In [ ]:

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

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/monthly-sunspots.csv"
df = pd.read_csv(url, usecols=[0,1], skiprows=1, names=['Month', 'Sunspots'], index_col=0)

scaler = MinMaxScaler(feature_range=(0, 1))
data = scaler.fit_transform(df)

def create_dataset(data, window_size):
    X, Y = [], []
    for i in range(len(data)-window_size):
        X.append(data[i:i+window_size])
        Y.append(data[i+window_size])
    return np.array(X), np.array(Y)

window_sizes = [5, 8, 12, 15]

# Split the dataset into training and testing sets
split_fraction = 0.8
split_index = int(split_fraction * len(data))
train_data = data[:split_index]
test_data = data[split_index:]

# Train and evaluate the models for each window size
for window_size in window_sizes:

    # Create the windowed dataset
    X_train, Y_train = create_dataset(train_data, window_size)
    X_test, Y_test = create_dataset(test_data, window_size)

    # Define the RNN model
    rnn_model = Sequential()
    rnn_model.add(SimpleRNN(units=3, activation='relu', input_shape=(window_size, 1)))
    rnn_model.add(Dense(units=1))
    rnn_model.compile(optimizer='adam', loss='mse')
    rnn_model.summary()

    # Train the RNN model
    rnn_history = rnn_model.fit(X_train, Y_train, epochs=100, batch_size=16, validation_split=0.2)

    # Define the LSTM model
    lstm_model = Sequential()
    lstm_model.add(LSTM(units=64, activation='relu', input_shape=(window_size, 1)))
    lstm_model.add(Dense(units=1))
    lstm_model.compile(optimizer='adam', loss='mse')
    lstm_model.summary()

    # Train the LSTM model
    lstm_history = lstm_model.fit(X_train, Y_train, epochs=100, batch_size=16, validation_split=0.

    # Evaluate the models
    rnn_train_score = rnn_model.evaluate(X_train, Y_train, verbose=0)
    rnn_test_score = rnn_model.evaluate(X_test, Y_test, verbose=0)
    lstm_train_score = lstm_model.evaluate(X_train, Y_train, verbose=0)
    lstm_test_score = lstm_model.evaluate(X_test, Y_test, verbose=0)

    print("Window size:", window_size)
    print("RNN training score:", rnn_train_score)
    print("RNN testing score:", rnn_test_score)
    print("LSTM training score:", lstm_train_score)
    print("LSTM testing score:", lstm_test_score)
    print()
```

```
Epoch 87/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0034 - val_loss: 0.0032
Epoch 88/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0033 - val_loss: 0.0032
Epoch 89/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0033 - val_loss: 0.0031
Epoch 90/100
112/112 [==============================] - 1s 11ms/step - loss: 0.0034 - val_loss: 0.0032
Epoch 91/100
112/112 [==============================] - 1s 13ms/step - loss: 0.0033 - val_loss: 0.0033
Epoch 92/100
112/112 [==============================] - 1s 12ms/step - loss: 0.0033 - val_loss: 0.0032
Epoch 93/100
112/112 [==============================] - 1s 11ms/step - loss: 0.0033 - val_loss: 0.0031
Epoch 94/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0033 - val_loss: 0.0032
Epoch 95/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0034 - val_loss: 0.0031
Epoch 96/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0033 - val_loss: 0.0032
Epoch 97/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0033 - val_loss: 0.0033
Epoch 98/100
112/112 [==============================] - 1s 11ms/step - loss: 0.0033 - val_loss: 0.0032
Epoch 99/100
112/112 [==============================] - 1s 10ms/step - loss: 0.0034 - val_loss: 0.0032
Epoch 100/100
112/112 [==============================] - 1s 11ms/step - loss: 0.0033 - val_loss: 0.0031
Window size: 15
RNN training score: 0.021588481962680817
RNN testing score: 0.06089344993233681
LSTM training score: 0.003228412475436926
LSTM testing score: 0.005670263897627592
```