PENGENALAN POLA
“Klasifikasi menggunakan SVM (Support Vector Machine)”

TAUFIK HIDAYAT
F 551 19 158

PROGRAM STUDI S1 TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS TADULAKO
2022

A. Tujuan

Aplikasi yang dapat Mengklasifikasikan antara Buah **jeruk biasa / Orange** dan **jeruk bali / Grapefruit** dengan Machine Learning menggunakan algoritma Supervised Learning yaitu SVM (Support Vector Machine) memakai bahasa pemrograman Python.

Adapun Pembahasannya dibagi menjadi beberapa point:

1. Pengantar Machine Learning
2. Pengantar SVM
3. Kernel trick
4. Dataset deskripsi
5. import library
6. import dataset
7. Exploratory data analysis
8. Label Encoding
9. Deklarasikan vektor fitur dan variabel target
10. Split data menjadi Data Train dan Data Test
11. Feature scaling
12. Jalankan SVM dengan kernel RBF
13. Confusion Matrix
14. Classification Report
15. Hasil Visualisasi klasifikasi Model SVM

B. Teori Dasar

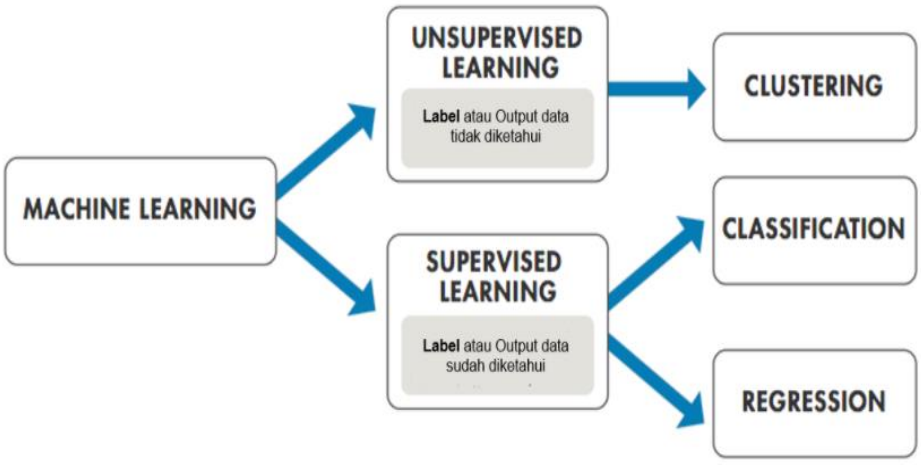
Machine Learning bermula diperkenalkan oleh Arthur Samuel pada tahun 1959, menurut Arthur Samuel

jika diterjemahkan “Pembelajaran Mesin adalah bidang studi yang memberikan komputer kemampuan untuk belajar tanpa diprogram secara eksplisit ”. Namun sudah ada beberapa pendapat yang mendefinisikan istilah machine learning :

Dalam pembuatan model machine learning tentunya dibutuhkan data. Sekumpulan data yang digunakan dalam machine learning disebut **DATASET**, yang kemudian dibagi/di-split menjadi training dataset dan test dataset.

TRAINING DATASET digunakan untuk membuat/melatih model machine learning, sedangkan **TEST DATASET** digunakan untuk menguji performa/akurasi dari model yang telah dilatih/di-training.

Machine Learning itu terbagi menjadi 2 tipe yaitu **Supervised Learning dan Unsupervised Learning**. Jika LABEL/CLASS dari dataset sudah diketahui maka dikategorikan sebagai supervised learning, dan jika Label belum diketahui maka dikategorikan sebagai unsupervised learning



Contoh algoritma Supervised Learning dan Unsupervised Learning :

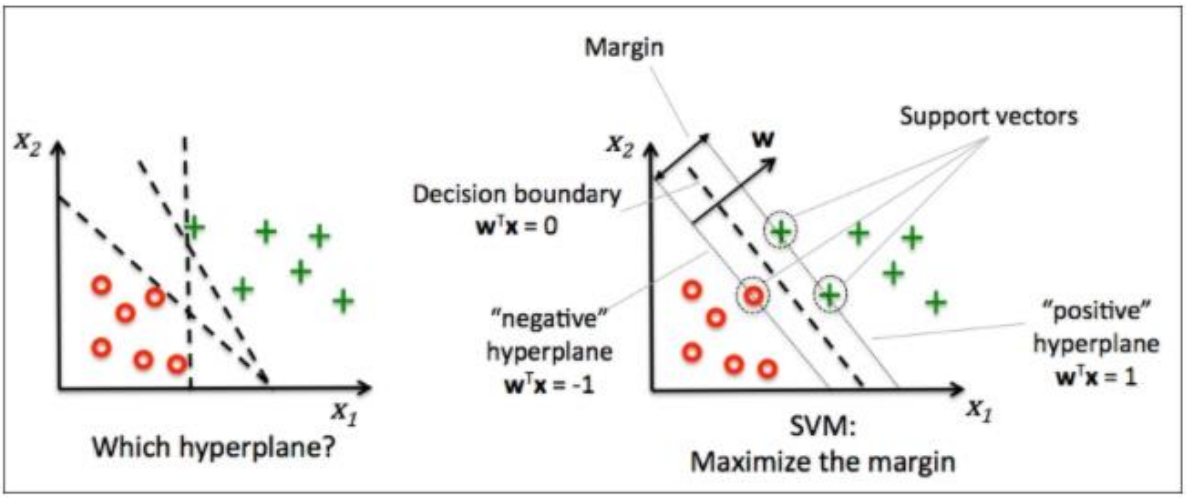
- **Supervised Learning :** Linear Regression, Naive Bayes, Decision Tree, SVM.
- **Unsupervised Learning :** K-Means, Hierarchical clustering, DBSCAN, dan Fuzzy C-Means.

Support Vector Machines (singkatnya SVM) adalah algoritma pembelajaran mesin yang digunakan untuk tujuan klasifikasi dan regresi. SVM adalah salah satu algoritma pembelajaran mesin yang kuat untuk tujuan klasifikasi, regresi, dan deteksi outlier/pencilan. Pengklasifikasi SVM membuat model yang menetapkan titik data baru ke salah satu kategori yang diberikan. Dengan demikian, ini dapat dipandang sebagai pengklasifikasi linear biner non-probabilistik.

SVM dapat digunakan untuk tujuan klasifikasi linier. Selain melakukan klasifikasi linear, SVM dapat secara efisien melakukan klasifikasi non-linier menggunakan **trik kernel**. Ini memungkinkan kita untuk secara implisit memetakan input ke dalam ruang fitur berdimensi tinggi.

terdapat beberapa terminologi SVM yang harus kita ketahui, ini merupakan hal yang sangat penting, karena dasar untuk pemahaman model SVM, perhatikan bagian berikut :

- **Support Vector** : merupakan titik data (data point) yang paling dekat dengan batas (boundary line). Titik-titik data ini akan menentukan garis pemisah atau hyperplane dengan lebih baik dengan menghitung margin.
- **Hyperplane** : garis pemisah antara class. Pengklasifikasi SVM memisahkan titik data menggunakan bidang-hiper dengan jumlah margin maksimum
- **Boundary line/margin** : dua buah garis di antara hyperplane yang membentuk margin menyinggung titik data (data point).



bentuk persamaan umum dari gambar di atas, secara sederhana dapat dituliskan sebagai persamaan linear atau fungsi berikut :

y = wx + b atau ***f(x) = (w, x) + b***

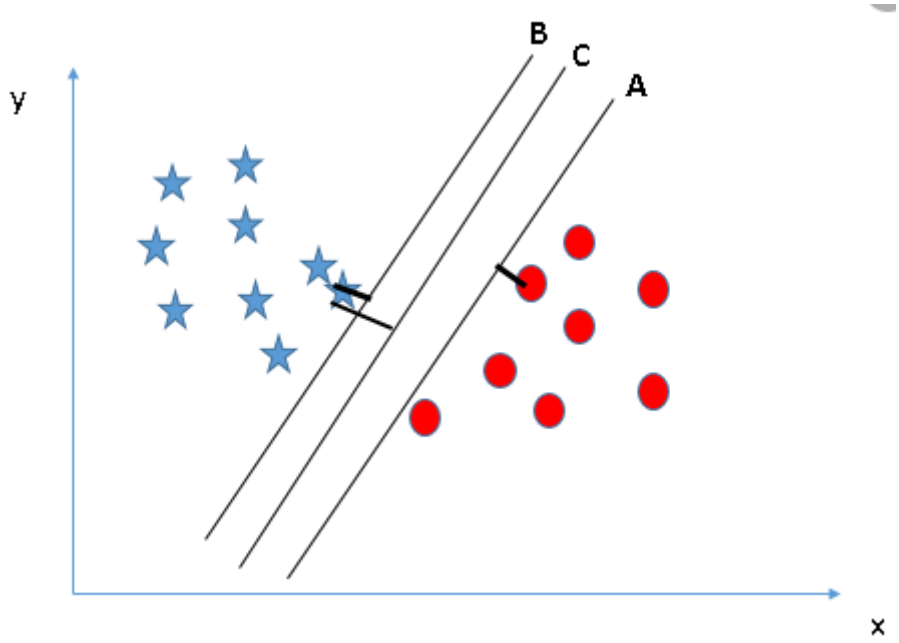
Kernel Trick

Dalam prakteknya, algoritma SVM diimplementasikan dengan menggunakan kernel. yang disebut teknik **trik kernel**. Dengan kata sederhana, kernel hanyalah fungsi yang memetakan data ke dimensi yang lebih tinggi dimana data dapat dipisahkan. Kernel mengubah ruang data masukan berdimensi rendah menjadi ruang berdimensi lebih tinggi. Jadi, ini mengubah masalah yang dapat dipisahkan non-linier menjadi masalah yang dapat dipisahkan linier dengan menambahkan lebih banyak dimensi ke dalamnya. Jadi, trik kernel membantu kita membuat pengklasifikasi yang lebih akurat. Karenanya, ini berguna dalam masalah pemisahan non-linier. function kernel dapat didefinisikan sebagai berikut :

$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \le 1 \\ 0 & \text{otherwise} \end{matrix}$$

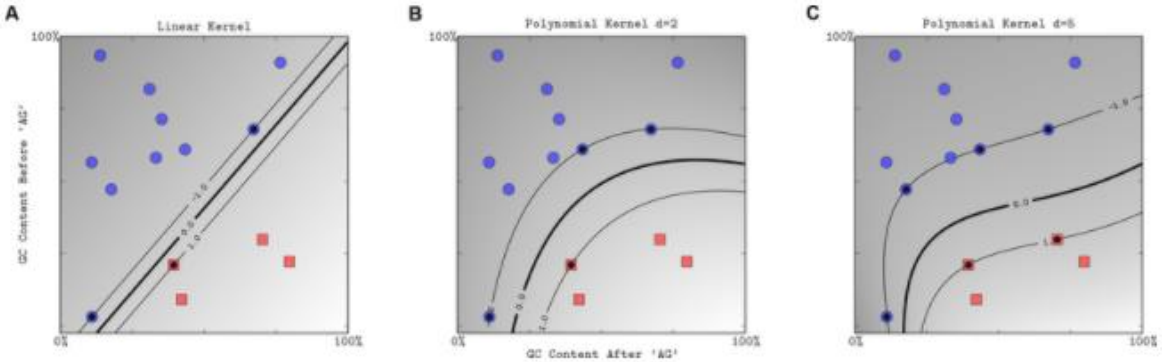
Dalam konteks SVM, terdapat 4 kernel populer, : **Linear kernel,Polynomial kernel dan Radial Basis Function (RBF) kernel (Gaussian kernel)**

Kernel linier — dapat digunakan sebagai perkalian titik normal antara dua pengamatan yang diberikan. Hasil perkalian antara dua vektor adalah hasil perkalian setiap pasang nilai masukan. Berikut adalah persamaan kernel linier. **linear kernel : K(xi , xj) = xiT xj**



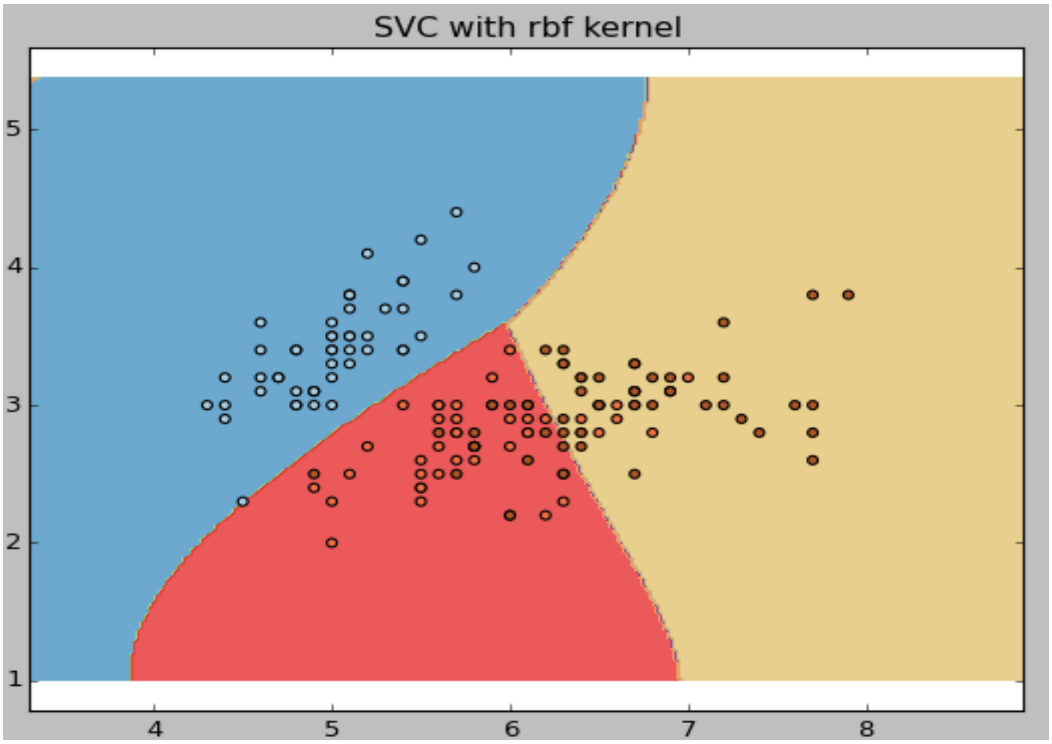
Polynomial Kernel — Ini adalah bentuk kernel linier yang agak umum. dapat membedakan ruang input melengkung atau nonlinier. Berikut adalah persamaan kernel polinomial.

Polynomial kernel : K(xi , xj) = (γxiT xj + r)d , γ > 0



Kernel RBF/Gaussian— Kernel fungsi basis radial biasanya digunakan dalam klasifikasi SVM, dapat memetakan ruang dalam dimensi tak terbatas. Berikut adalah persamaan kernel RBF.

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$



Dalam konten kali ini saya menggunakan **RBF/ Gaussian Kernel** dalam penyelesaianya

Deskripsi Dataset

saya disini menggunakan dataset yang saya ambil dari Kaggle, yaitu " **Orange vs Grapefruit"**. Tugas untuk memisahkan jeruk dan jeruk bali cukup jelas bagi rata — rata manusia, tetapi biasanya dengan pengamatan manual pun masih ada sedikit kesalahan atau sulit membedakan antara jeruk dan jeruk bali. Kumpulan data kali ini mengambil **'colors : RGB'**,**'weight'**, dan **'Diameter'** jeruk dan jeruk bali. Menghasilkan kumpulan data yang lebih besar yang berisi berbagai macam nilai, yaitu "jeruk" dan "jeruk bali".

Informasi Atribut:

1. Name
2. Diameter
3. Weight
4. Red
5. Green
6. Blue

C. Praktikum

Import Library

Pertama disini saya memulai dengan meng import Library Python yang umum searing kali di pakai

```
# Mengimpor library
import numpy as np #
import matplotlib.pyplot as plt # untuk visualisasi data
import pandas as pd # data processing, I/O file CSV
import seaborn as sns # untuk statistical Visual
```

Import Dataset

load data yang telah di download di kaggle, lalu kita tampilkan 4 dataset teratas

```
dataset = pd.read_csv('/content/citrus.csv')
dataset.head()
```

	name	diameter	weight	red	green	blue
0	orange	2.96	86.76	172	85	2
1	orange	3.91	88.05	166	78	3
2	orange	4.42	95.17	156	81	2
3	orange	4.47	95.60	163	81	4
4	orange	4.48	95.76	161	72	9

Kita dapat melihat, terdapat 6 variabel dalam dataset tersebut. 5 adalah variabel kontinu dan 1 variabel diskrit. Variabel diskrit adalah variabel "**name**", itu juga merupakan variabel target

Exploratory data analysis

sekarang kita akan melihat dimensi datasetnya :

```
dataset.shape
```

```
(10000, 6)
```

ukuran dimensi dari dataset tersebut terdapat 10000 instance/baris dan 6 variabel/kolom

selanjutnya kita akan melihat tipe data dari setiap variabel tersebut dan melihat apakah ada data yang NA / data kosong. Jika terdapat data yang kosong, maka perlu dilakukan handling missing values.

```
dataset.info()
print("")
dataset.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   name      10000 non-null  object 
 1   diameter  10000 non-null  float64
 2   weight    10000 non-null  float64
 3   red       10000 non-null  int64  
 4   green     10000 non-null  int64  
 5   blue      10000 non-null  int64  
dtypes: float64(2), int64(3), object(1)
memory usage: 468.9+ KB

name        0
diameter    0
weight      0
red         0
green       0
blue        0
dtype: int64
```

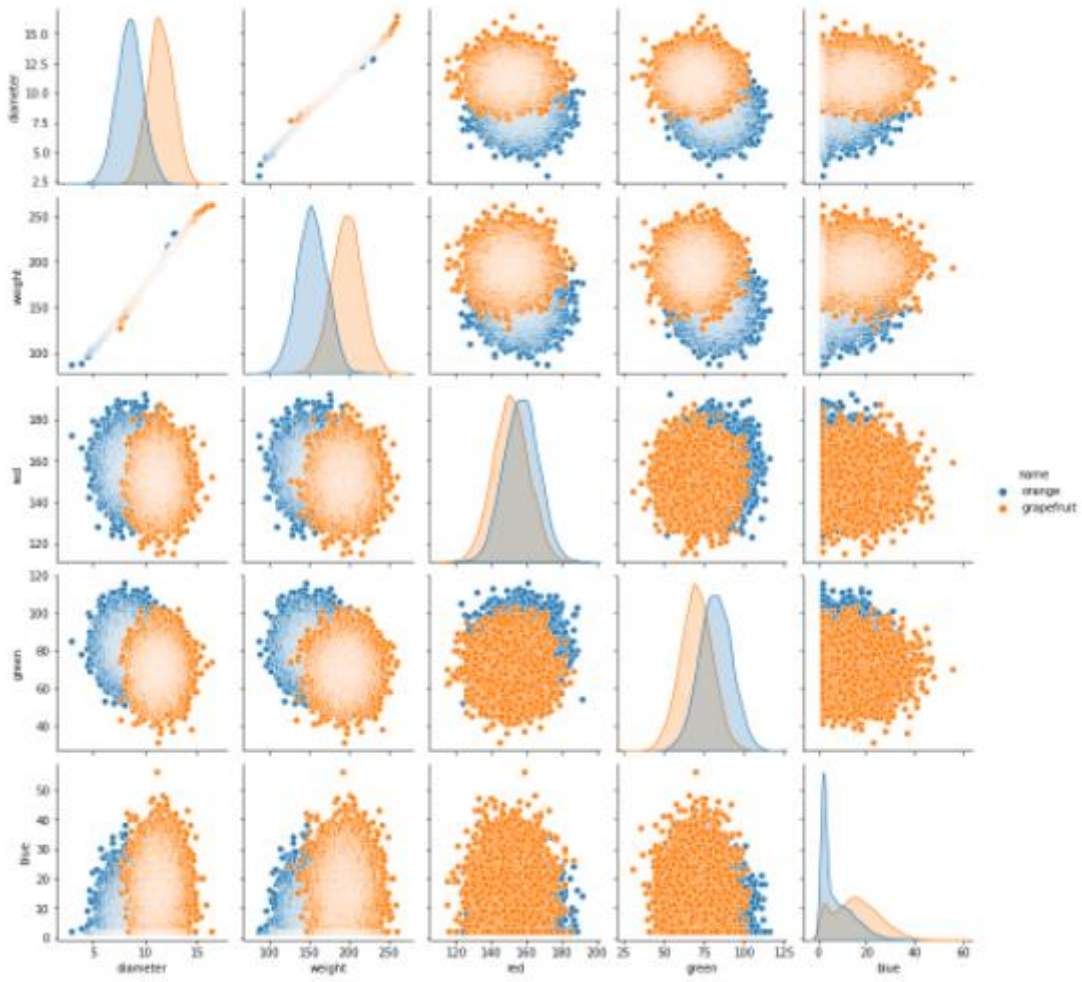
Dapat kita lihat, bahwasanya tidak terdapat data yang NA/kosong, di setiap variabelnya terisi semua, dan untuk tipe datanya terdapat 2 tipe data float, 3 tipe data integer dan 1 objek/string.

Ringkasan :

- Terdapat 6 variabel
- 5 merupakan variabel kontinu dan 1 merupakan variabel diskrit/objek
- variabel diskrit terdapat pada kolom “name”, sekaligus merupakan target variabel
- Tidak terdapat missing value

sekarang kita akan melihat bentuk persebaran datanya dengan visualisasi.

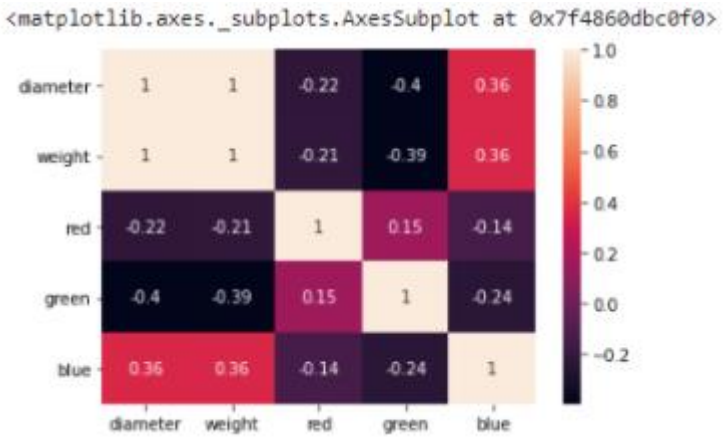
```
sns.pairplot(data=dataset, hue = 'name', kind='scatter')
```



Dapat dilihat dari hasil visualisasinya, warna biru merupakan perwakilan dari “orange”/jeruk dan warna oren perwakilan dari “grapefruit”/jeruk bali. Hasil visualisasi tersebut membandingkan antara setiap variabelnya, yang dimana terdapat 5 variabel bertipe data numerik. Pada dasarnya dataset tersebut sudah baik sekali, bahkan hanya dilihat seperti biasa pun kita dapat mengetahui persebaran datanya, seakan — akan terdapat garis pemisah antara biru dan oren. Pada model SVM nanti akan dibuat sebuah hyperplane atau titik terjauh yang akan memisahkan antara “Orange” dan “Grapefruit”.

berikutnya kita akan melihat korelasi antara variabelnya :

```
sns.heatmap(dataset[["diameter",'weight', 'red','green', 'blue']].corr(), annot=True)
```



Dapat diketahui bahwasanya jika hasil korelasi mendekati **nilai 1**, maka **korelasi tersebut dikatakan baik,** namun jika mendekati **nilai -1** maka korelasi **dikatakan buruk**. Dapat dilihat dari hasil visualisasi di atas, yang memiliki nilai korelasi baik terdapat pada variabel "diameter" dan "weight" yaitu memiliki nilai hasil korelasinya 1, adapun nilai korelasi yang buruk terdapat antara variabel "green" dan "diameter", yaitu memiliki nilai -0,39, jika hasil korelasi yang berbentuk diagonal menyamping tersebut itu merupakan korelasi variabel dengan dirinya sendiri.

Label Encoding

Sebelum menuju ke langkah split data, terlebih dahulu kita konversi nilai kategori yang bertipe data objek menjadi bentuk tipe data numerik, yaitu menjadi **1** dan **0**, dimana, nilai **1 perwakilan dari "Orange"** dan **0 perwakilan dari "Grapefruit"** :

```
from sklearn import preprocessinglabel_encoder =
preprocessing.LabelEncoder()dataset['name']=
label_encoder.fit_transform(dataset['name'])dataset['name'].unique()
```

```
array([1, 0])
```

Deklarasikan vektor fitur dan variabel target

Disini data akan di split, yang memisahkan antara variabel input dan variabel output/target, untuk lebih jelasnya perhatikan program berikut

```
X = dataset.iloc[:, 1:6].values
y = dataset.iloc[:, 0].values
```

X = merupakan variabel input, yaitu : “diameter”, “weight”, “red”, “green”, “blue”. **y** = merupakan variabel output/target, : “name”.

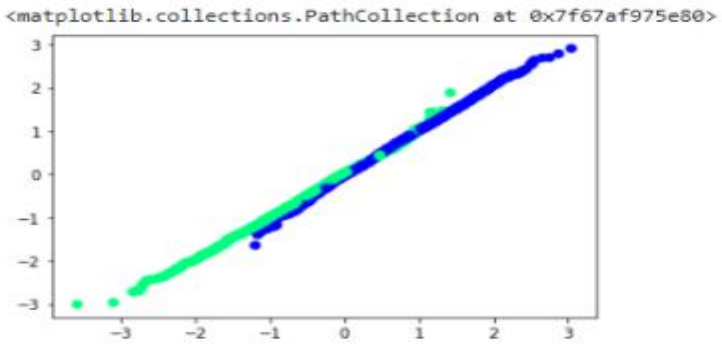
Split data menjadi Data Train dan Data Test

setelah itu kita split lagi menjadi data Training set dan Test set. Untuk perbandingannya disini saya menggunakan 75% : 25%, 80% untuk Training set dan 20% untuk Test set.

```
from sklearn.model_selection import train_test_splitX_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

sekarang kita coba visualisasikan Training set nya :

```
plt.scatter(X_train[:,0], X_train[:, 1], c=y_train, cmap='winter')
```



pada hasil visualisasi tersebut, nanti akan kita mencoba untuk mengklasifikasikan menggunakan algoritma SVM menggunakan kernel RBF. hasilnya nanti akan kelihatan terdapat garis yang memisah antara warna hijau dan biru.

Fitur Scaling

Fitur scaling adalah class dari sklearn yang digunakan untuk normalisasi data agar simpangan datanya tidak terlalu besar. Hal ini akan kita terapkan ke data training dan data testing

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

Menerapkan Algoritma SVM menggunakan kernel RBF

Sebelum menerapkan modelnya, pertama kita akan mengimpor modul SVM dari sklearn untuk membuat pengklasifikasi vektor dukungan di svc (). Dengan meneruskan kernel argumen sebagai kernel RBF dan parameter regulasi (C) nya adalah 100, parameter C tersebut digunakan untuk menghindari overfitting dengan memilih fungsi yang sesuai dengan data. setelah itu kita terapkan modelnya ke data training

```
# Membuat model SVM terhadap Training setfrom sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state=0, C=100)
classifier.fit(X_train, y_train)
```

```
SVC(C=100, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
    verbose=False)
```

Tanda di atas merupakan notif bahwa kita telah berhasil menerapkan Algoritma SVM ke model, selanjutnya kita prediksi hasil akurasi dari model yang telah kita buat untuk diterapkan ke data testing.

```
from sklearn import metricsy_pred = classifier.predict(X_test)
print("hasil akurasi :", metrics.accuracy_score(y_test,y_pred))
```

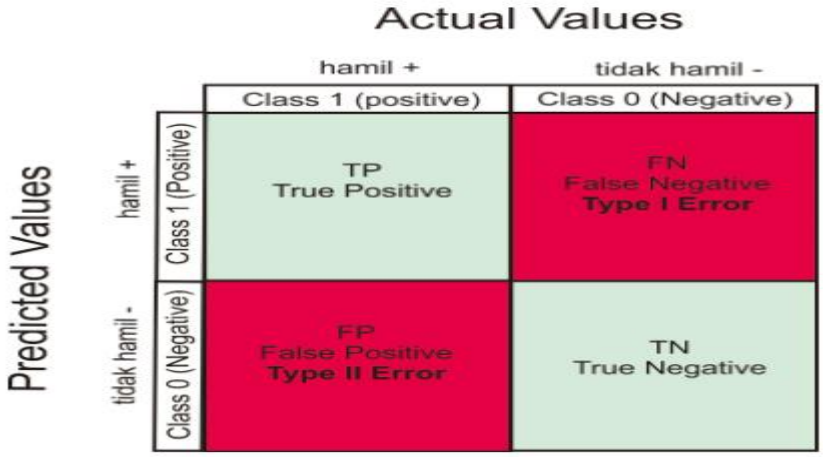
```
hasil akurasi : 0.9636
```

Dapat dilihat hasil akurasi dari model yang telah kita terapkan ke data testing sangat baik sekali, yaitu memiliki nilai akurasi 0.9636. Skor akurasi dikatakan bagus jika rata — ratanya di atas 0.7

Confusion Matrix

Sekarang kita akan melihat hasil klasifikasi dengan bentuk tabel matrik dengan Confusion Matrix. Sebelum lanjut, kita pahami terlebih dahulu tentang pengertian, kegunaan dan cara membaca tabel dari Confusion Matrix, karena ini sedikit rumit untuk memahaminya. Jadi apa sih Confusion Matrix itu ?, Confusion matrix sering juga disebut error matrix. Pada dasarnya confusion matrix memberikan informasi hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. Confusion matrix berbentuk tabel yang digunakan untuk

mendeskripsikan performa dari sebuah model classification pada sebuah set dari data test dimana true values diketahui. Dimana tabel tersebut memiliki 4 kombinasi, yaitu kombinasi nilai prediksi dan nilai aktual yang berbeda. perhatikan gambar di bawah ini



Mari kita lihat tabel di atas, terdapat ACTUAL VALUES (data sebenarnya) sebagai kolom dan PREDICTED VALUES (data hasil prediksi) sebagai baris. Ada 2 class, yaitu class **Positive** atau **1** dan **Negatif** atau **0.** Lalu disitu terdapat istilah TP, TN, FP, FN. Untuk lebih jelasnya perhatikan point berikut.

- **TP (True Positive)** : hasil positif yang diprediksi dibenarkan. Contoh orang itu hamil (class 1), dari hasil prediksi orang tersebut memang benar hamil (class 1)
- **TN (True False)** : hasil negatif yang diprediksi dibenarkan. Contoh Orang itu tidak hamil (class 0), dari hasil prediksi orang tersebut memang tidak hamil (class 0)
- **FP (False Positive) — Type I error** : hasil negatif tetapi dianggap sebagai data positif. Contoh, Orang itu tidak hamil (class 0), dari hasil prediksi, orang tersebut dianggap hamil (class 1).
- **FN (False Negative) — Type II error** : hasil positif tetapi dianggap sebagai data negatif. Contoh, Orang itu hamil (class 1), dari hasil prediksi, orang tersebut dianggap tidak hamil (class 0).

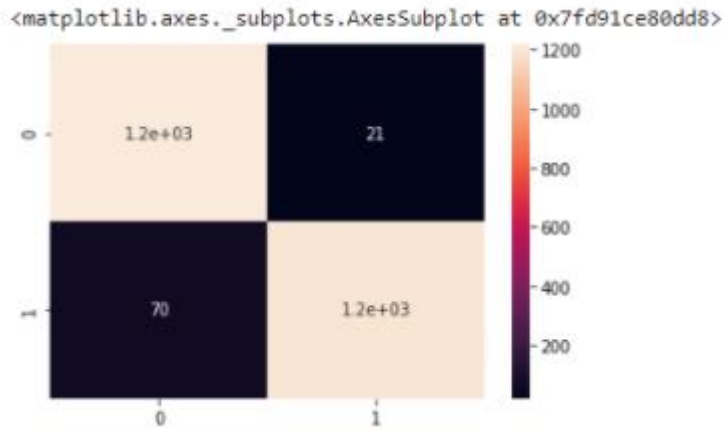
Seperti itu kurang lebih penjelasan tentang Confusion Matrix, sekarang kita lanjut ke koding, bagaimana hasil klasifikasi dari model yang telah kita buat di atas tadi dengan menggunakan Confusion matrix. Pertama yang kita butuhkan import library confusion matrix dari sklearn, lalu kita terapkan fungsi confusion matrix ke data testing dan prediksi.

```
from sklearn.metrics import confusion_matrixcm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1216   21]
 [  70 1193]]
```

supaya lebih menarik, kita buat visualisasi menggunakan heatmap

```
sns.heatmap(cm, annot=True)
```



Perlu diingat, **class 0** mewakili "**G rapefruit/Jeruk bali**" dan **class 1** mewakili "**Orange/Jeruk**". dimana sumbu x mewakili **"Actual Values"** dan sumbu y **"Predicted Values**". Sekarang kita jabarkan.

- TP : Dimana status aslinya adalah **"Grapefruit/jeruk bali" (class 0**), ternyata model kita berhasil menyatakan bahwa status tersebut memang benar — benar **"Grapefruit/jeruk bali"(class 0)**. berhasil memprediksi sebanyak **1216**

- TF : Dimana status aslinya adalah **"Orange/jeruk" (class 1**), ternyata model kita berhasil menyatakan bahwa status tersebut memang benar — benar **"Orange/jeruk"(class 1)**. berhasil memprediksi sebanyak **1193**

- **FP :** Status aslinya adalah “Orange/jeruk” (Class 1), tetapi model kita memprediksi bahwa status tersebut adalah **“Grapefruit/jeruk bali” (Class 0)**. dengan nilai prediksi **21**

- FN : Status aslinya adalah **“Grapefruit/jeruk bali”** (Class 0), tetapi model kita memprediksi bahwa status tersebut adalah **“Orange/jeruk” (Class 1)**. dengan nilai prediksi **70**

Kesimpulan : model kita berhasil memprediksi sebanyak 1216 (**Grapefruit/jeruk bali**) dan 1193 (**Orange/jeruk**), total 2409. dan prediksi yang gagal sebanyak 21 **(FP)** + 70 **(FN)** = 91.

Perbandingan model yang **berhasil** dan yang **gagal** adalah 2409 : 91.

Classification Report

Classification Report digunakan untuk mengukur kualitas prediksi dari algoritma klasifikasi. Perhatikan hasil dari Classification Report

```
from sklearn.metrics import classification_report#accuracy
print("acuracy:", metrics.accuracy_score(y_test,y_pred))
#precision score
print("precision:", metrics.precision_score(y_test,y_pred))
#recall score
print("recall" , metrics.recall_score(y_test,y_pred))
print(classification_report(y_test, y_pred))
```

```
acuracy: 0.9636
precision: 0.9827018121911038
recall 0.9445764053840063
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      1237
           1       0.98      0.94      0.96      1263

    accuracy                           0.96      2500
   macro avg       0.96      0.96      0.96      2500
weighted avg       0.96      0.96      0.96      2500
```

Precision — Akurasi prediksi positif.
Recall — Fraksi positif yang diidentifikasi dengan benar.

Skor F1 — adalah rata-rata presisi dan recall harmonik tertimbang sehingga skor terbaik adalah 1.0 dan yang terburuk adalah 0.0.

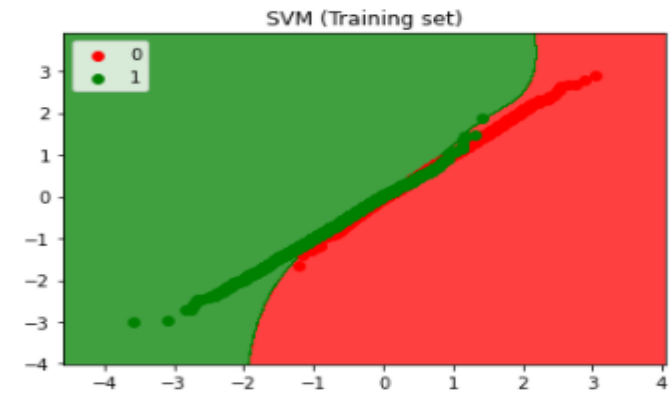
Support — adalah jumlah kemunculan aktual kelas dalam kumpulan data yang ditentukan.

Dari hasil di atas, kita mendapatkan nilai akurasi, presisi, dan recall yaitu 0,96, 0,98, dan 0,94 yang sangat tidak mungkin. Karena kumpulan data yang kita gunakan cukup deskriptif ,sehingga kita dapat memperoleh hasil yang sangat akurat.

Hasil Visualisasi klasifikasi Model SVM

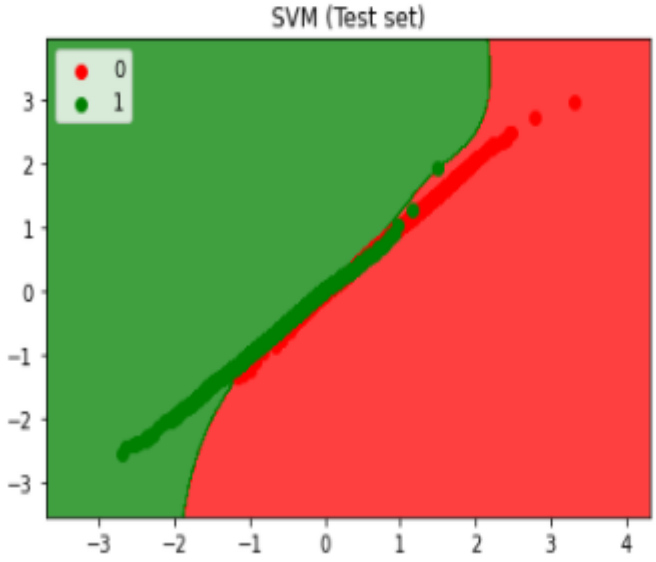
Terakhir, kita akan melihat hasil visualisasi data training dan data test yang sudah kita terapkan model SVM dengan kernel RBF.

```
from matplotlib.colors import ListedColormapX_set, y_set = X_train, y_trainX1, 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))Xpred = np.array([X1.ravel(), X2.ravel()] + [np.repeat(0, X1.ravel().size) for _
in range(3)]).T
pred = classifier.predict(Xpred).reshape(X1.shape)
plt.contourf(X1, X2, pred,
alpha = 0.75, cmap = ListedColormap(('red', 'green')))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(('red', 'green'))(i), label = j)plt.title('SVM (Training set)')
plt.legend()
plt.show()
```



Sekarang kita lihat untuk Test set nya :

```
# Visualisasi model SVM terhadap Test setfrom matplotlib.colors import
ListedColormapX_set, y_set = X_test, y_testX1, 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))Xpred = np.array([X1.ravel(), X2.ravel()] + [np.repeat(0, X1.ravel().size) for _
in range(3)]).Tpred = classifier.predict(Xpred).reshape(X1.shape)plt.contourf(X1, X2,
pred,
alpha = 0.75, cmap = ListedColormap(('red', 'green')))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(('red', 'green'))(i), label = j)plt.title('SVM (Test set)')
plt.legend()
plt.show()
```




```
[[1216   21]
 [  70 1193]]
```

Confusion Matrix

Pada gambar di atas, bisa dilihat bahwa ada 91 dari 2500 titik yang salah dideteksi oleh SVM (akurasi 96%), meskipun titik yang tersebut tidak kelihatan, namun dapat dilihat dari hasil confusion matrix. Artinya bahwa model SVM yang menggunakan kernel RBF/ Gaussian sudah cukup baik untuk membantu mengklasifikasi atau membedakan antara “Orange/jeruk” dan “GrapeFruit/ jeruk bali”

D. Hasil Percobaan

1. Klasifikasi menggunakan SVM (Support Vector Machine) di Python pada (Orange vs Grapefruit)

```python
# Mengimpor library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

dataset = pd.read_csv('C:\Users\taufi\PycharmProjects\UTS_SVM\citrus.csv')
dataset.head()
dataset.shape
dataset.info()
print("")
dataset.isnull().sum()

sns.pairplot(data=dataset, hue='name', kind='scatter')
sns.heatmap(dataset[["diameter",'weight', 'red','green', 'blue']].corr(), annot=True)

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
dataset['name']= label_encoder.fit_transform(dataset['name'])
dataset['name'].unique()

X = dataset.iloc[:, 1:6].values
y = dataset.iloc[:, 0].values

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)
plt.scatter(X_train[:,0], X_train[:, 1], c=y_train, cmap='winter')

# Feature Scaling
from sklearn.preprocessing import StandardScaler
```

```python
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Membuat model SVM terhadap Training set
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state=0, C=100)
classifier.fit(X_train, y_train)

from sklearn import metrics
y_pred = classifier.predict(X_test)
print("hasil akurasi :", metrics.accuracy_score(y_test,y_pred))

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
sns.heatmap(cm, annot=True)

from sklearn.metrics import classification_report
#accuracy
print("acuracy:", metrics.accuracy_score(y_test,y_pred))
#precision score
print("precision:", metrics.precision_score(y_test,y_pred))
#recall score
print("recall", metrics.recall_score(y_test,y_pred))
print(classification_report(y_test, y_pred))

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),
```

```python
Xpred = np.array([X1.ravel(), X2.ravel()] + [np.repeat(0, X1.ravel().size) for _ in range(3)]).T
pred = classifier.predict(Xpred).reshape(X1.shape)
plt.contourf(X1, X2, pred,
alpha_=_0.75, cmap_=_ListedColormap(('red', 'green')))
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(('red', 'green'))(i), label_=_j)
plt.title('SVM (Training set)')
plt.legend()
plt.show()

# Visualisasi model SVM terhadap Test set
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))
Xpred = np.array([X1.ravel(), X2.ravel()] + [np.repeat(0, X1.ravel().size) for _ in range(3)]).T
pred = classifier.predict(Xpred).reshape(X1.shape)
plt.contourf(X1, X2, pred,
alpha_=_0.75, cmap_=_ListedColormap(('red', 'green')))
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(('red', 'green'))(i), label_=_j)
plt.title('SVM (Test set)')
plt.legend()
plt.show()
```

2. Hasil Akurasi dari Klasifikasi menggunakan SVM (Support Vector Machine) di Python pada (Orange vs Grapefruit) adalah sebesar 96,3%

```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      10000 non-null  object
 1   diameter  10000 non-null  float64
 2   weight    10000 non-null  float64
 3   red       10000 non-null  int64
 4   green     10000 non-null  int64
 5   blue      10000 non-null  int64
dtypes: float64(2), int64(3), object(1)
memory usage: 468.9+ KB

hasil akurasi : 0.9636
[[1216   21]
 [  70 1193]]
```

```
acuracy: 0.9636
precision: 0.9827018121911038
recall 0.9445764053840063
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      1237
           1       0.98      0.94      0.96      1263

    accuracy                           0.96      2500
   macro avg       0.96      0.96      0.96      2500
weighted avg       0.96      0.96      0.96      2500
```