Source code :

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split

# Load data from local CSV file
file_path = '/content/zomato.csv'
data_df = pd.read_csv(file_path)

# Display first few rows of the data
print(data_df.head())

# Use 'Close' price for prediction (assuming your CSV has a 'Close' column)
# If your CSV uses a different column name, change 'Close' to the appropriate column
data = data_df[['Close']]

# Normalize the data (Min-Max Scaling)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)

# Prepare data for LSTM (using the past 60 days to predict the next day's closing price)
def create_dataset(data, time_step=60):
    X, y = [], []
    for i in range(time_step, len(data)):
        X.append(data[i-time_step:i, 0])  # past 'time_step' days
        y.append(data[i, 0])  # next day's closing price
    return np.array(X), np.array(y)

X, y = create_dataset(scaled_data)

# Reshape data for LSTM [samples, time_steps, features]
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Build LSTM model
model = Sequential()

# First LSTM layer with 50 units, return sequences set to True to stack LSTM layers
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.2))

# Second LSTM layer
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))

# Fully connected output layer
model.add(Dense(units=1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Predict stock prices
predicted_prices = model.predict(X_test)

# Inverse transform to get the original scale
predicted_prices = scaler.inverse_transform(predicted_prices)
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

# Visualize the results
plt.figure(figsize=(14, 8))
plt.plot(y_test_actual, color='blue', label='Actual Price')
plt.plot(predicted_prices, color='red', label='Predicted Price')
plt.title('Price Prediction using LSTM')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()

# Save the model
model.save('zomato_price_lstm_model.h5')
print("Model saved as 'zomato_price_lstm_model.h5'.")

# Predict future price (next day) based on last 60 days of data
last_60_days = data[-60:].values
last_60_days_scaled = scaler.transform(last_60_days.reshape(-1, 1))

# Prepare input for prediction
X_input = last_60_days_scaled.reshape(1, -1, 1)

# Predict the next day's price
predicted_next_price = model.predict(X_input)
predicted_next_price = scaler.inverse_transform(predicted_next_price)

print(f"Predicted next day's price: {predicted_next_price[0][0]}")
```