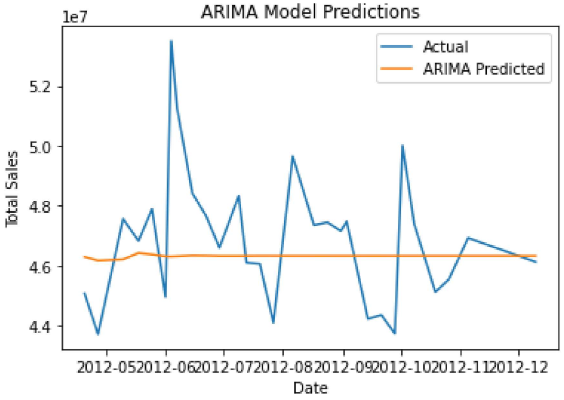
```
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('/content/Walmart.csv')

# Sort the data by date
df = df.sort_values(by=['Date'])

# Group the data by date and calculate total sales for each day
df['Date'] = pd.to_datetime(df['Date'])
df = df.groupby('Date')['Weekly_Sales'].sum().reset_index()

# Split the data into training and test sets
train_size = int(len(df) * 0.8)
train, test = df[:train_size], df[train_size:]

# Check for stationarity using ADF test
from statsmodels.tsa.stattools import adfuller
result = adfuller(train['Weekly_Sales'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
if result[1] > 0.05:
    print('Data is not stationary')
else:
    print('Data is stationary')

# ARIMA model
# Fit ARIMA model to the data
model = ARIMA(train['Weekly_Sales'].values, order=(5, 1, 0))
model_fit = model.fit()

# Generate predictions on test set
predictions_arima= model_fit.forecast(steps=len(test))

# Evaluate performance using metrics such as MSE and RMSE
mse_arima = mean_squared_error(test['Weekly_Sales'], predictions_arima)
rmse_arima = sqrt(mse_arima)
print('ARIMA RMSE: %.3f' % rmse_arima)



# Plot the ARIMA results
```

```
# Plot the ARIMA results
plt.plot(test['Date'], test['Weekly_Sales'], label='Actual')
plt.plot(test['Date'], predictions_arima, label='ARIMA Predicted')
plt.title('ARIMA Model Predictions')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.legend()
AR = plt.show()

```

```
ADF Statistic: -8.767550
p-value: 0.000000
Data is stationary
ARIMA RMSE: 2294028.669
```




```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout

#Load the Walmart sales dataset
sales_data = pd.read_csv('/content/Walmart.csv')

#Convert the date column to datetime format
sales_data['Date'] = pd.to_datetime(sales_data['Date'])

#Sort the data by date
sales_data = sales_data.sort_values('Date')

#Drop the Store and Holiday_Flag columns, as we are predicting for the total sales
sales_data = sales_data.drop(['Store', 'Holiday_Flag'], axis=1)

#Group the data by date and calculate the total sales for each day
daily_sales = sales_data.groupby('Date')['Weekly_Sales'].sum().reset_index()
```

```

#Split the data into train and test sets
train_size = int(len(daily_sales) * 0.8)
train_data = daily_sales[:train_size]['Weekly_Sales'].values.reshape(-1,1)
test_data = daily_sales[train_size:]['Weekly_Sales'].values.reshape(-1,1)

#Scale the data using MinMaxScaler
scaler = MinMaxScaler()
train_data_scaled = scaler.fit_transform(train_data)
test_data_scaled = scaler.transform(test_data)

#Define the RNN model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(train_data_scaled.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))

#Compile the model and fit it to the training data
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(train_data_scaled, train_data_scaled, epochs=100, batch_size=32)

#Generate predictions on the test data
test_inputs = test_data_scaled
test_inputs = np.reshape(test_inputs, (test_inputs.shape[0], test_inputs.shape[1], 1))
predicted_sales = model.predict(test_inputs)
predicted_sales = scaler.inverse_transform(predicted_sales)

#Plot the results
plt.figure(figsize=(10,6))
plt.plot(test['Date'],test_data, label='Actual')
plt.plot(test['Date'],predicted_sales, label='Predicted')
plt.legend()
plt.show()
```

```
Epoch 1/100
/usr/local/lib/python3.8/dist-packages/tensorflow/python/data/ops/structured_function.py:256: UserWarni
  warnings.warn(
4/4 [==============================] - 1s 190ms/step - loss: 0.0527
Epoch 2/100
4/4 [==============================] - 1s 190ms/step - loss: 0.0478
Epoch 3/100
4/4 [==============================] - 1s 183ms/step - loss: 0.0426
Epoch 4/100
4/4 [==============================] - 1s 194ms/step - loss: 0.0377
Epoch 5/100
4/4 [==============================] - 1s 131ms/step - loss: 0.0333
Epoch 6/100
4/4 [==============================] - 1s 132ms/step - loss: 0.0281
Epoch 7/100
4/4 [==============================] - 1s 137ms/step - loss: 0.0246
Epoch 8/100
4/4 [==============================] - 1s 149ms/step - loss: 0.0213
Epoch 9/100
4/4 [==============================] - 1s 120ms/step - loss: 0.0199
Epoch 10/100
4/4 [==============================] - 0s 103ms/step - loss: 0.0184
Epoch 11/100
4/4 [==============================] - 0s 96ms/step - loss: 0.0181
Epoch 12/100
4/4 [==============================] - 0s 93ms/step - loss: 0.0198
Epoch 13/100
4/4 [==============================] - 0s 92ms/step - loss: 0.0176
Epoch 14/100
4/4 [==============================] - 0s 95ms/step - loss: 0.0163
Epoch 15/100
4/4 [==============================] - 0s 94ms/step - loss: 0.0166
Epoch 16/100
4/4 [==============================] - 0s 96ms/step - loss: 0.0157
Epoch 17/100
4/4 [==============================] - 0s 90ms/step - loss: 0.0149
Epoch 18/100
4/4 [==============================] - 0s 95ms/step - loss: 0.0152
Epoch 19/100
4/4 [==============================] - 0s 102ms/step - loss: 0.0146
Epoch 20/100
4/4 [==============================] - 0s 97ms/step - loss: 0.0131
Epoch 21/100
4/4 [==============================] - 0s 101ms/step - loss: 0.0119
Epoch 22/100
4/4 [==============================] - 0s 100ms/step - loss: 0.0109
Epoch 23/100
4/4 [==============================] - 0s 97ms/step - loss: 0.0107
Epoch 24/100
4/4 [==============================] - 0s 101ms/step - loss: 0.0083
Epoch 25/100
4/4 [==============================] - 0s 100ms/step - loss: 0.0076
Epoch 26/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0050
Epoch 27/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0050
```

```
4/4 [==============================] - 0s 98ms/step - loss: 0.0030
Epoch 28/100
4/4 [==============================] - 0s 100ms/step - loss: 0.0025
Epoch 29/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0030
Epoch 30/100
4/4 [==============================] - 0s 95ms/step - loss: 0.0019
Epoch 31/100
4/4 [==============================] - 1s 163ms/step - loss: 0.0013
Epoch 32/100
4/4 [==============================] - 1s 259ms/step - loss: 9.8660e-04
Epoch 33/100
4/4 [==============================] - 2s 418ms/step - loss: 0.0014
Epoch 34/100
4/4 [==============================] - 1s 364ms/step - loss: 0.0016
Epoch 35/100
4/4 [==============================] - 1s 201ms/step - loss: 0.0024
Epoch 36/100
4/4 [==============================] - 1s 167ms/step - loss: 0.0015
Epoch 37/100
4/4 [==============================] - 1s 224ms/step - loss: 7.8172e-04
Epoch 38/100
4/4 [==============================] - 1s 102ms/step - loss: 0.0011
Epoch 39/100
4/4 [==============================] - 0s 96ms/step - loss: 0.0023
Epoch 40/100
4/4 [==============================] - 0s 94ms/step - loss: 0.0013
Epoch 41/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0014
Epoch 42/100
4/4 [==============================] - 0s 102ms/step - loss: 0.0014
Epoch 43/100
4/4 [==============================] - 0s 102ms/step - loss: 9.2370e-04
Epoch 44/100
4/4 [==============================] - 0s 100ms/step - loss: 9.9479e-04
Epoch 45/100
4/4 [==============================] - 0s 96ms/step - loss: 0.0014
Epoch 46/100
4/4 [==============================] - 0s 112ms/step - loss: 8.6688e-04
Epoch 47/100
4/4 [==============================] - 0s 99ms/step - loss: 7.4811e-04
Epoch 48/100
4/4 [==============================] - 0s 105ms/step - loss: 7.2715e-04
Epoch 49/100
4/4 [==============================] - 0s 106ms/step - loss: 8.5154e-04
Epoch 50/100
4/4 [==============================] - 0s 96ms/step - loss: 7.9618e-04
Epoch 51/100
4/4 [==============================] - 0s 108ms/step - loss: 4.5595e-04
Epoch 52/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0017
Epoch 53/100
4/4 [==============================] - 0s 109ms/step - loss: 0.0016
Epoch 54/100
4/4 [==============================] - 0s 99ms/step - loss: 8.3918e-04
Epoch 55/100
4/4 [==============================] - 0s 95ms/step - loss: 9.9422e-04
```

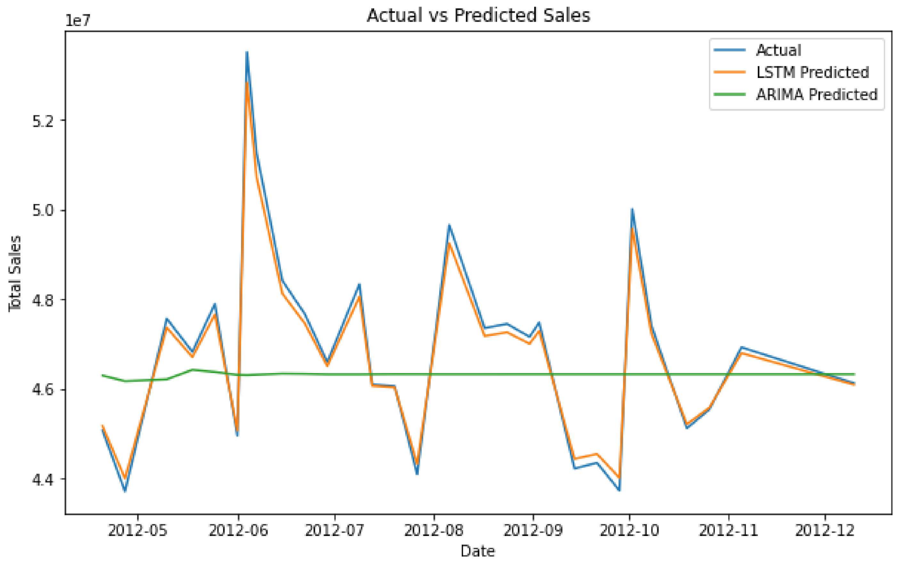
```
4/4 [==============================] - 0s 95ms/step - loss: 9.9422e-04
Epoch 56/100
4/4 [==============================] - 1s 148ms/step - loss: 6.0447e-04
Epoch 57/100
4/4 [==============================] - 1s 131ms/step - loss: 8.5996e-04
Epoch 58/100
4/4 [==============================] - 1s 148ms/step - loss: 7.8568e-04
Epoch 59/100
4/4 [==============================] - 1s 150ms/step - loss: 5.8693e-04
Epoch 60/100
4/4 [==============================] - 1s 142ms/step - loss: 7.7709e-04
Epoch 61/100
4/4 [==============================] - 0s 97ms/step - loss: 7.2330e-04
Epoch 62/100
4/4 [==============================] - 0s 102ms/step - loss: 9.4591e-04
Epoch 63/100
4/4 [==============================] - 0s 103ms/step - loss: 0.0015
Epoch 64/100
4/4 [==============================] - 0s 100ms/step - loss: 0.0014
Epoch 65/100
4/4 [==============================] - 0s 92ms/step - loss: 8.7571e-04
Epoch 66/100
4/4 [==============================] - 0s 92ms/step - loss: 0.0011
Epoch 67/100
4/4 [==============================] - 0s 96ms/step - loss: 0.0016
Epoch 68/100
4/4 [==============================] - 0s 100ms/step - loss: 5.0024e-04
Epoch 69/100
4/4 [==============================] - 0s 96ms/step - loss: 7.4762e-04
Epoch 70/100
4/4 [==============================] - 0s 94ms/step - loss: 5.9176e-04
Epoch 71/100
4/4 [==============================] - 0s 94ms/step - loss: 0.0012
Epoch 72/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0011
Epoch 73/100
4/4 [==============================] - 0s 98ms/step - loss: 5.0332e-04
Epoch 74/100
4/4 [==============================] - 0s 94ms/step - loss: 0.0018
Epoch 75/100
4/4 [==============================] - 0s 93ms/step - loss: 9.2534e-04
Epoch 76/100
4/4 [==============================] - 0s 91ms/step - loss: 7.9642e-04
Epoch 77/100
4/4 [==============================] - 0s 91ms/step - loss: 7.5608e-04
Epoch 78/100
4/4 [==============================] - 0s 88ms/step - loss: 0.0013
Epoch 79/100
4/4 [==============================] - 0s 90ms/step - loss: 5.0925e-04
Epoch 80/100
4/4 [==============================] - 0s 91ms/step - loss: 6.2419e-04
Epoch 81/100
4/4 [==============================] - 0s 86ms/step - loss: 4.9852e-04
Epoch 82/100
4/4 [==============================] - 0s 87ms/step - loss: 8.0134e-04
Epoch 83/100
```

```
4/4 [==============================] - 0s 89ms/step - loss: 9.3379e-04
Epoch 84/100
4/4 [==============================] - 0s 100ms/step - loss: 8.1607e-04
Epoch 85/100
4/4 [==============================] - 0s 95ms/step - loss: 6.8973e-04
Epoch 86/100
4/4 [==============================] - 0s 98ms/step - loss: 0.0013
Epoch 87/100
4/4 [==============================] - 1s 151ms/step - loss: 7.8305e-04
Epoch 88/100
4/4 [==============================] - 1s 132ms/step - loss: 5.2695e-04
Epoch 89/100
4/4 [==============================] - 1s 130ms/step - loss: 5.4740e-04
Epoch 90/100
4/4 [==============================] - 1s 150ms/step - loss: 6.8735e-04
Epoch 91/100
4/4 [==============================] - 1s 133ms/step - loss: 6.3957e-04
Epoch 92/100
4/4 [==============================] - 0s 99ms/step - loss: 0.0010
Epoch 93/100
4/4 [==============================] - 0s 94ms/step - loss: 4.2268e-04
Epoch 94/100
4/4 [==============================] - 0s 96ms/step - loss: 7.0202e-04
Epoch 95/100
4/4 [==============================] - 0s 94ms/step - loss: 5.5185e-04
Epoch 96/100
4/4 [==============================] - 0s 98ms/step - loss: 4.0964e-04
Epoch 97/100
4/4 [==============================] - 0s 104ms/step - loss: 2.7903e-04
Epoch 98/100
4/4 [==============================] - 0s 94ms/step - loss: 8.1451e-04
Epoch 99/100
4/4 [==============================] - 0s 99ms/step - loss: 4.2672e-04
Epoch 100/100
4/4 [==============================] - 0s 98ms/step - loss: 6.6720e-04
1/1 [==============================] - 0s 57ms/step
```


```
# Evaluate performance using metrics such as RMSE
mse_arima = mean_squared_error(test['Weekly_Sales'], predictions_arima)
rmse_arima = sqrt(mse_arima)
print('ARIMA RMSE: %.3f' % rmse_arima)

# Evaluate performance using metrics such as RMSE
mse_lstm = mean_squared_error(test_data, predicted_sales)
rmse_lstm = sqrt(mse_lstm)
print('LSTM RMSE: %.3f' % rmse_lstm)
```

```
ARIMA RMSE: 2294028.669
LSTM RMSE: 324160.427
```

```python
# Plot the results
plt.figure(figsize=(10,6))
plt.plot(test['Date'], test_data, label='Actual')
plt.plot(test['Date'], predicted_sales, label='LSTM Predicted')
plt.plot(test['Date'], predictions_arima, label='ARIMA Predicted')
plt.legend()
plt.title('Actual vs Predicted Sales')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()

```



0s completed at 00:06