Nama : Indira Septianita Larasati

NIM : H071191023

Support Vector Machine (SVM) merupakan salah satu metode dalam *supervised learning* yang digunakan untuk klasifikasi seperti *support vector classification* dan regresi *support vector regression*. Dalam pemodelan klasifikasi, SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. SVM juga dapat mengatasi masalah klasifikasi dan regresi dengan *linear* maupun *non linear*.

SVM digunakan untuk mencari *hyperplane* dengan memaksimalkan jarak antar kelas. H*yperplane* adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas. Dalam 2-D fungsi yang digunakan untuk klasifikasi antar kelas disebut sebagai *line whereas*, fungsi yang digunakan untuk klasifikasi antas kelas dalam 3-D disebut *plane similarly*, sedangan fungsi yang digunakan untuk klasifikasi di dalam ruang kelas dimensi yang lebih tinggi di sebut *hyperplane*. Terdapat beberapa kernel yang dapat digunakan yakni linear kernel, polinomial Kernel, radial basis function kernel. Pengklasifikasi pada SVM menawarkan akurasi tinggi dan bekerja dengan baik dengan ruang dimensi tinggi. SVM pengklasifikasi pada dasarnya menggunakan subset dari poin pelatihan sehingga hasilnya menggunakan memori yang sangat sedikit.

Berikut implementasi contoh code dari SVM menggunakan dataset Iris:

```
[51] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn import svm
     # import module bagian pengukuran kinerja
     from sklearn.metrics import classification_report, cohen_kappa_score, fbeta_score, roc_auc_score
     from sklearn.metrics import confusion_matrix
```

```
# untuk import data dan dijadikan dataframe
df = pd.read_csv("Iris.csv")
df.head(10)#menampilksan sampel data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```
label_enc = {"Species": {"Iris-setosa":1.0, "Iris-versicolor":2.0, "Iris-virginica":3.0}}
df.replace(label_enc, inplace=True)
```

```
[30] #menampilkan jumlah record adalah dengan menggunakan property shape sehingga kita engetahui dimensi dari dataframe iris

     df.shape

     (150, 6)
```

```
[47] # melihat struktur dari data
     df.dtypes

     Id                int64
     SepalLengthCm     float64
     SepalWidthCm      float64
     PetalLengthCm     float64
     PetalWidthCm      float64
     Species           float64
     dtype: object
```

```
[50] #Informasi lebih detail mengenai struktur DataFrame iris dapat dilihat menggunakan fungsi info()

     df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 150 entries, 0 to 149
     Data columns (total 6 columns):
      #   Column         Non-Null Count  Dtype
     ---  ------         --------------  -----
      0   Id             150 non-null    int64
      1   SepalLengthCm  150 non-null    float64
      2   SepalWidthCm   150 non-null    float64
      3   PetalLengthCm  150 non-null    float64
      4   PetalWidthCm   150 non-null    float64
      5   Species        150 non-null    float64
     dtypes: float64(5), int64(1)
     memory usage: 7.2 KB
```

```
[49] #Informasi statistik untuk setiap kolom seperti nilai minimum, nilai maksimum, standar deviasi, rata-rata dan sebagainya, dapat ditampilkan dengan mengikuti perintah
     df.describe(include='all')
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667	2.000000
std	43.445368	0.828066	0.433594	1.764420	0.763161	0.819232
min	1.000000	4.300000	2.000000	1.000000	0.100000	1.000000
25%	38.250000	5.100000	2.800000	1.600000	0.300000	1.000000
50%	75.500000	5.800000	3.000000	4.350000	1.300000	2.000000
75%	112.750000	6.400000	3.300000	5.100000	1.800000	3.000000
max	150.000000	7.900000	4.400000	6.900000	2.500000	3.000000

```
[45] #mendapatkan baris (atau kolom) pada posisi tertentu dalam indeks yg hanya membutuhkan bilangan bulat pada X
     X = df.iloc[:,0:4]
     X
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4
...	...	...	...	...
145	146	6.7	3.0	5.2
146	147	6.3	2.5	5.0
147	148	6.5	3.0	5.2
148	149	6.2	3.4	5.4
149	150	5.9	3.0	5.1

150 rows × 4 columns

```
[32] ##mendapatkan baris (atau kolom) pada posisi tertentu dalam indeks yg hanya membutuhkan bilangan bulat pada
     y = df.iloc[:,-1]
     y
```

```
0      1.0
1      1.0
2      1.0
3      1.0
4      1.0
      ...
145    3.0
146    3.0
147    3.0
148    3.0
149    3.0
Name: Species, Length: 150, dtype: float64
```

```
[46] #Dataset dibagi ke dalam data latih dan data uji, data latih digunakan untuk melatih model yang telah dibuat sedangkan evaluasinya akan dilakukan pada data uji.
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.33, stratify = y)
```

```
[34] # menjelaskan ukuran data setelah dibagi
     len(X_train), len(X_test)
```

```
(100, 50)
```

```
[35] list_kernel = ["linear", "rbf", "poly", "sigmoid"]
```

```
[36] list_model_svm = []
     for i in list_kernel :
       temporary_model = svm.SVC(kernel=i)
       temporary_model.fit(X_train, y_train)
       list_model_svm.append(temporary_model)
```

```
list_model_svm
```

```
[SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False),
 SVC(C=1.0, 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=None, shrinking=True,
     tol=0.001, verbose=False),
 SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False),
 SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False)]
```

```
[38] # membangun model
     model_svr = svm.SVR(kernel="linear")
     model_svm = svm.SVC(kernel = "poly")

     # melatih model dengan data train(latihan)
     model_svm.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
[39] # membuat variabel hasil prediksi pada data test uji
     predict_svm = model_svm.predict(X_test)

     predict_svm
```

```
array([1., 3., 1., 3., 2., 3., 2., 2., 3., 1., 2., 1., 2., 1., 1., 1., 3.,
       3., 1., 2., 3., 2., 3., 3., 3., 2., 2., 2., 3., 1., 2., 1., 2., 2.,
       3., 1., 3., 3., 2., 1., 2., 3., 1., 1., 1., 1., 1., 2., 3., 1.])
```

```
[40] print(classification_report(y_test, predict_svm))
```

```
              precision    recall  f1-score   support

         1.0       0.89      1.00      0.94        16
         2.0       0.94      0.88      0.91        17
         3.0       1.00      0.94      0.97        17

    accuracy                           0.94        50
   macro avg       0.94      0.94      0.94        50
weighted avg       0.94      0.94      0.94        50
```

```
[41] print(confusion_matrix(y_test, predict_svm))
```

```
[[16  0  0]
 [ 2 15  0]
 [ 0  1 16]]
```

Support Vector Regression (**SVR**) adalah salah satu metode yang bisa digunakan dalam melakukan peramalan. Konsep SVR didasarkan pada risk minimization, yaitu untuk mengestimasi suatu fungsi dengan cara meminimalkan batas atas dari generalization error, sehingga SVR mampu mengatasi overfitting. VR bertujuan untuk menemukan sebuah fungsi *f(x)* sebagai suatu *hyperplane* (garis pemisah) berupa fungsi regresi yang mana sesuai dengan semua input data dengan membuat *error* (ε) sekecil mungkin.

```
[59] import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

```
[60] ## untuk import data dan dijadikan dataframe
     df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

```
[62] ##Informasi lebih detail mengenai struktur DataFrame iris dapat dilihat menggunakan fungsi info()
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64  
 2   creatinine_phosphokinase  299 non-null    int64  
 3   diabetes                  299 non-null    int64  
 4   ejection_fraction         299 non-null    int64  
 5   high_blood_pressure       299 non-null    int64  
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64  
 9   sex                       299 non-null    int64  
 10  smoking                   299 non-null    int64  
 11  time                      299 non-null    int64  
 12  DEATH_EVENT               299 non-null    int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

```
[44] # memisahkan atribut pada dataset dan menyimpannya pada sebuah variabel
     X = df[df.columns[:8]]

     # memisahkan label pada dataset dan menyimpannya pada sebuah variabel
     y = df['DEATH_EVENT']
```

```
[64] from sklearn.preprocessing import StandardScaler
```

```
[63]    # standarisasi nilai-nilai dari dataset
     scaler = StandardScaler()
     scaler.fit(X)
     X = scaler.transform(X)
```

```
# memisahkan data untuk training dan testing
#Dataset dibagi ke dalam data latih dan data uji, data latih digunakan untuk melatih model
#yang telah dibuat sedangkan evaluasinya akan dilakukan pada data uji.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

```
[66] # membuat objek SVC dan memanggil fungsi fit untuk melatih model
     clf = SVC()
     clf.fit(X_train, y_train)
```

```
SVC(C=1.0, 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=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
[67] # Menampilkan skor akurasi prediksi
     clf.score(X_test, y_test)
```

```
0.6777777777777778
```