PRÁCTICA DE MODELO CLASIFICADOR DE IMÁGENES

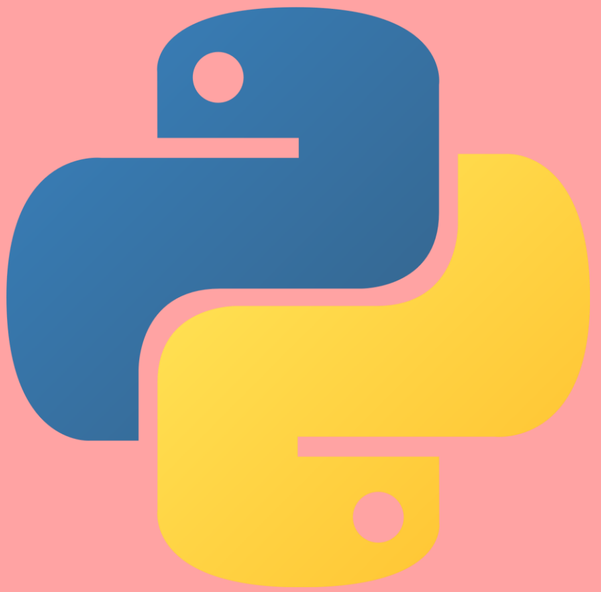
Índice

Introducción

Este programa utiliza Python como lenguaje de programación y aprovecha bibliotecas como scikit-learn y matplotlib para cargar, entrenar y evaluar un modelo de clasificación utilizando Support Vector Machines (SVM) con un kernel lineal.

Definición de terminos

- Scikit-learn: Realiza tareas de aprendizaje automático
- Matplotlib: Utilizada para la creación de gráficos estáticos, animados e interactivos.
- Support Vector Machines (SVM): Es un modelo de aprendizaje supervisado utilizado para clasificación y regresión.
- Kernel Lineal: Función matemática que transforma los datos en un espacio de mayor dimensión, lo que permite a SVM encontrar separaciones no lineales en el espacio original.



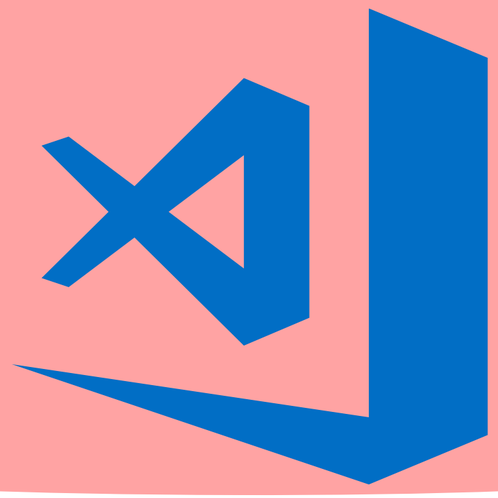
Objetivo 1

Entrenar el modelo para que identifique correctamente dígitos escritos a mano (0–9) y analizar su desempeño mediante métricas como precisión y un informe de clasificación,

Pasos

1

Crearemos un archivo, en este caso creado en Visual Studio Code, con terminación .py



Pasos

2

Posteriormente importaremos las bibliotecas.

Cargaremos el conjunto de datos

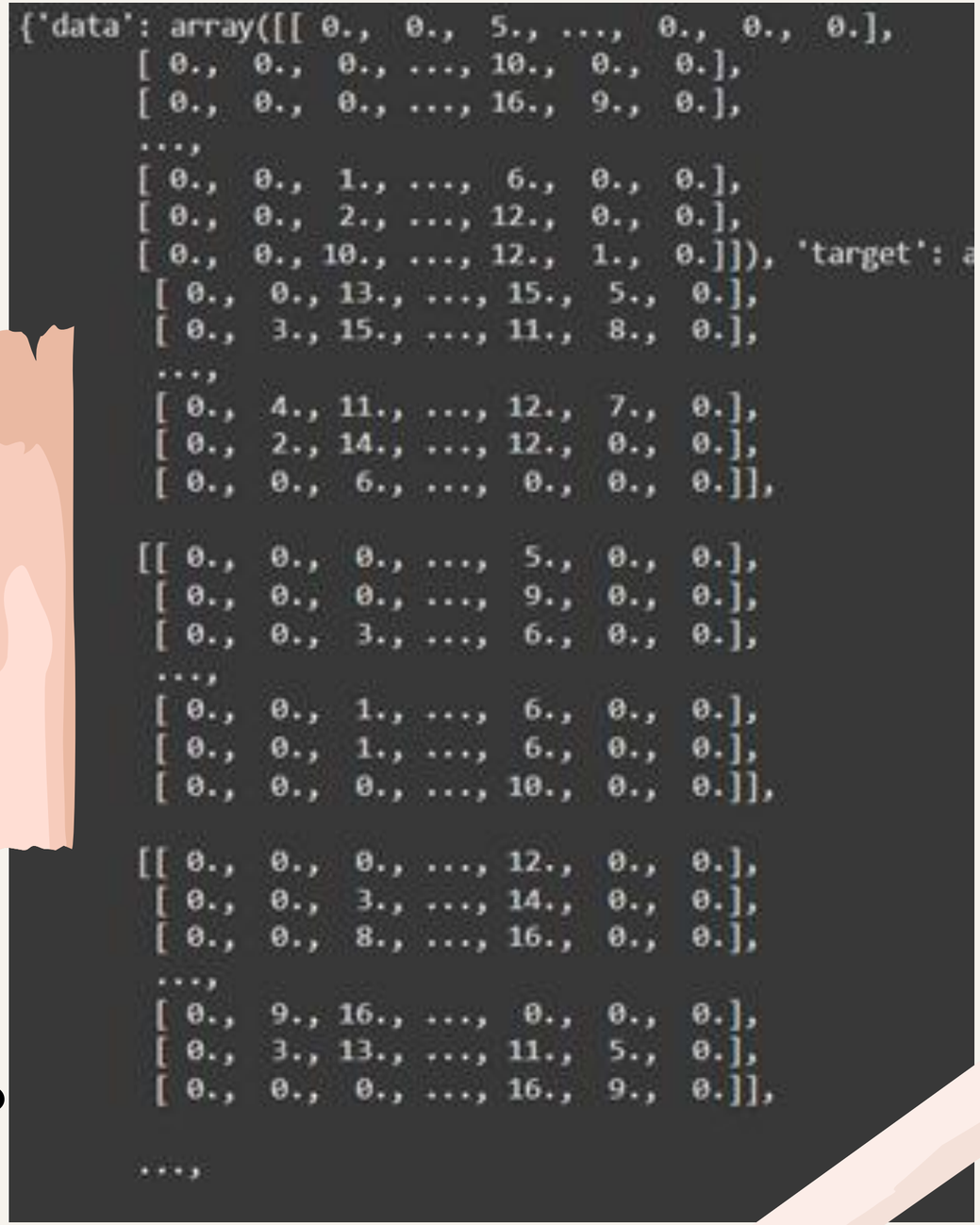
```
# Importar las bibliotecas necesarias
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
import matplotlib as plt

# Cargar el conjunto de datos de digitos
digits = datasets.load_digits()
```

Pasos

3

Los digitos se deben mostrar en forma de arreglos

Pueden seguir el siguiente ejemplo:



Pasos

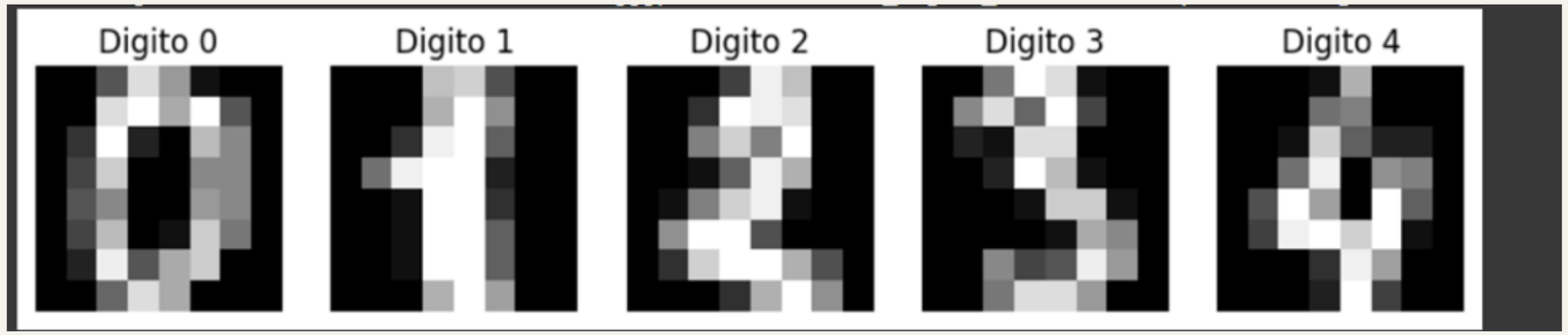
4

Se debe de dar la instrucción de separar los datos y visualizarlos, en este último se mostraran una fila de 5 imágenes en escala de grises con etiquetas como "Dígito 0", "Dígito 1", etc

```
# Dividir los datos en características (X) y etiquetas (y)
X = digits.images
y = digits.target


# Visualizar una muestra de las imágenes y etiquetas
n_muestras = 5
plt.figure(figsize = (10, 2))
for i in range(n_muestras):
    plt.subplot(1, n_muestras, i + 1)
    plt.imshow(X(i), cmap='gray')
    plt.title(f"Digito" (y[i]))
    plt.axis('off')
plt.show()
```

Resultados

En la consola se imprimirá la siguiente imagen:



Pasos

5

Debido a que el modelo SVM espera vectores planos como entrada, las imágenes originales de 8x8 píxeles (formato 2D) se convierten arreglos unidimensionales de 64 valores.

```
# Preprocesar las imágenes a formato 16
X = X.reshape([X.shape[0],-1])

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=43)

# Crear un modelo de SVM con el kernel lineal
modelo = SVC(kernel='linear')

# Entrenar el modelo con los datos de entrenamiento
modelo.fit(X_train, y_train)

# Predecir las etiquetas para el conjunto de pruebas
y_pred = modelo.predict(X_test)
```

Pasos

6

A continuación, hay que asegurar el porcentaje de precisión

Y para finalizar, debe generar un reporte detallado con métricas

```
# Calcular la precisión del modelo
precision = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo SVM con kernellineal: {precision: 2f}")
```

```
Precisión del modelo SVM con kernellineal: 0.98
```

```
# Mostrar un informe de clasificación
print("Informe de clasificación")
print(classification_report(y_test, y_pred))
```

Resultados

En la consola se imprimirá la siguiente imagen:

Informe de clasificación

	precision	recall	f1-score	support
0	1.00	1.00	1.00	42
1	0.91	1.00	0.96	32
2	1.00	1.00	1.00	25
3	1.00	0.96	0.98	25
4	1.00	0.98	0.99	44
5	0.94	1.00	0.97	30
6	1.00	1.00	1.00	41
7	0.97	1.00	0.99	39
8	1.00	0.93	0.96	43
9	0.95	0.92	0.94	39
accuracy			0.98	360
macro avg	0.98	0.98	0.98	360
weighted avg	0.98	0.98	0.98	360

CONCLUSIONES

La práctica demuestra que el modelo SVM con kernel lineal es altamente efectivo para clasificar imágenes de dígitos escritos a mano, alcanzando una precisión cercana al 98%. Esto destaca la capacidad de SVM para manejar tareas de clasificación en conjuntos de datos con características bien definidas. Además, la implementación ilustra la importancia del preprocesamiento, la división de datos y la evaluación del modelo para obtener resultados confiables.

MUCHAS GRACIAS

www.unsitiogenial.es