

Máquina de soporte vectorial - svm

By: Jared Isaías Monje Flores



30 de marzo de 2024

centro universitario de ciencias exactas e ingenierías

UdeG

**Máquina de soporte vectorial – SVM**

SVM (siglas en inglés de Support Vector Machine) es un algoritmo de aprendizaje supervisado utilizado en inteligencia artificial para resolver problemas de clasificación y regresión.

En la clasificación, SVM busca un hiperplano que separe de la mejor manera posible dos clases diferentes de puntos de datos. Este hiperplano maximiza el margen entre las dos clases, creando una separación clara entre ellas.

En la regresión, SVM busca un hiperplano que se ajuste lo mejor posible a los datos, minimizando el error de predicción.

Home\_data

Linear Regresión:

Train: 0.6926109343701519

Train: 0.7329563273174166

Stars

SVC

Model1

Train: 0.9404761904761905

Test: 0.8611111111111112

Model2

MLP

Train: 0.9345238095238095

Test: 0.8611111111111112

Código fuente:

Home data

"""Home\_sol By: Dexne

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1XBEjebh2rjOr\_QVmWFVVpYqdexoh37zp

"""

# Importamos paquetes

import numpy as np

import pandas as pd

from sklearn.svm import SVR

from sklearn.svm import SVC

from sklearn.neural\_network import MLPRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

# Leemos los datos

data = pd.read\_csv('home\_data.csv')

# Selección de variables

# Eliminamos los registros [ 'id', 'price', 'date', 'zipcode' ]

# Ya que no son muy relevantes para el análisis

x = np.array(data.drop(columns=['id', 'price', 'date', 'zipcode']))[:5000,:]

y = np.array(data[['price']])[:5000,:]

# Normalizamos los datos

x = MinMaxScaler().fit\_transform(x)

# Train test split

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.1)

## Crear modelo - Seleccionamos Regresión lineal

#model = SVR(gamma=0.001, C=0.01, kernel='rbf')

model = LinearRegression()

# Entrenar modelo

model.fit( xtrain, ytrain.ravel() )

# Metricas de desempeño

print('Train: ', model.score( xtrain, ytrain.ravel() ))

print('Train: ', model.score( xtest, ytest.ravel() ))

"""

Results:

Train:  0.6926109343701519

Train:  0.7329563273174166

"""

Stars

"""Star\_sol.py By: Dexne

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1H87No1lVuTBILU2FzGxokBQwkaztGmb2

"""

# Importamos paquetería

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.pipeline import Pipeline

# Leemos los datos

data = pd.read\_csv('Stars.csv')

# Codify

le = LabelEncoder()

# Codificación de la variable categórica 'Spectral Class'

data[['Spectral Class']] = le.fit\_transform(np.array(data[['Spectral Class']]).ravel()).reshape(-1,1)

labels = le.classes\_

le = LabelEncoder()

# Almacenar las etiquetas codificadas para la variable 'Spectral Class'

data[['Star color']] = le.fit\_transform(np.array(data[['Star color']]).ravel()).reshape(-1,1)

# Seleccionamos las variables independientes (X) y la variable dependiente (Y)

x = np.asanyarray(data.drop(columns=['Star category', 'Spectral Class']))

y = np.asanyarray(data[['Spectral Class']])

# Dividir los datos en conjuntos de entrenamiento y prueba

xtrain, xtest, ytrain, ytest = train\_test\_split( x, y, test\_size=0.3 )

# Creamos dos modelos: SVC (Support Vector Machine) y MLP (Multilayer Perceptron)

model1 = Pipeline([('scaler', StandardScaler()),

                   ('SVM', SVC(gamma=1))])

model2 = Pipeline([('scaler', StandardScaler()),

                   ('mlp', MLPClassifier(hidden\_layer\_sizes=(100,),

                                         max\_iter=500))])

# Entrenar y evaluar el modelo SVC

model1.fit(xtrain, ytrain.ravel())

print('SVC')

print('Train: ', model1.score( xtrain, ytrain.ravel() ))

print('Test: ', model1.score( xtest, ytest.ravel() ))

# Entrenar y evaluar el modelo MLP

model2.fit( xtrain, ytrain.ravel() )

print('MLP')

print('Train: ', model2.score( xtrain, ytrain.ravel() ))

print('Test: ', model2.score( xtest, ytest.ravel() ))

"""

    Results

Model1 - SVC

Train:  0.9404761904761905

Test:  0.8611111111111112

Model2 - MLP

Train:  0.9345238095238095

test:  0.8611111111111112

"""