```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lag
from pyspark.sql.window import Window
from pyspark.ml.feature import VectorAssembler, MinMaxScaler

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.metrics import mean_squared_error

# Initialize Spark session
spark = SparkSession.builder.appName("StockMarketAnalysis").getOrCreate()

# Load stock data
df = spark.read.csv("/content/infolimpioavanzadoTarget.csv", header=True, inferSchema=True)

# Display first few rows
df = df.select("Date", "Open", "High", "Low", "Close", "Volume")
df = df.orderBy("Date")

# Create lag feature: previous day's close
windowSpec = Window.orderBy("Date")
df = df.withColumn("Prev_Close", lag("Close").over(windowSpec))
df = df.na.drop()

# Feature vector assembly
feature_cols = ["Open", "High", "Low", "Volume", "Prev_Close"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
```

```
df = assembler.transform(df)

# Feature scaling
scaler = MinMaxScaler(inputCol="features", outputCol="scaled_features")
scaler_model = scaler.fit(df)
df = scaler_model.transform(df)

# Convert to Pandas
pandas_df = df.select("scaled_features", "Close").toPandas()

# Prepare X and y
X = np.array([np.array(x) for x in pandas_df["scaled_features"]])
y = pandas_df["Close"].values

# Train-test split
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Reshape for LSTM: (samples, time_steps, features)
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

# LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))

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

```
# Train the model
model.fit(X_train, y_train, epochs=1, batch_size=16, validation_data=(X_test, y_test), verbose=1)


# Predict
y_pred = model.predict(X_test)


# Evaluation
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)


# Plot results
plt.figure(figsize=(12, 6))
plt.plot(y_test, label='Actual Prices', color='blue')
plt.plot(y_pred, label='Predicted Prices', color='orange')
plt.title("Stock Price Prediction - Actual vs Predicted")
plt.xlabel("Time Steps")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```