IMPORT DATA

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.over_sampling import SMOTE
from collections import Counter
```

```
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
file_path = '/content/drive/MyDrive/Skripsi/Amikom.csv'
df = pd.read_csv(file_path)
```

EDA

```
print("Data awal")
display(df.head())
```

Data awal

	THA	Prodi	Hashed NPM	Ipk	SksTotal	IPS1	IPS2	IPS3	IPS4	IPS5	IPS6	IPS7	JenisKelamin	UsiaMasuk	PredikatKelulusa
0	2018	S1-Informatika	1836506135	3.72	144	3.75	3.50	3.42	3.75	3.92	4.00	0.0	L	20	Cum Lauc
1	2018	S1-Informatika	1836506132	3.49	144	3.42	3.25	3.50	3.75	3.33	2.67	0.0	L	19	Sangat Memuaska
2	2018	S1-Informatika	1836506133	3.97	144	3.92	4.00	3.92	4.00	4.00	3.00	4.0	L	19	Cum Lauc
3	2018	S1-Informatika	1836506131	3.57	144	3.25	3.42	3.42	3.83	3.50	3.50	0.0	L	19	Sangat Memuaska
4	2018	S1-Informatika	1836506122	3.82	144	3.83	3.83	4.00	3.67	3.92	3.56	0.0	P	22	Cum Lauc

```
df
```

	THA	Prodi	Hashed NPM	Ipk	SksTotal	IPS1	IPS2	IPS3	IPS4	IPS5	IPS6	IPS7	JenisKelamin	UsiaMasuk	PredikatKelu
0	2018	S1-Informatika	1836506135	3.72	144	3.75	3.50	3.42	3.75	3.92	4.00	0.00	L	20	Cum
1	2018	S1-Informatika	1836506132	3.49	144	3.42	3.25	3.50	3.75	3.33	2.67	0.00	L	19	Sangat Memu
2	2018	S1-Informatika	1836506133	3.97	144	3.92	4.00	3.92	4.00	4.00	3.00	4.00	L	19	Cum
3	2018	S1-Informatika	1836506131	3.57	144	3.25	3.42	3.42	3.83	3.50	3.50	0.00	L	19	Sangat Memu
4	2018	S1-Informatika	1836506122	3.82	144	3.83	3.83	4.00	3.67	3.92	3.56	0.00	P	22	Cum
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4528	2020	S1-Ilmu Komunikasi	1703339747	3.84	146	4.00	3.82	3.75	3.67	3.92	3.80	1.60	P	18	Cum
4529	2020	S1-Ilmu Komunikasi	1703339745	3.56	146	3.36	3.55	3.42	3.17	3.58	3.92	3.75	L	19	Cum
4530	2020	S1-Ilmu Komunikasi	1703339758	3.64	146	3.73	3.55	3.50	3.33	3.83	3.80	4.00	P	19	Cum
4531	2020	S1-Ilmu Komunikasi	1703339763	3.58	146	3.91	3.45	3.50	3.17	3.75	3.60	1.60	P	20	Cum
4532	2020	S1-Ilmu Komunikasi	1703339761	3.44	146	3.36	3.64	3.50	3.50	3.33	3.30	4.00	L	20	Sangat Memu

4533 rows × 16 columns

```
print("\nInformasi Data:")
df.info()
```

```
Informasi Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4533 entries, 0 to 4532
Data columns (total 16 columns):
```

```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   THA                 4533 non-null   int64
 1   Prodi               4533 non-null   object
 2   Hashed NPM          4533 non-null   int64
 3   Ipk                 4533 non-null   float64
 4   SksTotal            4533 non-null   int64
 5   IPS1                4533 non-null   float64
 6   IPS2                4533 non-null   float64
 7   IPS3                4533 non-null   float64
 8   IPS4                4533 non-null   float64
 9   IPS5                4533 non-null   float64
 10  IPS6                4533 non-null   float64
 11  IPS7                4533 non-null   float64
 12  JenisKelamin        4533 non-null   object
 13  UsiaMasuk           4533 non-null   int64
 14  PredikatKelulusan   4533 non-null   object
 15  Rerata Kehadiran    4533 non-null   int64
dtypes: float64(8), int64(5), object(3)
memory usage: 566.8+ KB
```

```
print("\nInformasi Data:")
display(df.describe(include='all'))
```

Informasi Data:

	THA	Prodi	Hashed NPM	Ipk	SksTotal	IPS1	IPS2	IPS3	IPS4	IP
count	4533.000000	4533	4.533000e+03	4533.000000	4533.000000	4533.000000	4533.000000	4533.000000	4533.000000	4533.000
unique	NaN	13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
top	NaN	S1-Informatika	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
freq	NaN	1189	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
mean	2018.822413	NaN	1.752274e+09	3.522813	144.680124	3.512226	3.454763	3.423446	3.459133	3.444
std	0.773577	NaN	7.081891e+07	0.299745	1.859601	0.376609	0.506525	0.551509	0.534763	0.537
min	2018.000000	NaN	1.648805e+09	2.140000	144.000000	0.500000	0.000000	0.000000	0.000000	0.000
25%	2018.000000	NaN	1.699137e+09	3.360000	144.000000	3.330000	3.330000	3.250000	3.330000	3.250
50%	2019.000000	NaN	1.703337e+09	3.580000	144.000000	3.580000	3.580000	3.580000	3.580000	3.580
75%	2019.000000	NaN	1.836506e+09	3.740000	144.000000	3.750000	3.750000	3.750000	3.750000	3.830
max	2020.000000	NaN	1.836514e+09	4.000000	162.000000	4.000000	4.000000	4.000000	4.000000	4.000

```
print("\nMissing Values (jumlah):")
print(df.isnull().sum())
```

```
Missing Values (jumlah):
THA              0
Prodi            0
Hashed NPM       0
Ipk              0
SksTotal         0
IPS1             0
IPS2             0
IPS3             0
IPS4             0
IPS5             0
IPS6             0
IPS7             0
```

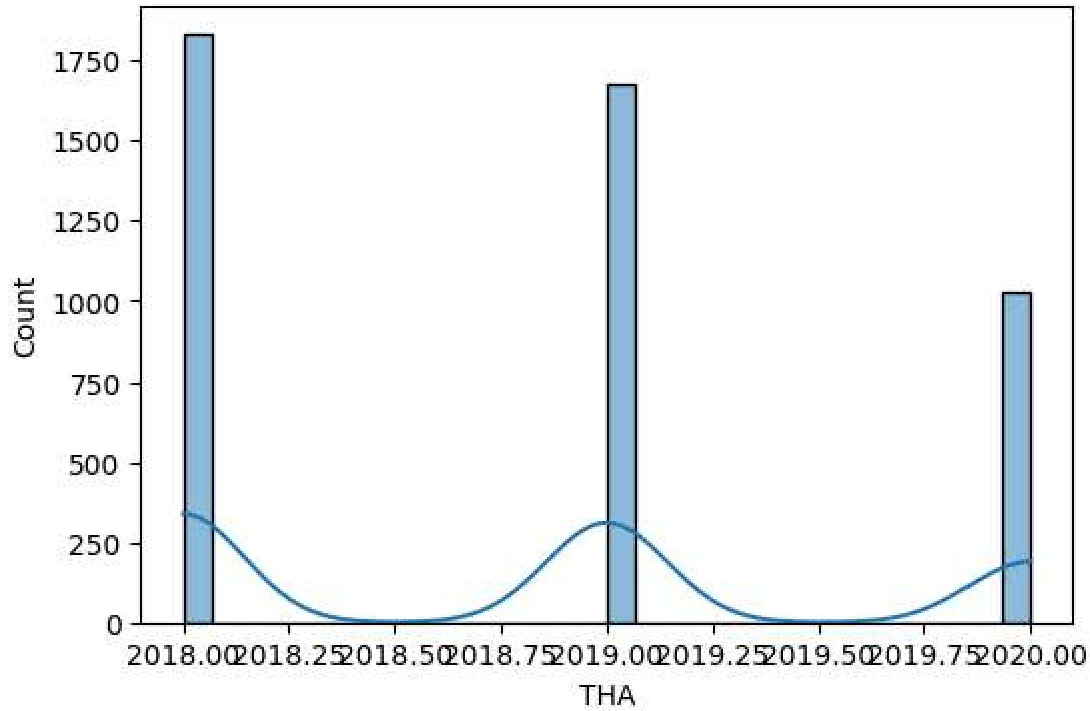
```
JenisKelamin         0
UsiaMasuk            0
PredikatKelulusan    0
Rerata Kehadiran     0
dtype: int64
```

```
print("\nJumlah Data Duplikat:")
print(df.duplicated().sum())
```

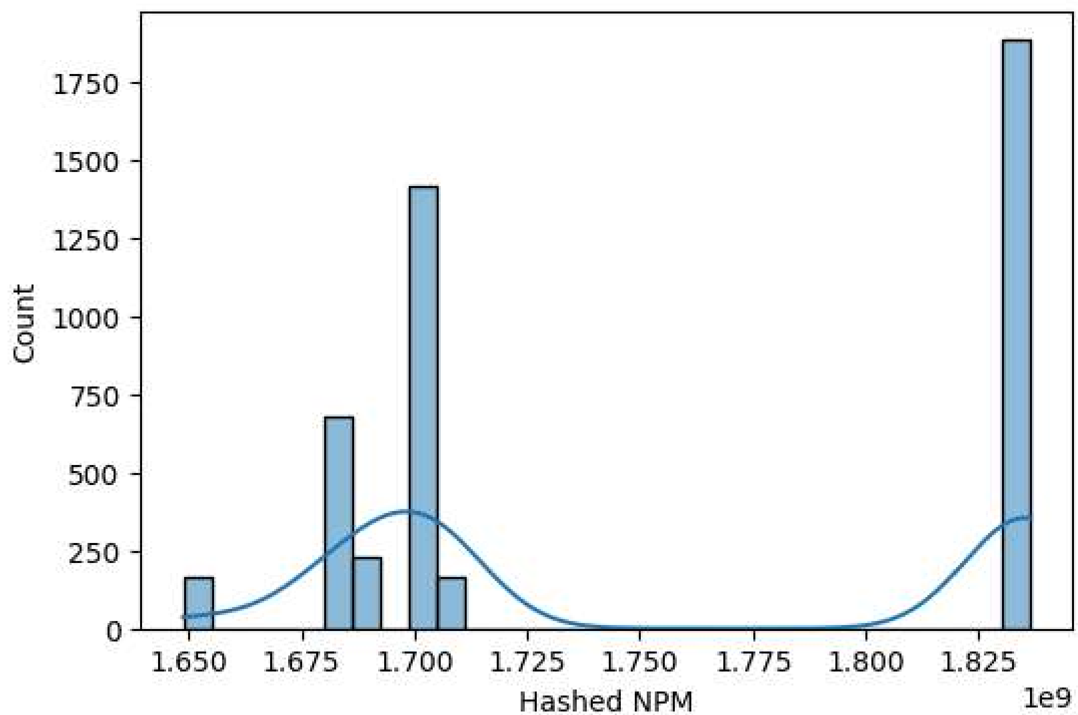
```
Jumlah Data Duplikat:
0
```

```
numerik = df.select_dtypes(include=np.number).columns
for col in numerik:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col].dropna(), kde=True, bins=30)
    plt.title(f'Distribusi Fitur: {col}')
    plt.show()
```

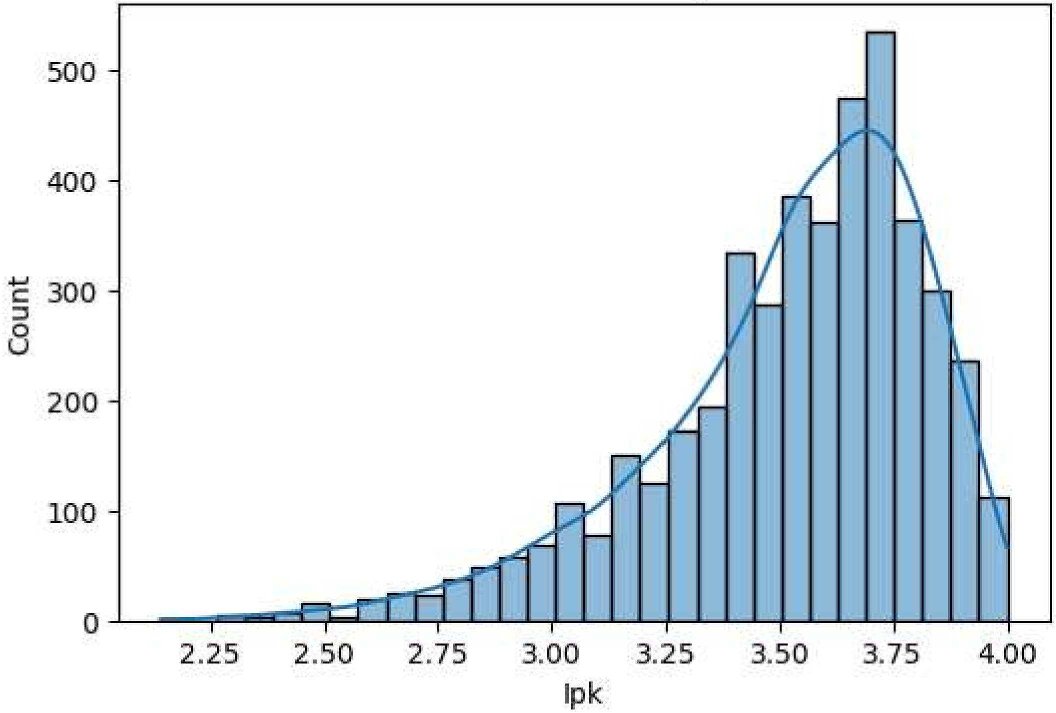
Distribusi Fitur: THA



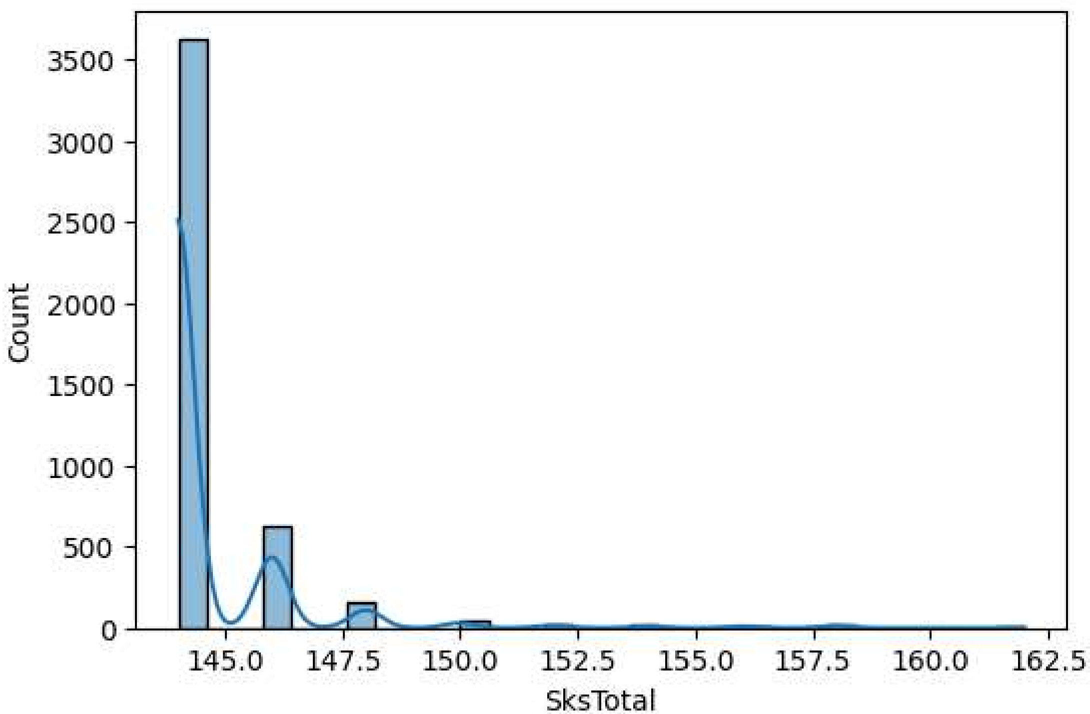
Distribusi Fitur: Hashed NPM



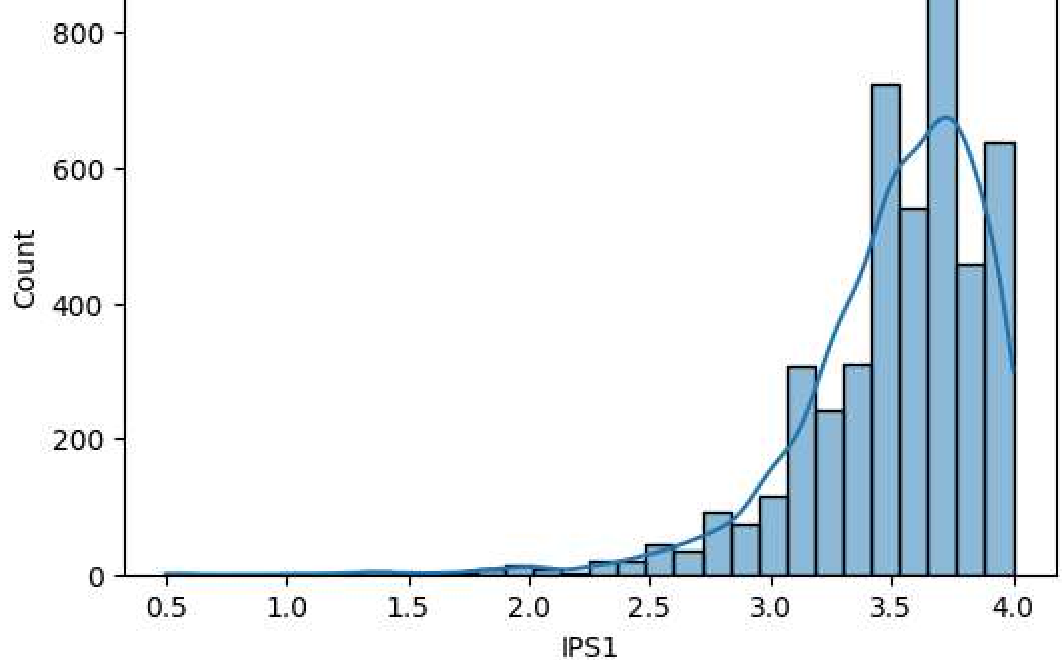
Distribusi Fitur: Ipk



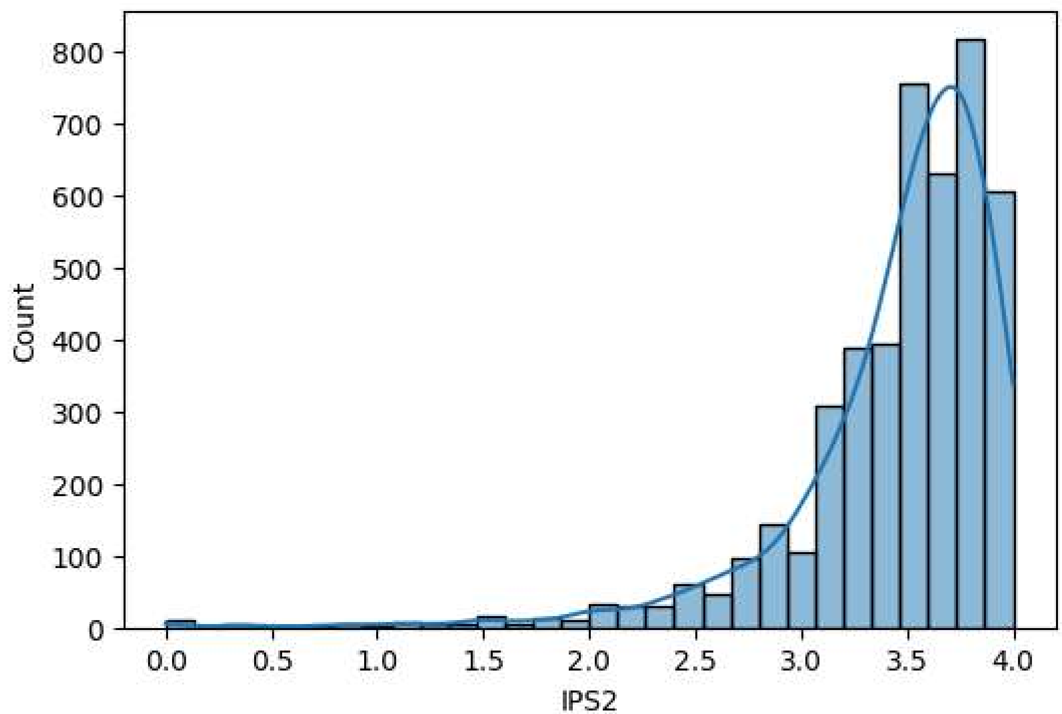
Distribusi Fitur: SksTotal



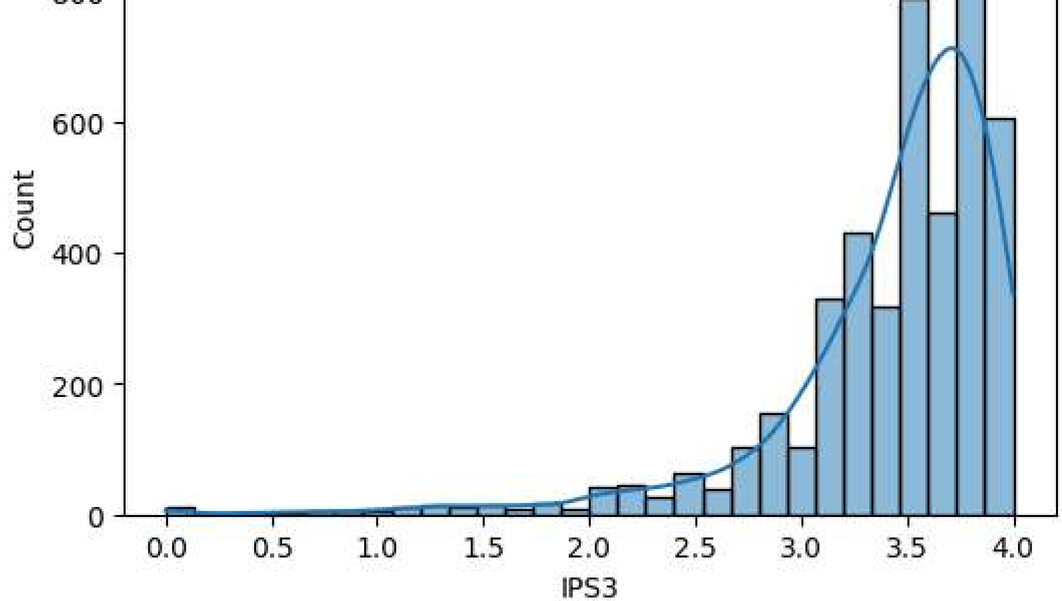
Distribusi Fitur: IPS1



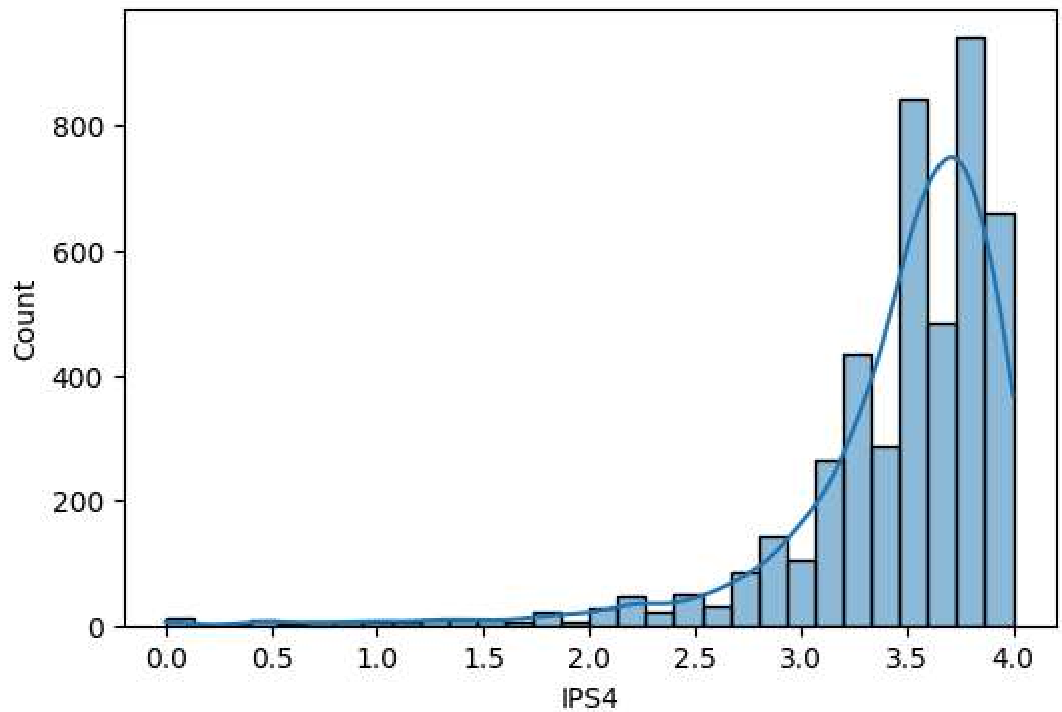
Distribusi Fitur: IPS2



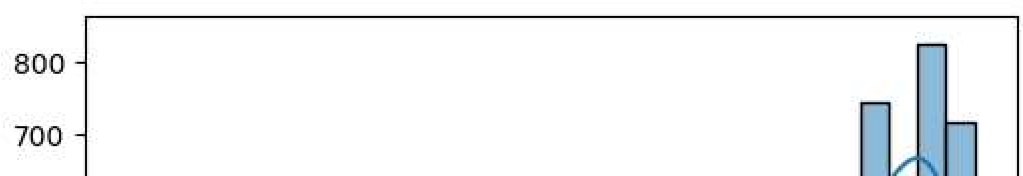
Distribusi Fitur: IPS3

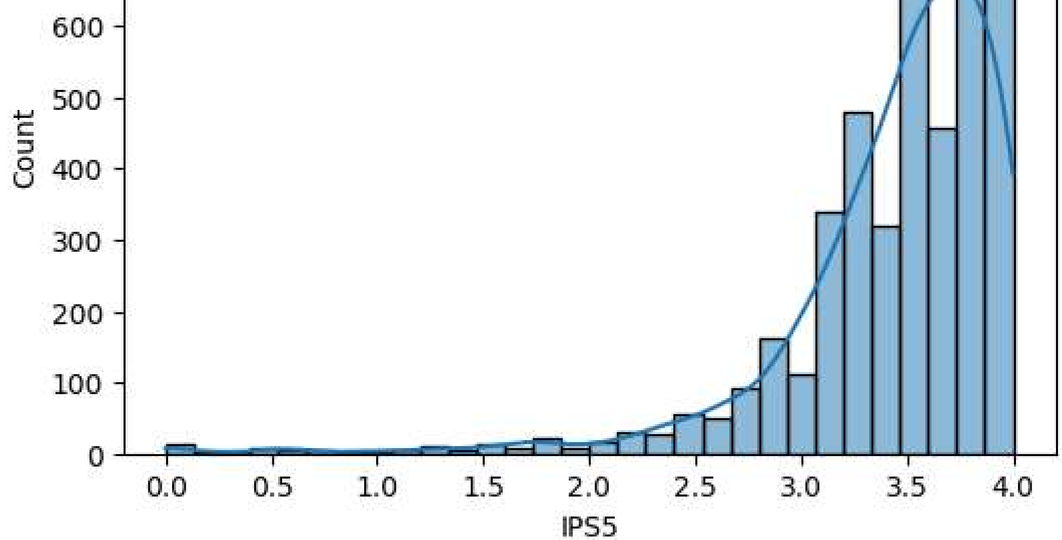


Distribusi Fitur: IPS4

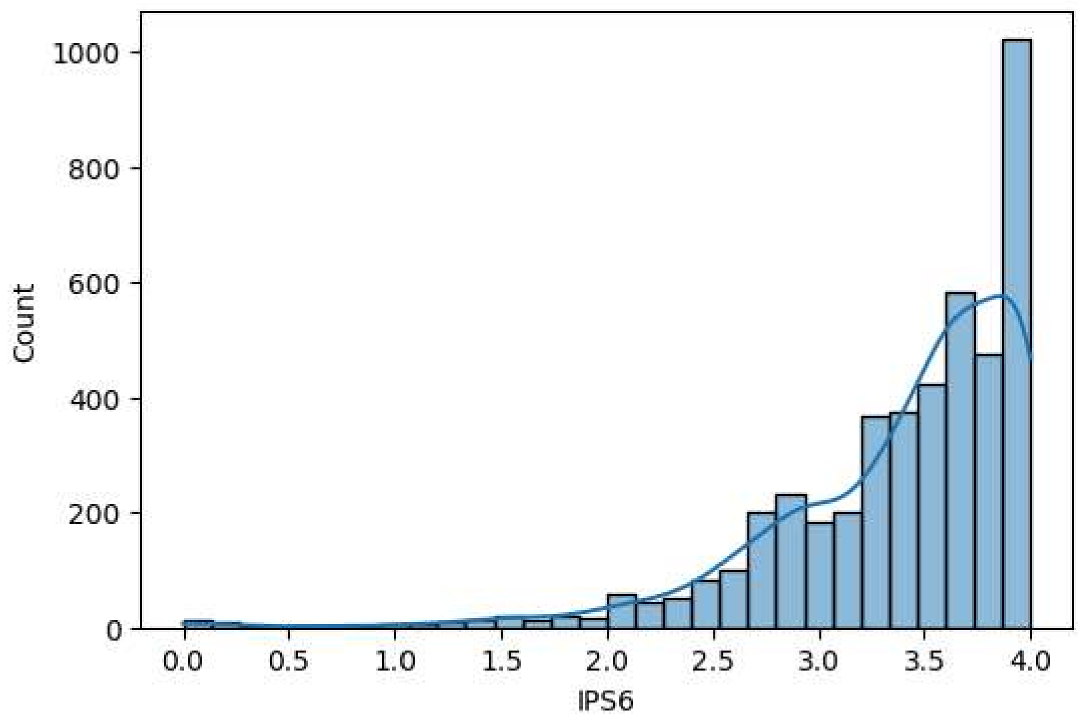


Distribusi Fitur: IPS5

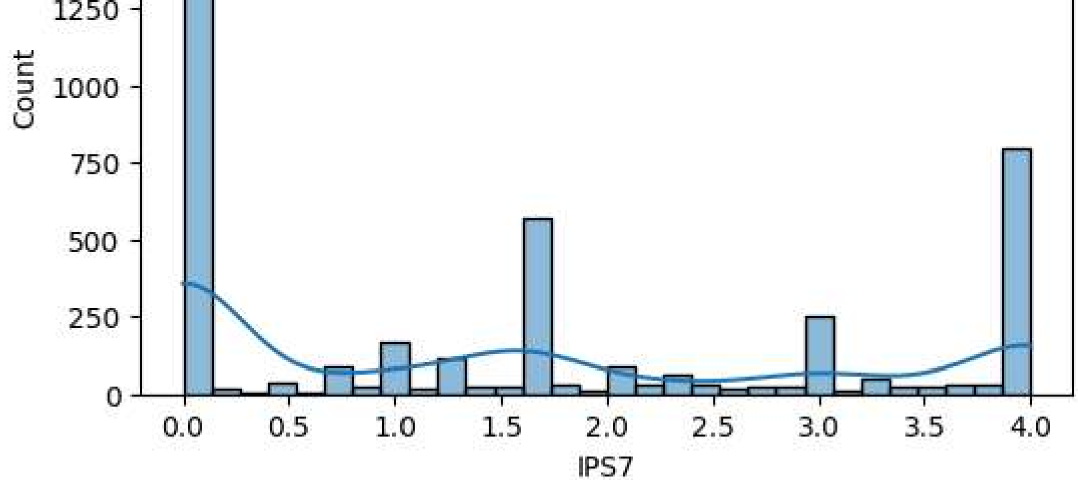




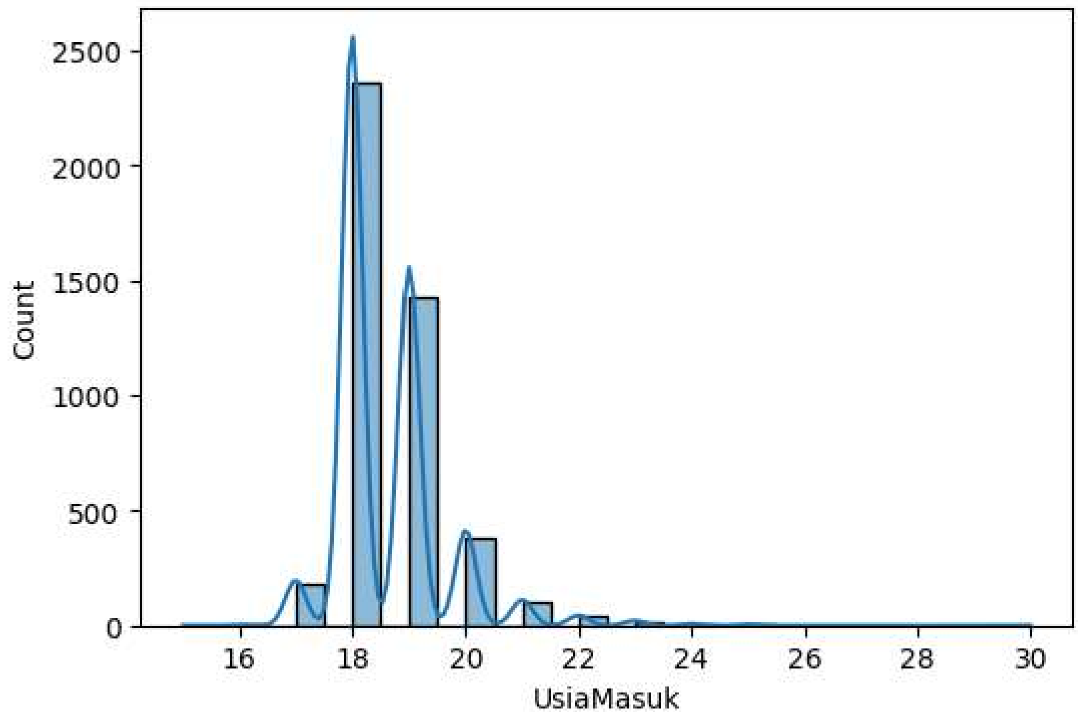
Distribusi Fitur: IPS6



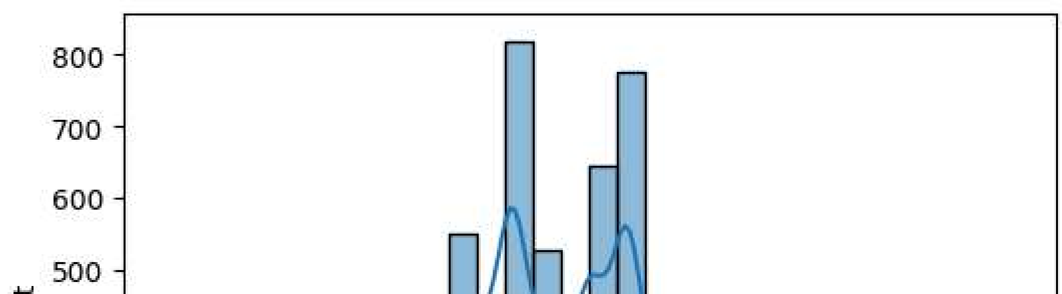
Distribusi Fitur: IPS7

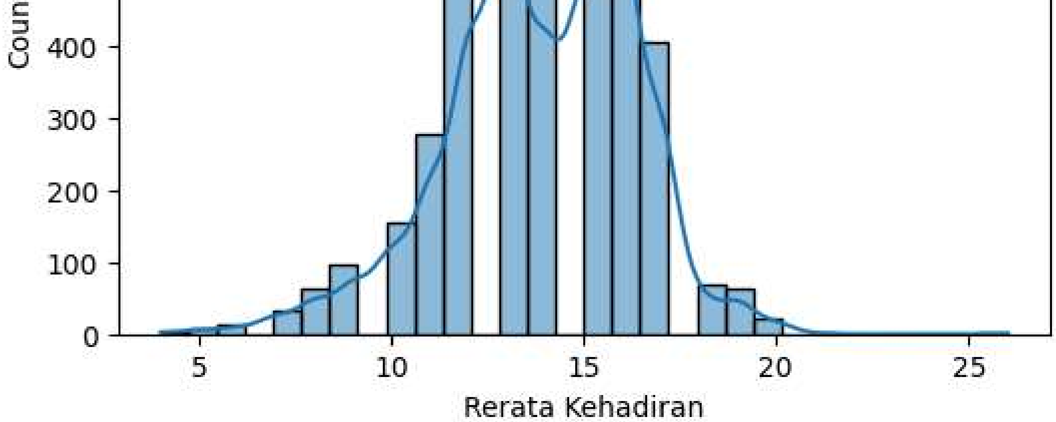


Distribusi Fitur: UsiaMasuk



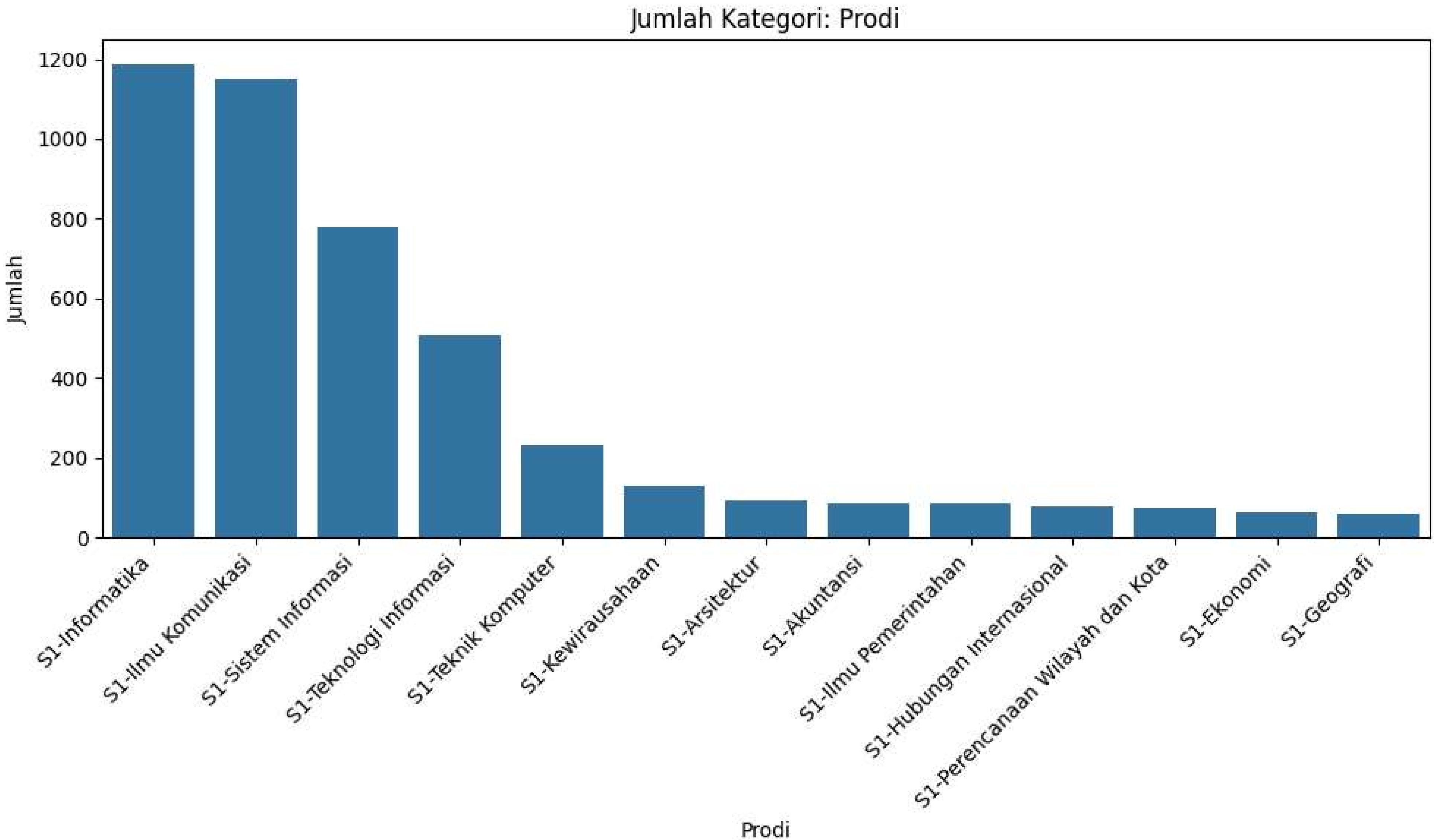
Distribusi Fitur: Rerata Kehadiran

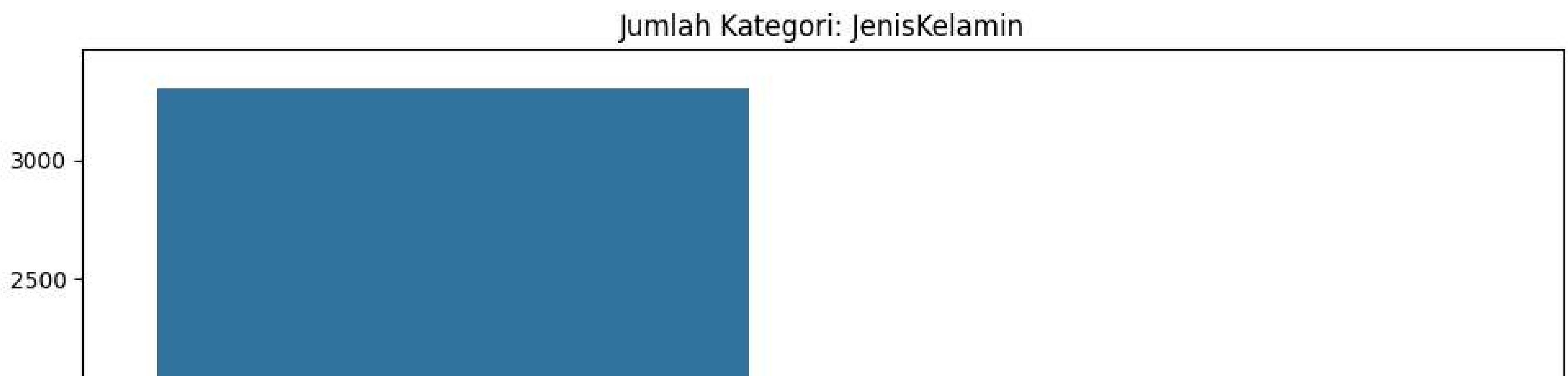


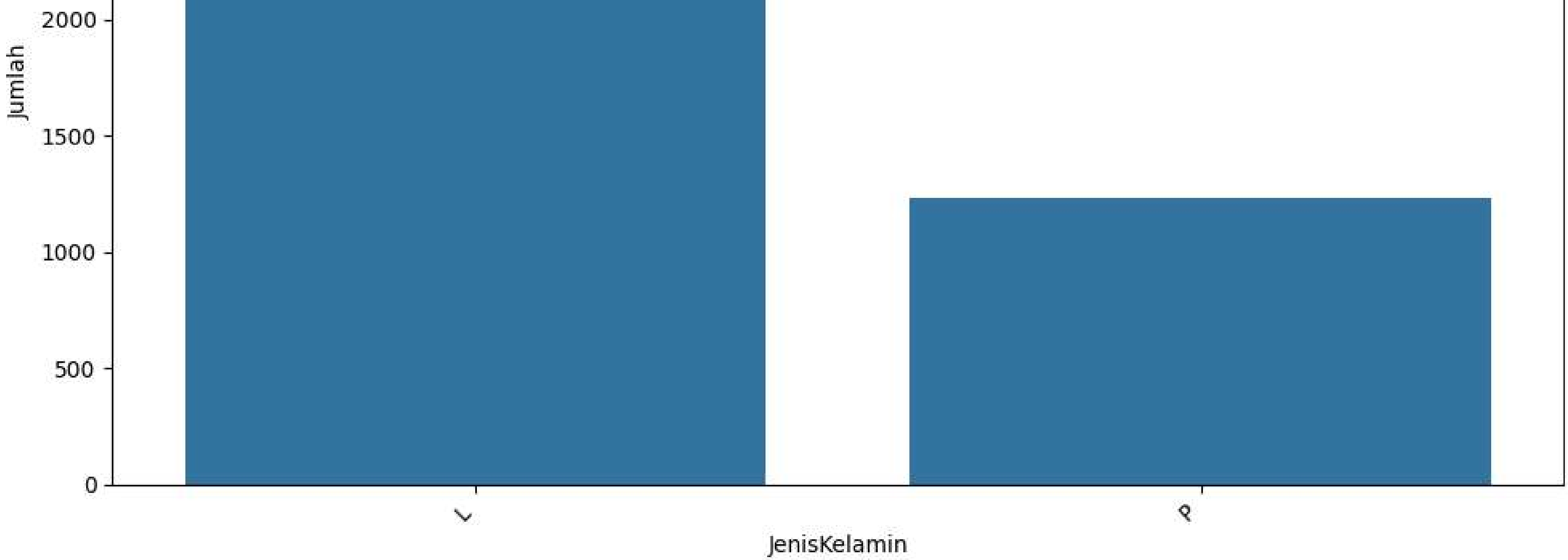


Coun
400
300
200
100
0
5
10
15
20
25
Rerata Kehadiran

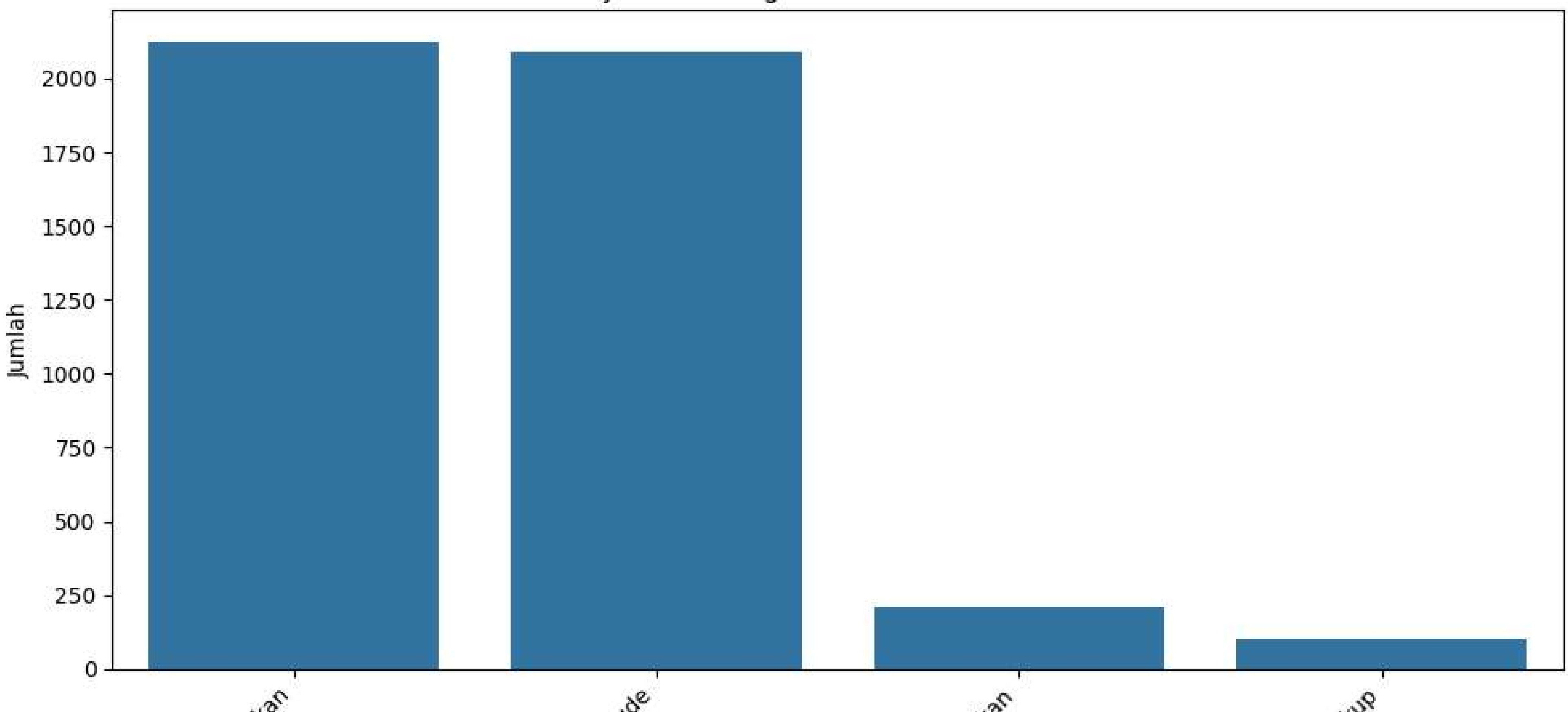
```
kategori = df.select_dtypes(include='object').columns

for col in kategori:
    plt.figure(figsize=(10, 6))
    ax = sns.countplot(data=df, x=col, order=df[col].value_counts().index)
    plt.title(f'Jumlah Kategori: {col}')
    plt.xlabel(col)
    plt.ylabel('Jumlah')


    plt.xticks(rotation=45, ha='right')


    plt.tight_layout()
    plt.show()
```


Jumlah Kategori: Prodi
Jumlah
0
200
400
600
800
1000
1200
S1-Informatika
S1-Ilmu Komunikasi
S1-Sistem Informasi
S1-Teknologi Informasi
S1-Teknik Komputer
S1-Kewirausahaan
S1-Arsitektur
S1-Akuntansi
S1-Ilmu Pemerintahan
S1-Hubungan Internasional
S1-Perencanaan Wilayah dan Kota
S1-Ekonomi
S1-Geografi
Prodi

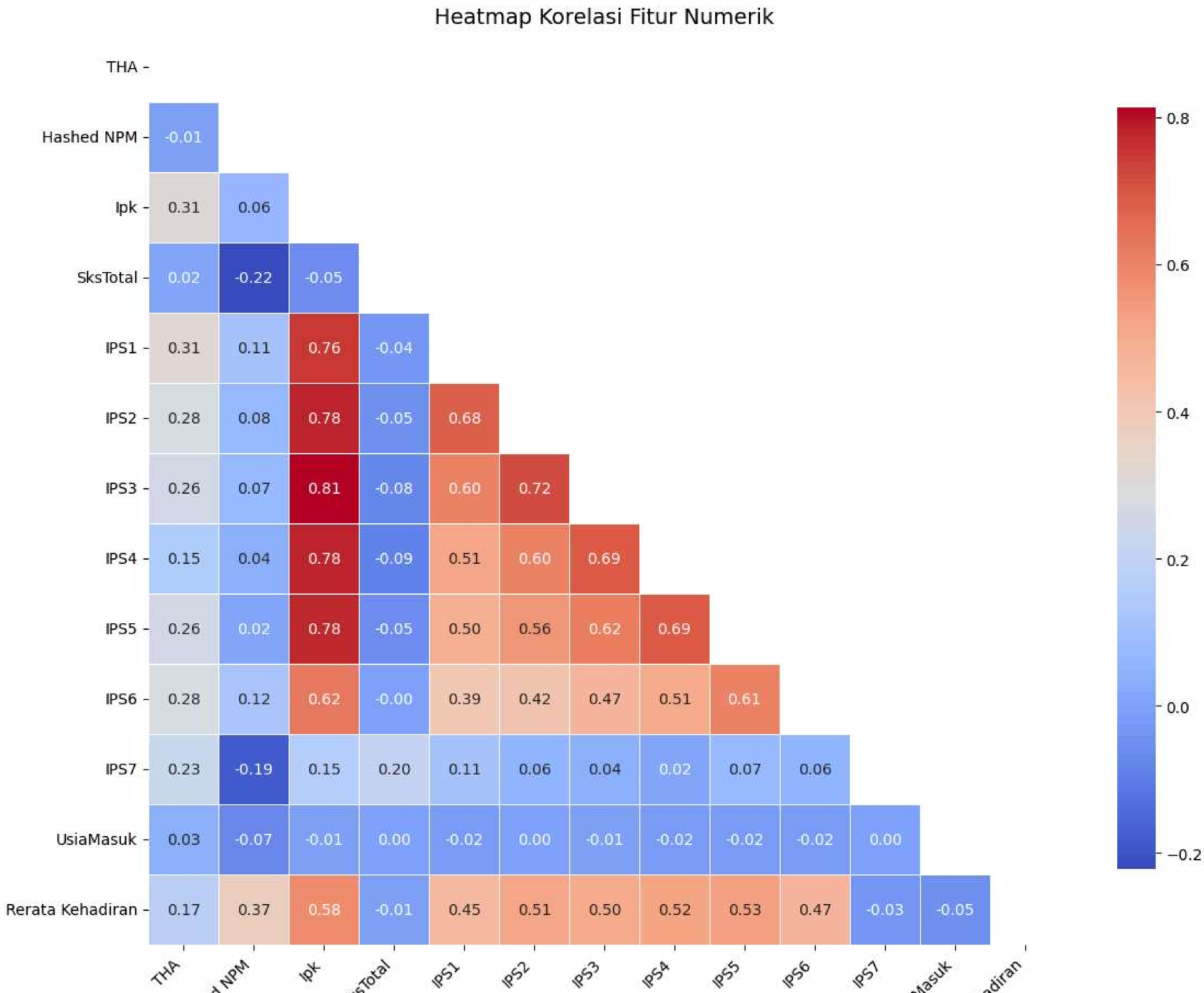

Jumlah Kategori: JenisKelamin
2500
3000



Jumlah Kategori: PredikatKelulusan



```
corr_matrix = df.corr(numeric_only=True)


mask = np.triu(np.ones_like(corr_matrix, dtype=bool))


# Plot heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix,
            annot=True,
            fmt=".2f",              # Format angka dua digit desimal
            cmap='coolwarm',
            mask=mask,              # Masking segitiga atas
            square=True,            # Biar kotak
            linewidths=.5,          # Garis antar kotak
            cbar_kws={"shrink": .8})  # Ukuran color bar

plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.title('Heatmap Korelasi Fitur Numerik', fontsize=14)
plt.tight_layout()
plt.show()
```



Heatmap Korelasi Fitur Numerik
THA
Hashed NPM -0.01
Ipk 0.31 0.06
SksTotal 0.02 -0.22 -0.05
IPS1 0.31 0.11 0.76 -0.04
IPS2 0.28 0.08 0.78 -0.05 0.68
IPS3 0.26 0.07 0.81 -0.08 0.60 0.72
IPS4 0.15 0.04 0.78 -0.09 0.51 0.60 0.69
IPS5 0.26 0.02 0.78 -0.05 0.50 0.56 0.62 0.69
IPS6 0.28 0.12 0.62 -0.00 0.39 0.42 0.47 0.51 0.61
IPS7 0.23 -0.19 0.15 0.20 0.11 0.06 0.04 0.02 0.07 0.06
UsiaMasuk 0.03 -0.07 -0.01 0.00 -0.02 0.00 -0.01 -0.02 -0.02 -0.02 0.00
Rerata Kehadiran 0.17 0.37 0.58 -0.01 0.45 0.51 0.50 0.52 0.53 0.47 -0.03 -0.05
THA
Ipk
IPS1
IPS2
IPS3
IPS4
IPS5
IPS6
IPS7
0.8
0.6
0.4
0.2
0.0
−0.2

Alasan menggunakan heatmap korelasi seperti diatas :

1. Hanya menampilkan segitiga bawah dari matriks korelasi, sehingga tidak menampilkan informasi yang berulang.(tanpa duplikasi)
2. Membuat tampilannya lebih ringkas dan fokus.
3. Nilai-nilai korelasi ditulis dengan dua desimal, jadi pengguna bisa tahu secara kuantitatif seberapa kuat hubungan antar fitur.
4. Label sumbu X dan Y ditampilkan secara jelas dan tidak bertabrakan. Sangat penting untuk identifikasi variabel yang saling berkorelasi.

Analisis dari heatmap diatas :

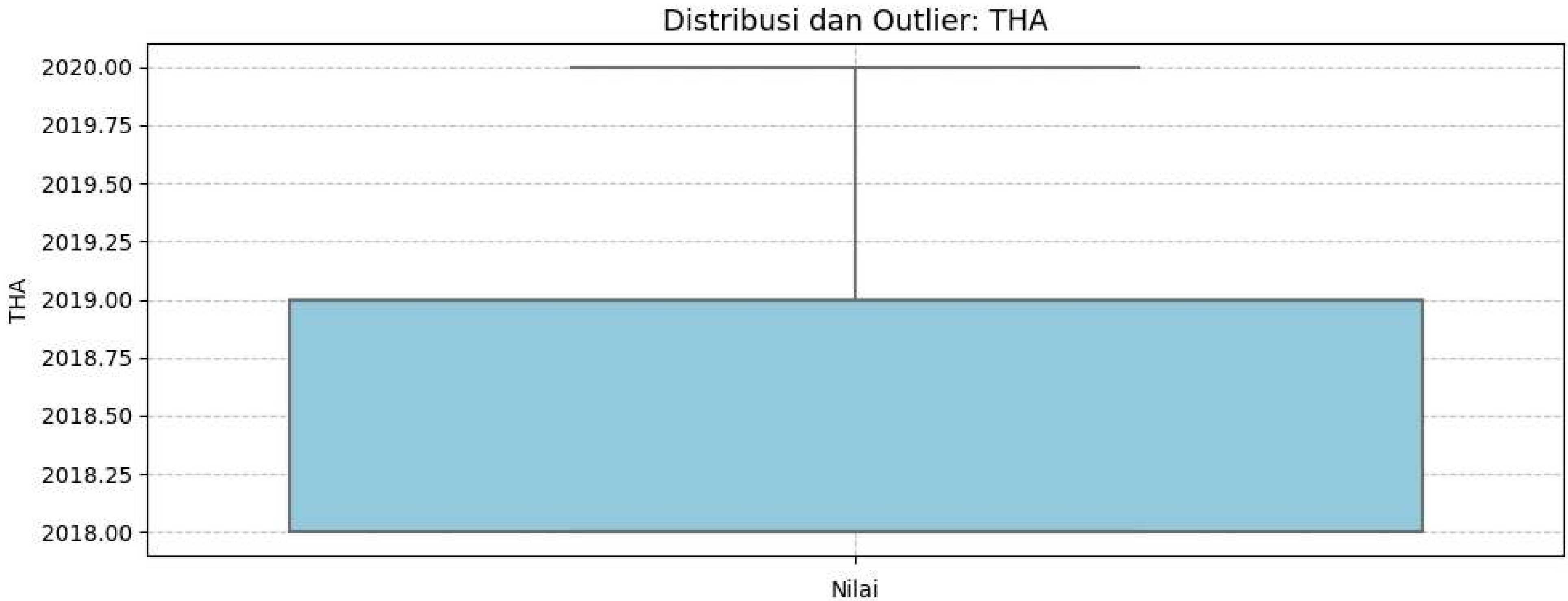
1. IPS1 hingga IPS5 sangat berkorelasi tinggi satu sama lain (nilai korelasi sekitar 0.76 – 0.81), ini logis karena performa akademik antar semester sering konsisten.

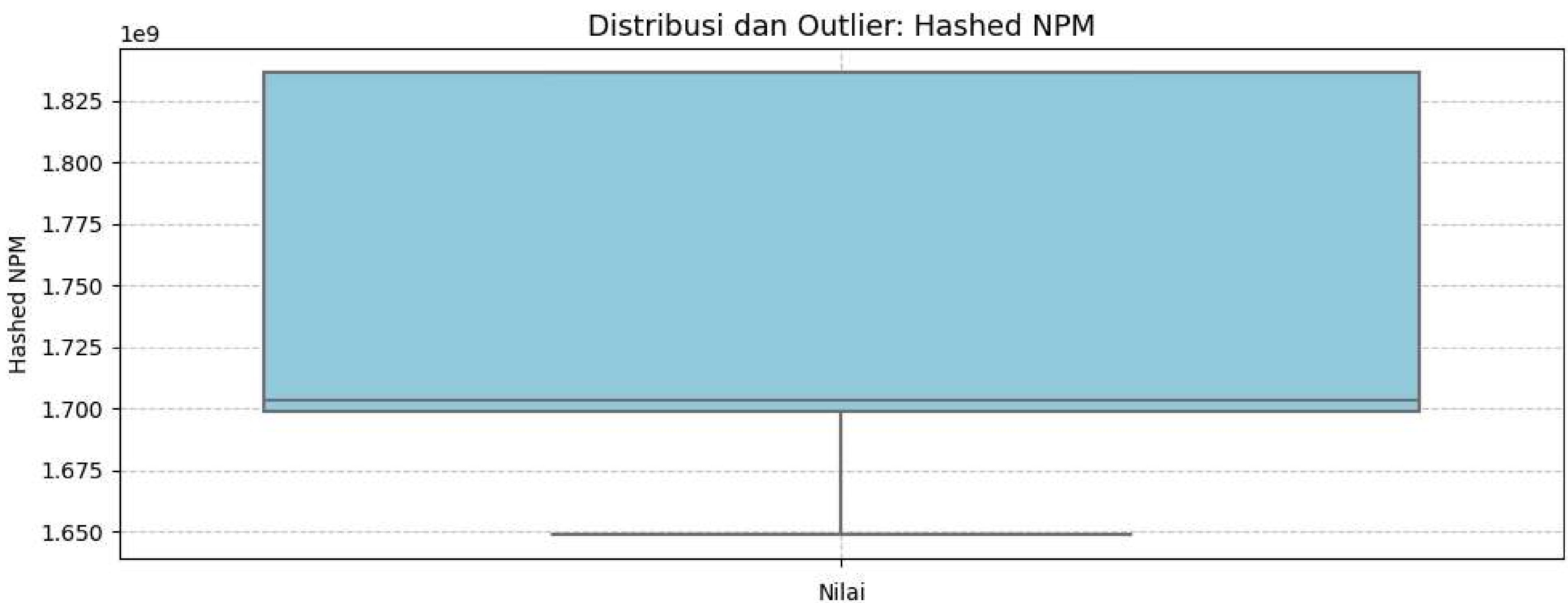
2. IPK juga memiliki korelasi tinggi dengan nilai IPS semester, terutama IPS3 (0.81) dan IPS2 (0.78) → artinya, performa di semester awal-mid punya pengaruh besar terhadap IPK.

3. Rerata Kehadiran punya korelasi sedang-tinggi dengan IPK (0.58) → ini menarik: mahasiswa yang rajin hadir cenderung punya IPK lebih tinggi.

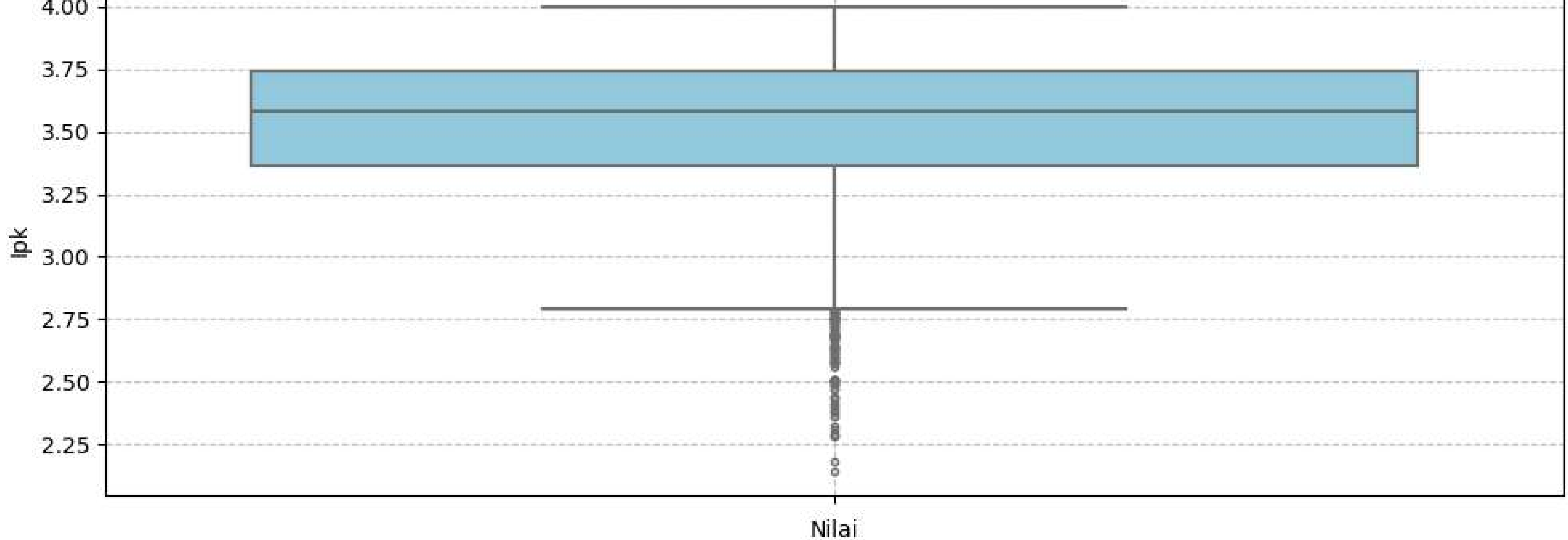
4. UsiaMasuk, THA, dan Hashed NPM tidak menunjukkan korelasi kuat dengan fitur lain (korelasi mendekati 0).

```
for col in numerik:
    plt.figure(figsize=(10, 4))  # Ukuran horizontal agar muat
    sns.boxplot(data=df, y=col, color='skyblue', fliersize=3, linewidth=1.5)
    plt.title(f'Distribusi dan Outlier: {col}', fontsize=13)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.xlabel('Nilai')
    plt.ylabel(col)
    plt.tight_layout()
    plt.show()
```

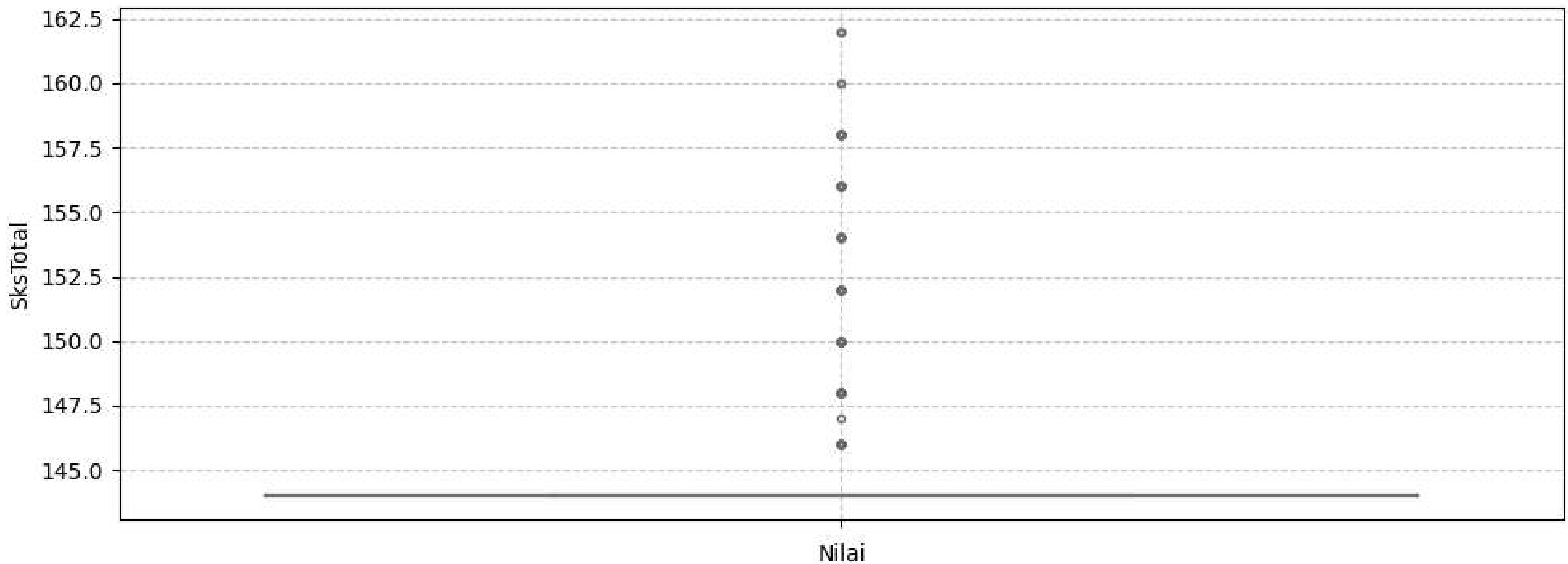




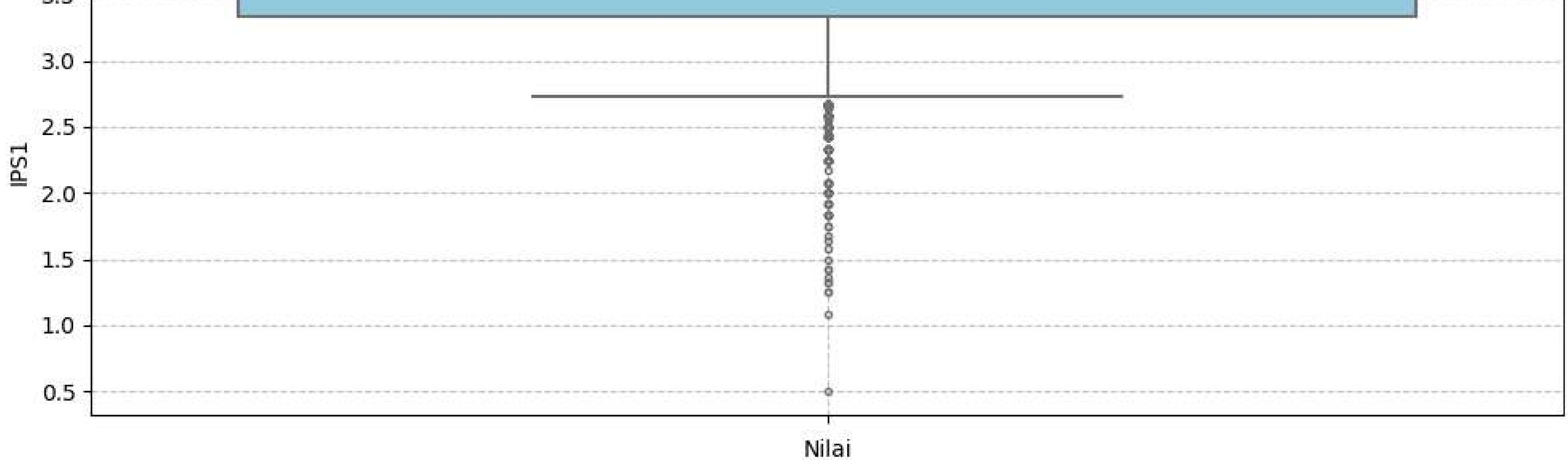
Distribusi dan Outlier: Ipk

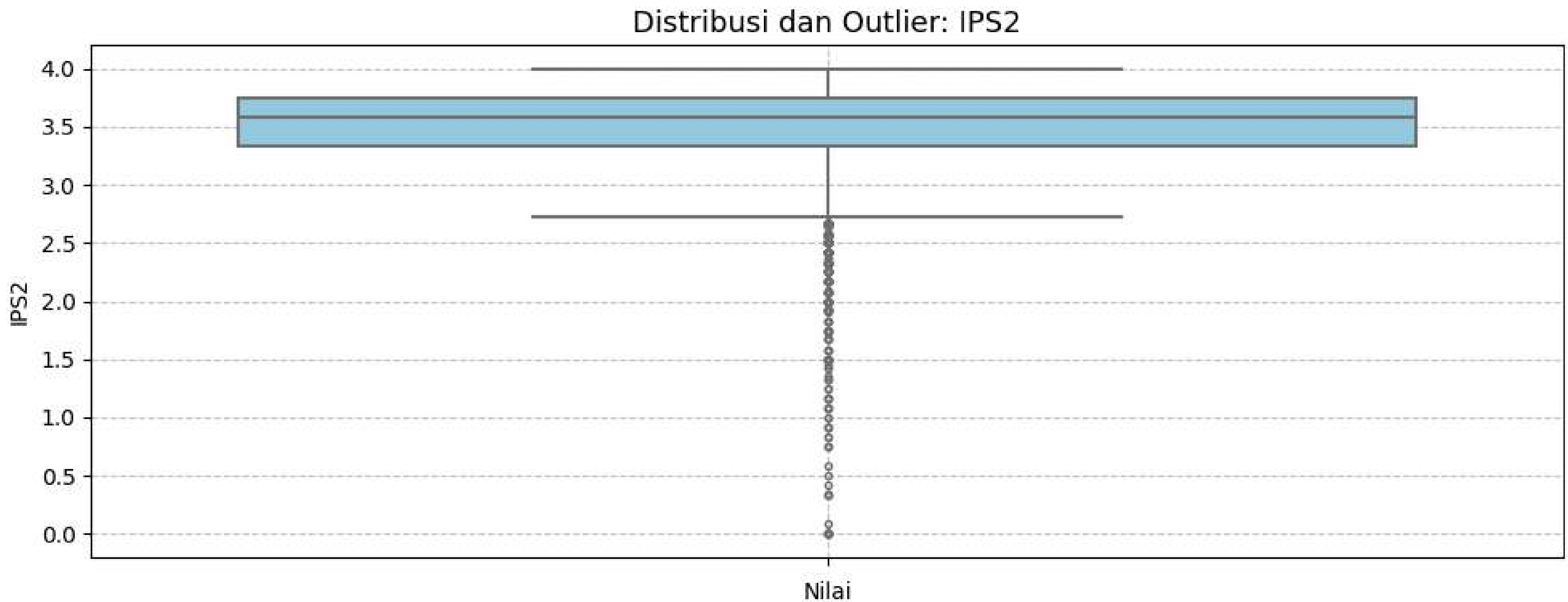


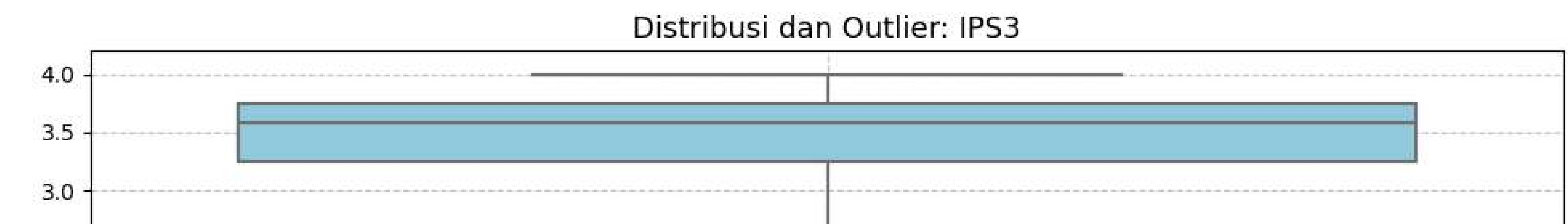
Distribusi dan Outlier: SksTotal

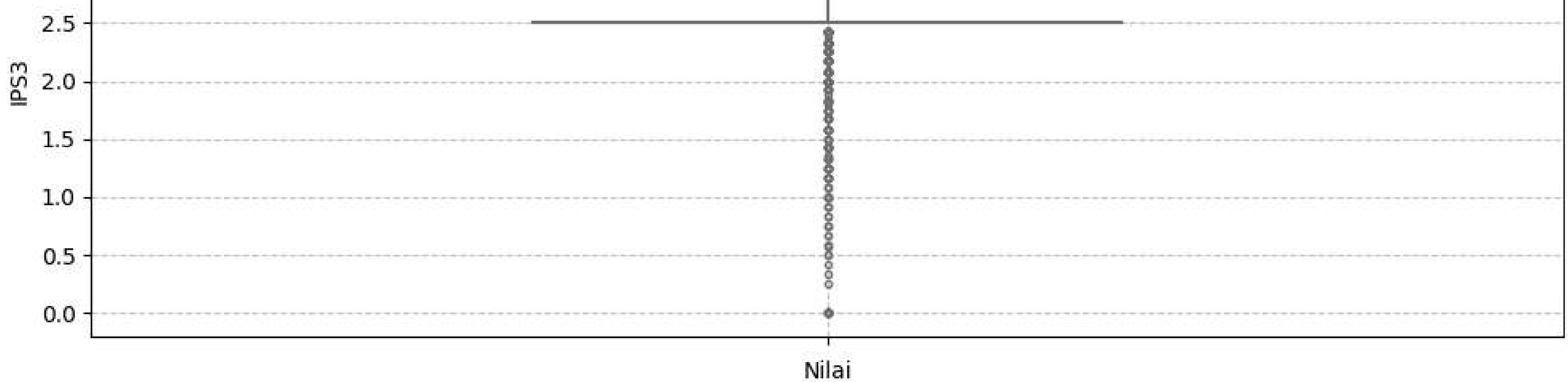


Distribusi dan Outlier: IPS1

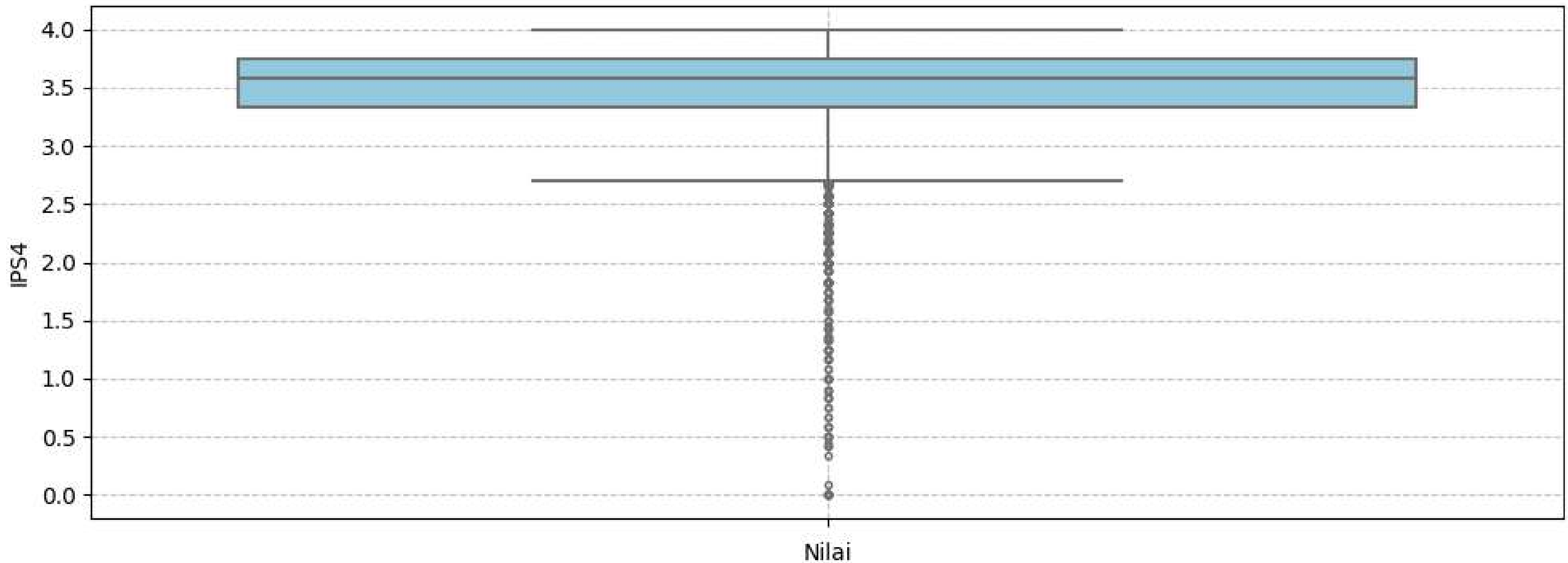

IPS1
3.0
2.5
2.0
1.5
1.0
0.5
Nilai


Distribusi dan Outlier: IPS2
IPS2
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Nilai

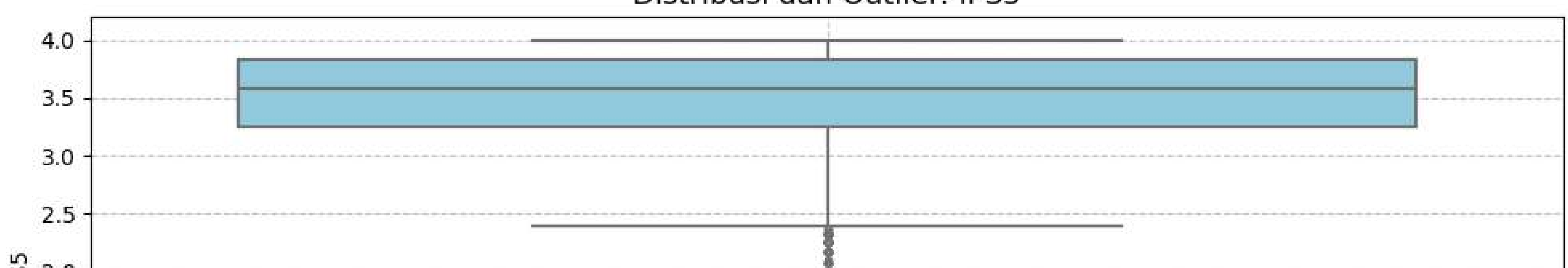

Distribusi dan Outlier: IPS3
4.0
3.5
3.0

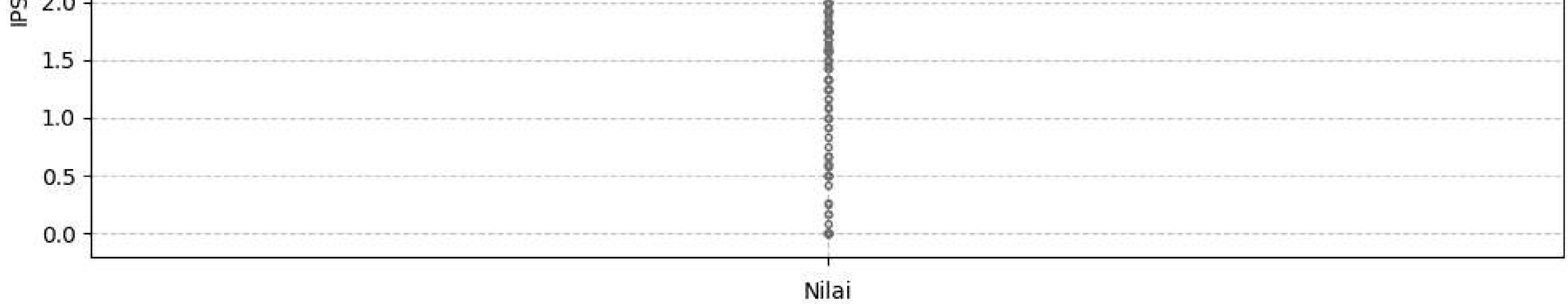


Distribusi dan Outlier: IPS4

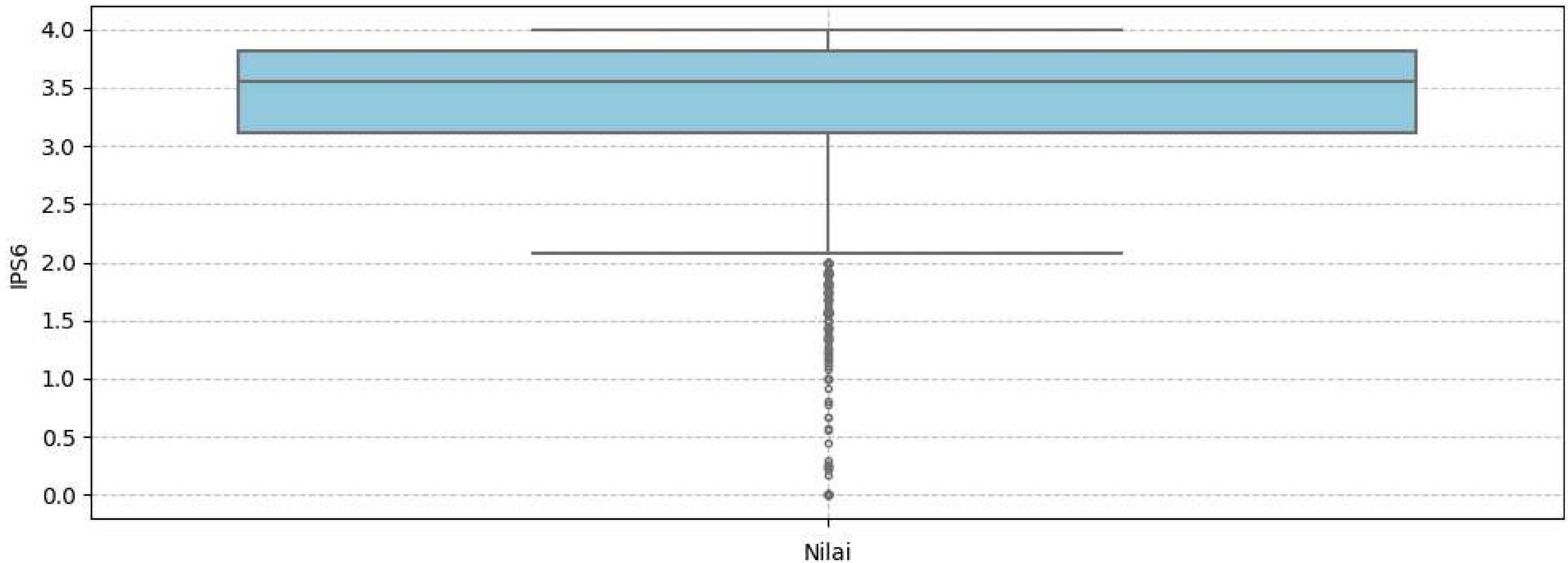


Distribusi dan Outlier: IPS5

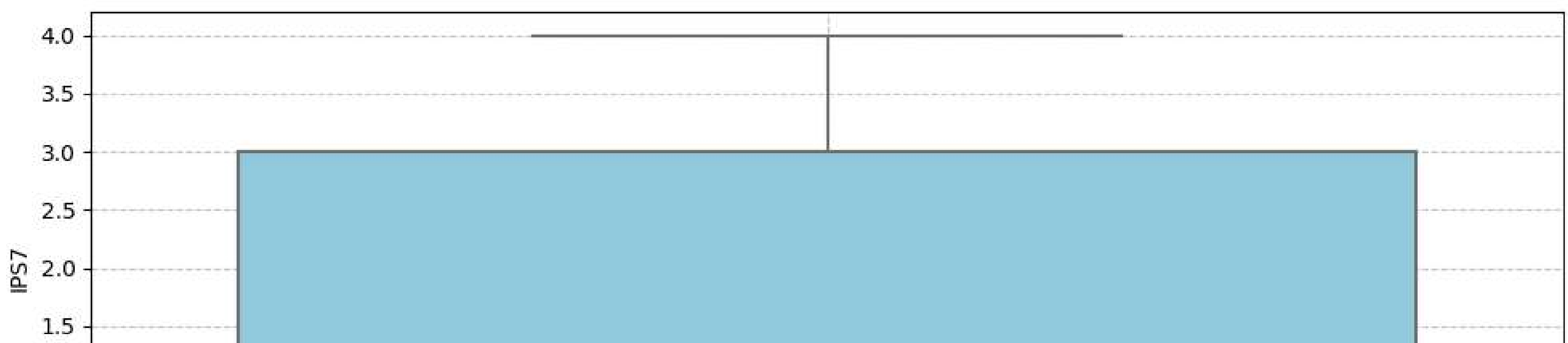


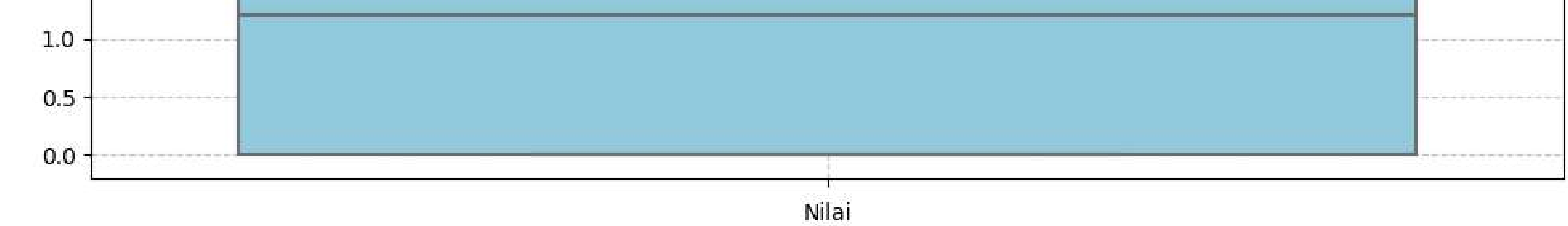


Distribusi dan Outlier: IPS6

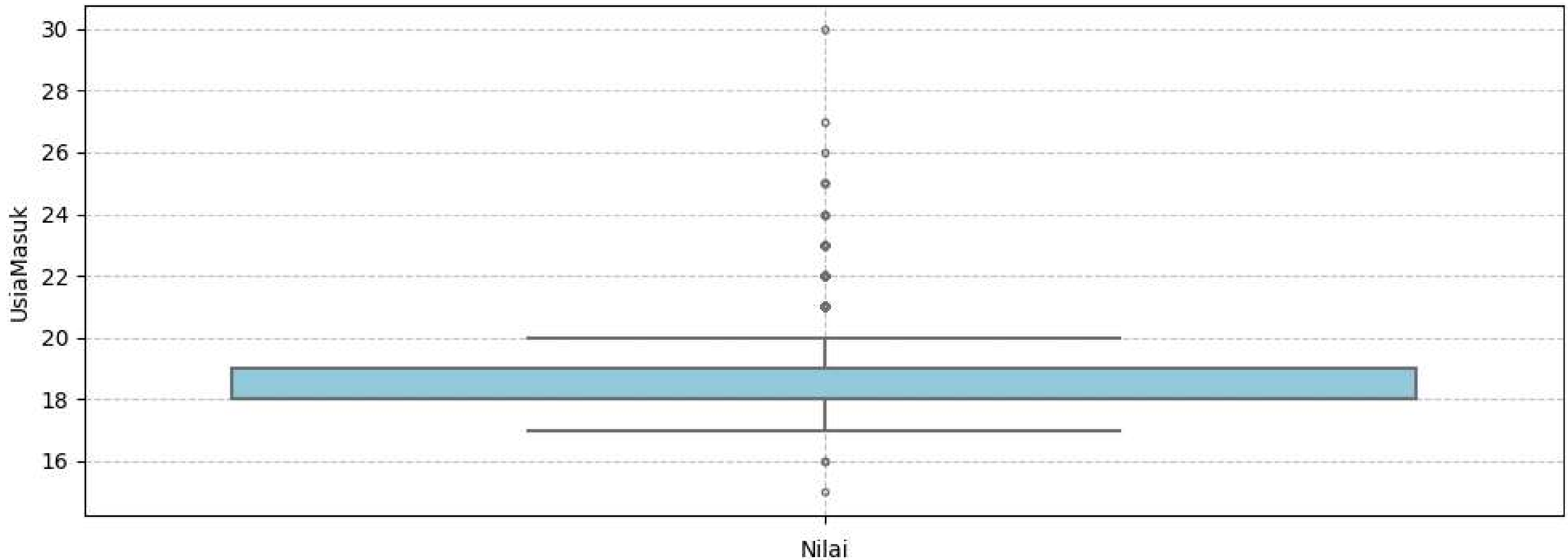


Distribusi dan Outlier: IPS7

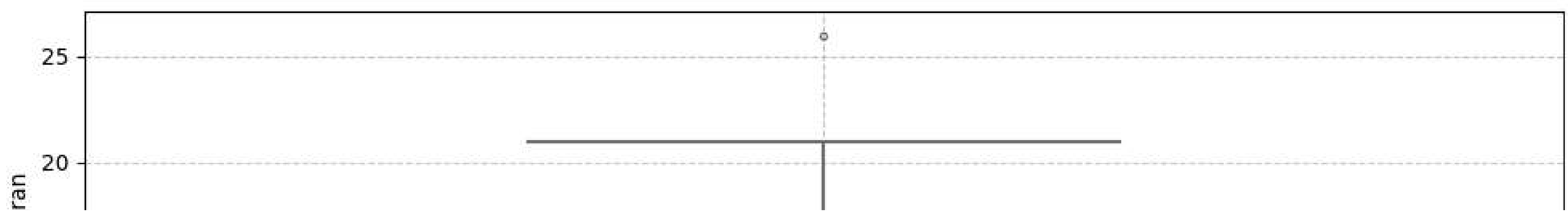




Distribusi dan Outlier: UsiaMasuk



Distribusi dan Outlier: Rerata Kehadiran



Preprocessing Data

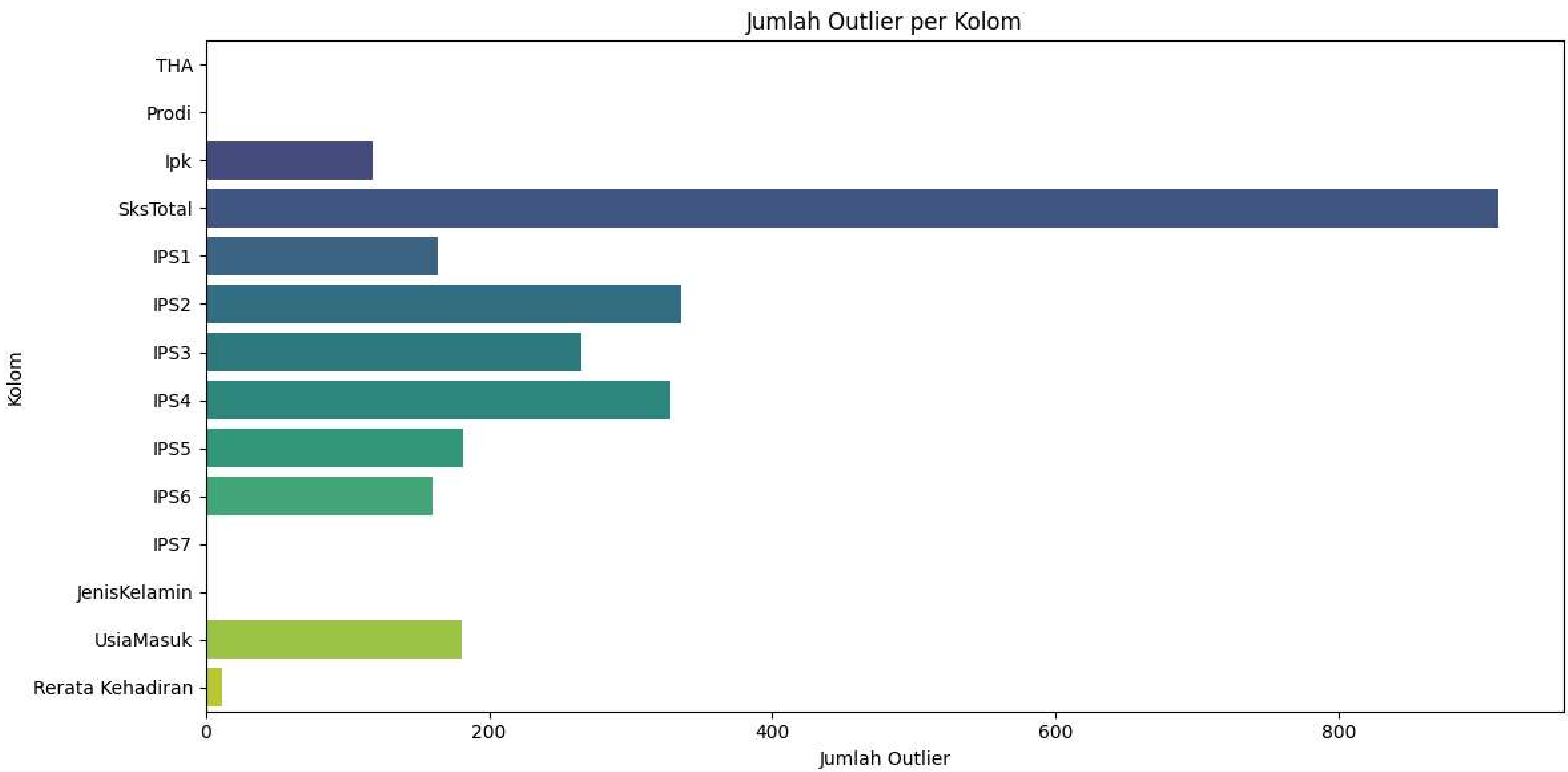
```
# Gabungkan kelas 'Memuaskan' → 'Sangat Memuaskan' (agar tidak minor)
df['PredikatKelulusan'] = df['PredikatKelulusan'].replace({
    'Memuaskan': 'Sangat Memuaskan'
  })


kolom_dihapus = ['Hashed NPM']
df = df.drop(columns=kolom_dihapus)


# Label encode fitur kategori
kategori = df.select_dtypes(include='object').columns.drop('PredikatKelulusan')
label_encoders = {}
for col in kategori:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Outlier Dectection & Removal

```
# Data jumlah outlier per kolom
data_outlier = {
    'Kolom': [
        'THA', 'Prodi', 'Ipk', 'SksTotal', 'IPS1', 'IPS2', 'IPS3',
        'IPS4', 'IPS5', 'IPS6', 'IPS7', 'JenisKelamin', 'UsiaMasuk', 'Rerata Kehadiran'
    ],
    'Jumlah Outlier': [
        0, 0, 118, 913, 164, 336, 265, 328, 182, 160, 0, 0, 181, 12
    ]
}

# Membuat DataFrame
df_outlier = pd.DataFrame(data_outlier)

# Visualisasi bar chart
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='Jumlah Outlier', y='Kolom', data=df_outlier, palette='viridis')

plt.title('Jumlah Outlier per Kolom')
plt.xlabel('Jumlah Outlier')
plt.ylabel('Kolom')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-20-3134798558.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x='Jumlah Outlier', y='Kolom', data=df_outlier, palette='viridis')
```




```
# Fungsi deteksi dan pembersihan outlier (IQR)
def detect_outliers_iqr(data, col):
```

```
    Q1, Q3 = data[col].quantile([0.25, 0.75])
    IQR = Q3 - Q1
    return data[(data[col] < Q1 - 1.5 * IQR) | (data[col] > Q3 + 1.5 * IQR)]

def remove_outliers_iqr(data, col):
    Q1, Q3 = data[col].quantile([0.25, 0.75])
    IQR = Q3 - Q1
    return data[(data[col] >= Q1 - 1.5 * IQR) & (data[col] <= Q3 + 1.5 * IQR)]



# Kolom numerik yang akan dicek outlier-nya
kolom_numerik = ['Ipk', 'SksTotal', 'IPS1', 'IPS2', 'IPS3', 'IPS4', 'IPS5', 'IPS6', 'UsiaMasuk']


# Simpan data asli
df_original = df.copy()

# Buat salinan untuk pembersihan
df_clean = df.copy()
for col in kolom_numerik:
    df_clean = remove_outliers_iqr(df_clean, col)


print(f"Jumlah data asli: {len(df_original)}")
print(f"Jumlah data bersih: {len(df_clean)}")
print(f"Outlier dibuang: {len(df_original) - len(df_clean)}")
```

```
Jumlah data asli: 4533
Jumlah data bersih: 2885
Outlier dibuang: 1648
```

```
print(df_clean.columns)
print(df_clean['SksTotal'].dtype)
print(df_clean['SksTotal'].isnull().sum())
```

```
Index(['THA', 'Prodi', 'Ipk', 'SksTotal', 'IPS1', 'IPS2', 'IPS3', 'IPS4',
       'IPS5', 'IPS6', 'IPS7', 'JenisKelamin', 'UsiaMasuk',
       'PredikatKelulusan', 'Rerata Kehadiran'],
      dtype='object')
int64
0
```

```
# Cek indeks data yang termasuk outlier (dibuang setelah pembersihan)
indeks_outlier = df_original.index.difference(df_clean.index)

# Tampilkan baris-baris yang merupakan outlier
outlier_dibuang = df_original.loc[indeks_outlier]

# Tampilkan jumlah dan data outlier yang dibuang
print(f"Jumlah outlier yang dibuang: {len(outlier_dibuang)}")
display(outlier_dibuang)
```

```
Jumlah outlier yang dibuang: 1648
```

	THA	Prodi	Ipk	SksTotal	IPS1	IPS2	IPS3	IPS4	IPS5	IPS6	IPS7	JenisKelamin	UsiaMasuk	PredikatKelulusan	Rerata Kehadiran
4	2018	7	3.82	144	3.83	3.83	4.00	3.67	3.92	3.56	0.00	1	22	Cum Laude	16
15	2018	7	3.25	144	2.67	2.42	2.92	2.75	3.42	3.73	1.40	0	18	Sangat Memuaskan	13
20	2018	7	3.06	144	3.33	2.42	3.33	2.83	3.42	2.78	0.00	0	20	Sangat Memuaskan	12
21	2018	7	3.84	146	3.75	3.92	4.00	3.83	3.67	3.80	0.00	0	19	Cum Laude	16
22	2018	7	2.63	144	3.00	2.58	2.83	2.00	2.58	1.55	0.00	0	19	Cukup	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4528	2020	5	3.84	146	4.00	3.82	3.75	3.67	3.92	3.80	1.60	1	18	Cum Laude	13
4529	2020	5	3.56	146	3.36	3.55	3.42	3.17	3.58	3.92	3.75	0	19	Cum Laude	13
4530	2020	5	3.64	146	3.73	3.55	3.50	3.33	3.83	3.80	4.00	1	19	Cum Laude	12
4531	2020	5	3.58	146	3.91	3.45	3.50	3.17	3.75	3.60	1.60	1	20	Cum Laude	13
4532	2020	5	3.44	146	3.36	3.64	3.50	3.50	3.33	3.30	4.00	0	20	Sangat Memuaskan	13

1648 rows × 15 columns

```
for col in kolom_numerik:
    outliers_col = detect_outliers_iqr(df_original, col)
    outliers_col = outliers_col.loc[outliers_col.index.difference(df_clean.index)]
    print(f"\nOutlier yang dibuang dari kolom {col} ({len(outliers_col)} data):")
    display(outliers_col[[col]])
```

```
Outlier yang dibuang dari kolom Ipk (118 data):
```

	Ipk
22	2.63
84	2.71
109	2.78
187	2.29
230	2.63
...	...
3442	2.71
3502	2.41
3607	2.76
3804	2.75
4085	2.78

118 rows × 1 columns

```
Outlier yang dibuang dari kolom SksTotal (913 data):
```

	SksTotal
21	146
33	146
69	148
79	146
90	146
...	...
4528	146
4529	146
4530	146
4531	146

4532	146

913 rows × 1 columns

```
Outlier yang dibuang dari kolom IPS1 (164 data):
```

	IPS1
15	2.67
163	2.67
190	2.25
230	2.42
240	2.67
...	...
4050	1.64
4085	2.45
4104	2.64
4121	2.55
4213	2.55

164 rows × 1 columns

```
Outlier yang dibuang dari kolom IPS2 (336 data):
```

	IPS2
15	2.42
20	2.42
22	2.58
39	2.42
57	2.67
...	...
4121	2.09
4167	2.55

```
outlier_index_all = set()

for col in kolom_numerik:
    outlier_index = detect_outliers_iqr(df_original, col).index
    outlier_index_all.update(outlier_index)

print(f"Total baris unik yang merupakan outlier di minimal satu kolom: {len(outlier_index_all)}")
```

```
Total baris unik yang merupakan outlier di minimal satu kolom: 1536
```

Outlier yang dibuang dari kolom IPS3 (265 data):

	IPS3
109	2.33
164	1.50
198	2.42
...	...
4168	2.17
4173	2.42
4182	2.33
4273	2.42

```
print(f"Total data yang dibuang: {len(df_original) - len(df_clean)}")
```

```
Total data yang dibuang: 1648
```

```
df_clean['SksTotal'].value_counts()
```

SksTotal	count
144	2885

dtype: int64

Split Data & SMOTE

```
# Pisahkan fitur dan target
X = df_clean.drop(columns='PredikatKelulusan')
y = df_clean['PredikatKelulusan']
le_y = LabelEncoder()
y_encoded = le_y.fit_transform(y)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, stratify=y_encoded, random_state=42
)
```

160 2.17

```
# Standarisasi numerik
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Terapkan SMOTE (pastikan k_neighbors < jumlah sample kelas terkecil)
min_class = min(Counter(y_train).values())
k = min(5, min_class - 1)
smote = SMOTE(random_state=42, k_neighbors=k)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)


# Distribusi label sebelum dan sesudah SMOTE
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
sns.countplot(x=y_train, ax=axes[0], palette="pastel")
axes[0].set_title("Distribusi Label Sebelum SMOTE")
sns.countplot(x=y_train_smote, ax=axes[1], palette="pastel")
axes[1].set_title("Distribusi Label Setelah SMOTE")
plt.tight_layout()
plt.show()
```

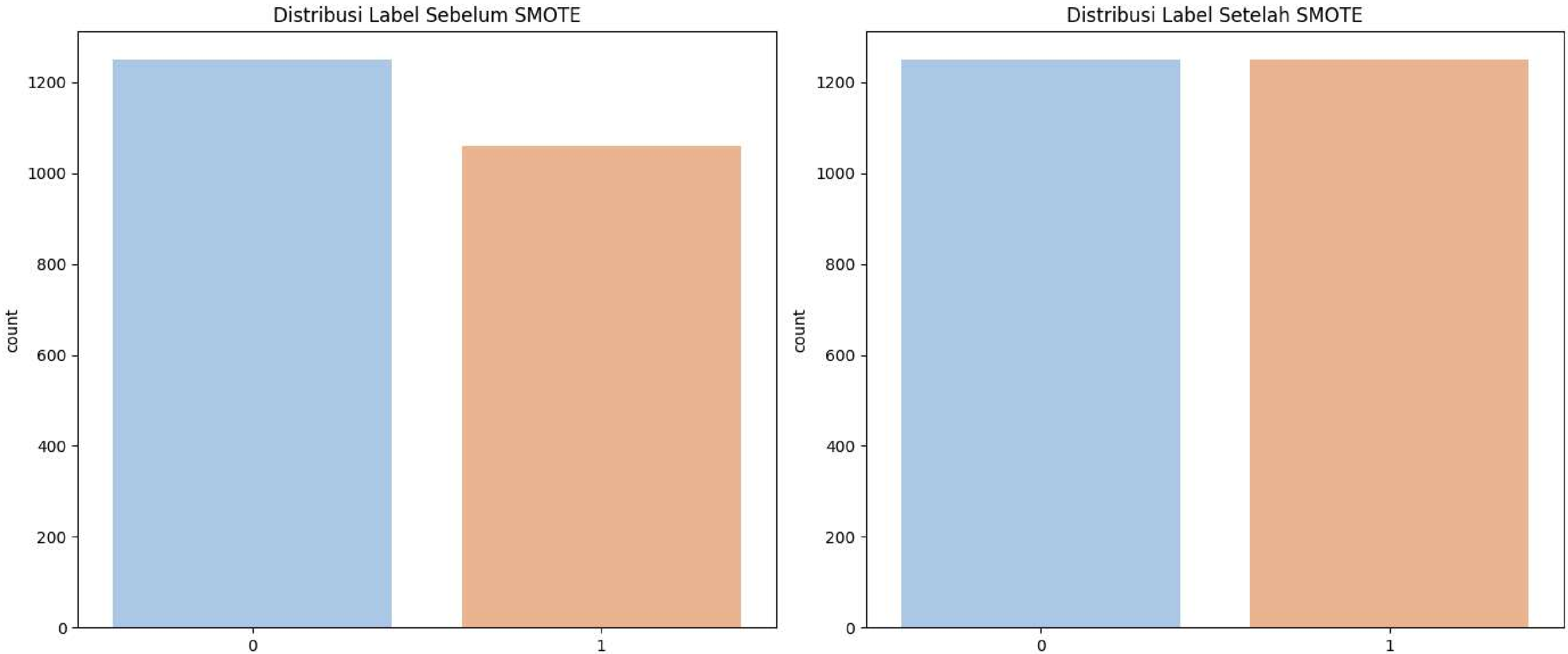
109	2.00
...	...
3481	2.33
3607	1.75
3621	2.00
3804	1.80
4167	2.33

182 rows × 1 columns

```
Outlier yang dibuang dari kolom IPS6 (160 data):
```

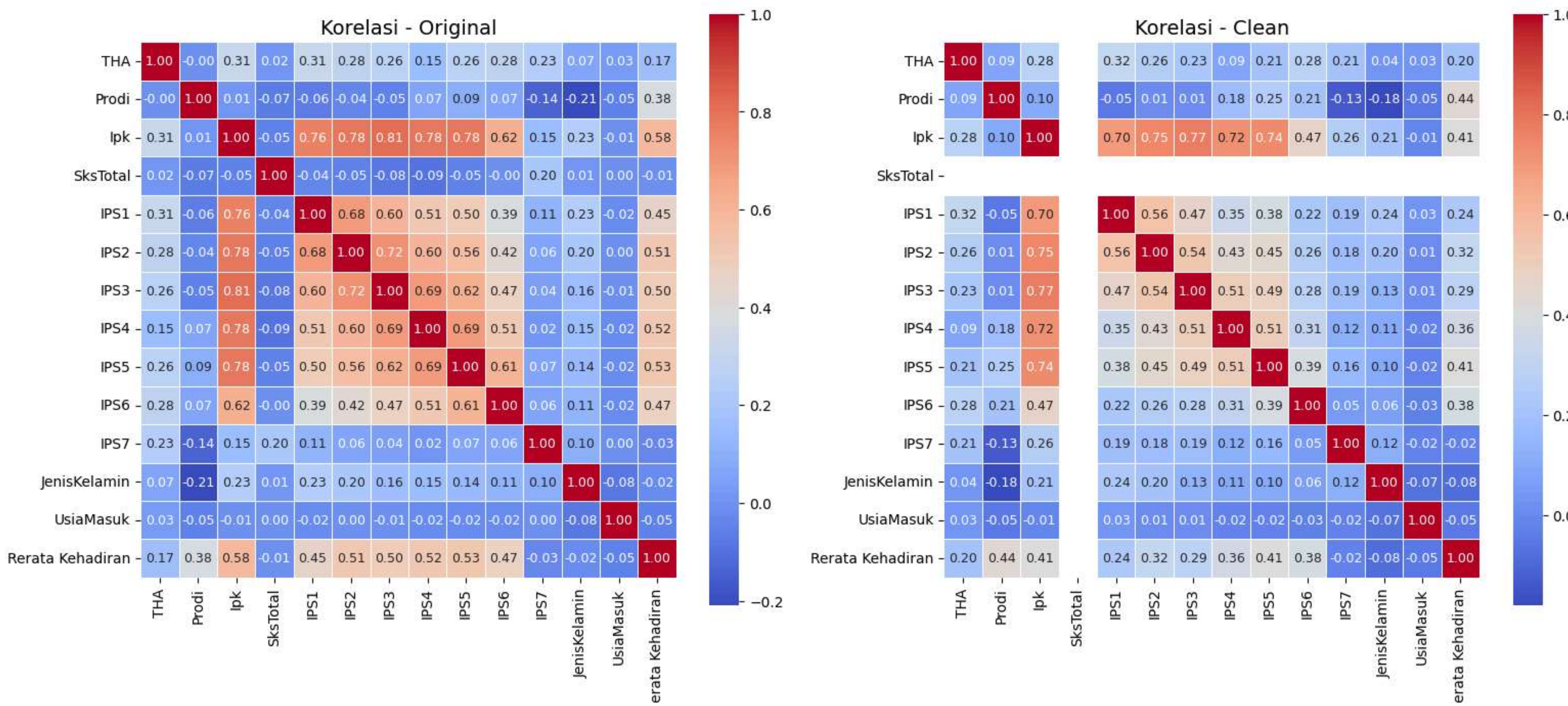
	IPS6
22	1.55
84	0.22
99	0.00

```
<ipython-input-17-119b9793e9a2>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.countplot(x=y_train, ax=axes[0], palette="pastel")
<ipython-input-17-119b9793e9a2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.countplot(x=y_train_smote, ax=axes[1], palette="pastel")
```



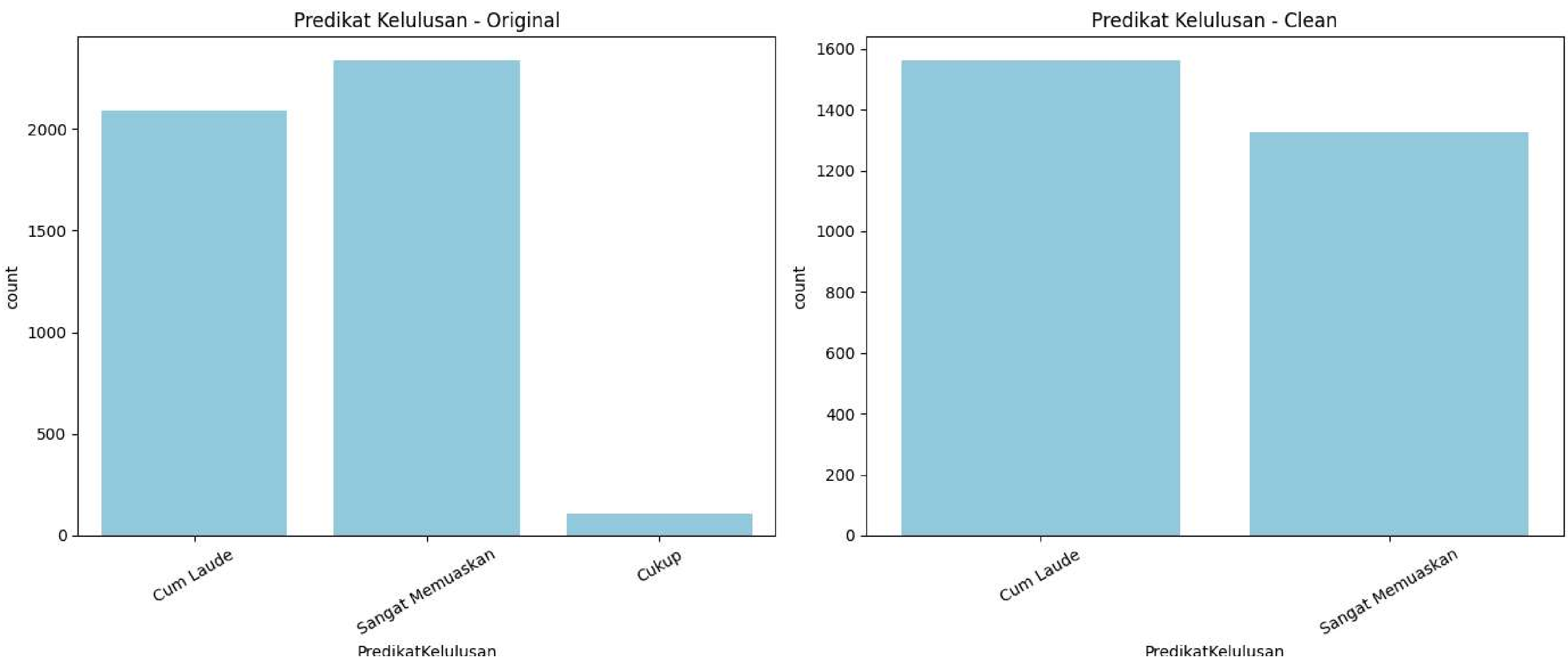

```
# Heatmap Korelasi
plt.figure(figsize=(16, 7))

plt.subplot(1, 2, 1)
sns.heatmap(
    df_original.corr(numeric_only=True).round(2),
    annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5, square=True,
    annot_kws={"size": 9}
)
```

```
plt.title('Korelasi - Original', fontsize=14)

plt.subplot(1, 2, 2)
sns.heatmap(
    df_clean.corr(numeric_only=True).round(2),
    annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5, square=True,
    annot_kws={"size": 9}
)
plt.title('Korelasi - Clean', fontsize=14)

plt.tight_layout()
plt.show()
```



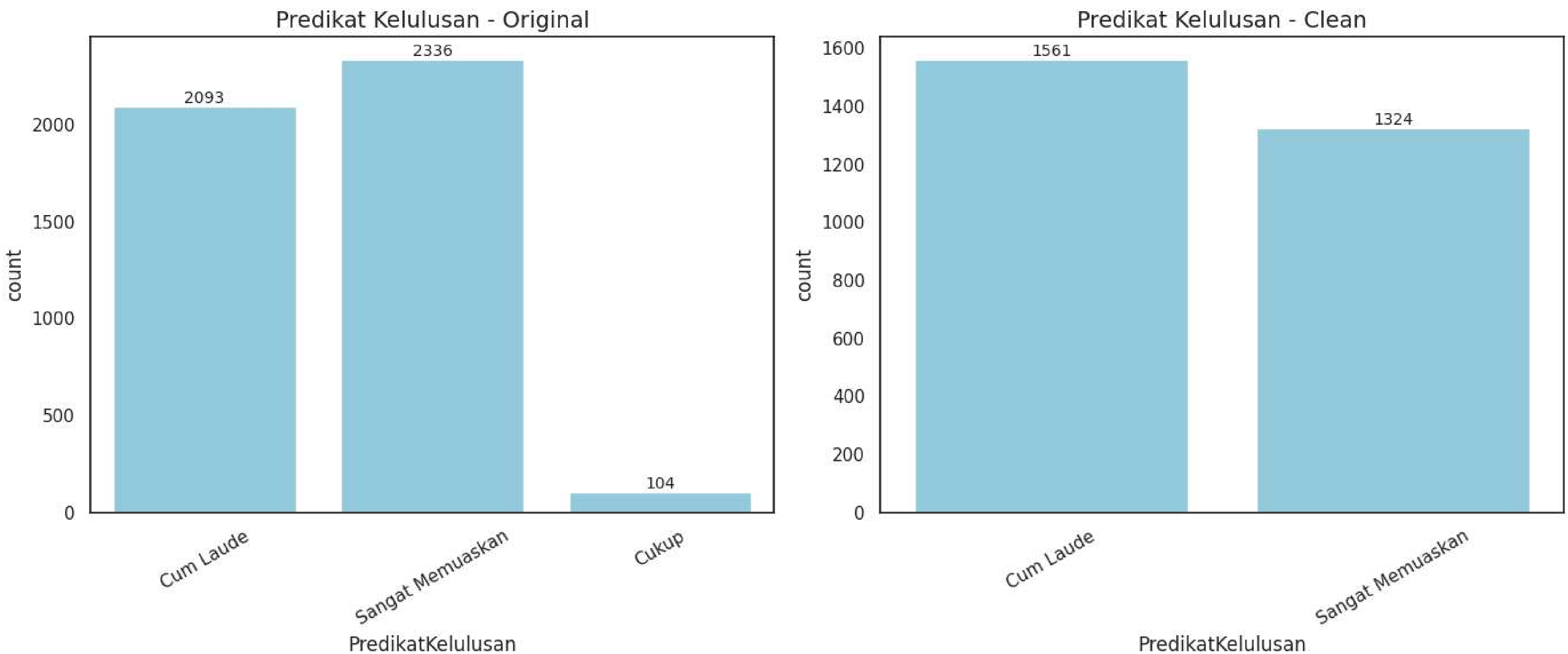

```
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.countplot(data=df_original, x='PredikatKelulusan', color='skyblue')
plt.title('Predikat Kelulusan - Original')
```

```
plt.xticks(rotation=30)

plt.subplot(1, 2, 2)
sns.countplot(data=df_clean, x='PredikatKelulusan', color='skyblue')
plt.title('Predikat Kelulusan - Clean')
plt.xticks(rotation=30)

plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(14, 6))

# Plot untuk data original
plt.subplot(1, 2, 1)
ax1 = sns.countplot(data=df_original, x='PredikatKelulusan', color='skyblue')
plt.title('Predikat Kelulusan - Original', fontsize=14)
plt.xticks(rotation=30)
for p in ax1.patches:
    ax1.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2, p.get_height()),
                 ha='center', va='bottom', fontsize=10)
```

```
# Plot untuk data clean
plt.subplot(1, 2, 2)
ax2 = sns.countplot(data=df_clean, x='PredikatKelulusan', color='skyblue')
plt.title('Predikat Kelulusan - Clean', fontsize=14)
plt.xticks(rotation=30)
for p in ax2.patches:
    ax2.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2, p.get_height()),
                 ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
```



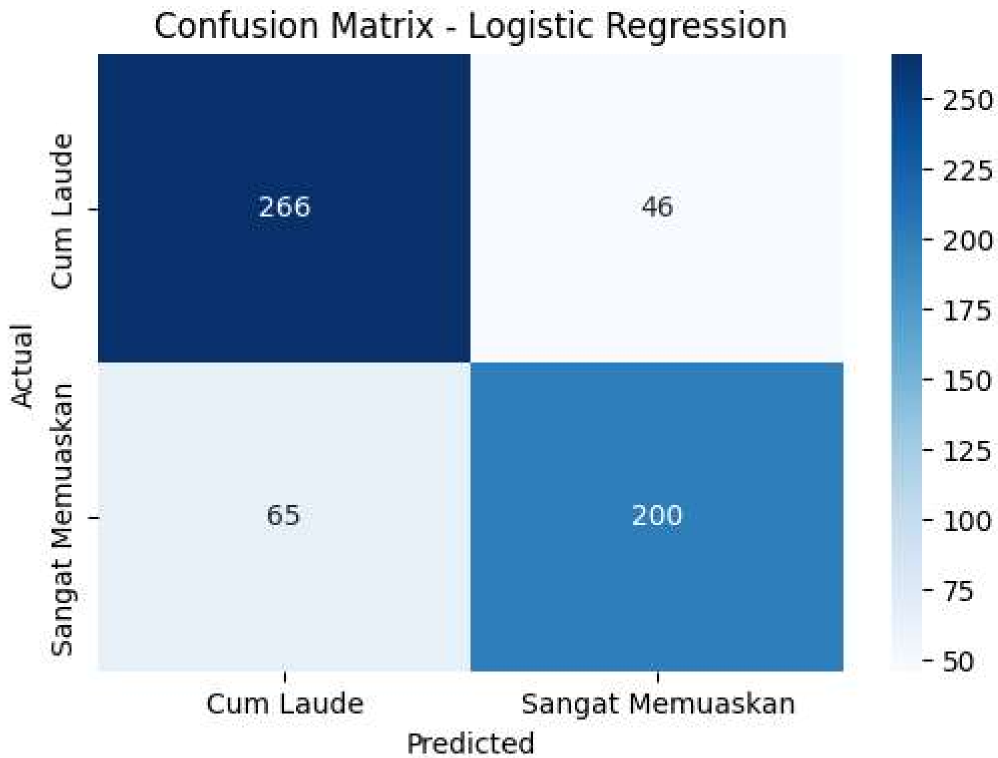
Modeling

```
# Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train_smote, y_train_smote)
```

```
y_pred_lr = lr_model.predict(X_test_scaled)


print("=== Logistic Regression ===")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Classification Report:")
print(classification_report(y_test, y_pred_lr, target_names=le_y.classes_))
```

```
=== Logistic Regression ===
Accuracy: 0.8076256499133448
Classification Report:
                  precision    recall  f1-score   support

       Cum Laude       0.80      0.85      0.83       312
Sangat Memuaskan       0.81      0.75      0.78       265

        accuracy                           0.81       577
       macro avg       0.81      0.80      0.81       577
    weighted avg       0.81      0.81      0.81       577
```

```
# Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap='Blues',
            xticklabels=le_y.classes_, yticklabels=le_y.classes_)
plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
# Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train_smote, y_train_smote)

y_pred_nb = nb_model.predict(X_test_scaled)


print("\n=== Naive Bayes ===")
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Classification Report:")
print(classification_report(y_test, y_pred_nb, target_names=le_y.classes_))
```

```
=== Naive Bayes ===
Accuracy: 0.7972270363951474
Classification Report:
                  precision    recall  f1-score   support

       Cum Laude       0.79      0.86      0.82       312
Sangat Memuaskan       0.81      0.72      0.77       265
```