Universitas Telkom

S-1 Informatika

KLASIFIKASI SENSOR IMU PADA PONSEL MENGGUNAKAN METODE K-NEAREST NEIGHBOR

KELOMPOK 7 (IF-46-09)

RAFIANTO TRI AUSHAF 1301223274

AGIL DITYA RAFIAZMI 301223044

MUHAMMAD FARHAN EDITYA 1301223077

PENJELASAN DATASET

Dataset ini dikumpulkan pada tahun 2022 di Universitas King Saud di Riyadh untuk mengenali aktivitas manusia menggunakan sensor IMU (Inertial Measurement Unit) pada ponsel, yaitu akselerometer dan giroskop. Dataset ini digunakan untuk menganalisis dan mengklasifikasikan aktivitas pengguna ponsel menjadi dua kategori: berdiri diam (standing/stop) dan berjalan (walking).

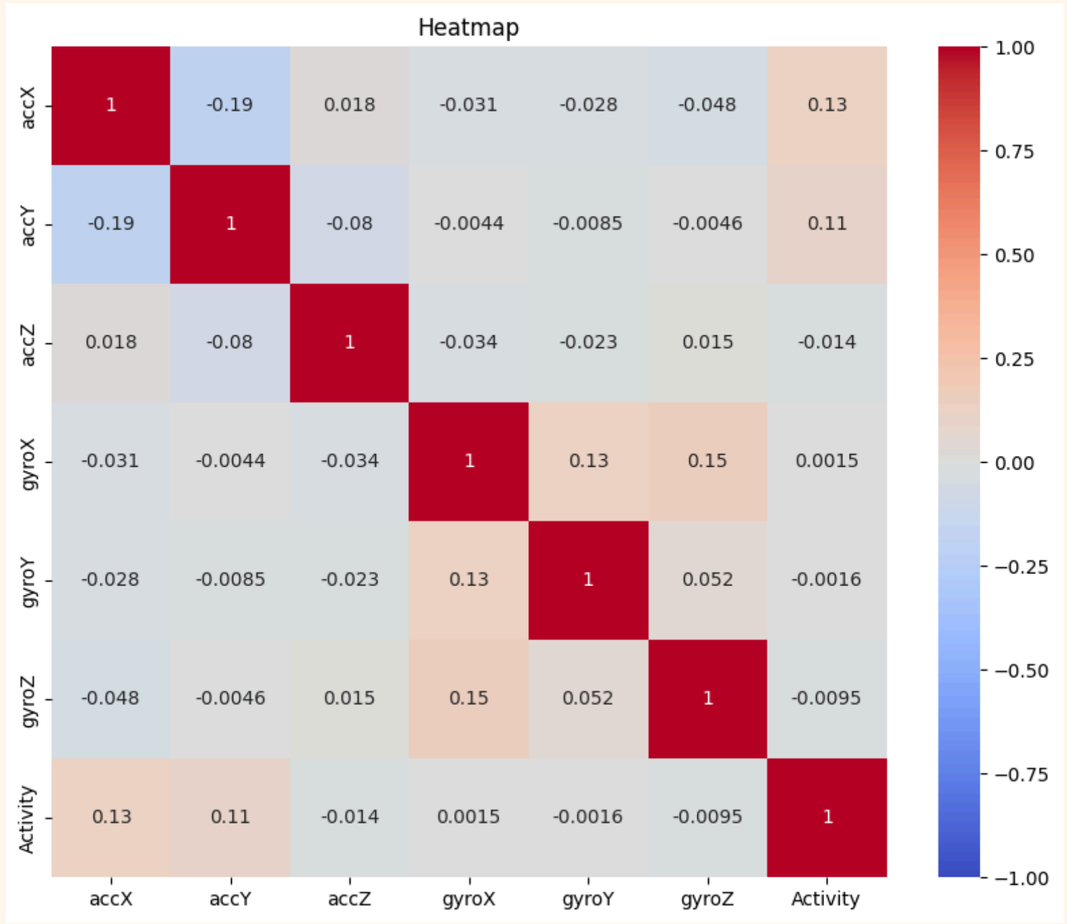
Pada dataset kami, terdapat data fitur dan juga data target. Data fitur berisi informasi terkait tentang accelerometer dan gyroscope pada sumbu X,Y, dan Z yang menjadi fitur. Sedangkan data target yaitu "Activity" berisi informasi terkait Kode numerik yang merepresentasikan aktivitas yang dilakukan saat data diambil, di mana "0" menunjukkan aktivitas berdiri diam (standing/stop) dan "1" menunjukkan aktivitas berjalan (walking).

DATASET INFORMATION

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31991 entries, 0 to 31990
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   accX       31991 non-null  float64
 1   accY       31991 non-null  float64
 2   accZ       31991 non-null  float64
 3   gyroX      31991 non-null  float64
 4   gyroY      31991 non-null  float64
 5   gyroZ      31991 non-null  float64
 6   timestamp  31991 non-null  object
 7   Activity   31991 non-null  int64
```

Data tidak memiliki null value.

HEATMAP



Tidak terlihat korelasi yang kuat antar fitur, tidak dilakukan seleksi fitur.

IMBALANCE DATA TARGET

```
arr = []
for i in df["Activity"]:
  arr.append(i)

print(set(arr))



counts = df["Activity"].value_counts()
count_0 = counts.get(0, 0)

count_1 = counts.get(1,0)
print(counts)
print(f"Jumlah angka 0: {count_0}")
print(f"Jumlah angka 1: {count_1}")
print(count_1%count_0, count_0 % count_1)
```

```
{0, 1}
Activity
1    31420
0      571
Name: count, dtype: int64
Jumlah angka 0: 571
Jumlah angka 1: 31420
15 571
```

DATA TARGET FILTER

```
df_0 = df.loc[df['Activity'] == 0].copy()
df_1 = df.loc[df['Activity'] == 1].copy()
df_1,df_0
```

Show hidden output

```
[235] df_1_train = df_1.iloc[:571].copy()
      df_1_train
```

Melakukan filter data target Activity yang bernilai 1, menjadi sama dengan jumlah activity dengan nilai 0.

MEMISAHKAN FITUR DAN TARGET


```
train_df = pd.concat([df_0, df_1_train])
```

```
[238] #Pisahkan kolom fitur dan target untuk train data

      x_train_features = train_df[['accX', 'accY', 'accZ', 'gyroX', 'gyroY', 'gyroZ']].copy()
      y_train = train_df['Activity'].copy()
```

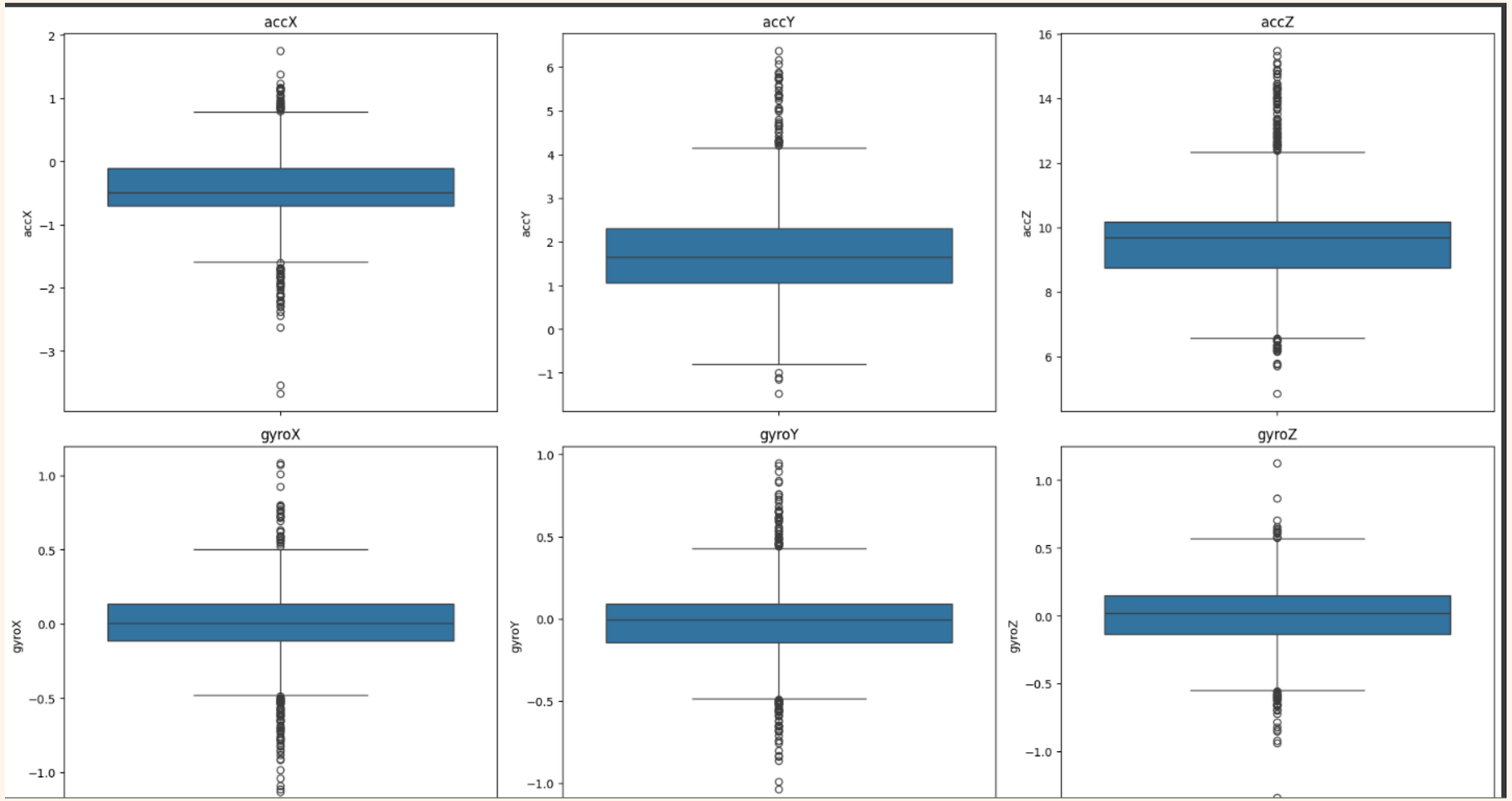
```
[239] x_train_features.describe()
```

	accX	accY	accZ	gyroX	gyroY	gyroZ
count	1142.000000	1142.000000	1142.000000	1142.000000	1142.000000	1142.000000
mean	-0.437748	1.784142	9.648687	-0.007887	-0.018681	0.001032
std	0.610533	1.124043	1.661451	0.276000	0.250366	0.265887
min	-3.673361	-1.477577	4.849016	-1.229589	-1.141854	-1.408300
25%	-0.702714	1.065709	8.742607	-0.113387	-0.142107	-0.133197
50%	-0.485590	1.654360	9.673634	0.001623	-0.004599	0.016293
75%	-0.103510	2.311418	10.188451	0.135222	0.089823	0.149198
max	1.755250	6.377039	15.480209	1.080128	0.944855	1.124462

Universitas Telkom

VISUALISASI BOXPLOT DENGAN OUTLIER

S-1 Informatika



NORMALISASI DATA

Normalisasi data

```
[244] sc = StandardScaler()
      x_normal = sc.fit_transform(x_train_features)
```

Normalisasi data menggunakan standardScaler dari library sklearn.

SPLIT DATA


```
# Splitting data into train (80%) and test (20%)

X=x_normal
y_arr = y_train.to_numpy()
x_train,x_test,y_train,y_test=train_test_split(X,y_arr,test_size=0.2,random_state=42)
```

```
[269] print("jumlah data train fitur:",x_train.shape)
      print("jumlah data test fitur:" ,x_test.shape)
      print("jumlah data train target",y_train.shape)
      print("jumlah data test target",y_test.shape)
```

```
jumlah data train fitur: (913, 6)
jumlah data test fitur: (229, 6)
jumlah data train target (913,)
jumlah data test target (229,)
```

Split daya menjadi train dan test menggunakan metode 80% train dan 20% test.
Jumlah data train = 913 data dan jumlah data test = 229 data

K-NEAREST NEIGHBORS

Metode yang digunakan untuk mengolah dataset ini adalah K-NN, dengan parameter nilai K. Metode ini akan menghitung nilai dari tetangga terdekat dari titik-titik data dengan nilai k sebagai parameter dalam penentuan label.

ALGORITMA YANG DIGUNAKAN

1.Euclidean distance untuk mencari jarak antar titik

```
[246] #euclidean distance
      import numpy as np
      def euclidean(p1,p2):
        return np.sqrt(np.sum((p1-p2)**2))
```

2.fungsi nearest_neighbors untuk mencari nilai terbesar berdasarkan parameter nilai k

```
def nearest_neighbors(x_train, y_train, test, k):
    jarak = []
    for i in range(len(x_train)):
        jarak = euclidean(x_train[i], test)
        jarak.append((y_train[i], jarak))
    jarak.sort(key=lambda x: x[1])
    neighbors = jarak[:k]
    return [neighbor[0] for neighbor in neighbors]
```

ALGORITMA YANG DIGUNAKAN

```python
from collections import Counter
def predict(x_train, y_train, test_instance, k):
    neighbors = nearest_neighbors(x_train, y_train, test_instance, k)
    most_common_label = Counter(neighbors).most_common(1)[0][0]
    return most_common_label
```

```python
def get_accuracy(y_test, predictions):
    true = 0
    for x in range(len(y_test)):
        if y_test[x] == predictions[x]:
            true += 1
    return (true / float(len(y_test))) * 100.0
```

```python
[250] def KNN(x_train, y_train, x_test, y_test, k):
    prediksi_data = []
    for i in x_test:
        prediksi_datas = predict(x_train, y_train, i, k)
        prediksi_data.append(prediksi_datas)

    accuracy = get_accuracy(y_test, predictions)
    return accuracy,predictions
```

3 fungsi predict() digunakan untuk memprediksi sebuah label.

4.get_accuracy() digunakan untuk meendapatkan akurasi dari prediksi model.

5.Model KNN, akan memprediksi label dari setiap data dan menghitung akurasi dari prediksi.

ALGORITMA YANG DIGUNAKAN

```python
from sklearn.model_selection import KFold
def cross_validation(X_features, Y_target, k, num_folds=5):
    kf = KFold(n_splits=num_folds)
    accuracies = []
    for train_index, valid_index in kf.split(X_features):
        Xtrain, Xvalid = X_features[train_index], X_features[valid_index]
        Ytrain, Yvalid = Y_target[train_index], Y_target[valid_index]
        accuracy, _ = KNN(Xtrain, Ytrain, Xvalid, Yvalid, k)
        accuracies.append(accuracy)
    return np.mean(accuracies)
```

6.Cross_validation merupakan sebuah algoritma untuk melakukan pemvalidasian terhadap data training. Data di bagi menjadi 5 folds untuk kemudian terus dilatih untuk mendapatkan akurasi yang maksimal.

ALGORITMA YANG DIGUNAKAN

```python
def find_bestK(k_values, x_train, y_train):
    best_k = 0
    score = 0.0
    for k in k_values:
        avg_accuracy = cross_validation(x_train, y_train, k)
        print(f'k={k}, Akurasi rata-rata ={avg_accuracy}')
        if avg_accuracy > score:
            score = avg_accuracy
            best_k = k
    print(f"Nilai k terbaik: {best_k}")
    return best_k
```

6.Mencari nilai K terbaik berdasarkan hasil akurasi dari cross_vali.dation

TRAINING MODEL


```
k_values = [2, 3, 5, 7]
best_k = find_bestK(k_values, x_train, y_train)
```

Mencari nilai K terbaik diantara [2,3,5,7]

```
k=2, Rata-rata Akurasi=86.85161832702816
k=3, Rata-rata Akurasi=89.04401609319643
k=5, Rata-rata Akurasi=88.49876899057227
k=7, Rata-rata Akurasi=88.17029964570949
Nilai k terbaik: 3
```

Hasil pelatihan model terhadap data train menggunakan cross validation dengan mengambil akurasi terbaik dari setiap folds nya pada setiap nilai K. Kemudian dipilih nilai K dengan rata-rata akurasi terbaik yang akan digunakan dalam testing.

TESTING MODEL


```
#Model testing
accuracy, predictions = KNN(x_train, y_train, x_test, y_test, best_k)
print(f'Akurasi k terbaik ({best_k}): {accuracy}%')
```

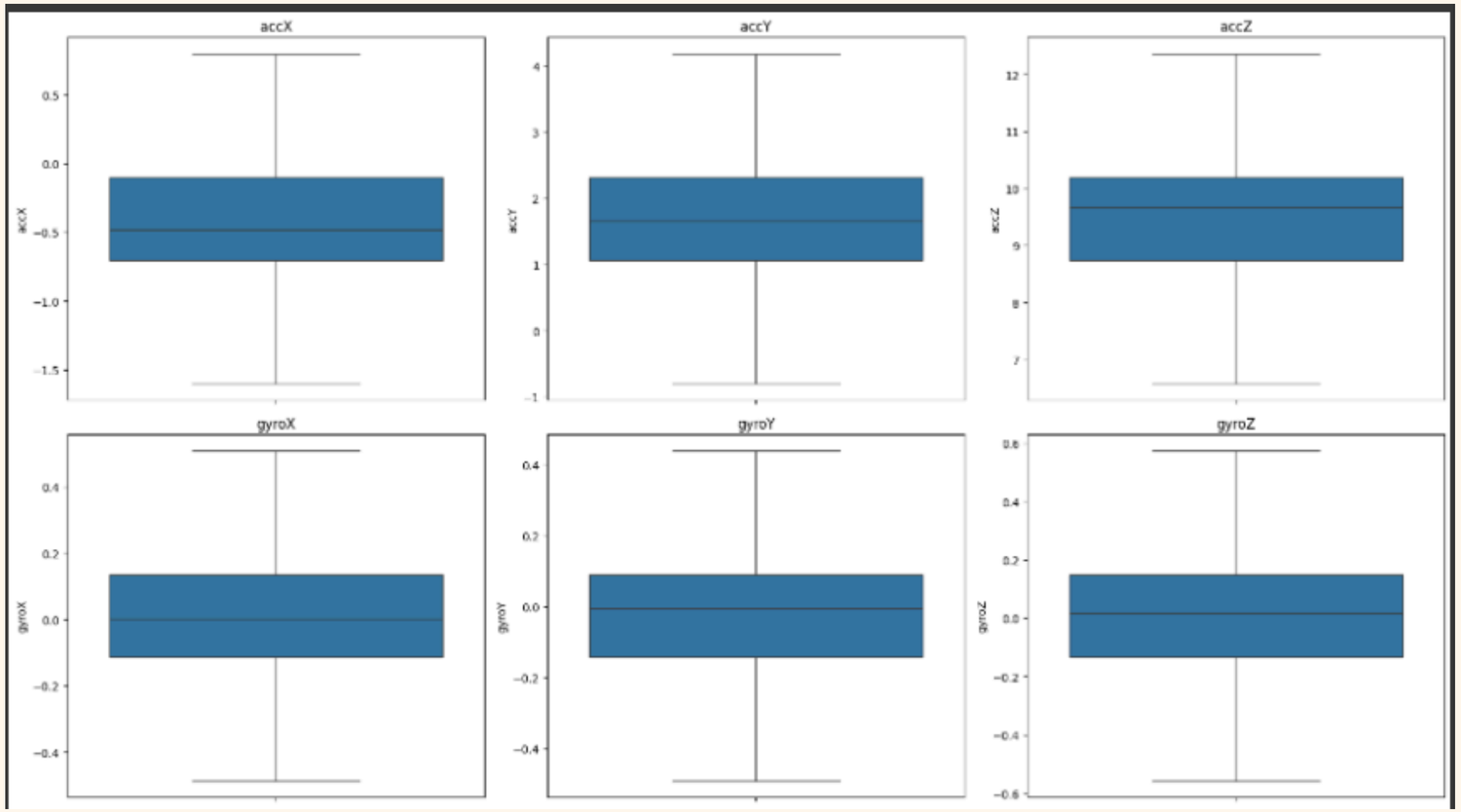
Melakukan testing terhadap model dengan nilai K terbaik berdasarkan data training.

```
Akurasi k terbaik (3): 89.08296943231441%
```

Dari hasil testing didapat akurasi sekitar 89% untuk model. pada data yang digunakan, masih terdapat outlier.

BAGAIMANA DENGAN DATA YANG SUDAH DILAKUKAN HANDLE OUTLIER?

VISUALISASI BOXPLOT SETELAH OUTLIER DI HANDLE



MENGGUNAKAN METODE IQR DALAM HANDLE OUTLIER

```python
# Handle outliers menggunakan metode IQR
for col in x_train_features_cls.columns:
    Q1 = x_train_features_cls[col].quantile(0.25)
    Q3 = x_train_features_cls[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    x_train_features_cls.loc[x_train_features_cls[col] < lower_bound, col] = lower_bound
    x_train_features_cls.loc[x_train_features_cls[col] > upper_bound, col] = upper_bound
```

Metode Interquartile Range (IQR) adalah salah satu teknik yang digunakan untuk mendeteksi dan menangani outliers dalam data. IQR adalah ukuran statistik yang digunakan untuk menggambarkan penyebaran nilai dalam satu set data, khususnya yang berhubungan dengan median (nilai tengah) dari data.

SPLIT DATA DAN TRAINING MODEL

```
X_cls=x_normal_cls
y_arr_cls = y_train_cls.to_numpy()
x_train,x_test,y_train,y_test=train_test_split(X_cls,y_arr_cls,test_size=0.2,random_state=42)

k_values = [2, 3, 5, 7]
best_k = find_bestK(k_values, x_train, y_train)
```

Hasil training

```
k=2, Rata-rata Akurasi=86.7411277247343
k=3, Rata-rata Akurasi=88.49936948297605
k=5, Rata-rata Akurasi=88.39128085029725
k=7, Rata-rata Akurasi=87.95292139554434
Nilai k terbaik: 3
```

TESTING MODEL

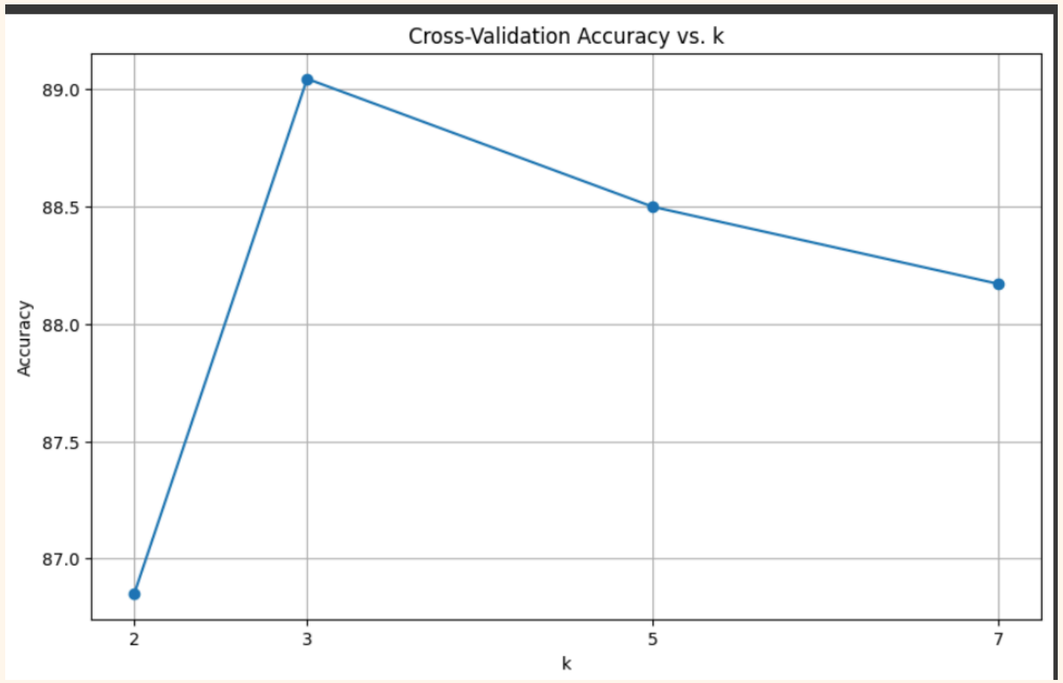
```
accuracy, predictions = KNN(x_train, y_train, x_test, y_test, best_k)
```

Hasil testing menggunakan nilai K terbaik, didapatkan akurasi sekitar **88.2%**

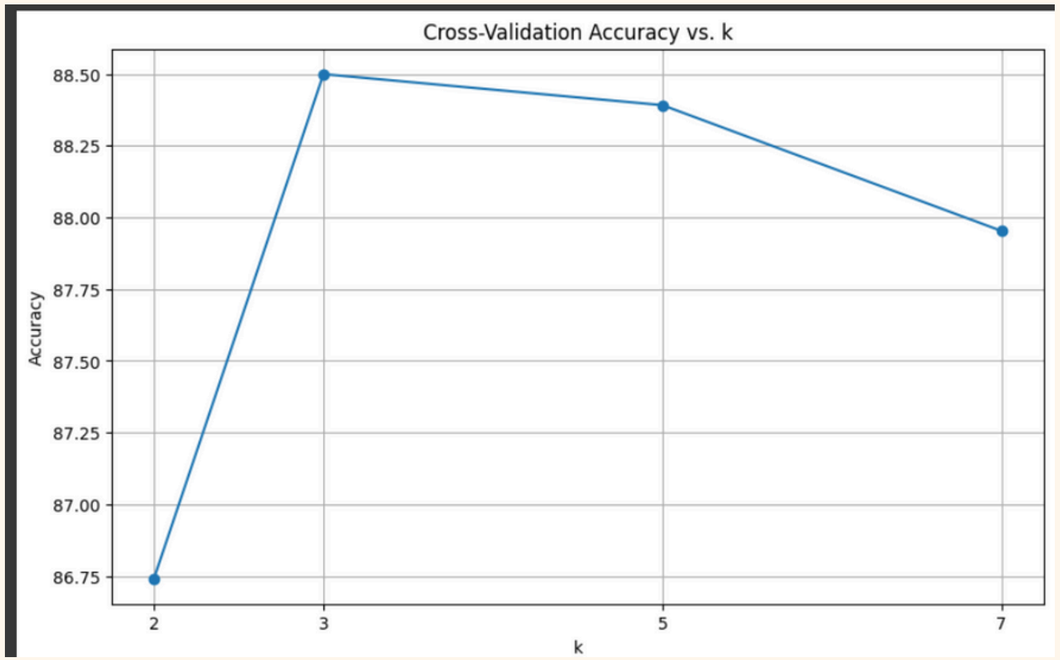
```
Akurasi k terbaik (3): 88.20960698689956%
```

HASIL DAN ANALISIS

NILAI K TERBAIK DARI MASING-MASING KASUS

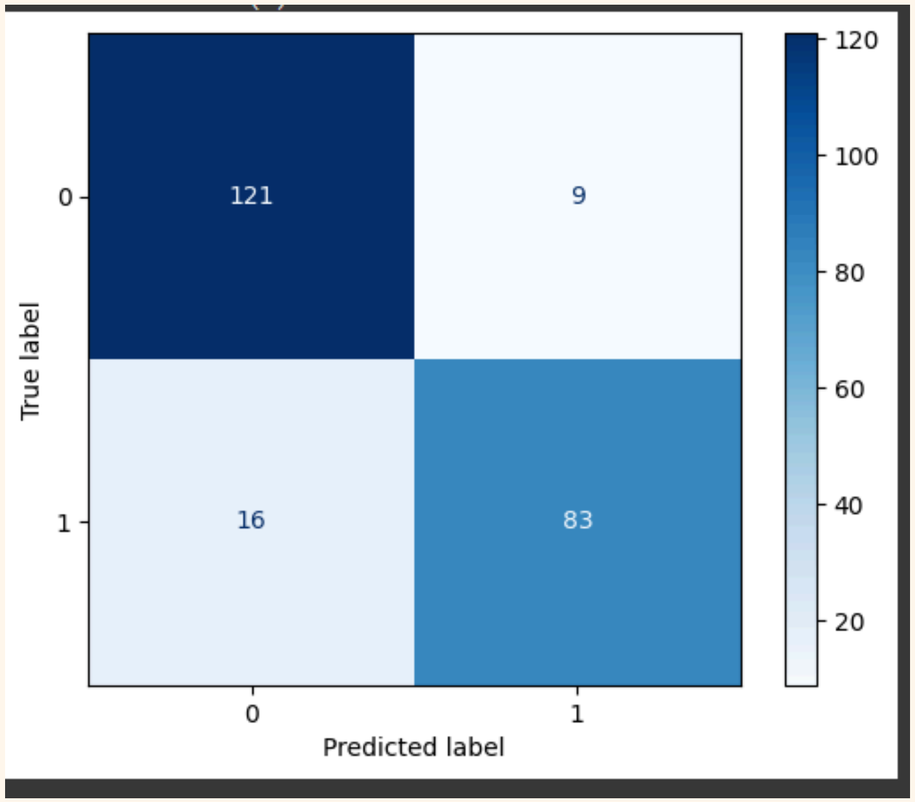


Grafik ini menunjukkan bahwa tanpa outlier handle nilai k=3 memberikan akurasi tertinggi mendekati 89%, menandakan nilai k optimal. Setelah k=3, akurasi menurun secara bertahap, lebih jelas pada k=5 dan terus menurun hingga k=7. Rentang k yang dianalisis adalah dari k=2 hingga k=7.

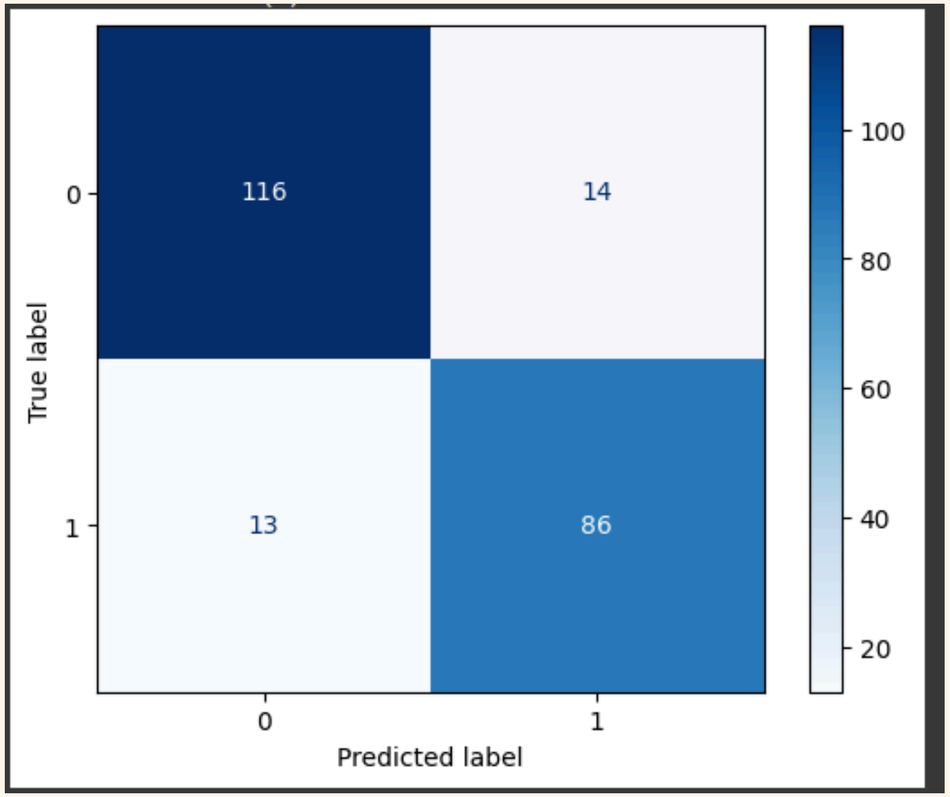


Grafik ini menunjukkan bahwa dengan outlier handler nilai k=3 memberikan akurasi tertinggi mendekati 88.5%, tidak jauh berbeda dengan ketika tidak menggunakan outlier handler dalam konteks ini.

CONFUSSION MATRIX MASING-MASING KASUS



Pada data yang tidak mengguanakan outlier handler pada confussion matrix,terdapat
83 TRUE POSITVE, 121 TRUE NEGATIVE, 16 FALSE POSITIVE, DAN 9 FALSE NEGATIVE.



Pada data yang mengguanakan outlier handler pada confussion matrix,terdapat
86 TRUE POSITVE, 116 TRUE NEGATIVE, 13 FALSE POSITIVE, DAN 14 FALSE NEGATIVE.

KESIMPULAN

- ***Meskipun terdapat sedikit perbedaan dalam akurasi antara data yang dilakukan pembersihan outlier dan yang tidak, perbedaan tersebut tidak signifikan.***
- ***Pembersihan outlier tampaknya tidak memberikan peningkatan yang substansial dalam performa model klasifikasi pada dataset ini.***
- ***Pemilihan nilai k terbaik adalah 3, yang mungkin merupakan nilai k yang optimal untuk model ini berdasarkan hasil cross-validation.***
- ***Model klasifikasi k-NN memiliki kinerja yang stabil dan cukup baik dalam memprediksi kategori target pada dataset ini, baik dengan atau tanpa pembersihan outlier.***

Universitas Telkom

S-1 Informatika

TERIMA KASIH