

Machine Learning

Kelompok 3

PENERAPAN ALGORITMA RANDOM FOREST CLASSIFIER UNTUK PREDIKSI STOK PENJUALAN PRODUK BOLEN CRISPY



KELOMPOK 3

Muhamad Azi Sudarya

1

Nadia Chusnul Ikromah

Muhamad Ridho Alfarizi

2

Ravina Indriyani

Khaisar Rizki Maulana

3

Rosmawati

TABLE OF CONTENT

- 1 Latar Belakang**
- 2 Tujuan**
- 3 Random Forest Classifier**
- 4 Preprocessing Data**

LATAR BELAKANG



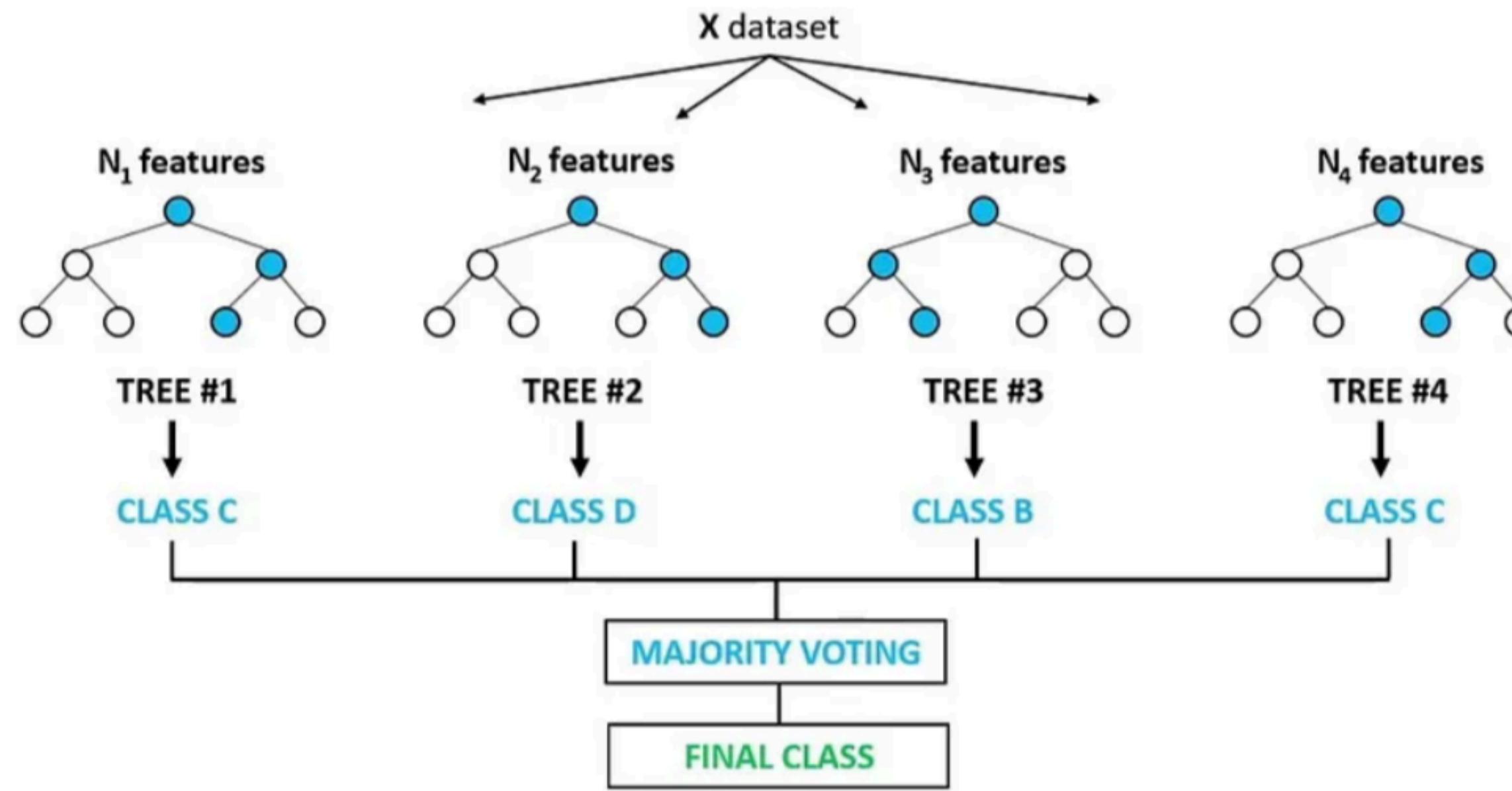
UMKM seperti Bolen Crispy menghadapi tantangan dalam mengelola stok akibat permintaan yang fluktuatif. Pengelolaan yang masih manual sering menyebabkan overstock atau understock, berdampak pada pemborosan dan kerugian penjualan. Untuk mengatasi hal ini, dibutuhkan sistem prediktif berbasis data yang akurat.

Algoritma Random Forest Classifier dipilih karena mampu menangani data kompleks dan memberikan hasil klasifikasi yang andal. Berdasarkan kajian jurnal dan penelitian, algoritma ini terbukti efektif dalam meningkatkan efisiensi stok dan prediksi penjualan.

TUJUAN

- 01** Menerapkan algoritma Random Forest untuk memprediksi stok penjualan produk Bolen Crispy.
- 02** Mengembangkan sistem prediksi berbasis web untuk mendukung pengambilan keputusan manajemen persediaan.
- 03** Menilai akurasi dan performa algoritma Random Forest dalam sistem prediksi.

RANDOM FOREST CLASSIFIER



Random Forest Classifier adalah algoritma machine learning berbasis ensemble yang digunakan untuk klasifikasi data. Algoritma ini membangun banyak pohon keputusan (decision tree) dan menggabungkan hasil klasifikasinya menggunakan voting mayoritas untuk menentukan kelas akhir.

IDENTIFIKASI STRUKTUR DATASET

01 Jumlah kolom dan baris

```
▶ # Menampilkan jumlah kolom dan baris dalam DataFrame  
print("Dataset ini berisi %.f kolom dan %.f baris" %(df.shape[1], df.shape[0]))
```

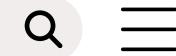
→ Dataset ini berisi 6 kolom dan 1096 baris

02 Data Duplikat

```
[ ] # Menampilkan jumlah data duplikat dalam DataFrame  
print('Data ini memiliki %.f data duplikat' %df.duplicated().sum())
```

→ Data ini memiliki 0 data duplikat

IDENTIFIKASI STRUKTUR DATASET



03 Missing Value

→ Menghitung missing value:

```
Tanggal      0
Hari         0
Nama Produk  0
Harga Satuan 0
Jumlah Terjual 0
Stok Produk   0
dtype: int64
```

04 Informasi Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1096 entries, 0 to 1095
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Tanggal          1096 non-null   object  
 1   Hari              1096 non-null   object  
 2   Nama Produk      1096 non-null   object  
 3   Harga Satuan     1096 non-null   int64  
 4   Jumlah Terjual    1096 non-null   int64  
 5   Stok Produk       1096 non-null   int64  
dtypes: int64(3), object(3)
memory usage: 51.5+ KB
```

KONVERSI TIPE DATA

- **Tanggal**

```
# Mengubah tipe data kolom tanggal menjadi datetime
df['Tanggal'] = pd.to_datetime(df['Tanggal'], dayfirst=True)

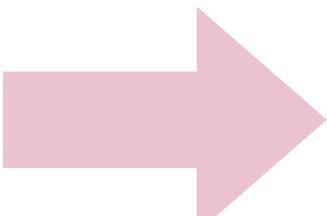
# Mengubah tipe data kolom tanggal menjadi integer
df['Tanggal Int'] = df['Tanggal'].astype('int64') // 10**6
```

- **Hari**

```
▶ # Konversi nama hari jadi angka
mapping_hari = {
    'Senin': 1,
    'Selasa': 2,
    'Rabu': 3,
    'Kamis': 4,
    'Jumat': 5,
    'Sabtu': 6,
    'Minggu': 7
}

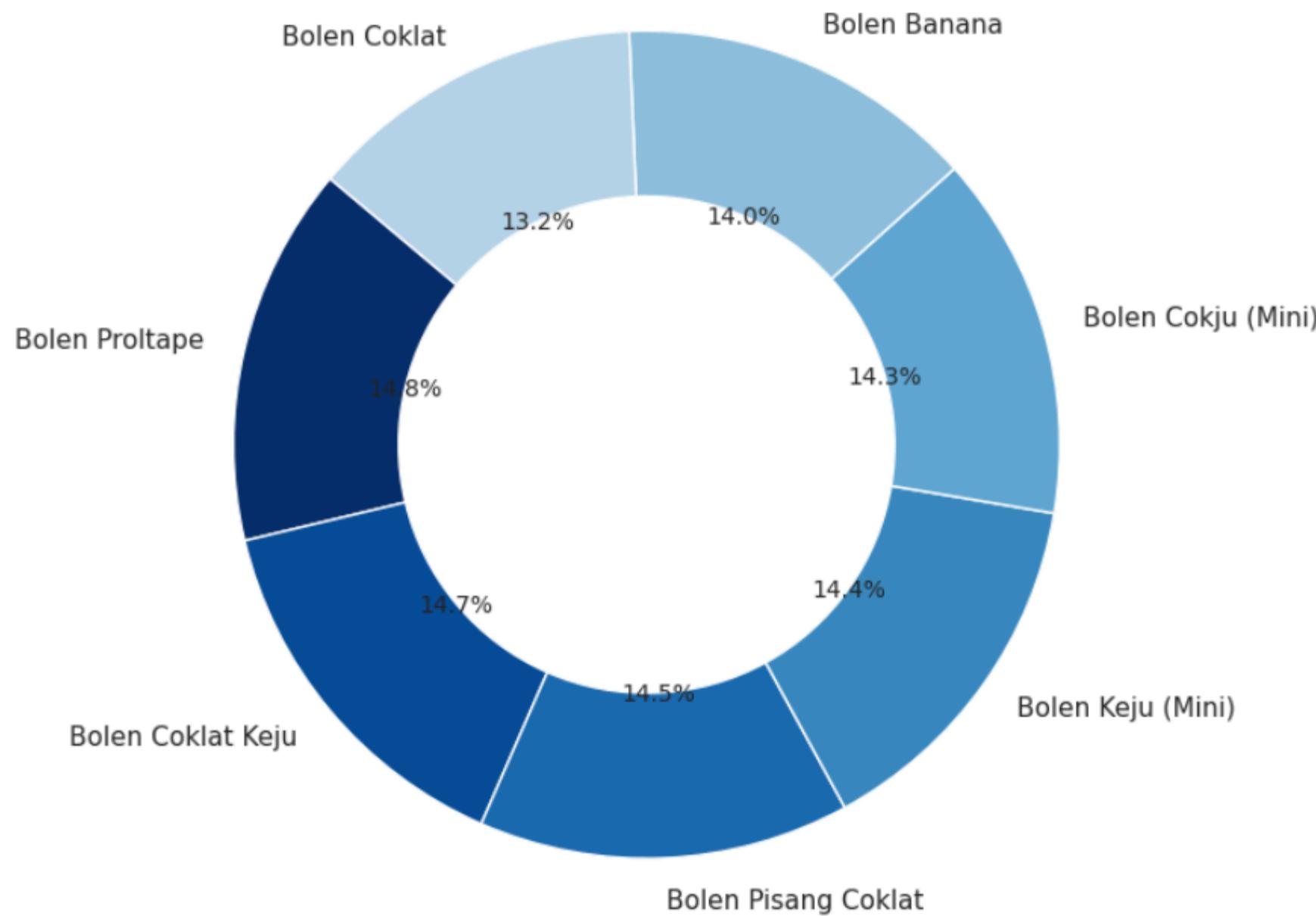
df['Hari'] = df['Hari'].map(mapping_hari)

print(df[['Hari']].head(7))
```



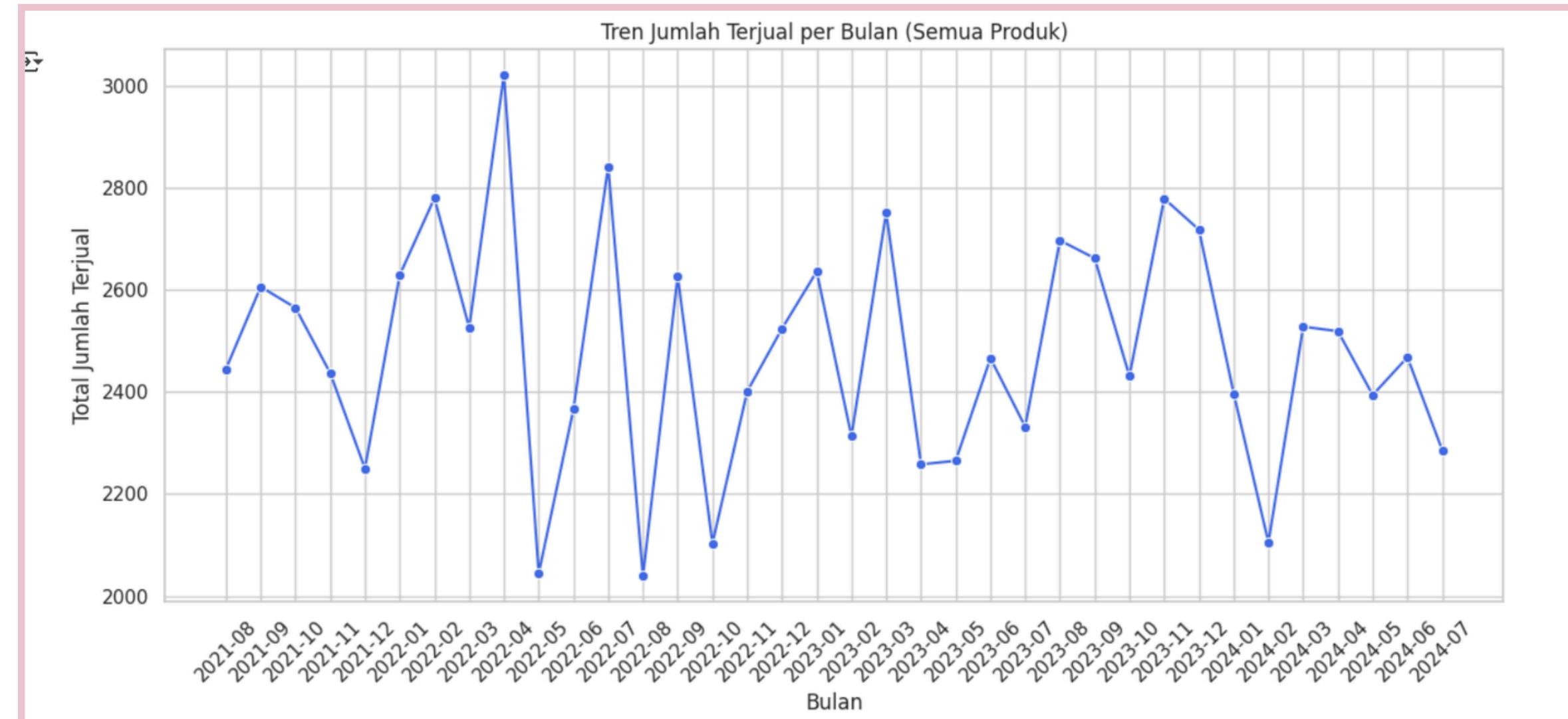
	Hari
0	7
1	1
2	2
3	3
4	4
5	5
6	6

PENJUALAN PRODUK



Berdasarkan visualisasi disamping, terlihat bahwa semua varian produk bolen memiliki proporsi penjualan yang cukup merata, dengan Bolen Proltape menjadi yang paling laris sebesar 14,8%. Perbedaan antar produk tidak terlalu signifikan, yang menunjukkan bahwa seluruh varian bolen memiliki minat pasar yang relatif seimbang. Hal ini dapat menjadi peluang untuk mempertahankan ketersediaan semua varian secara konsisten.

TREN PENJUALAN PRODUK PER BULAN



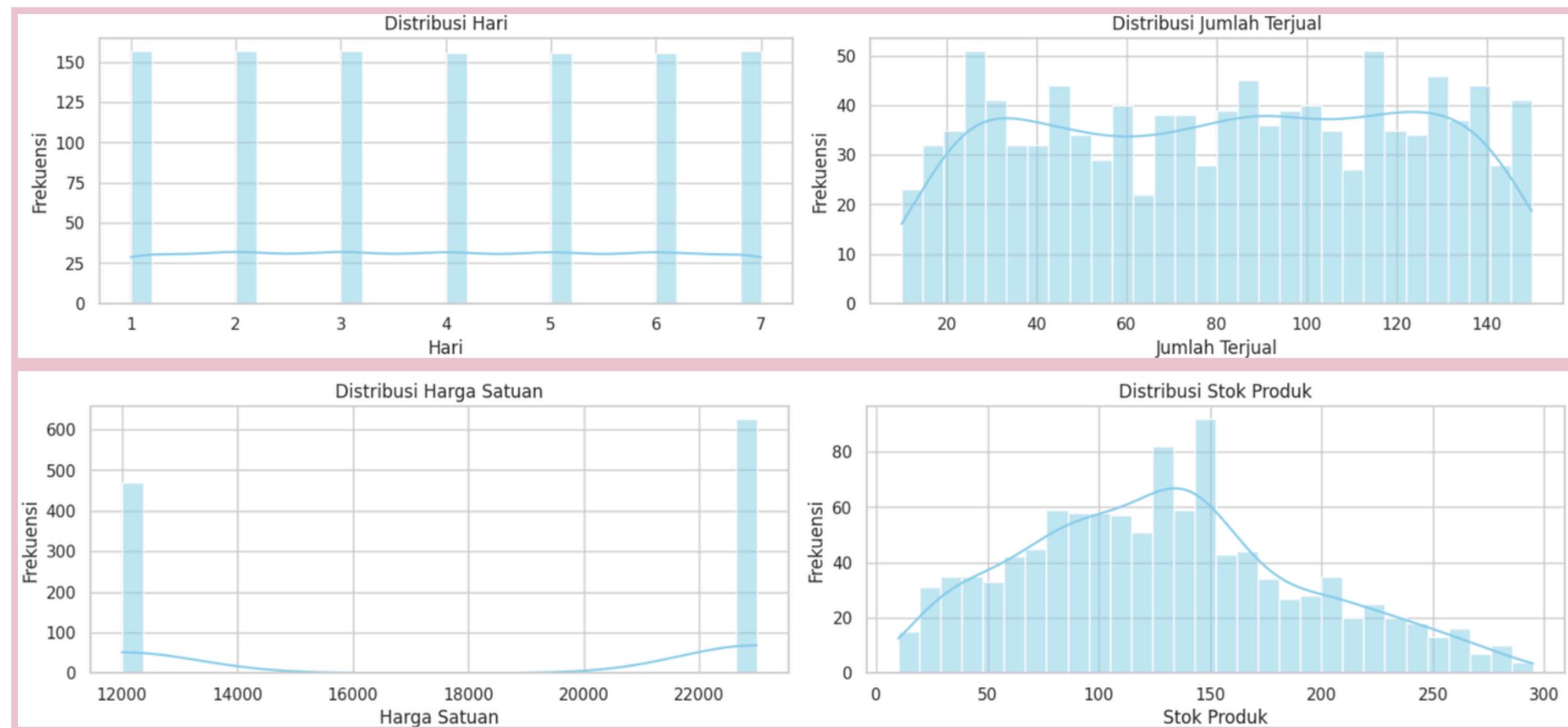
STATISTIK DESKRIPTIF

	Tanggal	Hari	Harga Satuan	Jumlah Terjual	Stok Produk	Tanggal Int
count	1096	1096.000000	1096.000000	1096.000000	1096.000000	1.096000e+03
mean	2023-01-30 12:00:00	3.997263	18292.883212	81.397810	129.081204	1.675080e+12
min	2021-08-01 00:00:00	1.000000	12000.000000	10.000000	10.000000	1.627776e+12
25%	2022-05-01 18:00:00	2.000000	12000.000000	45.000000	82.000000	1.651428e+12
50%	2023-01-30 12:00:00	4.000000	23000.000000	83.000000	127.000000	1.675080e+12
75%	2023-10-31 06:00:00	6.000000	23000.000000	117.000000	168.000000	1.698732e+12
max	2024-07-31 00:00:00	7.000000	23000.000000	150.000000	295.000000	1.722384e+12
std	Nan	2.002508	5445.033302	40.354644	62.946754	2.734839e+10

STATISTIK DESKRIPTIF



Visualisasi distribusi data numerik dengan histogram

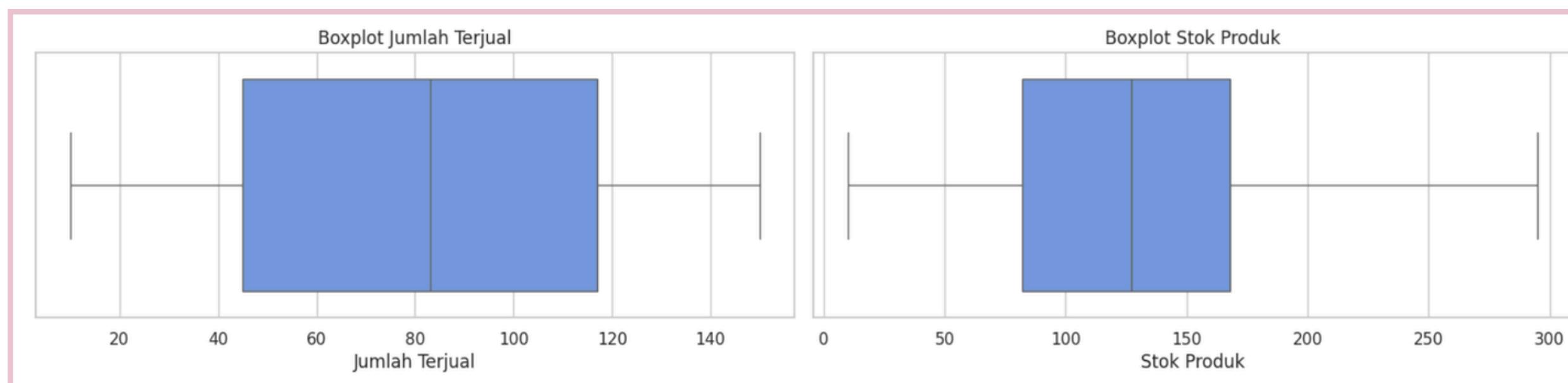
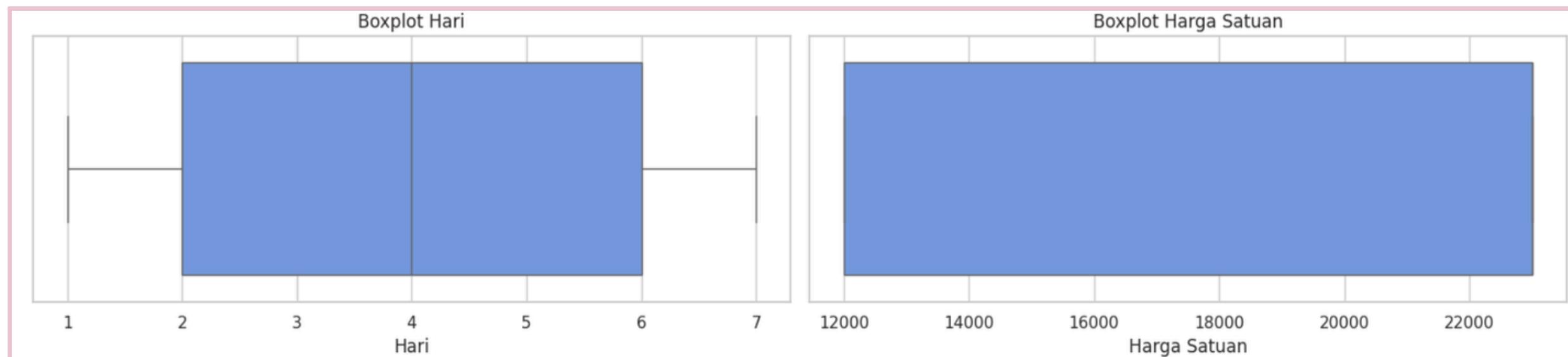


Visualisasi menunjukkan bahwa data tersebar merata sepanjang hari, penjualan produk umumnya berada pada rentang 40-140 unit, harga satuan didominasi dua titik yakni Rp12.000 dan Rp23.000, serta stok produk cenderung tinggi dengan distribusi condong ke kanan.

OUTLIER



1. Menampilkan boxplot



2. Output jumlah outlier

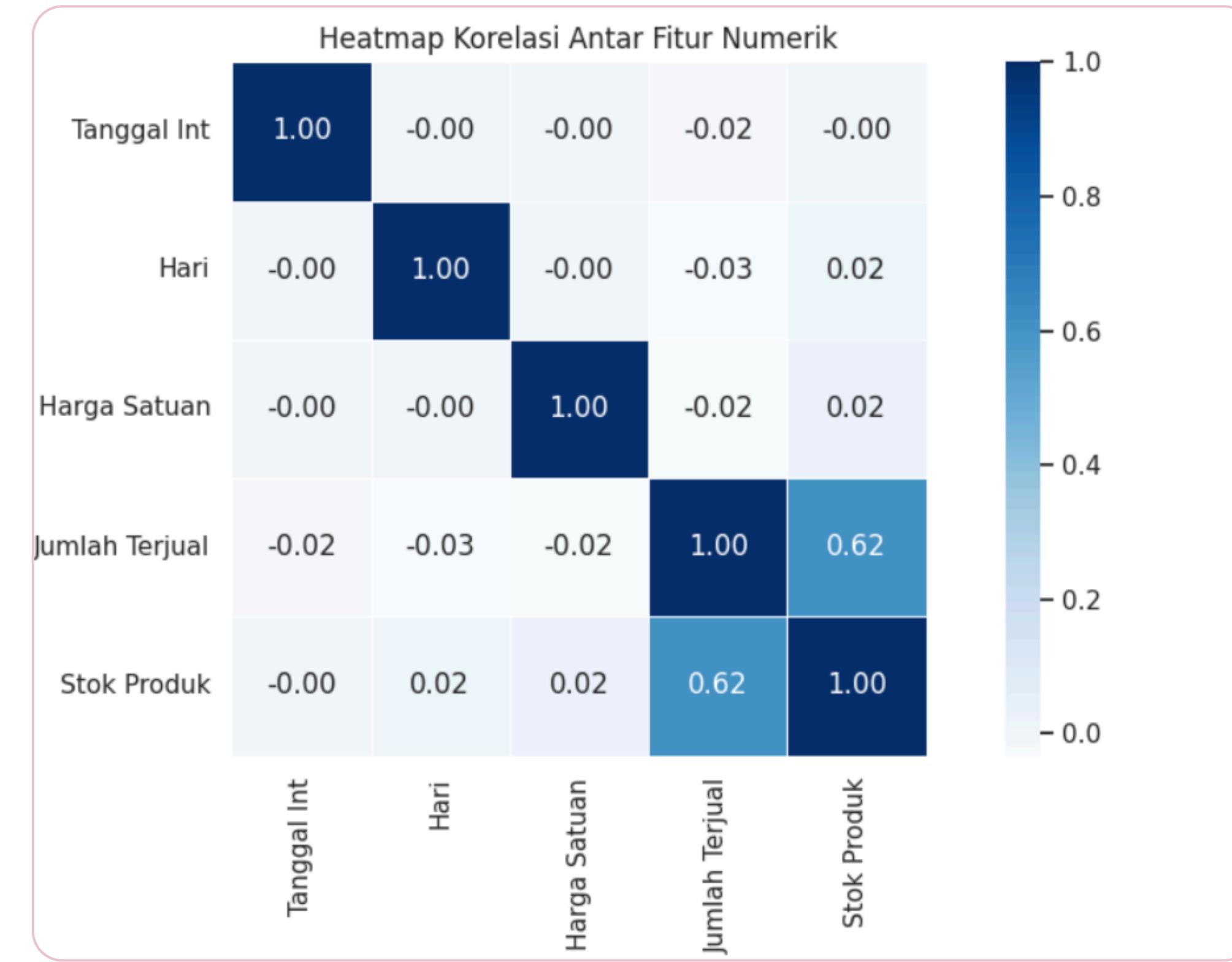
Kolom: Hari
Jumlah outlier: 0

Kolom: Harga Satuan
Jumlah outlier: 0

Kolom: Jumlah Terjual
Jumlah outlier: 0

Kolom: Stok Produk
Jumlah outlier: 0

CORELATION MATRIX



FEATURE ENGINEERING



- Membuat fitur baru: Status Stok

```
# Membuat kolom status stok berdasarkan selisih antara stok produk dan jumlah terjual
def get_status_stok(row):
    selisih = row['Stok Produk'] - row['Jumlah Terjual']
    if selisih <= 0:
        return 'Understock'
    elif selisih > 50:
        return 'Overstock'
    else:
        return 'Normal'

df['status stok'] = df.apply(get_status_stok, axis=1)

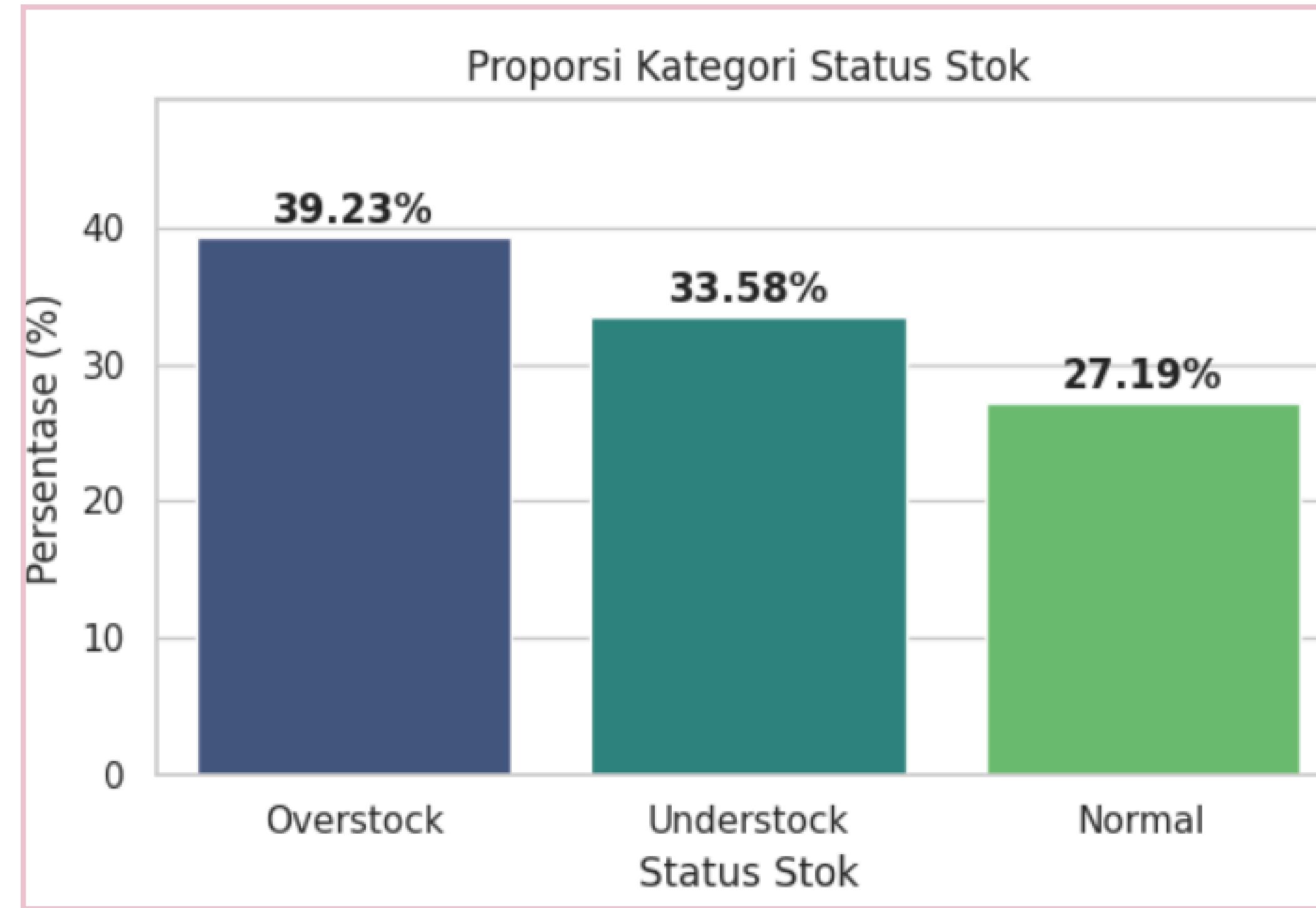
print(df[['Nama Produk', 'Jumlah Terjual', 'Stok Produk', 'status stok']].head())
```

	Nama Produk	Jumlah Terjual	Stok Produk	Status Stok
0	Bolen Coklat	38	130	Overstock
1	Bolen Banana	80	125	Normal
2	Bolen Proltape	45	89	Normal
3	Bolen Coklat Keju	149	150	Normal
4	Bolen Cokju (Mini)	118	118	Understock

FEATURE ENGINEERING



- Visualisasi Proporsi Status Stok



LABEL ENCODING



- Status Stok

```
# Mengubah fitur Status Stok menjadi nilai numerik menggunakan LabelEncoder  
le_status = LabelEncoder()  
df['Status Stok Encoded'] = le_status.fit_transform(df['Status Stok'])  
print(df[['Status Stok', 'Status Stok Encoded']].drop_duplicates())
```

	Status Stok	Status Stok Encoded
0	Overstock	1
1	Normal	0
4	Understock	2

- Nama Produk

```
# Mengubah fitur Nama Produk menjadi nilai numerik menggunakan LabelEncoder  
le = LabelEncoder()  
df['Nama Produk Encoded'] = le.fit_transform(df['Nama Produk'])  
print(df[['Nama Produk', 'Nama Produk Encoded']].drop_duplicates())
```

	Nama Produk	Nama Produk Encoded
0	Bolen Coklat	2
1	Bolen Banana	0
2	Bolen Proltape	6
3	Bolen Coklat Keju	3
4	Bolen Cokju (Mini)	1
5	Bolen Keju (Mini)	4
6	Bolen Pisang Coklat	5

DATA SPLIT

```
# Split data menjadi 80% data training dan 20% data testing
fitur_rfc = df[['Tanggal Int', 'Hari', 'Nama Produk Encoded', 'Harga Satuan', 'Stok Produk', 'Jumlah Terjual']]
target_rfc = df['Status Stok Encoded']

X_train_rfc, X_test_rfc, y_train_rfc, y_test_rfc = train_test_split(
    fitur_rfc,
    target_rfc,
    test_size=0.2,
    random_state=42,
    stratify=target_rfc
)
```

Data dibagi menjadi dua bagian: 80% untuk training dan 20% untuk testing. Proses ini dilakukan untuk memisahkan data yang akan digunakan untuk melatih model dan mengevaluasi performa model.

SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)

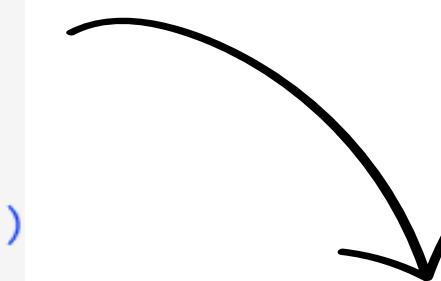


```
▶ from imblearn.over_sampling import SMOTE

# Menyeimbangkan data target dengan SMOTE dan cek distribusinya
smote = SMOTE(random_state=42)
x_train_smote, y_train_smote = smote.fit_resample(x_train_rfc, y_train_rfc)

print("Distribusi sebelum SMOTE:\n", pd.Series(y_train_rfc).value_counts())
print("\nDistribusi setelah SMOTE:\n", pd.Series(y_train_smote).value_counts())
```

Sebelum dilakukan SMOTE, distribusi kelas pada Status Stok Encoded tidak seimbang: kelas 1 (Overstock) mendominasi dengan 344 data, kelas 2 (Understock) 294 data, dan kelas 0 (Normal) hanya 238 data. Ketidakseimbangan ini dapat menyebabkan model bias terhadap kelas mayoritas. Setelah SMOTE dilakukan, jumlah data tiap kelas disamakan menjadi 344, sehingga dataset menjadi seimbang.



⬇ Distribusi sebelum SMOTE:
Status Stok Encoded
1 344
2 294
0 238
Name: count, dtype: int64

Distribusi setelah SMOTE:
Status Stok Encoded
1 344
2 344
0 344
Name: count, dtype: int64

DATA MODELING

```
# Melatih model dan melakukan prediksi pada data uji
model_rfc = RandomForestClassifier(random_state=42, class_weight='balanced')
model_rfc.fit(x_train_smote, y_train_smote)

y_pred_rfc = model_rfc.predict(x_test_rfc)
```

Data modeling dilakukan dengan Random Forest Classifier, menggunakan data hasil SMOTE dan class weight seimbang untuk mengatasi ketidakseimbangan kelas, lalu dilakukan prediksi pada data uji.

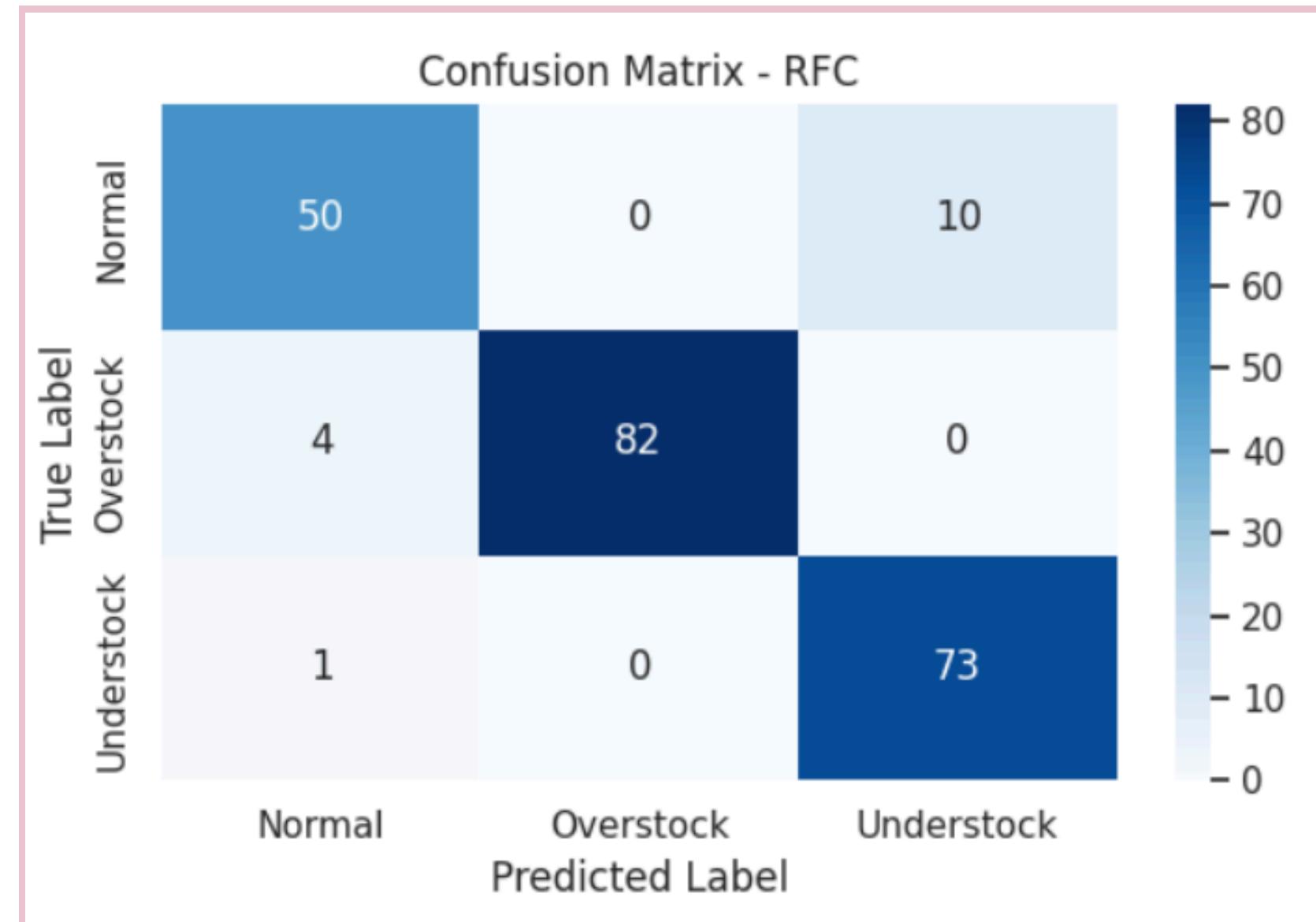
DATA EVALUATION

```
# Menampilkan metrik evaluasi model
accuracy = accuracy_score(y_test_rfc, y_pred_rfc)
precision = precision_score(y_test_rfc, y_pred_rfc, average='weighted')
recall = recall_score(y_test_rfc, y_pred_rfc, average='macro')
f1 = f1_score(y_test_rfc, y_pred_rfc, average='macro')

# Output hasil evaluasi
print("Accuracy :", round(accuracy, 2))
print("Precision:", round(precision, 2))
print("Recall   :", round(recall, 2))
print("F1 Score :", round(f1, 2))
```

```
Accuracy : 0.93
Precision: 0.93
Recall   : 0.92
F1 Score : 0.93
```

CONFUSION MATRIX



Model menunjukkan performa klasifikasi yang sangat baik, terlihat dari tingginya jumlah prediksi benar di diagonal confusion matrix:

- 50 data Normal, 82 Overstock, dan 73 Understock diklasifikasikan dengan benar.
- Kesalahan kecil terjadi: 10 Normal diprediksi sebagai Understock, 4 Overstock dan 1 Understock diprediksi sebagai Normal.

Secara keseluruhan, kesalahan prediksi tergolong sedikit dibanding total data.

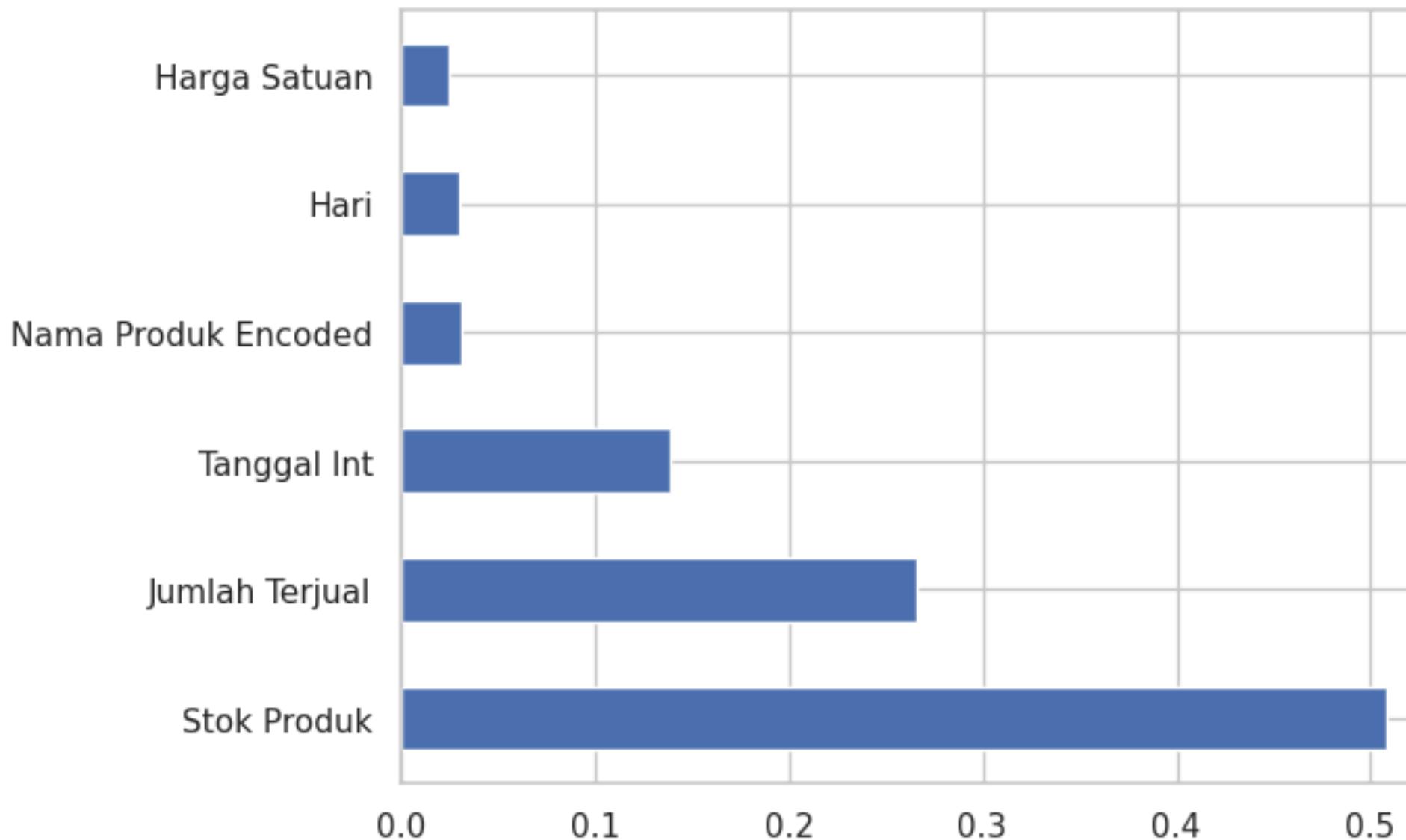
CLASIFICATION REPORT

```
# Menampilkan Classification Report dari model RFC  
print(classification_report(y_test_rfc, y_pred_rfc))
```

	precision	recall	f1-score	support
0	0.91	0.83	0.87	60
1	1.00	0.95	0.98	86
2	0.88	0.99	0.93	74
accuracy			0.93	220
macro avg	0.93	0.92	0.93	220
weighted avg	0.93	0.93	0.93	220

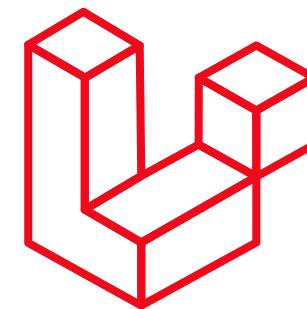
Classification report menunjukkan akurasi 93%, dengan nilai precision, recall, dan f1-score tinggi di semua kelas. Model bekerja konsisten dan seimbang.

FEATURE IMPORTANCE



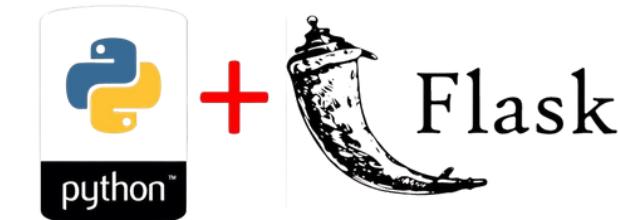
Berdasarkan grafik feature importance dari model Random Forest Classifier, fitur yang paling berpengaruh terhadap hasil prediksi adalah **Stok Produk** dan **Jumlah Terjual**, yang menunjukkan bahwa kondisi awal stok dan angka penjualan sangat menentukan status stok akhir (understock, normal, atau overstock).

TECH STACK



sweetalert2

REST API using Flask





KESIMPULAN

Penerapan algoritma Random Forest Classifier terbukti efektif dalam memprediksi status stok penjualan produk Bolen Crispy. Hasil evaluasi menunjukkan akurasi model mencapai 93%, dengan nilai precision, recall, dan f1-score yang tinggi dan seimbang di semua kelas stok (Normal, Overstock, Understock). Melalui teknik SMOTE dan feature engineering, kualitas data ditingkatkan sehingga membantu model dalam memberikan prediksi yang lebih akurat. Sistem ini dapat diintegrasikan ke dalam pengambilan keputusan manajemen persediaan agar lebih efisien, akurat, dan berbasis data.



Three



GPS Map Camera



Cheese!



THANK YOU!

DO YOU HAVE ANY QUESTION?

