PENERAPAN METODE KERNEL PADA DATASET OPP

KELOMPOK 2

Anggota:

1. 121450007 Dede Masita
2. 121450025 Ayu Erlinawati
3. 121450043 Sella Dianka Fitri
4. 121450073 Rizki Adrian Bennovry
5. 121450115 Saiful Haris Muhammad
6. 121450135 Muhammad Kaisar Firdaus

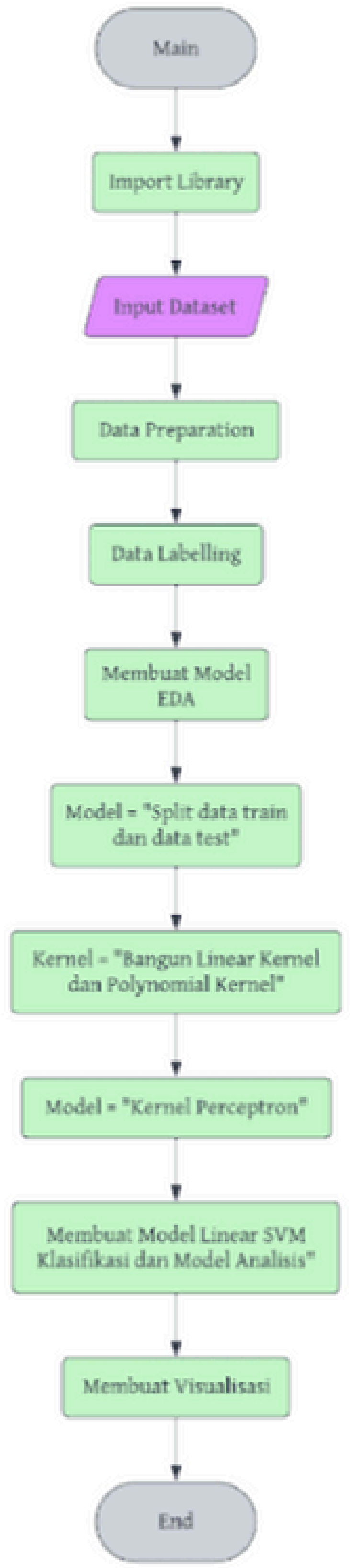
PENDAHULUAN

Pada era perkembangan teknologi saat ini, pembelajaran mesin atau machine learning menjadi bidang yang menarik perhatian karena kemampuannya dalam menghasilkan model prediktif dari data. Metode-metode dalam machine learning, terutama yang berbasis kernel, seperti Perceptron Kernel, memiliki peran penting dalam mengatasi permasalahan klasifikasi pada data yang tidak linier. Dalam konteks ini, tugas kami adalah menerapkan metode Perceptron Kernel pada dataset OPP (Opossum Possum), sebuah dataset yang menyajikan atribut-atribut fisik dari beberapa posum.

METODE

Perceptron Kernel adalah salah satu teknik dalam pembelajaran mesin yang digunakan untuk menangani pemisahan kelas atau klasifikasi pada dataset yang tidak linier. Metode ini memanfaatkan kernel untuk mentransformasikan data ke dimensi yang lebih tinggi, sehingga memungkinkan pemisahan kelas yang tidak dapat dipisahkan secara linier dalam dimensi rendah. Dengan memproyeksikan data ke ruang fitur yang lebih kompleks, Perceptron Kernel memungkinkan model untuk melakukan klasifikasi dengan lebih akurat.

FLOWCHART



Main
Import Library
Input Dataset
Data Preparation
Data Labelling
Membuat Model EDA
Model = "Split data train dan data test"
Kernel = "Bangun Linear Kernel dan Polynomial Kernel"
Model = "Kernel Perceptron"
Membuat Model Linear SVM Klasifikasi dan Model Analisis"
Membuat Visualisasi
End

Data Understanding & Processing

```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.svm import SVC
```

```
[ ] np.set_printoptions(suppress=True)
    df =pd.read_csv('opp.csv')
    df.head()
```

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
0	1	1	Vic	m	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36.0
1	2	1	Vic	f	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33.0
2	3	1	Vic	f	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34.0
3	4	1	Vic	f	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34.0
4	5	1	Vic	f	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33.0

Data Cleaning & Feature Engineering

```
df=df.dropna()
df=df.drop(['case','Pop','site'],axis=1)
df.head()
```

	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
0	m	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36.0
1	f	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33.0
2	f	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34.0
3	f	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34.0
4	f	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101 entries, 0 to 103
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sex       101 non-null    object
 1   age       101 non-null    float64
 2   hdlngth   101 non-null    float64
 3   skullw    101 non-null    float64
 4   totlngth  101 non-null    float64
 5   taill     101 non-null    float64
 6   footlgth  101 non-null    float64
 7   earconch  101 non-null    float64
 8   eye       101 non-null    float64
 9   chest     101 non-null    float64
 10  belly     101 non-null    float64
dtypes: float64(10), object(1)
memory usage: 9.5+ KB
```

Data Labeling

```
[ ] data_column_category = df.select_dtypes(exclude=[np.number]).columns

    data_column_category
```

```
Index(['sex'], dtype='object')
```

```
[ ] df[data_column_category].head()
```

	sex
0	m
1	f
2	f
3	f
4	f

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

for i in data_column_category:
    df[i] = label_encoder.fit_transform(df[i])

print("Label Encoded Data: ")
df.head()
```

Label Encoded Data:

	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
0	1	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36.0
1	0	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33.0
2	0	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34.0
3	0	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34.0
4	0	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33.0

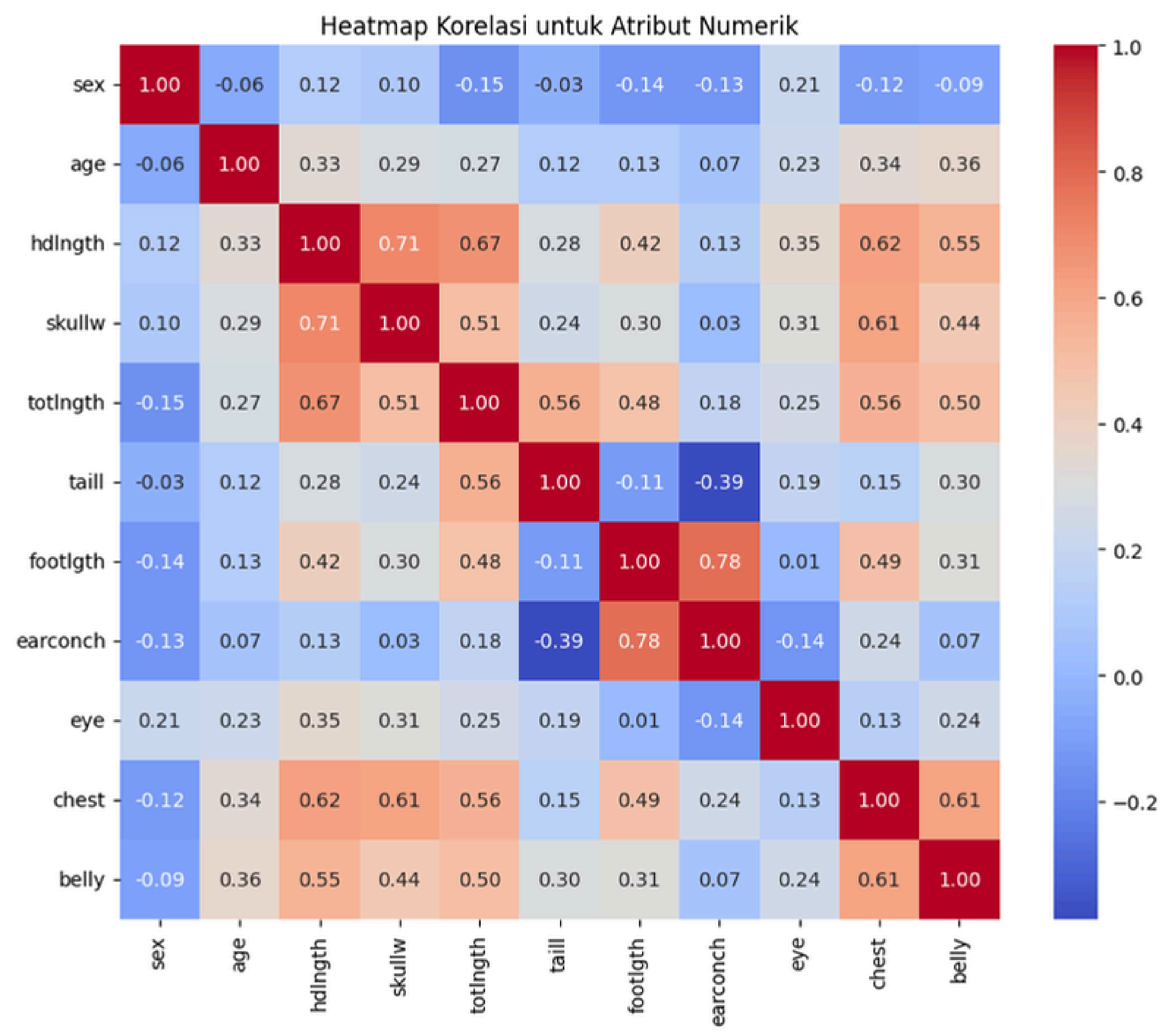
EDA

```
[ ] korelasi = df.corr()
    korelasi
```

	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
sex	1.000000	-0.057821	0.118705	0.104349	-0.152441	-0.029907	-0.137153	-0.133683	0.212078	-0.117863	-0.093835
age	-0.057821	1.000000	0.329505	0.285563	0.268297	0.120205	0.126190	0.066234	0.231857	0.335030	0.360816
hdlngth	0.118705	0.329505	1.000000	0.705901	0.670402	0.275155	0.415945	0.131576	0.354688	0.621068	0.545438
skullw	0.104349	0.285563	0.705901	1.000000	0.506382	0.241027	0.297197	0.025293	0.314319	0.613842	0.444216
totlngth	-0.152441	0.268297	0.670402	0.506382	1.000000	0.563586	0.483174	0.181230	0.247150	0.556094	0.500558
taill	-0.029907	0.120205	0.275155	0.241027	0.563586	1.000000	-0.114560	-0.387871	0.192341	0.152924	0.296206
footlgth	-0.137153	0.126190	0.415945	0.297197	0.483174	-0.114560	1.000000	0.782415	0.013869	0.486477	0.311970
earconch	-0.133683	0.066234	0.131576	0.025293	0.181230	-0.387871	0.782415	1.000000	-0.143869	0.241359	0.071309
eye	0.212078	0.231857	0.354688	0.314319	0.247150	0.192341	0.013869	-0.143869	1.000000	0.134730	0.242902
chest	-0.117863	0.335030	0.621068	0.613842	0.556094	0.152924	0.486477	0.241359	0.134730	1.000000	0.609757
belly	-0.093835	0.360816	0.545438	0.444216	0.500558	0.296206	0.311970	0.071309	0.242902	0.609757	1.000000

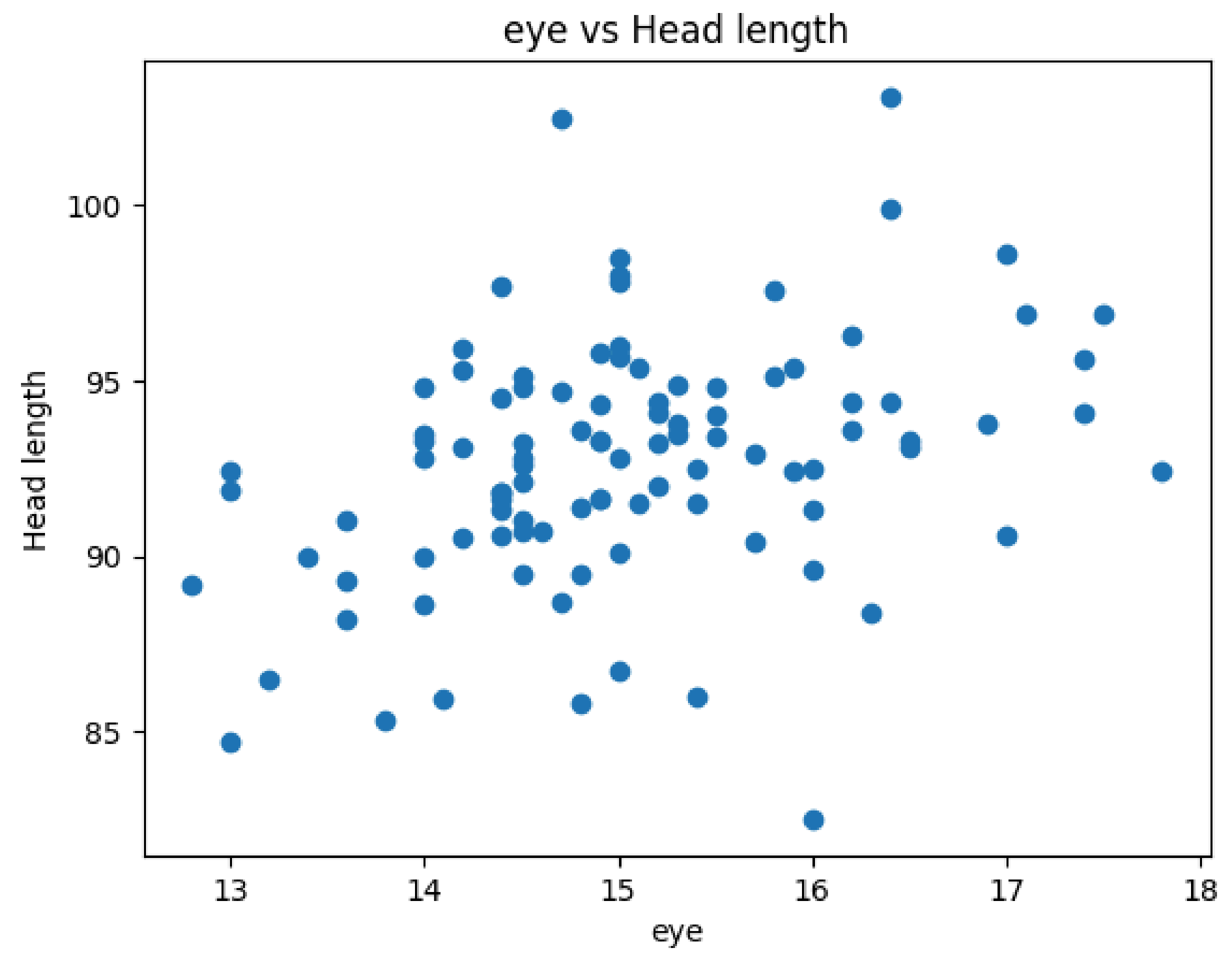
EDA

```python
korelasi = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(korelasi, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Heatmap Korelasi untuk Atribut Numerik')
plt.show()
```



EDA

```
plt.scatter(df['eye'], df['hdlngth']) # membuat scatter plot 'eye' dan juga 'hdlngth'
plt.title('eye vs Head length') # menambahkan judul
plt.xlabel('eye') #memberikan label x
plt.ylabel('Head length') # memberikan label y
plt.show() # menampilkan plot
```



Kernel Perceptron Modelling

Split Train n Test Data

```
[ ] X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values

    X = np.asarray(X,dtype=np.float)
    y = np.asarray(y,dtype=np.float)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

    print(X_train.shape,y_train.shape)
    print(X_test.shape,y_test.shape)
```

```
(80, 2) (80,)
(21, 2) (21,)
```

Linear Kernel & Polynomial Kernel

```
def linear_kernel(X,Y):
    assert X.shape[1] == Y.shape[1]
    K = np.matmul(X,Y.T)
    return K

def polynomial_kernel(X,Y,degree=2,gamma=None,a=1):
    assert X.shape[1] == Y.shape[1]
    if gamma is None:
        gamma = 1./X.shape[1]
    K = np.matmul(X,Y.T)
    K *= gamma
    K += a
    K **= degree
    return K
```

Kernel Perceptron

```python
def KernelPerceptron(X_train,y_train,X_test,y_test, max_iter=10, kernel='linear', degree=2,gamma=None,a=1):
    T = X_train.shape[0] # inisiasi jumlah data pelatihan 'T'
    alpha = np.zeros(T) # vektor alpha sebagai vektor nol dengan panjang T
    # Kernel Linear
    if kernel == 'linear':
        for it in range(max_iter):
            up = 0
            for t in range(T):
                X_t = np.reshape(X_train[t],(1,2))
                h = linear_kernel(X_train,X_t)
                y_t_hat = np.sign(np.matmul((alpha*y_train).T,h))
                if y_train[t] != y_t_hat:
                    alpha[t] += 1
                    up += 1
            print('iter:{},updates:{}'.format(it+1,up))
        h = linear_kernel(X_train,X_test)  # menghitung hasil prediksi pada data uji dan data pelatihan
        y_test_preds = np.sign(np.matmul((alpha*y_train).T,h))
        h = linear_kernel(X_train,X_train)
        y_train_preds = np.sign(np.matmul((alpha*y_train).T,h))

    elif kernel == 'poly': # kernel polynomial
        for it in range(max_iter):
            up = 0
            for t in range(T):
                X_t = np.reshape(X_train[t],(1,2))
                h = polynomial_kernel(X_train,X_t,degree=degree,gamma=gamma,a=a)
                y_t_hat = np.sign(np.matmul((alpha*y_train).T,h))
                if y_train[t] != y_t_hat:
                    alpha[t] += 1
                    up += 1
            print('iter:{},updates:{}'.format(it+1,up))
        h = polynomial_kernel(X_train,X_test,degree=degree,gamma=gamma,a=a)
        y_test_preds = np.sign(np.matmul((alpha*y_train).T,h))
        h = polynomial_kernel(X_train,X_train,degree=degree,gamma=gamma,a=a)
        y_train_preds = np.sign(np.matmul((alpha*y_train).T,h))
    else:
        print('Invalid Kernel !')
        return
    return alpha, np.mean(y_test == y_test_preds), np.mean(y_train == y_train_preds), [degree, gamma, a]

def predict_poly(alpha,X,Y,y,degree=2,gamma=None,a=1):
    h = polynomial_kernel(X,Y,degree=degree,gamma=gamma,a=a)
    return np.sign(np.matmul((alpha*y).T,h))
```

Kernel Perceptron

```python
def predict_linear(alpha,X,Y,y):
    h = linear_kernel(X,Y)
    return np.sign(np.matmul((alpha*y).T,h))

def predict(alpha,X,Y,y,kernel='linear',degree=2,gamma=None,a=1):
    if kernel == 'linear':
        return predict_linear(alpha,X,Y,y)
    elif kernel == 'poly':
        return predict_poly(alpha,X,Y,y,degree=degree,gamma=gamma,a=a)
    else:
        print('Invalid Kernel !')
        return

kernel = 'linear'
alpha,te_acc,tr_acc,params = KernelPerceptron(X_train,y_train,X_test,y_test,max_iter=100,kernel=kernel,degree=3)
print('test accuracy: {}, train accuracy: {}'.format(te_acc,tr_acc))
```

```
iter:96,updates:35
iter:97,updates:35
iter:98,updates:35
iter:99,updates:35
iter:100,updates:35
test accuracy: 0.6666666666666666, train accuracy: 0.5625
```

Linear SVM Classification and Model Analysis

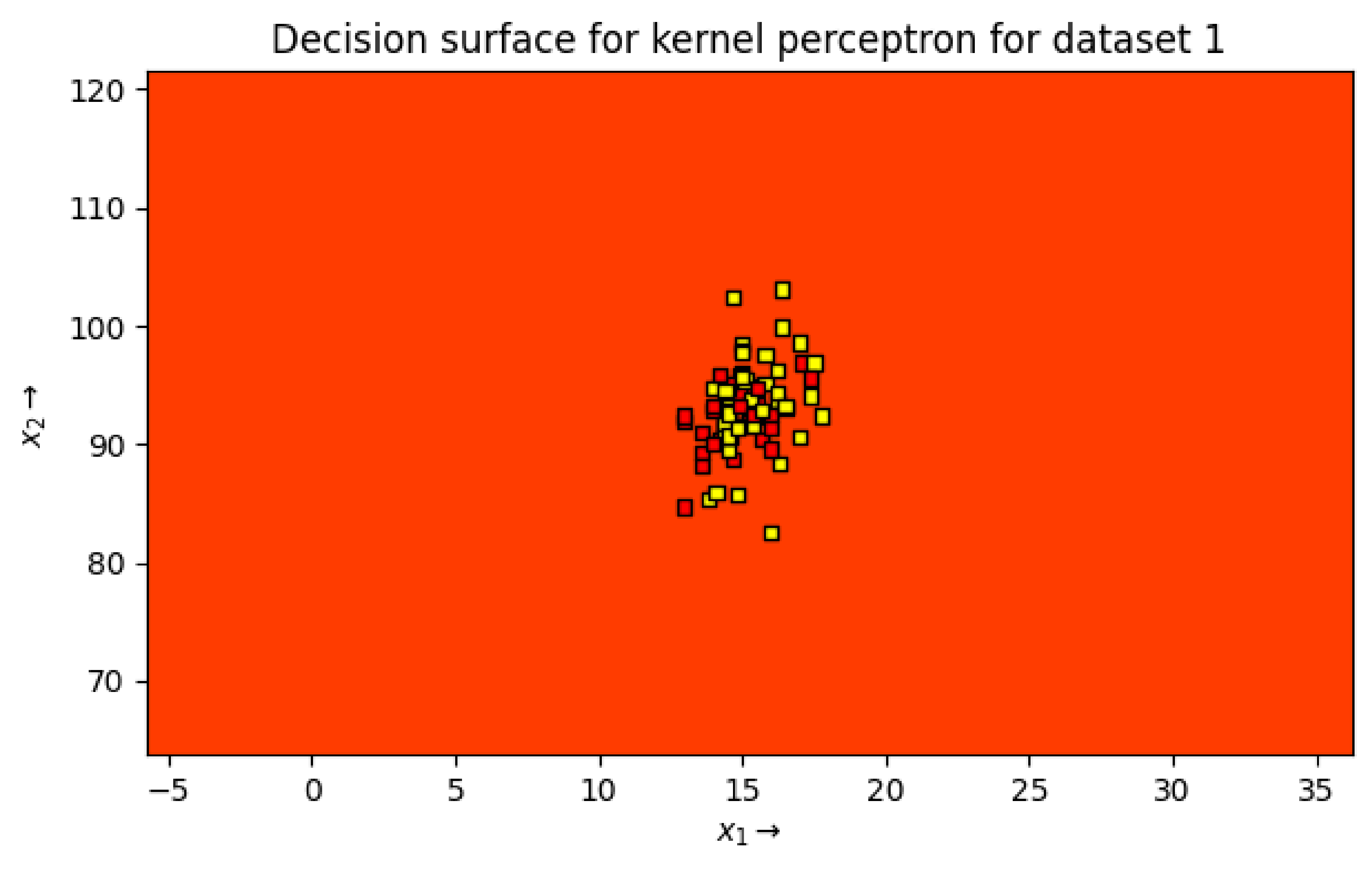
```
[ ]  R = np.linalg.norm(X_train,axis=1).max()
     clf = SVC(kernel='linear',C=1000,degree=3)
     clf.fit(X_train, y_train)
     print(f'Train Accuracy: {clf.score(X_train, y_train)}')
     print(f'Test Accuracy: {clf.score(X_test, y_test)}')

     w = clf.coef_
     w1 = w/np.linalg.norm(w)
     gamma = (y_train*np.matmul(w1,X_train.T)).min()
```

```
Train Accuracy: 0.55
Test Accuracy: 0.5714285714285714
```

Fungsi Visual

```
[ ] dataset = 1
    def make_meshgrid(x, y, dataset=3):
        h = 0.02 if dataset == 3 else 0.75
        x_min, x_max = x.min() - 25 * h, x.max() + 25 * h
        y_min, y_max = y.min() - 25 * h, y.max() + 25 * h
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
        return xx, yy

    def plot_contours(ax, alpha_, X_train, y, xx, yy, kernel='linear', degree=2, gamma=None, a=1, **params):
        X = np.c_[xx.ravel(), yy.ravel()]
        Z = predict(alpha_, X_train, X, y, kernel=kernel, degree=degree, gamma=gamma, a=a)
        Z = Z.reshape(xx.shape)
        out = ax.contourf(xx, yy, Z, **params)
        return out
```

```
fig, ax = plt.subplots(figsize=(7, 4))

X0, X1 = X_train[:, 0], X_train[:, 1]

xx, yy = make_meshgrid(X0, X1, dataset=dataset)

degree, gamma, a = params

plot_contours(ax, alpha_=alpha, X_train=X_train,
              y=y_train, xx=xx, yy=yy,
              kernel=kernel, degree=degree,
              gamma=gamma, a=a,
              cmap=plt.cm.autumn, alpha=1)

ax.scatter(X0, X1, c=y_train, cmap=plt.cm.autumn, s=20, edgecolors='k', marker='s')

ax.set_ylabel(r'$x_2\rightarrow$')
ax.set_xlabel(r'$x_1\rightarrow$')
ax.set_title('Decision surface for kernel perceptron for dataset 1')

plt.show()
```


Decision surface for kernel perceptron for dataset 1
120
110
100
90
80
70
$x_2 \rightarrow$
−5
0
5
10
15
20
25
30
35
$x_1 \rightarrow$

kesimpulan

Model Kernel Perception yang digunakan pada dataset opp menghasilkan nilai akurasi sebesar 0.6666666666666666 pada Train Accuracy dan 0.5625 pada Test Accuracy. Lalu pada model SVM Classification menghasilkan nilai akurasi sebesar 55% pada Train Accuracy dan 57,14%. Keakuratan model pada kumpulan data pengujian sedikit lebih tinggi dibandingkan pada kumpulan data pelatihan sehingga menunjukkan bahwa model tersebut mungkin tidak secara efektif menangkap pola dasar data.