Ejercicio: Predicción del clima usando una red neuronal

Objetivo

Entrenar una red neuronal que pueda predecir el estado del clima basándose en la temperatura y la fecha.

---

1. Crear un DataFrame con datos simulados

```
import pandas as pd
import numpy as np

# Generar datos aleatorios
np.random.seed(42)
n_samples = 1000

# Generar fechas aleatorias en 2024
dates = pd.date_range(start="2024-01-01", end="2024-12-31", periods=n_samples)

# Generar temperaturas aleatorias (entre -10°C y 40°C)
temperatures = np.random.uniform(-10, 40, n_samples)

# Generar estados del clima en función de la temperatura
# Soleado: temp > 25, Nublado: 10 <= temp <= 25, Lluvioso: temp < 10
conditions = np.where(temperatures > 25, "soleado",
                      np.where(temperatures >= 10, "nublado", "lluvioso"))

# Crear el DataFrame
data = pd.DataFrame({
  "fecha": dates,
  "temperatura": temperatures,
  "clima": conditions
})

# Convertir la columna 'fecha' en valores numéricos (para simplificar el modelo)
data['dia'] = data['fecha'].dt.dayofyear  # Día del año (1 a 365)
data.drop(columns="fecha", inplace=True)  # Eliminar la columna original de fechas

# Mostrar los primeros datos
print(data.head())
```

2. Preprocesar los datos y dividirlos

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Codificar el clima (soleado=0, nublado=1, lluvioso=2)
encoder = LabelEncoder()
data['clima_encoded'] = encoder.fit_transform(data['clima'])

# Variables de entrada (día y temperatura) y salida (clima codificado)
X = data[['dia', 'temperatura']].values
y = data['clima_encoded'].values

# Dividir en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Escalar los datos de entrada
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

---

3. Crear y entrenar la red neuronal

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Crear el modelo
model = Sequential([
  Dense(64, activation='relu', input_shape=(2,)),  # Entrada: día y temperatura
  Dense(32, activation='relu'),
  Dense(3, activation='softmax')  # Salida: 3 clases (soleado, nublado, lluvioso)
])

# Compilar el modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Entrenar el modelo
model.fit(X_train, y_train, epochs=30, validation_data=(X_test, y_test))
```

---

4. Evaluar el modelo

```
# Evaluar el modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Pérdida: {loss:.2f}, Precisión: {accuracy:.2%}")
```

---

5. Realizar predicciones

```
# Realizar predicciones
new_data = np.array([[120, 15],  # Día 120 del año, temperatura 15°C
                     [200, 30],  # Día 200, temperatura 30°C
                     [300, -5]])  # Día 300, temperatura -5°C

new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)

# Convertir predicciones a etiquetas
predicted_classes = np.argmax(predictions, axis=1)
predicted_labels = encoder.inverse_transform(predicted_classes)

# Mostrar resultados
for i, pred in enumerate(predicted_labels):
    print(f"Entrada: {new_data[i]} => Clima predicho: {pred}")
```

Tareas

1. Explica cómo los datos generados influyen en la predicción del modelo.
2. Modifica los rangos de temperatura y observa cómo cambian las predicciones.
3. Agrega una nueva característica, como la humedad, para mejorar el modelo.