COLOMBIA POTENCIA DE LA VIDA

TIC

Construcción Redes Neuronales LSTM en Series Temporales

TALENTO TECH

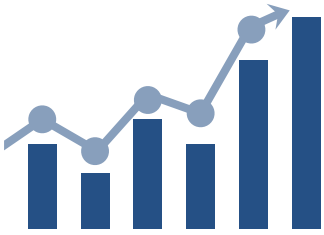
AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Modelado

Aprenderemos cómo implementar modelos de pronóstico utilizando redes neuronales LSTM en Keras, una popular biblioteca de aprendizaje profundo en Python.

Las redes neuronales LSTM son especialmente efectivas para capturar dependencias a largo plazo en datos secuenciales



Esto las convierte en una opción poderosa para el pronóstico de series de tiempo

Al finalizar esta actividad, los estudiantes estarán equipados con los conocimientos y habilidades necesarios para comprender, pronosticar e implementar modelos de pronóstico de series de tiempo utilizando redes neuronales LSTM en Keras.

Para construir un modelo de pronóstico de series de tiempo utilizando redes neuronales LSTM (Long Short-Term Memory) en Keras, primero se necesita tener claro el proceso general que seguirá nuestro modelo.

- **Preprocesamiento de datos**
- **Preparación de datos para LSTM**
- **Construcción del modelo LSTM**
- **Compilación del modelo**
- **Entrenamiento del modelo**
- **Evaluación del modelo**
- **Predicción**

El siguiente es un ejemplo básico para ilustrar el proceso, en la práctica, el procesamiento de datos y la construcción del modelo pueden ser mucho más complejos dependiendo de la naturaleza de los datos y los requisitos del problema. Además, es importante ajustar los hiperparámetros del modelo y validar su rendimiento adecuadamente.

Antes de empezar importamos las librería a usar:

```
from keras.models import Sequential
from keras.layers import LSTM, Dense
import numpy as np
import matplotlib.pyplot as plt
```

1 Preprocesamiento de datos:

Esto implica cargar los datos de la serie de tiempo, realizar cualquier limpieza necesaria y dividir los datos en conjuntos de entrenamiento y prueba.

```
# Paso 1: Preprocesamiento de datos (ejemplo básico)
# Supongamos que tenemos la siguiente serie de tiempo prueba
time_series = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
print("time_series: ",time_series)
train_size = int(len(time_series) * 0.6)  # Tamaño del conjunto de entrenamiento
train_data, test_data = time_series[:train_size], time_series[train_size:]
print("train_data:", train_data)
print("test_data",test_data)
```

Ejecutando esto se obtiene:

```
time_series:  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
train_data: [1 2 3 4 5 6 7 8 9]
test_data [10 11 12 13 14 15]
```

2 Preparación de datos para LSTM:

Dado que las redes LSTM requieren datos en un formato específico (es decir, secuencias), necesitamos transformar nuestros datos de series de tiempo en lotes de secuencias para que puedan ser consumidos por la red LSTM.

```
# Paso 2: Preparación de datos para LSTM (ejemplo básico)
# Por ejemplo, transformamos nuestros datos en secuencias
def create_dataset(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

# Definimos el número de pasos de tiempo para la serie temporal
seq_length = 4
# Creamos conjuntos de entrenamiento y prueba en formato 3D
X_train, y_train = create_dataset(train_data, seq_length)
X_test, y_test = create_dataset(test_data, seq_length)

# Imprimimos los conjuntos de datos en formato 3D
print("Conjunto de secuencias de entrenamiento:\n", X_train)
print("Etiquetas de entrenamiento:\n", y_train)
print("Conjunto de secuencias de prueba:\n", X_test)
print("Etiquetas de prueba:\n", y_test)
```

Ejecutando esto se obtiene:

```
Conjunto de secuencias de entrenamiento:
 [[1 2 3 4]
 [2 3 4 5]
 [3 4 5 6]
 [4 5 6 7]
 [5 6 7 8]]
Etiquetas de entrenamiento:
 [5 6 7 8 9]
Conjunto de secuencias de prueba:
 [[10 11 12 13]
 [11 12 13 14]]
Etiquetas de prueba:
 [14 15]
```

3 Construcción del modelo LSTM:

Luego, construimos el modelo LSTM en Keras, esto implica definir la arquitectura de la red, incluyendo el número de capas LSTM, el número de neuronas en cada capa, las funciones de activación, etc.

```
# Paso 3: Construcción del modelo LSTM
model = Sequential()
model.add(LSTM(16, activation='relu', input_shape=(seq_length, 1)))
model.add(Dense(1))
```

4 Compilación del modelo:

Después de construir el modelo, lo compilan especificando la función de pérdida, el optimizador y las métricas que se utilizarán para evaluar el rendimiento del modelo durante el entrenamiento.

```
# Paso 4: Compilacion del modelo
model.compile(optimizer='adam', loss='mse')
```

5 Entrenamiento del modelo:

A continuación, entrenan el modelo utilizando los datos de entrenamiento. Durante este paso, el modelo ajustará sus pesos para minimizar la función de pérdida especificada.

```
# Paso 5: Entrenamiento del modelo
model.fit(X_train, y_train, epochs=500)
```

6 Evaluación del modelo:

Una vez entrenado, evalúan el rendimiento del modelo utilizando nuestros datos de prueba para ver qué tan bien puede generalizar a datos no vistos.

```
# Paso 6: Evaluación del modelo
Evaluación = model.evaluate(X_test, y_test)
print(Evaluación)
```

7 Predicción:

Finalmente, usan el modelo entrenado para realizar predicciones sobre datos futuros y evalúan la precisión de estas predicciones.

```
# Paso 6: Predicción
print("X_test: \n",X_test)
print("y_test: \n",y_test)
y_pred = model.predict(X_test)
print("y_pred: \n",y_pred)
```

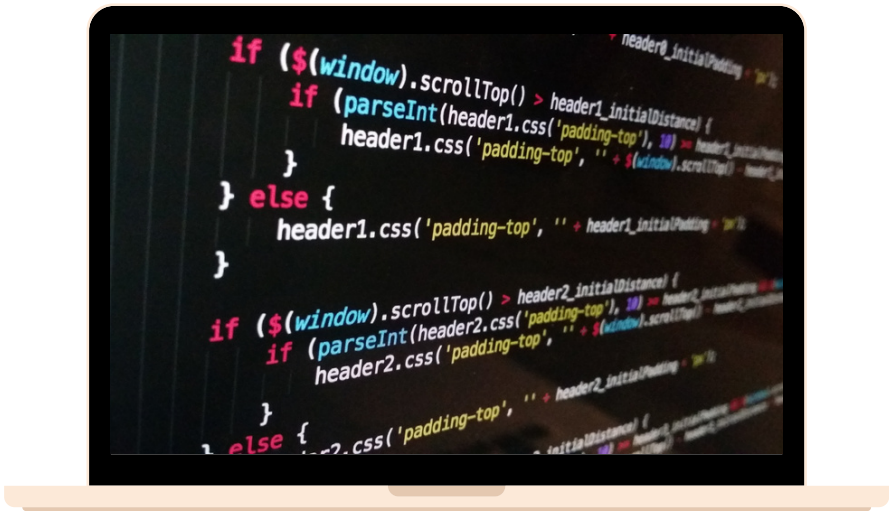
Ejecutando esto se obtiene:

```
X_test:
 [[10 11 12 13]
 [11 12 13 14]]
y_test:
 [14 15]
1/1 [==============================] - 0s 288ms/step
y_pred:
 [[13.988677]
 [15.085961]]
```

Actividad

para los estudiantes

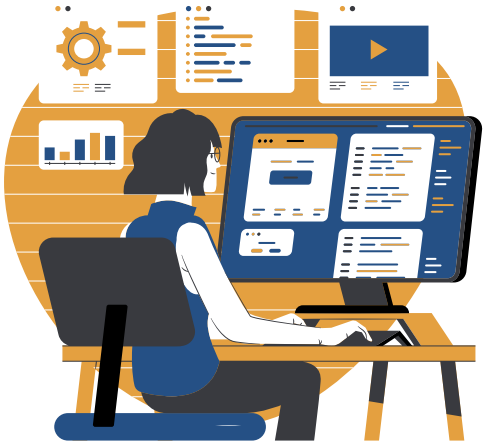
Preguntas relacionados con el código proporcionado



Preguntas de comprensión:

1. ¿Qué biblioteca se utiliza para implementar la red neuronal LSTM en Python?
2. ¿Cuál es el propósito del paso 1 en el código?
3. ¿Qué hace la función **create_dataset** en el paso 2?
4. ¿Qué significa el parámetro **seq_length** en la función **create_dataset**?
5. ¿Por qué es necesario transformar los datos en secuencias en el paso 2?

Ejercicios de exploración



1. Modifique el tamaño de la serie de tiempo (time_series) y observe cómo afecta a la predicción del modelo.
2. Experimente con diferentes valores para el parámetro seq_length y observe cómo afecta al desempeño del modelo.
3. Cambie la arquitectura de la red neuronal LSTM, por ejemplo, ajustando el número de unidades LSTM o la función de activación, y observe cómo afecta a la precisión del modelo.
4. Añada capas adicionales a la red neuronal LSTM y compare el desempeño del modelo con la arquitectura original.
5. Divida los datos de entrenamiento y pruebe de manera diferente (por ejemplo, 70% para entrenamiento y 30% para prueba) y observe cómo afecta al desempeño del modelo.

Estas preguntas y ejercicios ayudarán a los estudiantes a comprender mejor el funcionamiento de una red neuronal LSTM para predecir series de tiempo y les permitirán explorar diferentes aspectos del modelo para mejorar su comprensión y habilidades en el campo del aprendizaje profundo y el análisis de series temporales.