tampilan sistem prediksi pada Gambar 4.31 ini disesuaikan mengikuti tema perangkat pengguna dengan opsi latar belakang belap dan terang. Link sistem prediksi keluarga beresiko *stunting* dapat diakses pada link berikut:

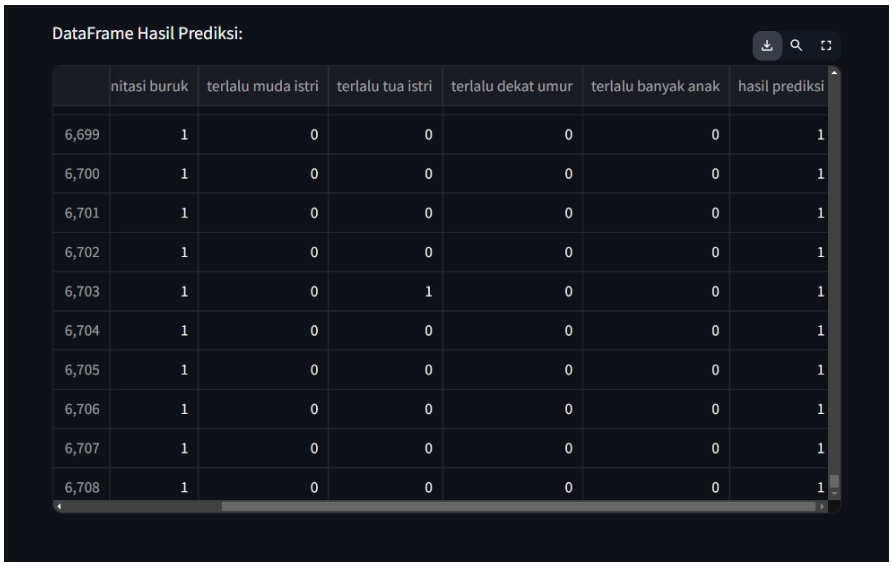
https://beresikostunting.streamlit.app/

Setelah sistem dibuat, tentu harus dilakukan pengujian sistem untuk memastikan sistem beroperasi dengan baik dan mengetahui hasil implementasi model prediksi pada sistem. Pada *sample* percobaan sistem prediksi ini, peneliti mengunggah dataset yang berada pada kecamatan Argapura dengan menghilangkan kolom beresiko *stunting* yang selanjutnya kolom tersebut akan terisi otomatis dengan hasil prediksi. Proses pengujian sistem dapat dilihat pada Gambar 4.32.



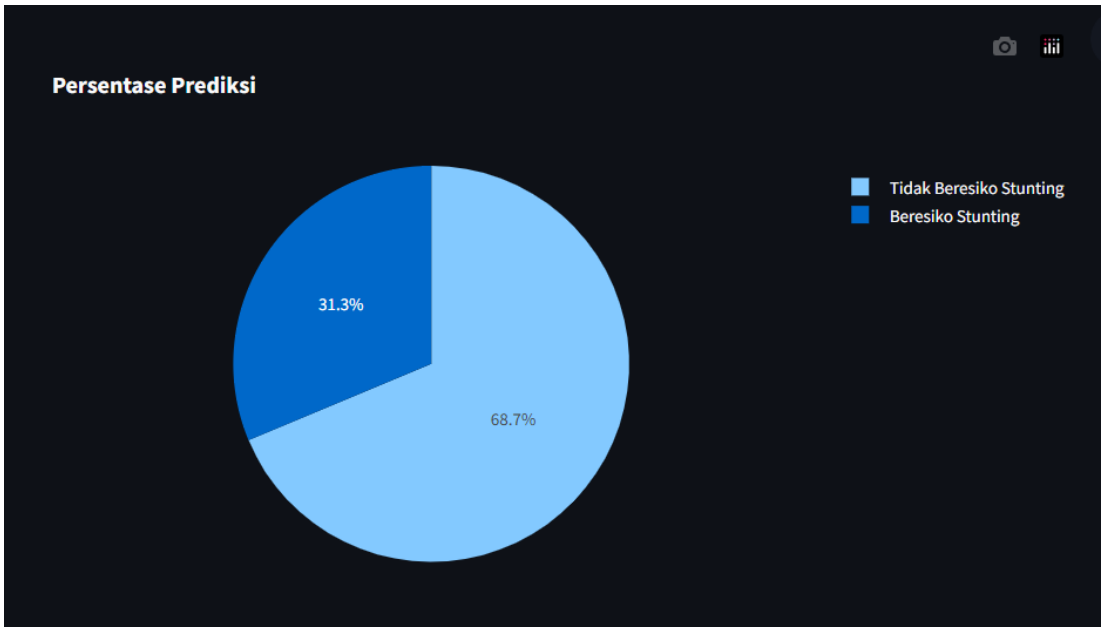
Gambar 4. 32 *Proses Pengolahan Dataset Uji Coba Sistem*

Setelah file diunggah, sistem akan menampilkan isi file berupa dataframe. Data sudah tersimpan dan dibaca oleh sistem sehingga dapat dilakukan proses prediksi dengan menekan tombol lakukan prediksi.



Gambar 4. 33 *Dataset Hasil Prediksi*

Setelah tombol lakukan prediksi ditekan, akan tampil dataframe yang menampilkan hasil prediksi seperti pada Gambar 4.33. kolom hasil prediksi akan otomatis terisi dengan nilai hasil prediksi. Dataframe hasil prediksi tersebut juga dapat diunduh langsung ke perangkat.



Gambar 4. 34 *Sebaran Data Hasil Prediksi*

Sistem ini juga menyediakan visualisasi sebaran data hasil prediksi berupa diagram lingkaran secara otomatis. Diagram ini tampil pada halaman paling bawah. Diagram ini juga dapat diunduh ke perangkat untuk pembuatan laporan atau untuk kebutuhan data lainnya.

Dengan ini dapat diambil kesimpulan bahwa model prediksi yang telah dibangun menggunakan Naïve Bayes pada dataset keluarga beresiko stunting sudah teruji dan layak diterapkan pada sistem sederhana prediksi keluarga beresiko *stunting*. Sistem prediksi keluarga beresiko *stunting* yang telah dibuat menggunakan streamlit juga dapat berjalan dengan baik dan alur dari sistem prediksi sederhana dan efektif sehingga dapat digunakan untuk mengolah data prediksi.