

UNIVERSIDAD CATÓLICA DE SAN PABLO
MAESTRÍA EN CIENCIA DE LA COMPUTACIÓN
SISTEMAS INTELIGENTES

Máquina de Vectores de Soporte (SVM)

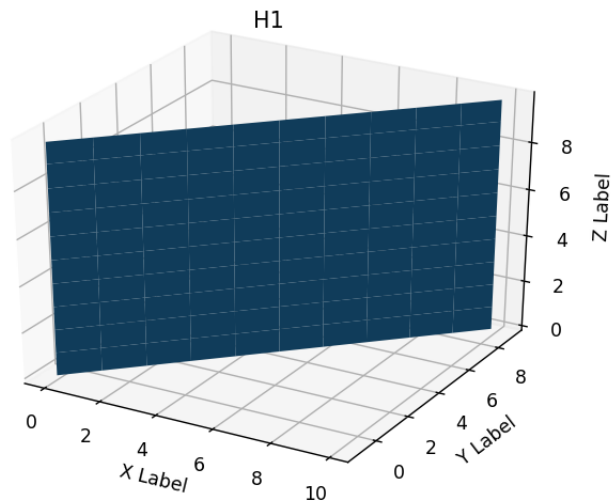
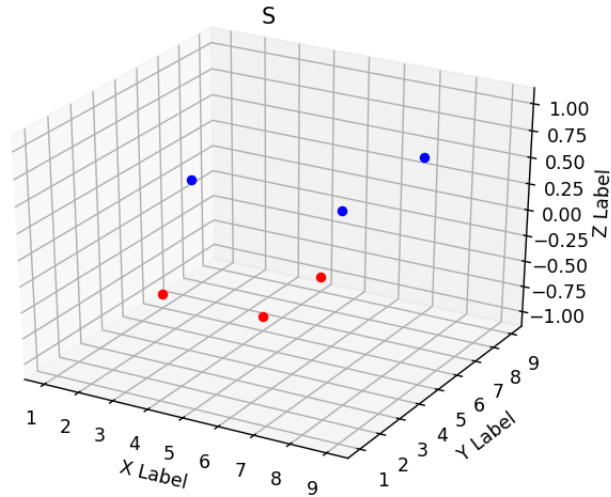
Prof. Graciela Meza Lovón

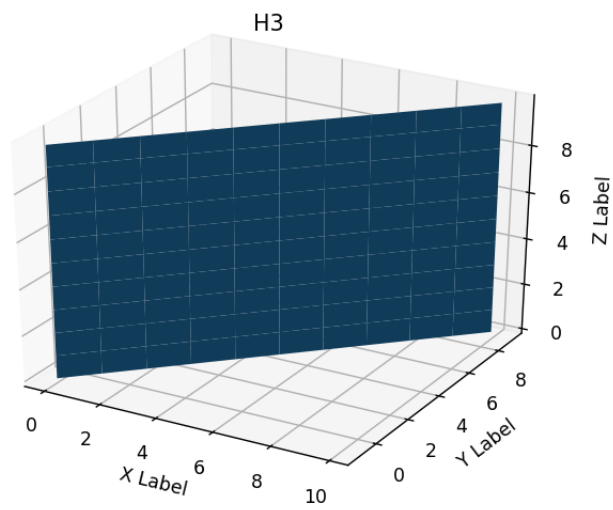
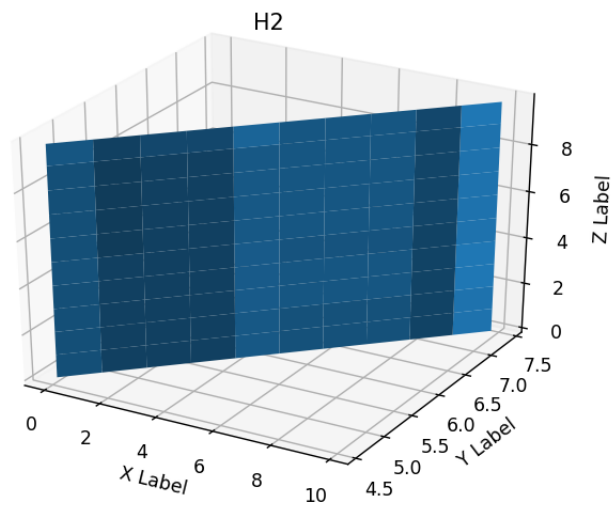
Palomino Paucar, Daniel Alfredo
Noviembre 26, 2018

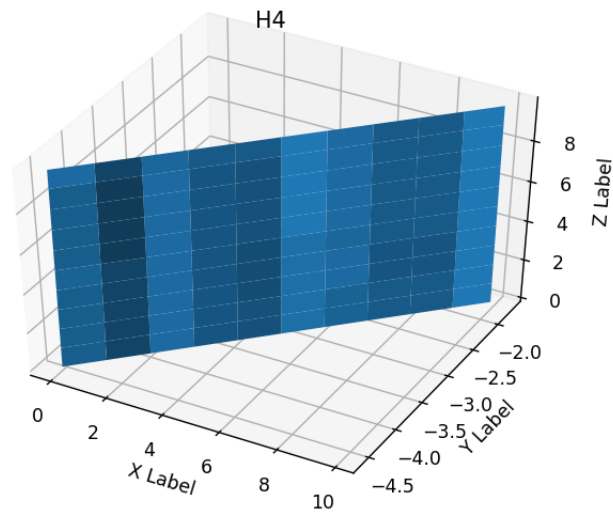
1. PREGUNTAS DE TEORÍA

Sea el conjunto $S = ((1, 6), -1), ((4, 9), -1), ((4, 6), -1), ((5, 1), 1), ((9, 5), 1), ((9, 1), 1)$ y un conjunto de cuatro hiperplanos $H = H_1, H_2, H_3, H_4$ definidos como: $H_1 : x_1 - x_2 - 1 = 0$, $H_2 : 2x_1 - 7x_2 + 32 = 0$, $H_3 : \sqrt{\frac{1}{2}}x_1 - \sqrt{\frac{1}{2}}x_2 - \sqrt{\frac{1}{2}} = 0$, $H_4 : 2x_1 - 7x_2 - 32 = 0$

1. Usando cualquier lenguaje de programación grafique S , H_1 , H_2 , H_3 y H_4 .







2. Encuentre los parámetros w y b que definen los hiperplanos H_1 , H_2 , H_3 y H_4 , y luego determine para H_1 , H_2 , H_3 y H_4 si son hiperplanos de separación. Fundamente.

Dadas la ecuación vectorial de un hiperplano: $W.X + b = 0$, se tiene al extender la ecuación: $w_1.x_1 + w_2.x_2 + w_3.w_3 + b = 0$.

Dando la forma a la ecuaciones dadas:

a) H_1 :

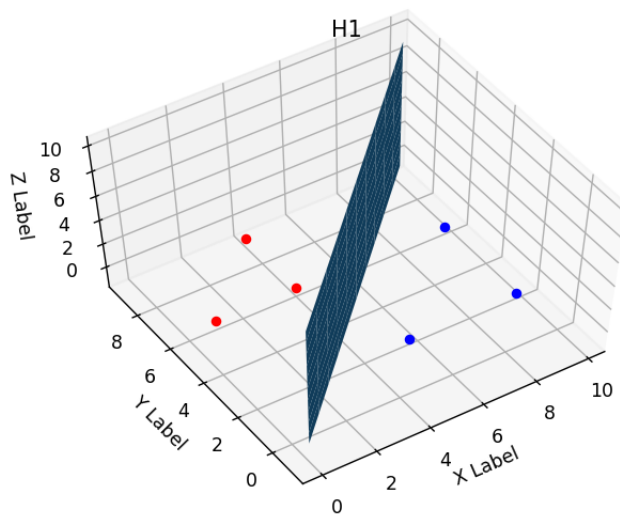
$$x_1 - x_2 - 1 = 0$$

$$(1, -1, 0).X - 1 = 0$$

Entonces:

$$W = (1, -1, 0) \wedge b = -1$$

Graficando:



Por lo tanto H_1 : Sí es un hiperplano de separación.

b) H_2 :

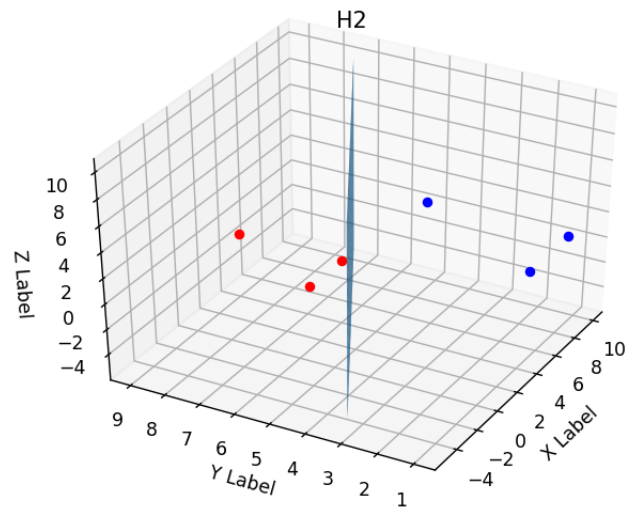
$$2x_1 - 7x_2 + 32 = 0$$

$$(2, -7, 0) \cdot X + 32 = 0$$

Entonces:

$$W = (2, -7, 0) \wedge b = 32$$

Graficando:



Por lo tanto H_2 : Sí es un hiperplano de separación.

c) H_3 :

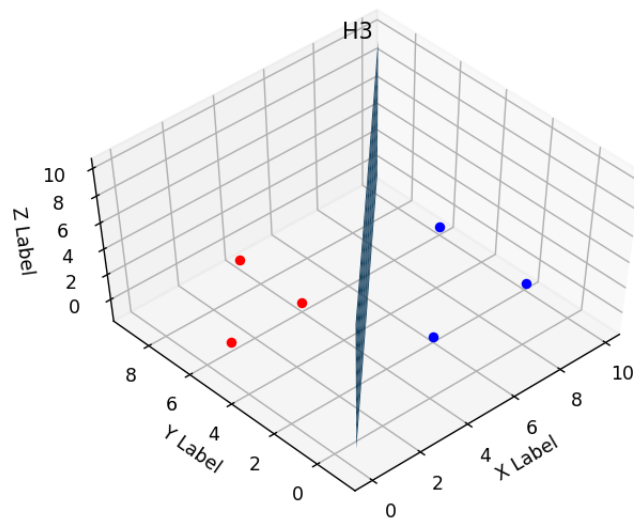
$$\sqrt{\frac{1}{2}}x_1 - \sqrt{\frac{1}{2}}x_2 - \sqrt{\frac{1}{2}} = 0$$

$$(1, -1, 0) \cdot X - 1 = 0$$

Entonces:

$$W = \left(\sqrt{\frac{1}{2}}, -\sqrt{\frac{1}{2}}, 0\right) \wedge b = -\sqrt{\frac{1}{2}}$$

Graficando:



Por lo tanto H_3 : Sí es un hiperplano de separación.

d) H_4 :

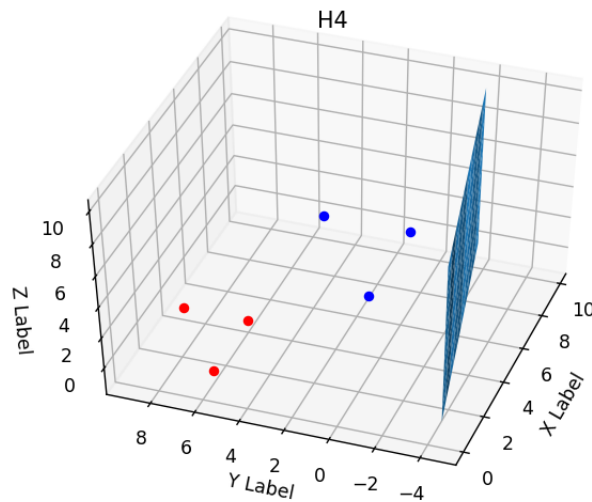
$$2x_1 - 7x_2 - 32 = 0$$

$$(2, -7, 0) \cdot X - 32 = 0$$

Entonces:

$$W = (2, -7, 0) \wedge b = -32$$

Graficando:



Por lo tanto H_4 : No es un hiperplano de separación.

3. En el conjunto H , ¿cuántos hiperplanos iguales existen?. En el caso de que existan ¿cuáles son éstos? Fundamente.

Existen 2 hiperplanos iguales H_1 y H_3 , dado que si a la ecuación de H_3 la dividimos entre $\sqrt{\frac{1}{2}}$ tendríamos la misma ecuación que H_1 .

4. Calcule el margen τ para cada hiperplano de separación. Luego, suponga que el conjunto H contiene al hiperplano óptimo H^* , ¿cuál sería H^* ? Fundamente.

a) H_1 :

$$\tau_1 = 2,1213$$

b) H_2 :

$$\tau_2 = 0,2742$$

c) H_3 :

$$\tau_3 = 2,1213$$

De acuerdo a los datos calculados, si el hiperplano H^* existiera en el conjunto H , este sería H_1 o H_3 (pues representan el mismo plano), dado que poseen el τ con mayor valor.

5. ¿Cuáles son los vectores de soporte del hiperplano H^* escogido en la pregunta anterior?. Fundamente. (No necesita encontrar los valores α)

Los vectores de soporte del hiperplano óptimo elegido (H_1 o H_3) serían:

$$(4, 6, -1), (5, 1, 1), (9, 5, 1)$$

pues se encuentran en el margen óptimo, teniendo cada vector el τ mínimo de 2,1313.

6. Demuestre la primera condición KKT, i.e. (Ec. 7 de las diapositivas) $\frac{\partial L}{\partial w}(w^*, b^*, \alpha) = w^* - \sum_{i=1}^m \alpha_i y^{(i)} x^i$

Sea:

$$L = \frac{1}{2} W \cdot W^T - \sum_{i=1}^m \alpha_i [y^{(i)} (W^T X^{(i)} + b) - 1]$$

Expandiendo los términos:

$$L = \frac{1}{2} W \cdot W^T - \sum_{i=1}^m \alpha_i [y^{(i)} W^T X^{(i)}] - \sum_{i=1}^m \alpha_i [y^{(i)} b] + \sum_{i=1}^m \alpha_i$$

Para encontrar los W^*, b^*, α^* óptimo, necesitamos derivar respecto de los parámetros, lo cual se describe en la ecuación 1:

$$\frac{\partial L(W^*, b^*, \alpha^*)}{\partial w_i} = 0, i = 1, \dots, n$$

Entonces, derivando L respecto de w_i tenemos:

$$\frac{\partial L(W^*, b^*, \alpha^*)}{\partial w_i} = \frac{\partial(\frac{1}{2} W \cdot W^T - \sum_{i=1}^m \alpha_i [y^{(i)} W^T X^{(i)}] - \sum_{i=1}^m \alpha_i [y^{(i)} b] + \sum_{i=1}^m \alpha_i)}{\partial w_i}$$

Separando los términos de la derecha:

$$\frac{\partial L(W^*, b^*, \alpha^*)}{\partial w_i} = \frac{\partial(\frac{1}{2} W \cdot W^T)}{\partial w_i} - \frac{\partial(\sum_{i=1}^m \alpha_i [y^{(i)} W^T X^{(i)}])}{\partial w_i} - \frac{\partial(\sum_{i=1}^m \alpha_i [y^{(i)} b])}{\partial w_i} + \frac{\partial(\sum_{i=1}^m \alpha_i)}{\partial w_i}$$

El tercer y cuarto término de la derecha no depende de w_i , por lo tanto al derivarse desaparecerán:

$$\frac{\partial L(W^*, b^*, \alpha^*)}{\partial w_i} = \frac{\partial(\frac{1}{2} W \cdot W^T)}{\partial w_i} - \frac{\partial(\sum_{i=1}^m \alpha_i [y^{(i)} W^T X^{(i)}])}{\partial w_i}$$

Finalmente calculando la derivada parcial:

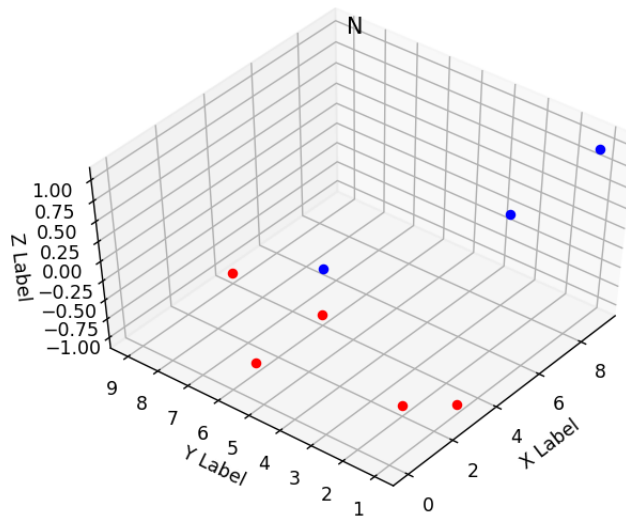
$$\frac{\partial L}{\partial w}(w^*, b^*, \alpha) = w^* - \sum_{i=1}^m \alpha_i y(i) x^i = 0$$

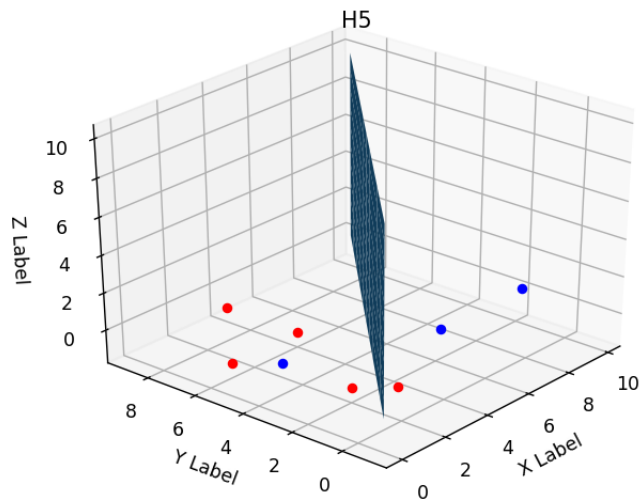
Quedando demostrado la ecuación 7 de las diapositivas.

Sea el conjunto $N = ((1, 6), -1), ((4, 9), -1), ((4, 6), -1), ((5, 1), 1), ((9, 1), 1), ((0, 3), 1), ((2, 2), -1), ((3, 1), -1)$ y el hiperplano H_1 definido anteriormente.

7. Usando cualquier lenguaje de programación grafique N y H_5 .

Graficando:





8. Identifique los ejemplos que son separables y los que no lo son. Luego, determine los ejemplos que son clasificados correctamente y los que no.

Para que un vector sea separable debe cumplir:

$$y_i(W^T X) \geq 1$$

Por lo tanto haciendo los cálculos para los ejemplos del espacio N y el hiperplano H_5 :

- a) $((1, 6), -1): 5$
- b) $((4, 9), -1): 5$
- c) $((4, 6), -1): 2$
- d) $((5, 1), 1): 4$
- e) $((9, 1), 1): 8$
- f) $((0, 3), 1): -3$
- g) $((2, 2), -1): 0$
- h) $((3, 1), -1): -2$

Por lo tanto: Los ejemplos separables son:

$((1, 6), -1), ((4, 9), -1), ((4, 6), -1), ((5, 1), 1), ((9, 1), 1)$

y los no separables son:

$((0, 3), 1), ((2, 2), -1), ((3, 1), -1)$

Así mismo, los ejemplos clasificados correctamente son:

$((1, 6), -1), ((4, 9), -1), ((4, 6), -1), ((5, 1), 1), ((9, 1), 1), ((2, 2), -1)$

y los no clasificados correctamente son:

$((0, 3), 1), ((3, 1), -1)$

9. Calcule la ... de los ejemplos no separables.

Enunciado NO definido.

2. PREGUNTAS DE INVESTIGACIÓN

1. Explique el significado de la constante C en el término $C \sum_{i=1} \epsilon_i$ que se agrega a la función objetivo en el caso de ejemplos casi linealmente separables. Luego, explique la influencia de C en la capacidad de generalización de una SVM.
2. Describa el significado del parámetro γ en el kernel gaussiano. Luego, explique la influencia de γ en la capacidad de generalización de una SVM.

3. IMPLEMENTACIÓN

1. Usando Scikit-learn de Python, implemente (comente su código) una svm que clasifique el conjunto de datos (por definir).

Como no se ha incluido el dataset, he propuesto trabajar el presente ejercicio con el siguiente dataset (banknote authentication):

https://drive.google.com/file/d/13nw-uRXPY8XIZQxKRNZ3yYlho-CYm_Qt/view

Lectura de Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Lectura del Dataset usando pandas
bankdata = pd.read_csv("bill_authentication.csv")
```

Exploración de Datos

```
# Dimensiones del Dataset
bankdata.shape
# Descripción del Dataset
bankdata.head()
```

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Preprocesamiento

```
# Separación de las columnas del dataset para crear los valores de entrada
# y el valor de salida
X = bankdata.drop('Class', axis=1)
y = bankdata['Class']
from sklearn.model_selection import train_test_split
```

```
# Separaci3n del dataset en valores de entranamiento y testeo
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Entrenamiento

```
from sklearn.svm import SVC
# Elecci3n del Kernel: linear
svclassifier = SVC(kernel='linear')
# Ejecuci3n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicci3n y Validaci3n

```
# Predicci3n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci3n de la matriz de confusi3n
print(confusion_matrix(y_test, y_pred))
# Creaci3n de reporte de resultados
print(classification_report(y_test, y_pred))
```

```
[[144  1]
 [ 1 129]]
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	145
1	0.99	0.99	0.99	130
avg / total	0.99	0.99	0.99	275

- Experimente y muestre resultados usando diferentes valores para los parámetros de los kernels: lineal, polinomial, gaussiano, y el parámetro C. Los resultados deben ser mostrados en el documento pdf.

Como no se ha incluido el dataset, he propuesto trabajar el presente ejercicio con el siguiente dataset (Iris Flower): <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

Lectura de Dataset

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases"
      "/iris/iris.data"
# Asignar columnas al Dataset para su posterior lectura
colnames = ['sepal-length', 'sepal-width',
            'petal-length', 'petal-width', 'Class']
# Lectura del Dataset usando pandas
irisdata = pd.read_csv(url, names=colnames)
```

Exploración de Datos

```
# Descripción del Dataset
bankdata.head()
```

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Preprocesamiento

```
# Separación de las columnas del dataset para crear los valores de entrada
# y el valor de salida
X = irisdata.drop('Class', axis=1)
y = irisdata['Class']
# Separación del dataset en valores de entrenamiento y testeo
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Lineal

Entrenamiento

```
from sklearn.svm import SVC
# Elección del Kernel: lineal
svclassifier = SVC(kernel='linear')
# Ejecución del entrenamiento
svclassifier.fit(X_train, y_train)
```


Predicción y Validación

```
# Predicción usando los valores de testeo
y_pred = svcclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creación de la matriz de confusión
print(confusion_matrix(y_test, y_pred))
# Creación de reporte de resultados
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 10  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	10
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Polinomial

Entrenamiento

```
from sklearn.svm import SVC
# Elección del Kernel: poly
svcclassifier = SVC(kernel='poly')
# Ejecución del entrenamiento
svcclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicción usando los valores de testeo
y_pred = svcclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creación de la matriz de confusión
print(confusion_matrix(y_test, y_pred))
# Creación de reporte de resultados
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  9  1]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.90	0.95	10
Iris-virginica	0.90	1.00	0.95	9
avg / total	0.97	0.97	0.97	30

Gaussiano

Entrenamiento

```
from sklearn.svm import SVC
# Elección del Kernel: rbf
svclassifier = SVC(kernel='rbf')
# Ejecución del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicción usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creación de la matriz de confusión
print(confusion_matrix(y_test, y_pred))
# Creación de reporte de resultados
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 10  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	10
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

3. Dentro de la sección de Implementación incluya una subsección donde indique las instrucciones para ejecutar el código.

La presente implementación ha sido realizada en python 3.6 usando anaconda y jupyter-lab como herramientas de virtualización y visualización.

El notebook desarrollado que incluye todos los pasos descritos, así como los datasets, ha sido publicado en mi cuenta de github:

<https://github.com/dpalominop/SistemasInteligentes/blob/master/semana7/homework/svm.ipynb>