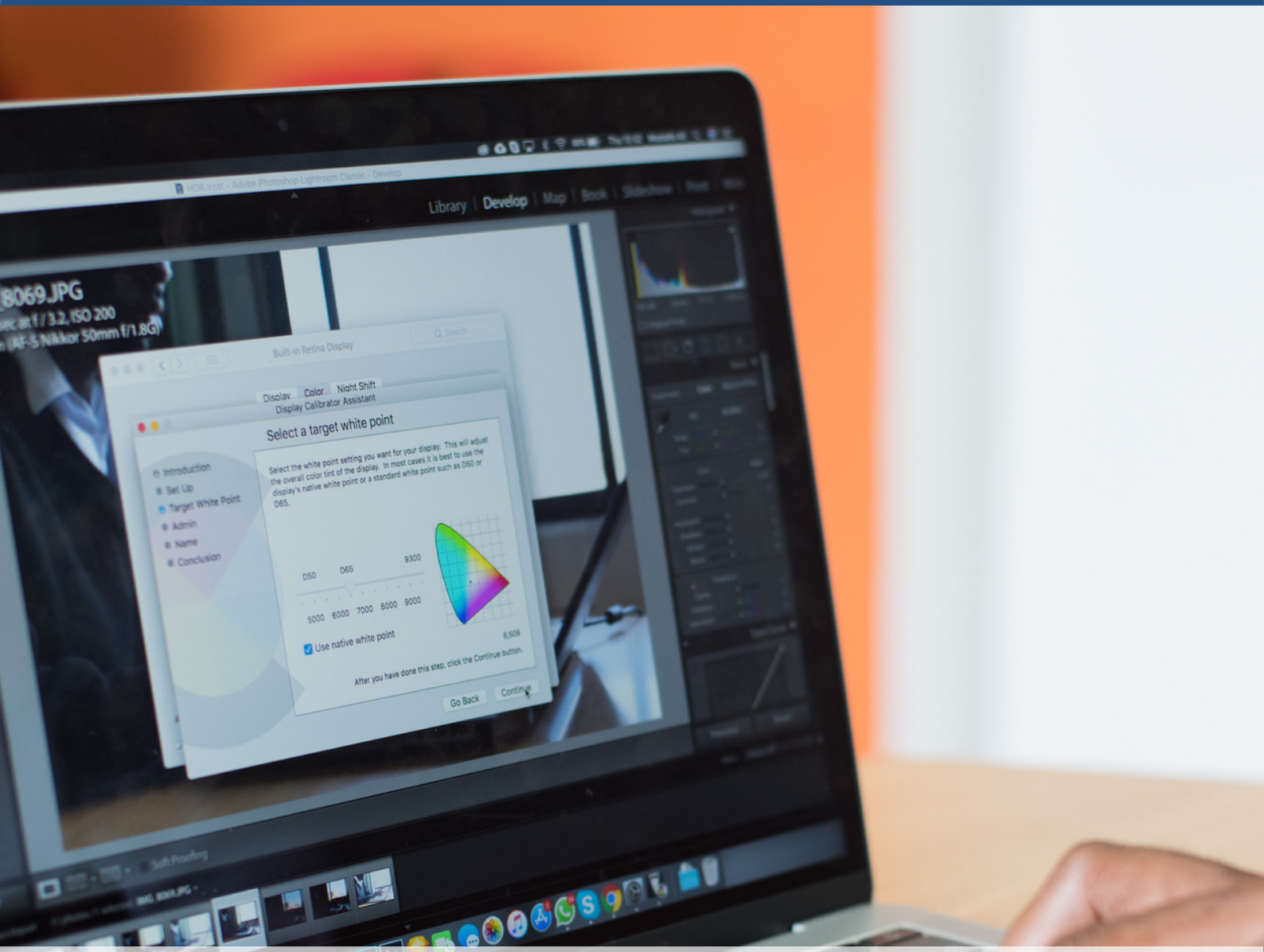
COLOMBIA POTENCIA DE LA VIDA

TIC




# Aplicación de Redes Neuronales LSTM en Series Temporales

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

# Aplicación de Redes Neuronales LSTM en Series Temporales

Ahora que los pasos para construir una red neuronal LSTM para pronosticar series de tiempo están claros, vamos a aplicarlos en un problema un poco más grande.

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

## 1. Generación de datos de serie de tiempo sintética:

Se generan datos de serie de tiempo sintética creciente con ruido aleatorio agregado.

```
# Generar datos para una serie de tiempo sintética
np.random.seed(0)  # Fijar semilla para reproducibilidad
n_samples = 200   # Número de puntos de datos en la serie de tiempo
t = np.arange(n_samples)  # Valores de tiempo

# Crear una serie de tiempo sintética (en este caso, creciente)
time_series = 0.1*t +np.random.randn(n_samples)
```

## 2. División de datos:

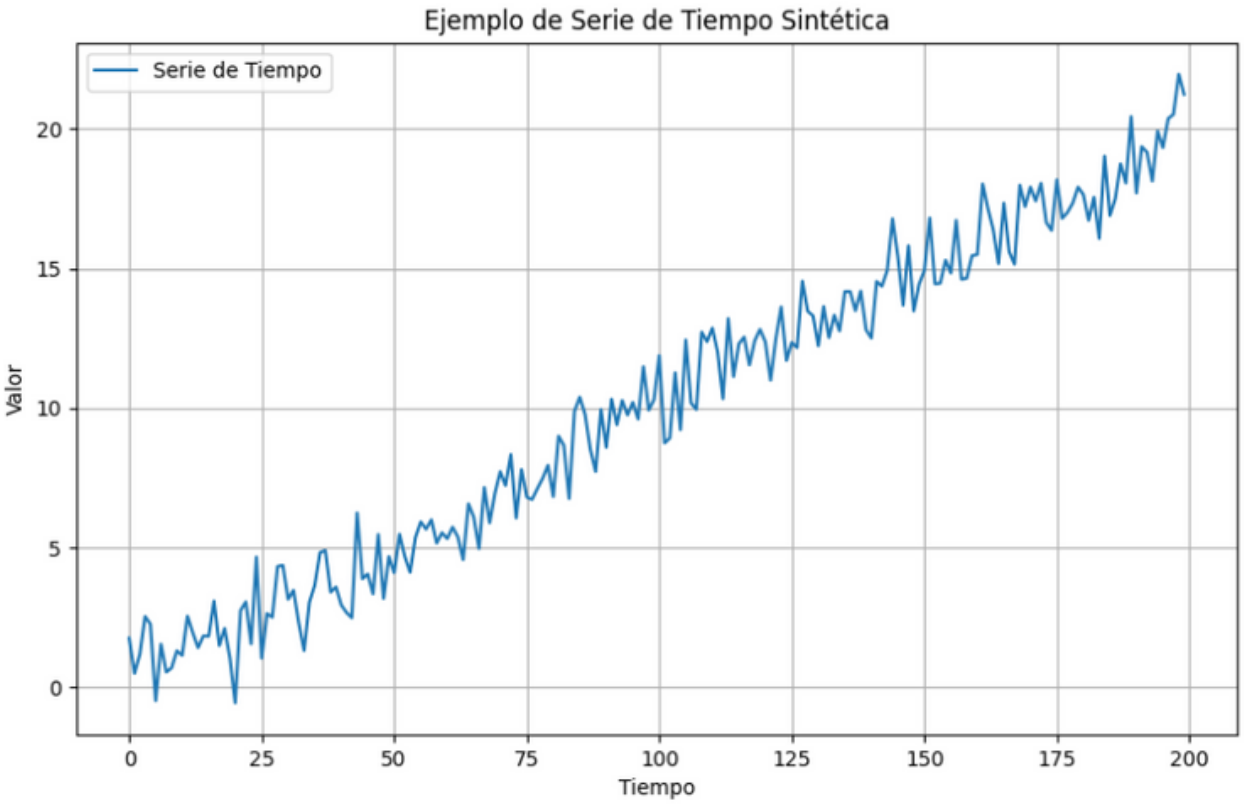
Los datos de serie de tiempo se dividen en conjuntos de entrenamiento y prueba, con un 80% de los datos destinados al entrenamiento y el 20% restante al prueba.

```
train_size = int(len(time_series) * 0.8)  # Tamaño del conjunto de entrenamiento
train_data, test_data = time_series[:train_size], time_series[train_size:]
```

## 3. Visualización de la serie de tiempo:

Se grafica la serie de tiempo sintética para visualizar los datos.

```
# Graficar la serie de tiempo
plt.figure(figsize=(10, 6))
plt.plot(t, time_series, label='Serie de Tiempo')
plt.title('Ejemplo de Serie de Tiempo Sintética')
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.legend()
plt.grid(True)
plt.show()
```



## 4. Preparación de datos para LSTM:

Se define una función llamada **create_dataset** que crea secuencias de datos de entrada y salida a partir de la serie de tiempo original, estas secuencias se utilizarán para entrenar el modelo LSTM.

```
# 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 = 16

# 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)
```

## 5. Construcción del modelo LSTM:

Se crea un modelo secuencial en Keras y se agregan capas LSTM con 32 unidades cada una, utilizando la función de activación ReLU. Se añade una capa densa para la salida.

```
# Construcción del modelo LSTM
model = Sequential()
model.add(LSTM(32,
               activation='relu',
               input_shape=(seq_length, 1),
               return_sequences=True))
model.add(LSTM(32, activation='relu'))
model.add(Dense(1))
```

## 6. Compilación del modelo:

El modelo se compila utilizando el optimizador Adam y la función de pérdida de error cuadrático medio (MSE).

```
model.compile(optimizer='adam', loss='mse')
```

## 7. Entrenamiento del modelo:

El modelo se entrena utilizando los datos de entrenamiento durante 100 épocas.

```
# Entrenamiento del modelo
model.fit(X_train, y_train, epochs=100)
```

## 8. Evaluación del modelo:

Se evalúa el desempeño del modelo utilizando los datos de prueba, calculando la pérdida obtenida por el modelo.

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

## 9. Predicción:

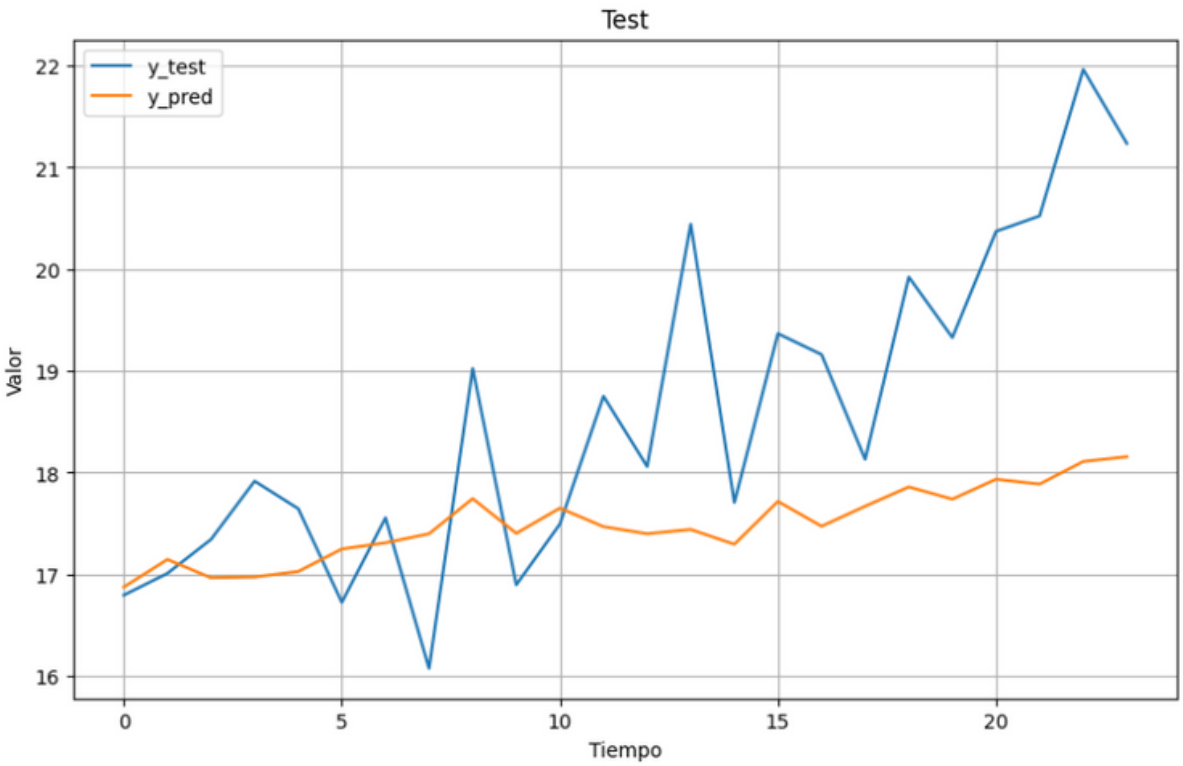
Se realizan predicciones utilizando el modelo entrenado en los datos de prueba.

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

## 10. Visualización de las predicciones:

Se grafican los datos de prueba (**y_test**) junto con las predicciones (**y_pred**) para comparar el rendimiento del modelo. Las dos series de tiempo se visualizan en el mismo gráfico para facilitar la comparación.

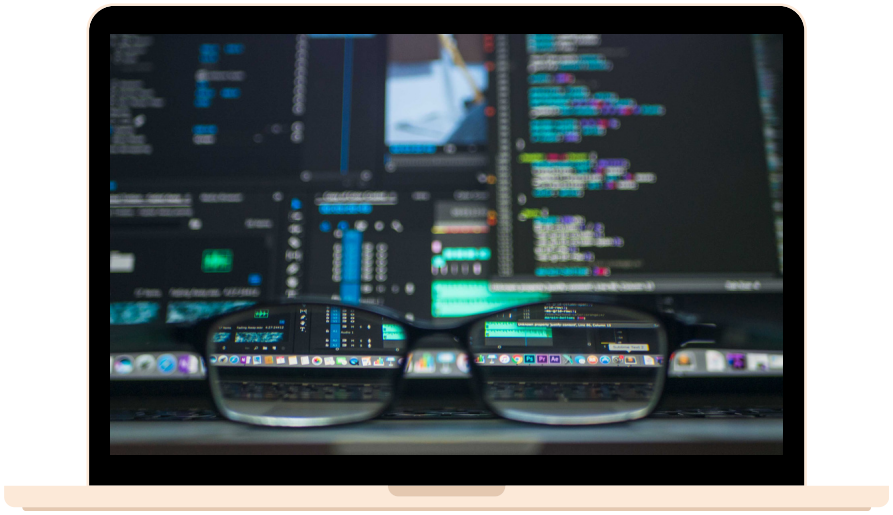
```
# Graficar de la data de test vs pronostico la serie de tiempo
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='y_test')
plt.plot(y_pred, label='y_pred')
plt.title('Test')
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.legend()
plt.grid(True)
plt.show()
```



# Actividad

## para los estudiantes

# Ejercicios y preguntas relacionadas con el ejercicio anterior proporcionado



## 1. Conceptos básicos de LSTM y series de tiempo:

- ¿Qué es una serie de tiempo y por qué es importante en el análisis de datos?
- Describa brevemente cómo funciona una red neuronal LSTM y cuál es su ventaja para modelar series de tiempo.

## 2. Preprocesamiento de datos:

- ¿Por qué es importante dividir los datos de serie de tiempo en conjuntos de entrenamiento y prueba?
- ¿Qué hace la función **create_dataset** en el código?

## 3. Construcción y entrenamiento del modelo:

- Explique la estructura del modelo LSTM creado en el código.
- ¿Qué significan el parámetros **epochs** en la función **fit**?

## 4. Evaluación del modelo:

- ¿Por qué es importante evaluar el modelo con datos de prueba después del entrenamiento?
- ¿Qué medida se utiliza para evaluar el rendimiento del modelo en este caso?

## 5. Predicción y visualización:

- ¿Qué representa la variable **y_pred** en el código?
- ¿Qué significan los gráficos resultantes de la predicción? ¿Qué conclusiones se pueden sacar de ellos?

## 6. Experimentación y ajuste del modelo:

- ¿Cómo podría mejorar el rendimiento del modelo LSTM en este caso?
- ¿Qué pasaría si aumentara el número de unidades LSTM o la cantidad de épocas de entrenamiento?

## 7. Aplicaciones prácticas:

- ¿Puede pensar en algunos escenarios del mundo real donde el pronóstico de series de tiempo con LSTM podría ser útil?
- ¿Cuáles son algunas industrias o campos que podrían beneficiarse especialmente de esta técnica?

## 8. Desafío adicional:

- Modifique los parámetros del modelo LSTM o la serie de tiempo de entrada y observe cómo cambian los resultados de la predicción.
- Intente implementar un modelo diferente para comparar su rendimiento con el LSTM. ¿Qué modelo obtiene mejores resultados y por qué?