Actividad final - Entrenar un modelo de clasificación en Machine Learning: Pasos para el entrenamiento del modelo

Los pasos para entrenar el modelo son los siguientes:

1. RECOPILAR Y PROCESAR LOS DATOS PARA ENTRENAR Y EVALUAR EL MODELO

Primero, crea y guarda un archivo de Python llamado “entrenamiento”.

Después, descarga la base de datos “Iris” pinchando en el siguiente enlace: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data. Guárdala en la misma carpeta que el archivo de Python que has creado. (No le cambies el nombre. Se descarga por defecto como iris.data).

En el entorno de Spyder, importa las bibliotecas necesarias y carga la base de datos Iris:

```
# Importa las bibliotecas necesarias para el análisis

import pandas as pd
import matplotlib.pyplot as plt

# Cargar la base de datos iris

dataframe = pd.read_csv('iris.data', header=None, names=['longitud_sepalo', 'ancho_sepalo', 'longitud_petalo',
'ancho_petalo', 'especies'])
X = dataframe.drop('especies', axis=1)
y = dataframe['especies']
```

2. EXPLORAR LOS DATOS

Genera dos gráficos de dispersión para observar la distribución de los datos y la relación entre las variables:

```
# Crear un gráfico de dispersión para visualizar la relación entre la longitud y el ancho del sépalo para cada clase de flor iris

colors = ['red', 'blue', 'green']
classes = dataframe['especies'].unique()
for i in range(len(classes)):
   plt.scatter(dataframe.loc[dataframe['especies'] == classes[i], 'longitud_sepalo'],
            dataframe.loc[dataframe['especies'] == classes[i], 'ancho_sepalo'],
            color=colors[i], label=classes[i])
plt.xlabel('longitud del sépalo')
plt.ylabel('ancho del sépalo')
plt.legend(loc='best')
plt.show()

# Crear un gráfico de dispersión para visualizar la relación entre la longitud y el ancho del pétalo para cada clase de flor iris

colors = ['red', 'blue', 'green']
classes = dataframe['especies'].unique()
for i in range(len(classes)):
   plt.scatter(dataframe.loc[dataframe['especies'] == classes[i], 'longitud_petalo'],
            dataframe.loc[dataframe['especies'] == classes[i], 'ancho_petalo'],
            color=colors[i], label=classes[i])
plt.xlabel('longitud del pétalo')
plt.ylabel('ancho del pétalo')
plt.legend(loc='best')
plt.show()
```

Pega los gráficos en tu actividad y coméntalos brevemente.

3. DIVIDIR LOS DATOS EN UN CONJUNTO DE ENTRENAMIENTO Y OTRO DE PRUEBA

La fase de dividir los datos en conjuntos de entrenamiento y de prueba es una parte crucial del proceso de entrenamiento de un modelo de ML, ya que permite evaluar su capacidad para generalizar con datos nuevos.

```
# Dividir los datos en conjuntos de entrenamiento y de prueba

train = dataframe.sample(frac=0.8, random_state=0)
test = dataframe.drop(train.index)
```

Con el código anterior, se divide la base de datos en dos subconjuntos: uno destinado al entrenamiento (train) y otro de prueba (test). Para obtener los datos de entrenamiento (train), se emplea el método “sample” de Pandas, con el que se obtiene una muestra aleatoria del 80% (frac=0.8) de la muestra.

4. ELEGIR EL ALGORITMO DE CLASIFICACIÓN

Existen muchos algoritmos de clasificación en el campo del ML, cada uno con sus propias ventajas y desventajas. Los más populares son los siguientes: K-vecinos más cercanos (KNN), árboles de decisión, regresión logística o máquinas de vectores de soporte, entre otros.

Para este ejercicio, vamos a aplicar el algoritmo K-vecinos más cercanos o K-Nearest Neighbors (KNN). El algoritmo KNN calcula la distancia entre el punto de datos desconocido y los puntos de datos conocidos del conjunto de entrenamiento, y luego selecciona los k puntos de datos más cercanos al punto desconocido. Después, asigna al punto desconocido la clase más común entre esos k vecinos más cercanos.

```
# Entrenar el modelo mediante el algoritmo de clasificación KNN

import numpy as np

def knn_classifier(X_train, y_train, X_test, k):
    y_pred = []
    for i in range(X_test.shape[0]):
        distances = np.sqrt(np.sum((X_train - X_test.iloc[i])**2, axis=1))
        nearest = y_train.iloc[np.argsort(distances)[:k]]
        y_pred.append(nearest.value_counts().index[0])
    return y_pred
```

5. EVALUAR LA PRECISIÓN DEL MODELO EN EL CONJUNTO DE PRUEBA

La precisión es una medida importante para evaluar la calidad del modelo, ya que indica su capacidad para generalizar con nuevos datos que no ha visto durante el entrenamiento. La precisión del modelo se refiere a la proporción de predicciones correctas que realiza el modelo en el conjunto de prueba. Por ejemplo, si la precisión es del 78%, significa que el modelo ha realizado correctamente el 78% de las predicciones en el conjunto de prueba.

Para calcular la precisión de nuestro modelo, inserta y ejecuta el siguiente código:

```
# Evaluar el modelo en el conjunto de prueba calculando la precisión

y_test_pred = knn_classifier(train.drop('especies', axis=1), train['especies'], test.drop('especies', axis=1), 5)
accuracy = (test['especies'] == y_test_pred).mean()
print(f'Accuracy: {accuracy:.2f}')
```

En la consola inferior derecha, verás la precisión (“accuracy”) obtenida. **Pégala en tu actividad y coméntala.**

6. HACER PREDICCIONES SOBRE NUEVOS DATOS

Esta fase es la última en el proceso de construcción de un modelo de aprendizaje automático. Después del entrenamiento, el modelo se utiliza para hacer predicciones sobre nuevos datos que no se usaron durante la etapa anterior.

El siguiente código proporciona un ejemplo de cómo usar el algoritmo KNN para hacer predicciones con nuevos datos. En este código, primero se ha creado un DataFrame de Pandas llamado "new_data", que contiene dos filas de datos nuevos, es decir, dos nuevas observaciones de flores. Cada fila recoge cuatro características (longitud_sepalo, ancho_sepalo, longitud_petalo, ancho_petalo) para las cuales se quieren hacer predicciones.

```
# Hacer predicciones sobre nuevos datos

new_data = pd.DataFrame({'longitud_sepalo': [5.1, 4.9], 'ancho_sepalo': [3.5, 3.0], 'longitud_petalo': [1.4, 1.4], 'ancho_petalo': [0.2, 0.2]})
y_pred = knn_classifier(X, y, new_data, 5)
print(f'Predictions: {y_pred}')
```

Al ejecutar este código, se imprime en la consola inferior derecha la predicción devuelta por la función "knn_classifier" en formato cadena de texto.

¿Cómo se interpreta la predicción? Tomemos como ejemplo la siguiente predicción hipotética: Predictions: ['Iris-virginica', 'Iris-versicolor']. Esta constata que el modelo ha predicho que los nuevos datos de la primera observación corresponden a la clase virginica, y los de la segunda a la versicolor.

Pega la predicción que se ha impreso en la consola inferior derecha en el documento de tu actividad, e interprétala.

Envía la actividad a tu tutor en un documento de pdf. En dicho documento, deberás incluir todos los elementos que estimes oportunos: texto explicativo, imágenes, volcados de pantalla, código de Python, etc., para demostrar que has resuelto las cuestiones solicitadas.