Ejercicio:
Predicción del tráfico vehicular basándose en la hora y el día de la semana

Entrenar una red neuronal que pueda predecir el nivel de tráfico vehicular (bajo, moderado o alto) según la hora y el día de la semana.

---

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 horas aleatorias (de 0 a 23) y días de la semana (0 a 6)
hours = np.random.randint(0, 24, n_samples)
days = np.random.randint(0, 7, n_samples)  # 0=Lunes, 6=Domingo

# Generar niveles de tráfico
# Alto: hora punta (7-9, 17-19) en días laborales (0-4)
# Moderado: horas intermedias o fines de semana
# Bajo: madrugada (0-6 horas)
conditions = np.where(((hours >= 7) & (hours <= 9) | (hours >= 17) & (hours <= 19)) &
(days < 5), "alto",
                np.where((hours >= 10) & (hours <= 16) | (days >= 5), "moderado",
"bajo"))

# Crear el DataFrame
data = pd.DataFrame({
   "hora": hours,
   "dia_semana": days,
   "trafico": conditions
})

# Mostrar las primeras filas del DataFrame
print(data.head())
```

---

2. Preprocesar los datos

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

# Codificar el nivel de tráfico (bajo=0, moderado=1, alto=2)
encoder = LabelEncoder()
data['trafico_encoded'] = encoder.fit_transform(data['trafico'])

# Variables de entrada (hora y día de la semana) y salida (tráfico codificado)
X = data[['hora', 'dia_semana']].values
y = data['trafico_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(32, activation='relu', input_shape=(2,)),  # Entrada: hora y día de la
semana
  Dense(16, activation='relu'),
  Dense(3, activation='softmax')  # Salida: 3 clases (bajo, moderado, alto)
])

# 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([[8, 2],   # Hora 8:00, Martes
                     [15, 6],  # Hora 15:00, Domingo
                     [23, 0]])  # Hora 23:00, Lunes

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: Hora={new_data[i, 0]}, Día={new_data[i, 1]} => Tráfico predicho:
{pred}")
```

Tareas

1. Analiza cómo las horas y los días afectan las predicciones del modelo.
2. Modifica las condiciones de tráfico para agregar más categorías (por ejemplo, "extremo").
3. Incluye una nueva característica, como eventos especiales (1 si hay evento, 0 si no), para observar su impacto.