DATA SCIENCE

KLASIFIKASI SUPPORT VECTOR MACHINE

OLEH :

NADHIFA SOFIA

(19/448721/PPA/05804)

MAGISTER ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

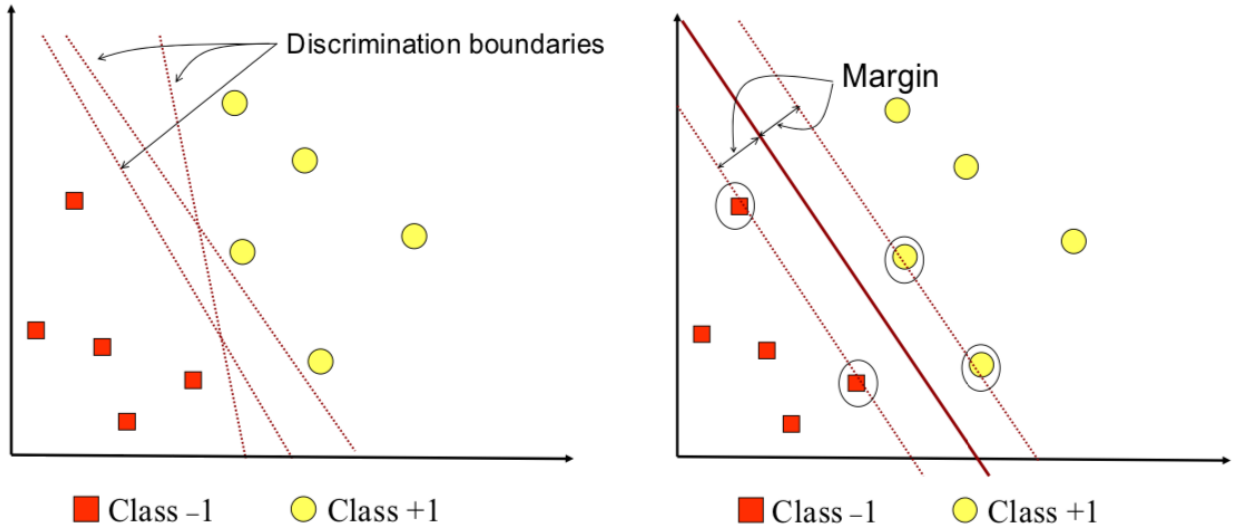
UNIVERSITAS GADJAH MADA

2020

Pendahuluan

Support Vector Machine (SVM) adalah salah satu metode supervised learning dalam machine learning yang sudah diberi pelabelan pada datanya. SVM merupakan suatu teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi [1]. Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari hyperplane terbaik yang berfungsi sebagai pemisah dua buah class pada input space. Gambar 1a memperlihatkan beberapa pattern yang merupakan anggota dari dua buah class : +1 dan –1. Pattern yang tergabung pada class –1 disimbolkan dengan warna merah (kotak), sedangkan pattern pada class +1, disimbolkan dengan warna kuning(lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis (hyperplane) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada gambar 1-a.

Hyperplane pemisah terbaik antara kedua class dapat ditemukan dengan mengukur *margin* hyperplane tsb. dan mencari titik maksimalnya. Margin adalah jarak antara hyperplane tersebut dengan pattern terdekat dari masing-masing class. Pattern yang paling dekat ini disebut sebagai *support vector*. Garis solid pada gambar 1-b menunjukkan hyperplane yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua class, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah support vector. Usaha untuk mencari lokasi hyperplane ini merupakan inti dari proses pembelajaran pada SVM [2].



(a) **(b)**

Gambar 1. SVM mencari hyperplane optimum untuk memisahkan class-1 dan class+1

Menurut [1] hyperplane klasifikasi linier SVM dinotasikan :

$$f(x) = \boldsymbol{w}^T \boldsymbol{x} + b$$

sehingga menurut [3] diperoleh persamaan

$$[(\boldsymbol{w}^T . \boldsymbol{x}_i) + b] \geq 1 \text{ untuk } y_i = +1$$

$$[(\boldsymbol{w}^T . \boldsymbol{x}_i) + b] \leq -1 \text{ untuk } y_i = -1$$

dengan, ***xi*** = himpunan data training, i = 1,2,...n dan yi = label kelas dari $\boldsymbol{x_i}$

Untuk mendapatkan hyperplane terbaik adalah dengan mencari hyperplane yang terletak di tengah-tengah antara dua bidang pembatas kelas dan untuk mendapatkan hyperplane terbaik itu, sama dengan memaksimalkan margin atau jarak antara dua set objek dari kelas yang berbeda [1]. Margin dapat dihitung dengan $\frac{2}{||w||}$.

Mencari hyperplane terbaik dapat digunakan metode *Quadratic Programming Problem (QP)* yaitu meminimalkan $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$

dengan syarat $y_i(\boldsymbol{w}^T . \boldsymbol{x}_i + b) \geq 1, \quad i = 1,2,3, \ldots n$

Solusi untuk mengoptimasi oleh [4] diselesaikan dengan menggunakan fungsi Lagrange sebagai berikut

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \{y_i[(\boldsymbol{w}^T . \boldsymbol{x}_i) + b] - 1\} \qquad (1)$$

dengan αi = pengganda fungsi Lagrange dan *i* =1, 2, …, n

Nilai optimal dapat dihitung dengan memaksimalkan L terhadap αi, dan meminimalkan L terhadap dan b. Hal ini seperti kasus dual problem

$$\max_\alpha W(\alpha) = \max_\alpha \left(\min_{w,b} L(w, b, \alpha)\right)$$

Nilai minimum dari fungsi lagrange tersebut diberikan oleh

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$
$$\frac{\partial L}{\partial w} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i y_i \quad (2)$$

Untuk menyederhanakannya persamaan (1) harus ditransformasikan ke dalam fungsi Lagrange Multiplier itu sendiri, sehingga menurut [1] persamaan (1) menjadi

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i (\boldsymbol{w}^T . \boldsymbol{x}_i) - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \quad (3)$$

Berdasarkan persamaan (2), maka persamaan (3) oleh [3] menjadi sebagai berikut

$$\alpha_i \geq 0, \quad i = 1,2,\dots,n \quad \text{dan} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

dan diperoleh dual problem

$$L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\, \alpha_j y_i\, y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

dengan batasan, $\max_\alpha L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\, \alpha_j y_i\, y_j x_i^T x_j$

Data training dengan terletak pada hyperplane disebut support vector. Data training yang tidak terletak pada hyperplane tersebut mempunyai . Setelah solusi permasalahan quadratic programming ditemukan (nilai αi), maka kelas dari data yang akan diprediksi atau data testing dapat ditentukan berdasarkan nilai fungsi berikut;

$$f(x_t) = \sum_{s=1}^{ns} \alpha_s y_s \boldsymbol{x}_s . \boldsymbol{x}_t + b$$

dengan

xt = data yang akan diprediksi kelasnya (data testing)

xs = data support vector, s = 1,2, ...ns

ns = banyak data support vector

Pada kasus linier non-separable beberapa data mungkin tidak bisa dikelompokkan secara benar atau terjadi misclassification [4]. Sehingga persamaan dimodifikasi dengan menambahkan variabel slack $\xi_i \geq 0$. Variabel slack ini merupakan sebuah ukuran kesalahan klasifikasi [4]. Berikut ini adalah pembatas yang sudah dimodifikasi oleh [5] untuk kasus non-separable:

$$y_i[(\boldsymbol{w}^T . \boldsymbol{x}_i) + b] \geq 1 - \xi_i, \quad i = 1,2,\dots,n$$

Sehingga persamaan (2.6) menjadi meminimalkan

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i$$

dengan $\xi_i \geq 0, i = 1,2,\ldots n$

Banyak teknik data mining atau machine learning yang dikembangkan dengan asumsi kelinieran, sehingga algoritma yang dihasilkan terbatas untuk kasus-kasus yang linier (Santosa, 2007). SVM dapat bekerja pada data non-linier dengan menggunakan pendekatan kernel pada fitur data awal himpunan data. Fungsi kernel yang digunakan untuk memetakan dimensi awal (dimensi yang lebih rendah) himpunan data ke dimensi baru (dimensi yang relatif lebih tinggi). Menurut Prasetyo (2012) macam fungsi kernel diantaranya:

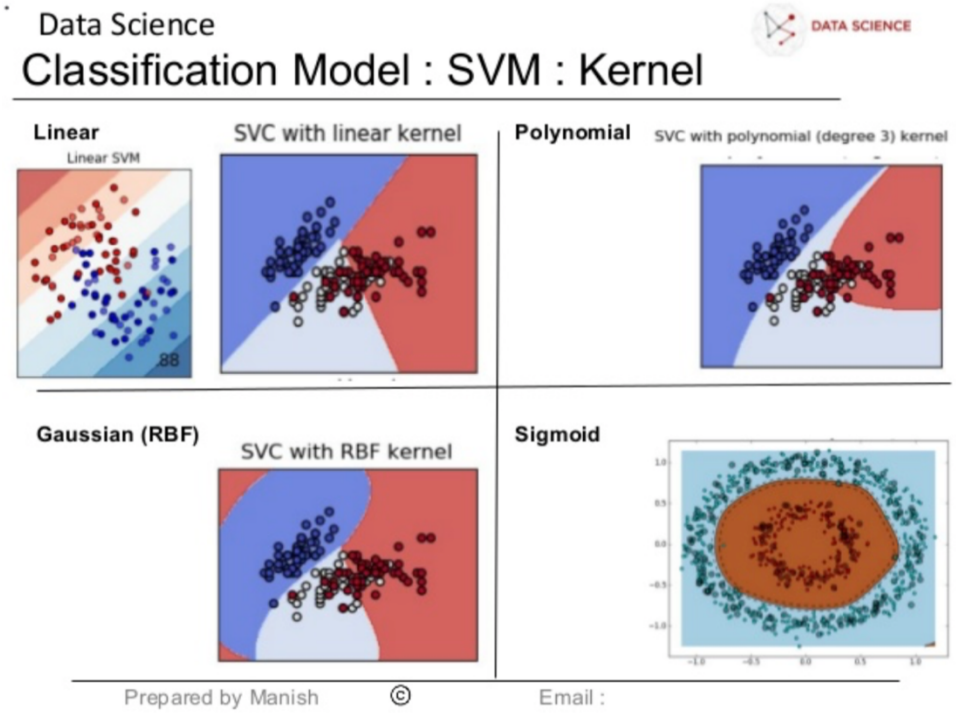
1. Kernel *Gaussian Radial Basic Function (RBF)*

$$\boldsymbol{K}\left(x_i, x_j\right) = exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

2. Kernel *Polynomial*

$$\boldsymbol{K}\left(x_i, x_j\right) = \left(\left(x_i . x_j\right) + c\right)^d$$

x_i dan x_j adalah pasangan dua data training. Parameter σ, c, d > 0 merupakan konstanta. Fungsi kernel harus digunakan untuk substitusi *dot product* di feature space sangat tergantung pada data karena fungsi kernel ini akan menentukan fitur baru di mana hyperplane akan dicari [1].



Gambar 2. Macam-macam kernel SVM

Implementasi

Untuk klasifikasi SVM ini menggunakan beberapa library Python seperti

- **numpy** untuk komputasi array dengan Python, penggunaan ilmiah, numpy juga dapat digunakan sebagai wadah multi-dimensi data generik yang efisien (https://pypi.org/project/numpy/)
- **matplotlib** untuk membuat visualisasi statis, animasi, dan interaktif dengan Python (https://pypi.org/project/matplotlib/)
- **pandas** untuk melakukan analisis data dalam Python, bekerja dengan data terstruktur (tabular, multidimensi, berpotensi heterogen) dan deret waktu mudah dan intuitif (https://pypi.org/project/pandas/)

```
# Classification template

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Dataset yang digunakan pada klasifikasi SVM ini adalah mengenai iklan jejaring social dimana terdapat informasi mengenai user_id, gender, age, estimated_salary, dan purchase. Untuk variable X sendiri menggunakan variable umur dan estimasi gaji seperti pada gambar 2.

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Gambar 3. Sampel dataset untuk klasifikasi SVM

Karena SVM ini termasuk supervised learning dalam machine learning, maka kita akan bagi dataset tersebut menjadi 2 jenis; training dan testing

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Selanjutnya adalah langkah *data preprocessing* yang diterapkan pada variabel independen atau fitur data. Ini pada dasarnya membantu untuk menormalkan data dalam rentang tertentu. Terkadang, ini juga membantu mempercepat perhitungan dalam suatu algoritma.

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Langkah berikutnya adalah untuk membuat classifier pada model SVM. Kernel function berguna untuk mengambil input 1 dimensi lalu ditransfer ke output 2 dimensi yang membuatnya sangat mudah untuk menggambar garis antara dua set data. Menentukan jenis kernel yang akan digunakan dalam algoritma. Itu harus salah satu dari 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' atau callable. Jika tidak ada yang diberikan, 'rbf' akan digunakan. Jika callable diberikan, ini digunakan untuk melakukan pre-compute matriks kernel dari matriks data; matriks itu harus berupa array bentuk (n_samples, n_samples).

Gamma merupakan koefisien kernel untuk 'rbf', 'poly' dan 'sigmoid'. Jika gamma = 'skala' (default) dilewatkan maka ia menggunakan 1 / (n_features * X.var ()) sebagai nilai gamma, jika 'otomatis', gunakan 1 / n_fitur.

C merupakan parameter pengaturan. Kekuatan regularisasi berbanding terbalik dengan C. Harus bernilai positif dan nilai defaultnya adalah 1.0

```
# Fitting classifier to the Training set
# Create your classifier here
from sklearn.svm import SVC
classifier = SVC(kernel='linear', C=1, gamma=1)
classifier.fit(X_train, y_train)
```

Lalu, dibuat fungsi untuk memprediksi data testing X dengan

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

Setelah itu membuat confusion matrix untuk mengevaluasi keakuratan klasifikasi

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Pada training set, dicari sebuah hyperplane yang memisahkan pola secara linear. Untuk pemisahan pola yang non-linier menggunakan penambahan fungsi kernel. Disini menggunakan fungsi meshgrid dengan margin -1 dan +1. Lalu dibuat untuk kelas pertama dengan warna

kuning serta kelas kedua dengan warna biru. Klasifikasi pada training set ini direpresentasikan dengan scatter plot. Begitu juga dengan dataset testing.

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('yellow', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('yellow', 'blue'))(i), label = j)
plt.title('Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('yellow', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('yellow', 'blue'))(i), label = j)
plt.title('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

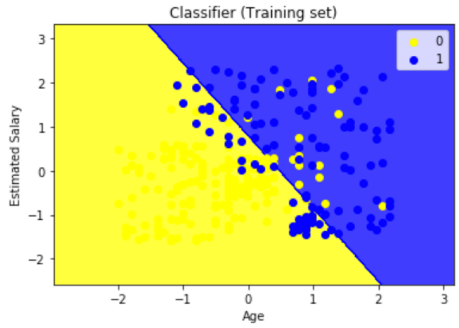
Jika dataset berada dalam ruang n-dimensi, direpresentasikan sebagai matriks X, asumsikan kernel sebagai interpolasi, trik untuk memetakannya dalam ruang dimensi yang lebih rendah Kemudian fungsi kernel yang berbeda dikalkulasikan dengan cara ini:

$$\text{linear: } z = y * x + b$$

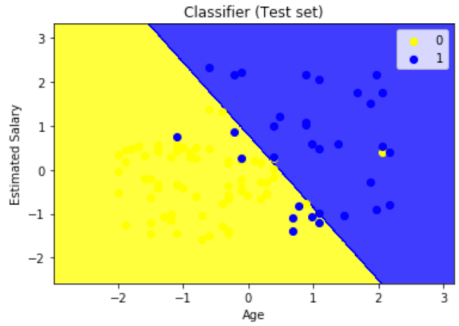
$$\text{polinominal: } z = (y * x + b)^2$$

$$\text{rbf: } z = \exp(-\gamma \| \chi - y \|^2)$$

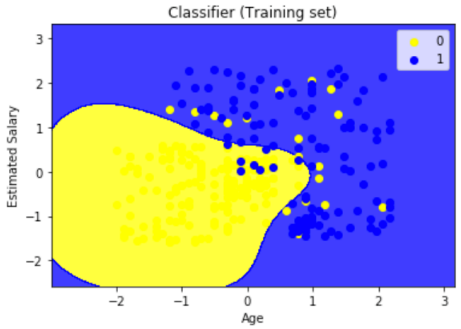
Kernel : Linear



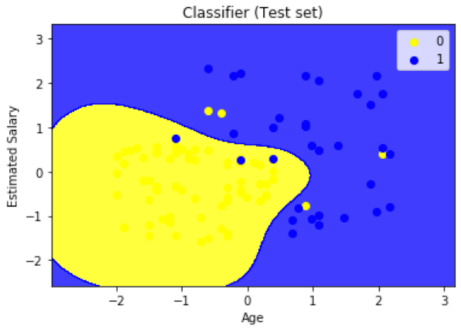

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
```



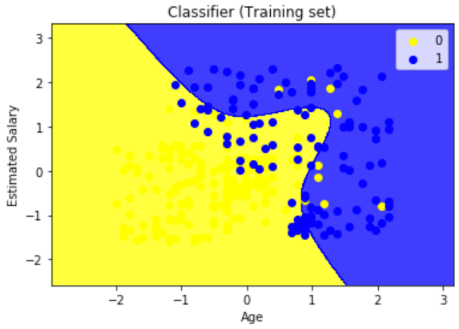
Kernel : ‘RBF’



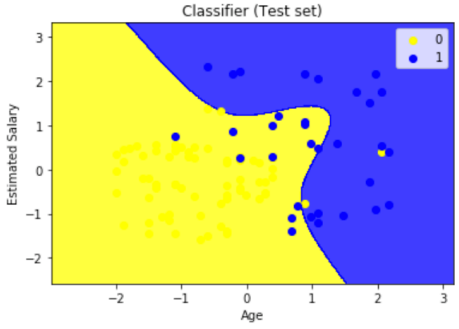

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
```



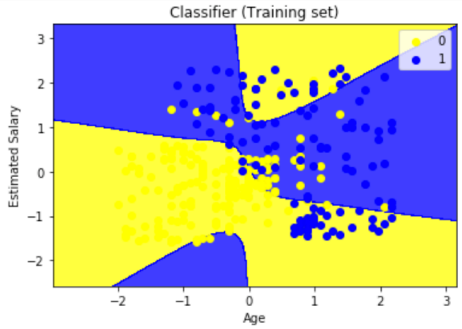
Kernel : Polynomial



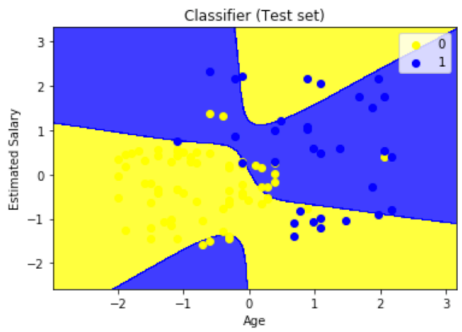

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
```



Kernel : Sigmoid




```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pr
ecedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to
specify the same RGB or RGBA value for all points.
```



Referensi

[1] Santosa, B., 2007, Data Mining Teknik Pemanfaatan Data untuk Keperluan Bisnis, Graha Ilmu : Yogyakarta.

[2] Nugroho, A. S., Witarto, A. B., dan Handoko, D., 2003, Support Vector Machine: Teori dan Aplikasinya dalam Bioinformatika, *Prosiding Indonesian Scientific Meeting in Central Japan..*

[3] Vapnik, V., dan Cortes, C, 1995, *Support Vector Networks*. *Machine Learning*, 20, 273-297.

[4] Novianti, F.A., dan Purnami, S.W., 2012, *Analisis Diagnosis Pasien Kanker Payudara Menggunakan Regresi Logistik dan Support Vector Machine (SVM) Berdasarkan Hasil Mamograf, Jurnal Sains dan Seni ITS*, Vol. 1, No. 1 ISSN : 2301-928X.

[5] Hsu, C.W., dan Lin, C.J., 2002, A Comparison of Methods for Multi-class Support Vector Machines, *IEEE Transaction on Neural Network*, 13(2) : 415- 425.