In [843…

```
import os
import sys
from support_functions import *
from tensorflow import keras
from datetime import datetime
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean
from matplotlib import pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [844…

```
# configure
n_seq = 30
n_test = 200
n_lag = 120
n_epochs = 20
n_batch = 1
n_neurons = 120

#index to the test case you want to plot
test_index = n_test-1

#Configure dataset
path = '/Users/torefossland/Documents/MAC0460/project/dataset/GlobalTemperat
SELECTED = "LandAverageTemperature"
```

In [845…

```
df = pd.read_csv(path)

#string to date format
df['Month'] = pd.to_datetime(df['dt'],infer_datetime_format=True)
df = df.set_index(['Month'])
df=df.dropna(axis=0)
display(df)
series = df[SELECTED]
display(series)
```

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemp
Month				
1850-01-01	1850-01-01	0.749	1.105	
1850-02-01	1850-02-01	3.071	1.275	
1850-03-01	1850-03-01	4.954	0.955	
1850-04-01	1850-04-01	7.217	0.665	
1850-05-01	1850-05-01	10.004	0.617	
...	...	...	...	
2015-08-01	2015-08-01	14.755	0.072	
2015-09-01	2015-09-01	12.999	0.079	
2015-10-01	2015-10-01	10.801	0.102	
2015-11-01	2015-11-01	7.433	0.119	
2015-12-01	2015-12-01	5.518	0.100	

1992 rows × 9 columns

```
Month
1850-01-01     0.749
1850-02-01     3.071
1850-03-01     4.954
1850-04-01     7.217
1850-05-01    10.004
               ...
2015-08-01    14.755
2015-09-01    12.999
2015-10-01    10.801
2015-11-01     7.433
2015-12-01     5.518
Name: LandAverageTemperature, Length: 1992, dtype: float64
```

In [846…

```
#Function to create the input data and the corresponding labels to be predic
def split_sequence(sequence, n_steps_in, n_steps_out):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out
        # check if we are beyond the sequence
        if out_end_ix > len(sequence):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix:out_end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

In [847…

```
# transform series into train and test sets for supervised learning
def prepare_data(series, n_test, n_lag, n_seq):
        raw_values = series.values

        # transform to stationary data
        diff_series = series.diff().dropna()
        diff_values = diff_series.values
        diff_values = diff_values.reshape(len(diff_values), 1)

        # rescale values
        scaler = MinMaxScaler(feature_range=(-1, 1))
        scaled_values = scaler.fit_transform(diff_values)
        scaled_values = scaled_values.reshape(len(scaled_values), 1)

        # transform into supervised learning problem X, y
        X, y = split_sequence(scaled_values, n_lag, n_seq)
        X_train, y_train = X[:-n_test], y[:-n_test]
        X_test, y_test = X[-n_test:], y[-n_test:]
        return scaler, X_train, y_train, X_test, y_test

scaler, X_train, y_train, X_test, y_test = prepare_data(series, n_test, n_la
```

In [865…

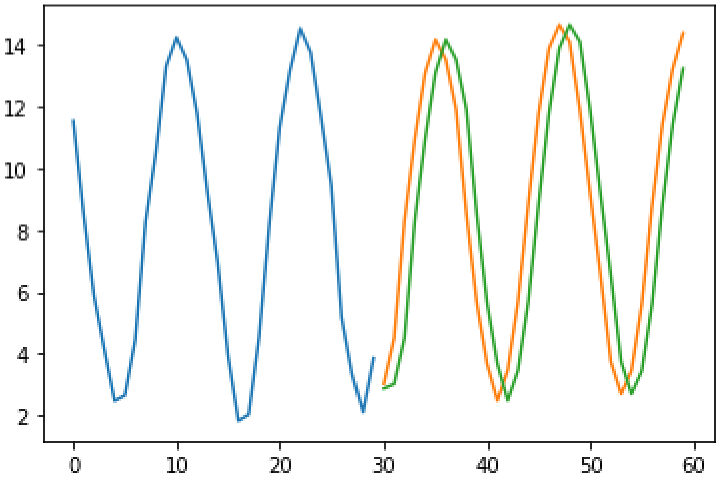
```
def evaluate_predictions(test, preds, n_lag, n_seq):
        rmse_list = []
        msle_list = []
        mae_list = []
        for i in range(n_seq):
            actual = [row[i] for row in test]
            predicted = [pred[i] for pred in preds]
            rmse = np.sqrt(mean_squared_error(actual, predicted))
            msle = mean_squared_log_error(actual, predicted)
            mae = mean_absolute_error(actual, predicted)
            rmse_list.append(rmse)
            msle_list.append(msle)
            mae_list.append(mae)
            print('t+%d RMSE: %f' % ((i+1), rmse), "\t", 't+%d MSLE: %f' % (
        return sum(rmse_list)/len(rmse_list), sum(msle_list)/len(msle_list),
```

In [869…

```
# split into train and test sets
values = pd.DataFrame(series)
values = values.set_index(series.index)

dataframe = pd.concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
print(dataframe.head(5))

X = dataframe.values
train_size = int(len(X) * 0.66)
train, test = X[1:train_size], X[train_size:]
train_X, train_y = train[:,0], train[:,1]
test_X, test_y = test[:,0], test[:,1]

# persistence model
def model_persistence(x):
    return x
# walk-forward validation
predictions = list()
for x in test_X:
        yhat = model_persistence(x)
        predictions.append(yhat)
```

```
# plot predictions and expected results
plt.plot(train_y[test_index:n_test+n_seq-1])
plt.plot([None for i in train_y[test_index:n_test+n_seq-1]] + [x for x in te
plt.plot([None for i in train_y[test_index:n_test+n_seq-1]] + [x for x in pr
plt.show()

rmse = np.sqrt(mean_squared_error(test_y, predictions))
msle = mean_squared_log_error(test_y, predictions)
mae = mean_absolute_error(test_y, predictions)
print('RMSE: ', rmse, "\t", 'MSLE: ', msle, "\t", 'MAE: ', mae)
```

```
              t-1     t+1
Month
1850-01-01    NaN   0.749
1850-02-01  0.749   3.071
1850-03-01  3.071   4.954
1850-04-01  4.954   7.217
1850-05-01  7.217  10.004
```



```
RMSE:  2.1662133904124867       MSLE:  0.06819431043262443      MAE:  1.921
379056047198
```

In [850…

```
# Create a simple LSTM model
def create_model(X_train, y_train, n_lag, n_seq, n_batch, nb_epoch, n_neuron
        X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
        # Create model topology
        model = Sequential()
        model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X_train.shape[
        model.add(Dense(units = y_train.shape[1]))
        if noise:
            print("Added gaussian noise to model. ")
            model.add(GaussianNoise(0.1))
        model.compile(loss='mean_squared_error', optimizer='adam', metrics=[
        return model
```

In [851…

```
# Make a single prediction
def predict_lstm(model, X, n_batch):
    X = X.reshape(1, 1, len(X))
    # make prediction
    pred = model.predict(X, batch_size=n_batch)
    return [x for x in pred[0, :]]
```

In [852…

```
#Function that loops through the test data and gets the predictions
def make_predictions(model, n_batch, X_test, n_lag, n_seq):
    preds = list()
    for i in range(X_test.shape[0]):
        pred = predict_lstm(model, X_test[i], n_batch)
        preds.append(pred)
    return preds
```

In [853…

```
# Undo the differencing done to make data stationary
def inverse_difference(last_ob, pred):
    pred_cumsum = pred.cumsum()
    inverted = [last_ob]*len(pred)
    inverted = [sum(x) for x in zip(inverted, pred_cumsum)]
    return inverted
```

In [854…

```
# Inversly transforms the data back to proper scale
def inverse_transform(series, preds, scaler, n_test):
    inverted = list()
    for i in range(len(preds)):
        # create array from pred
        pred = np.array(preds[i])
        pred = np.array(preds[i]).reshape(1, len(pred))

        # invert scaling
        inv_scale = scaler.inverse_transform(pred)
        inv_scale = inv_scale[0, :]

        # invert differencing
        index = len(series) - n_test + i - 1
        last_ob = series.values[index]
        inv_diff = inverse_difference(last_ob, inv_scale)
        # store
        inverted.append(inv_diff)
    return inverted
```

In [855…

```
# plot the predictions in the context of the original dataset
def plot_predictions(series, preds, n_test, test_index):
    assert test_index<len(preds) and test_index>=0, \
        f'Please input a valid test_index. Must be larger than 0 and less th

    x_lim = int(len(series) * 0.95)
    y_lim_upper = max(np.max(preds), np.max(series[x_lim:])) * 1.01
    y_lim_lower = min(np.min(series[x_lim:]), np.min(preds))*0.99

    # plot the entire dataset in blue
    plt.figure(figsize=(16,8))
    plt.plot(series.values, '-o', label='Actual')

    # plot the prediction
    start_index = len(series) - n_test + test_index - 1
    end_index = start_index + len(preds[test_index]) + 1
    xaxis = [x for x in range(start_index, end_index)]
    yaxis = [series.values[start_index]] + preds[test_index]
    plt.plot(xaxis, yaxis, '-o', color='orange', label='Predicted')

    # show the plot
    plt.axis([x_lim, len(series)+len(preds[0]), y_lim_lower, y_lim_upper])
    plt.legend()
    plt.show()
    return
```

In [856…

```
def make_prediction(X_train, y_train, n_lag, n_seq, n_batch, n_epochs, n_neu
    model = create_model(X_train, y_train, n_lag, n_seq, n_batch, n_epochs,

    #Have to rehsape the training data to fit the model [samples, 1, feature
    X_train_reshaped = X_train.reshape(X_train.shape[0], 1, X_train.shape[1]

    #Fit the model with the training data
    es = tf.keras.callbacks.EarlyStopping(patience=5, monitor='loss')
    for i in range(n_epochs):
        history = model.fit(X_train_reshaped, y_train, epochs=1, verbose=1,
        model.reset_states()

    # make predictions
    preds = make_predictions(model, n_batch, X_test, n_lag, n_seq)

    # inverse transform forecasts and test
    preds = inverse_transform(series, preds, scaler, n_test+n_seq-1)
    actual = [row.reshape(-1) for row in y_test]
    actual = inverse_transform(series, actual, scaler, n_test+n_seq-1)
    return actual, preds


actual, preds = make_prediction(X_train, y_train, n_lag, n_seq, n_batch, n_e
```

```
1642/1642 [==============================] - 1s 533us/step - loss: 0.0152 -
mean_absolute_error: 0.0895
1642/1642 [==============================] - 1s 537us/step - loss: 0.0101 -
mean_absolute_error: 0.0746
1642/1642 [==============================] - 1s 531us/step - loss: 0.0090 -
mean_absolute_error: 0.0712
1642/1642 [==============================] - 1s 531us/step - loss: 0.0086 -
mean_absolute_error: 0.0697
1642/1642 [==============================] - 1s 525us/step - loss: 0.0084 -
mean_absolute_error: 0.0687
1642/1642 [==============================] - 1s 527us/step - loss: 0.0082 -
mean_absolute_error: 0.0680
1642/1642 [==============================] - 1s 526us/step - loss: 0.0081 -
mean_absolute_error: 0.0674
1642/1642 [==============================] - 1s 529us/step - loss: 0.0079 -
mean_absolute_error: 0.0669
1642/1642 [==============================] - 1s 540us/step - loss: 0.0078 -
mean_absolute_error: 0.0665
1642/1642 [==============================] - 1s 532us/step - loss: 0.0077 -
mean_absolute_error: 0.0661
1642/1642 [==============================] - 1s 538us/step - loss: 0.0076 -
mean_absolute_error: 0.0657
1642/1642 [==============================] - 1s 536us/step - loss: 0.0076 -
mean_absolute_error: 0.0654
1642/1642 [==============================] - 1s 523us/step - loss: 0.0075 -
mean_absolute_error: 0.0650
1642/1642 [==============================] - 1s 528us/step - loss: 0.0074 -
mean_absolute_error: 0.0647
1642/1642 [==============================] - 1s 528us/step - loss: 0.0073 -
mean_absolute_error: 0.0644
1642/1642 [==============================] - 1s 524us/step - loss: 0.0072 -
mean_absolute_error: 0.0641
1642/1642 [==============================] - 1s 527us/step - loss: 0.0072 -
mean_absolute_error: 0.0638
1642/1642 [==============================] - 1s 526us/step - loss: 0.0071 -
mean_absolute_error: 0.0635
1642/1642 [==============================] - 1s 531us/step - loss: 0.0070 -
mean_absolute_error: 0.0632
1642/1642 [==============================] - 1s 526us/step - loss: 0.0070 -
mean_absolute_error: 0.0629
```

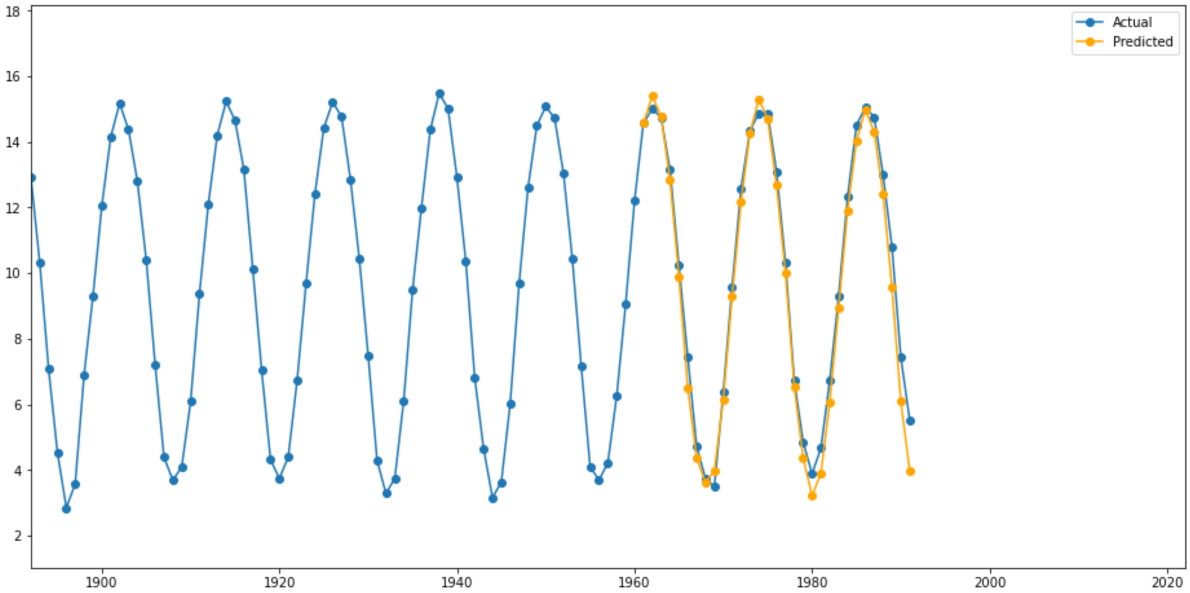
In [867…

```
# evaluate predictions
rmse_avg, msle_avg, mae_avg = evaluate_predictions(actual, preds, n_lag, n_s
print("rmse_avg: ",rmse_avg, "msle_avg: ", msle_avg,"mae_avg: ", mae_avg )
```

```
t+1 RMSE: 0.367811      t+1 MSLE: 0.003450      t+1 MAE: 0.281363
t+2 RMSE: 0.428876      t+2 MSLE: 0.004188      t+2 MAE: 0.335454
t+3 RMSE: 0.532505      t+3 MSLE: 0.005739      t+3 MAE: 0.403915
t+4 RMSE: 0.574479      t+4 MSLE: 0.005114      t+4 MAE: 0.421498
t+5 RMSE: 0.610699      t+5 MSLE: 0.004595      t+5 MAE: 0.424666
t+6 RMSE: 0.605909      t+6 MSLE: 0.004586      t+6 MAE: 0.415425
t+7 RMSE: 0.583887      t+7 MSLE: 0.005029      t+7 MAE: 0.433264
t+8 RMSE: 0.539991      t+8 MSLE: 0.004873      t+8 MAE: 0.415448
t+9 RMSE: 0.515914      t+9 MSLE: 0.005325      t+9 MAE: 0.411327
t+10 RMSE: 0.532353     t+10 MSLE: 0.005439     t+10 MAE: 0.403097
t+11 RMSE: 0.568601     t+11 MSLE: 0.006651     t+11 MAE: 0.400186
t+12 RMSE: 0.552522     t+12 MSLE: 0.009302     t+12 MAE: 0.392485
t+13 RMSE: 0.510662     t+13 MSLE: 0.008071     t+13 MAE: 0.384249
t+14 RMSE: 0.507149     t+14 MSLE: 0.006079     t+14 MAE: 0.393862
t+15 RMSE: 0.537629     t+15 MSLE: 0.005161     t+15 MAE: 0.433018
t+16 RMSE: 0.525889     t+16 MSLE: 0.004659     t+16 MAE: 0.428712
t+17 RMSE: 0.527424     t+17 MSLE: 0.004752     t+17 MAE: 0.413087
t+18 RMSE: 0.488741     t+18 MSLE: 0.004284     t+18 MAE: 0.376867
t+19 RMSE: 0.484175     t+19 MSLE: 0.004531     t+19 MAE: 0.372251
t+20 RMSE: 0.491875     t+20 MSLE: 0.004500     t+20 MAE: 0.385234
t+21 RMSE: 0.484699     t+21 MSLE: 0.004063     t+21 MAE: 0.380445
t+22 RMSE: 0.513318     t+22 MSLE: 0.004663     t+22 MAE: 0.382184
t+23 RMSE: 0.527114     t+23 MSLE: 0.005872     t+23 MAE: 0.371030
t+24 RMSE: 0.544426     t+24 MSLE: 0.009759     t+24 MAE: 0.362149
t+25 RMSE: 0.513475     t+25 MSLE: 0.010448     t+25 MAE: 0.354280
t+26 RMSE: 0.500584     t+26 MSLE: 0.005596     t+26 MAE: 0.389290
t+27 RMSE: 0.534977     t+27 MSLE: 0.005124     t+27 MAE: 0.451235
t+28 RMSE: 0.539630     t+28 MSLE: 0.005088     t+28 MAE: 0.433975
t+29 RMSE: 0.541186     t+29 MSLE: 0.005030     t+29 MAE: 0.426168
t+30 RMSE: 0.545673     t+30 MSLE: 0.005599     t+30 MAE: 0.423764
rmse_avg:  0.5244057082431859 msle_avg:  0.005585629831335857 mae_avg:  0.39
666431700730337
```

In [858…

```
# plot forecasts
plot_predictions(series, preds, n_test+n_seq-1, test_index)
```



In [859…

```
X_train_small = X_train[0:500]
y_train_small = y_train[0:500]
actual_small, preds_small = make_prediction(X_train_small, y_train_small, n_
500/500 [==============================] - 1s 560us/step - loss: 0.0292 - me
an_absolute_error: 0.1253
500/500 [==============================] - 0s 527us/step - loss: 0.0158 - me
an_absolute_error: 0.0937
500/500 [==============================] - 0s 536us/step - loss: 0.0137 - me
an_absolute_error: 0.0876
500/500 [==============================] - 0s 533us/step - loss: 0.0129 - me
an_absolute_error: 0.0852
500/500 [==============================] - 0s 536us/step - loss: 0.0124 - me
an_absolute_error: 0.0834
500/500 [==============================] - 0s 538us/step - loss: 0.0120 - me
an_absolute_error: 0.0820
500/500 [==============================] - 0s 538us/step - loss: 0.0117 - me
an_absolute_error: 0.0809
500/500 [==============================] - 0s 537us/step - loss: 0.0114 - me
an_absolute_error: 0.0798
500/500 [==============================] - 0s 527us/step - loss: 0.0112 - me
an_absolute_error: 0.0789
500/500 [==============================] - 0s 534us/step - loss: 0.0110 - me
an_absolute_error: 0.0781
500/500 [==============================] - 0s 537us/step - loss: 0.0108 - me
an_absolute_error: 0.0777
500/500 [==============================] - 0s 537us/step - loss: 0.0106 - me
an_absolute_error: 0.0771
500/500 [==============================] - 0s 546us/step - loss: 0.0103 - me
an_absolute_error: 0.0761
500/500 [==============================] - 0s 532us/step - loss: 0.0101 - me
an_absolute_error: 0.0753
500/500 [==============================] - 0s 533us/step - loss: 0.0098 - me
an_absolute_error: 0.0743
500/500 [==============================] - 0s 529us/step - loss: 0.0096 - me
an_absolute_error: 0.0737
500/500 [==============================] - 0s 572us/step - loss: 0.0093 - me
an_absolute_error: 0.0728
500/500 [==============================] - 0s 532us/step - loss: 0.0091 - me
an_absolute_error: 0.0719
500/500 [==============================] - 0s 532us/step - loss: 0.0089 - me
an_absolute_error: 0.0712
500/500 [==============================] - 0s 544us/step - loss: 0.0086 - me
an_absolute_error: 0.0702
```

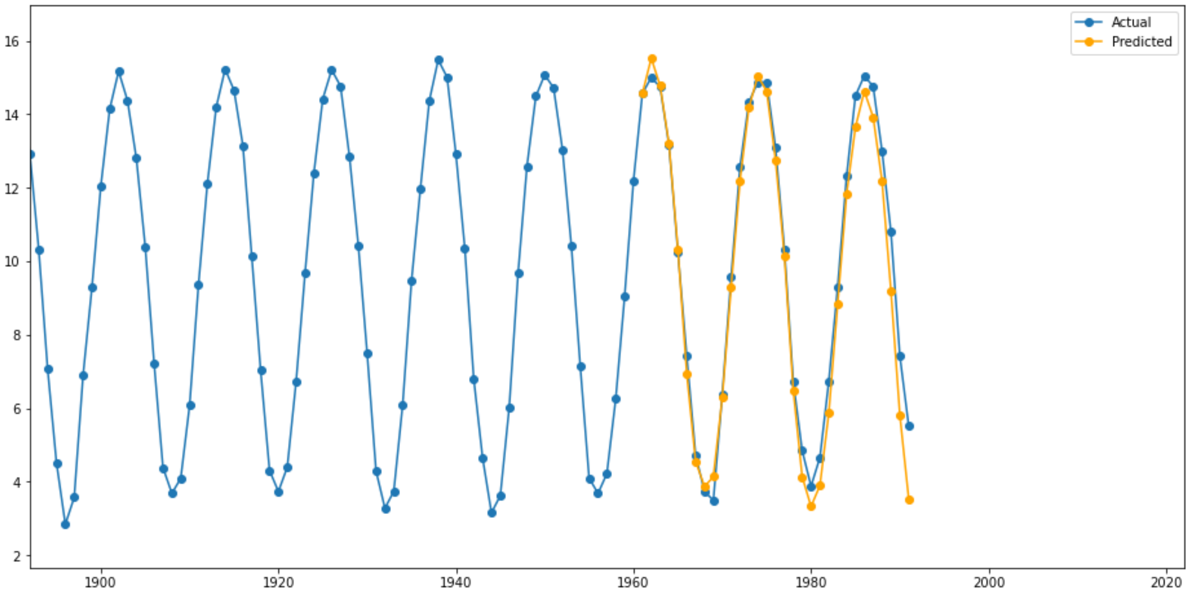
In [870…

```
# evaluate predictions
rmse_avg, msle_avg, mae_avg = evaluate_predictions(actual_small, preds_small
print("rmse_avg: ",rmse_avg, "msle_avg: ", msle_avg,"mae_avg: ", mae_avg )
```

```
t+1 RMSE: 0.389343       t+1 MSLE: 0.003752      t+1 MAE: 0.296569
t+2 RMSE: 0.461608       t+2 MSLE: 0.004332      t+2 MAE: 0.362169
t+3 RMSE: 0.522457       t+3 MSLE: 0.005300      t+3 MAE: 0.397003
t+4 RMSE: 0.525298       t+4 MSLE: 0.004856      t+4 MAE: 0.396546
t+5 RMSE: 0.590671       t+5 MSLE: 0.005472      t+5 MAE: 0.429351
t+6 RMSE: 0.617039       t+6 MSLE: 0.006380      t+6 MAE: 0.449587
t+7 RMSE: 0.536233       t+7 MSLE: 0.005046      t+7 MAE: 0.417170
t+8 RMSE: 0.534686       t+8 MSLE: 0.005396      t+8 MAE: 0.418516
t+9 RMSE: 0.514941       t+9 MSLE: 0.005480      t+9 MAE: 0.400526
t+10 RMSE: 0.554063      t+10 MSLE: 0.007266     t+10 MAE: 0.412052
t+11 RMSE: 0.568340      t+11 MSLE: 0.008047     t+11 MAE: 0.441944
t+12 RMSE: 0.609760      t+12 MSLE: 0.008185     t+12 MAE: 0.482826
t+13 RMSE: 0.608896      t+13 MSLE: 0.007644     t+13 MAE: 0.499742
t+14 RMSE: 0.578722      t+14 MSLE: 0.006476     t+14 MAE: 0.469307
t+15 RMSE: 0.606924      t+15 MSLE: 0.006186     t+15 MAE: 0.479585
t+16 RMSE: 0.638543      t+16 MSLE: 0.006577     t+16 MAE: 0.492356
t+17 RMSE: 0.649719      t+17 MSLE: 0.007913     t+17 MAE: 0.500780
t+18 RMSE: 0.632499      t+18 MSLE: 0.008470     t+18 MAE: 0.493064
t+19 RMSE: 0.576063      t+19 MSLE: 0.007365     t+19 MAE: 0.451912
t+20 RMSE: 0.581845      t+20 MSLE: 0.006637     t+20 MAE: 0.453019
t+21 RMSE: 0.589461      t+21 MSLE: 0.006022     t+21 MAE: 0.476804
t+22 RMSE: 0.650459      t+22 MSLE: 0.007058     t+22 MAE: 0.529454
t+23 RMSE: 0.703410      t+23 MSLE: 0.009223     t+23 MAE: 0.573475
t+24 RMSE: 0.763907      t+24 MSLE: 0.009191     t+24 MAE: 0.625356
t+25 RMSE: 0.825153      t+25 MSLE: 0.011007     t+25 MAE: 0.673190
t+26 RMSE: 0.862610      t+26 MSLE: 0.014035     t+26 MAE: 0.704281
t+27 RMSE: 0.914145      t+27 MSLE: 0.019293     t+27 MAE: 0.750920
t+28 RMSE: 0.933381      t+28 MSLE: 0.018887     t+28 MAE: 0.735517
t+29 RMSE: 0.929807      t+29 MSLE: 0.019298     t+29 MAE: 0.749349
t+30 RMSE: 0.914996      t+30 MSLE: 0.021257     t+30 MAE: 0.757616
rmse_avg:  0.6461658858374068 msle_avg:  0.00873502646349514 mae_avg:  0.510
6661998678844
```

In [861…

```
# plot forecasts
plot_predictions(series, preds_small, n_test+n_seq-1, test_index)
```



In [862…

```
actual_noise, preds_noise = make_prediction(X_train, y_train, n_lag, n_seq,
Added gaussian noise to model.
1642/1642 [==============================] - 1s 545us/step - loss: 0.0275 -
mean_absolute_error: 0.1283
1642/1642 [==============================] - 1s 536us/step - loss: 0.0219 -
mean_absolute_error: 0.1165
1642/1642 [==============================] - 1s 532us/step - loss: 0.0203 -
mean_absolute_error: 0.1125
1642/1642 [==============================] - 1s 534us/step - loss: 0.0198 -
mean_absolute_error: 0.1110
1642/1642 [==============================] - 1s 534us/step - loss: 0.0195 -
mean_absolute_error: 0.1102
1642/1642 [==============================] - 1s 543us/step - loss: 0.0190 -
mean_absolute_error: 0.1087
1642/1642 [==============================] - 1s 540us/step - loss: 0.0189 -
mean_absolute_error: 0.1084
1642/1642 [==============================] - 1s 532us/step - loss: 0.0189 -
mean_absolute_error: 0.1084
1642/1642 [==============================] - 1s 540us/step - loss: 0.0188 -
mean_absolute_error: 0.1080
1642/1642 [==============================] - 1s 535us/step - loss: 0.0187 -
mean_absolute_error: 0.1077
1642/1642 [==============================] - 1s 536us/step - loss: 0.0188 -
mean_absolute_error: 0.1083
1642/1642 [==============================] - 1s 527us/step - loss: 0.0183 -
mean_absolute_error: 0.1069
1642/1642 [==============================] - 1s 525us/step - loss: 0.0183 -
mean_absolute_error: 0.1068
1642/1642 [==============================] - 1s 557us/step - loss: 0.0183 -
mean_absolute_error: 0.1066
1642/1642 [==============================] - 1s 548us/step - loss: 0.0184 -
mean_absolute_error: 0.1071
1642/1642 [==============================] - 1s 532us/step - loss: 0.0184 -
mean_absolute_error: 0.1069
1642/1642 [==============================] - 1s 530us/step - loss: 0.0185 -
mean_absolute_error: 0.1069
1642/1642 [==============================] - 1s 550us/step - loss: 0.0184 -
mean_absolute_error: 0.1068
1642/1642 [==============================] - 1s 530us/step - loss: 0.0182 -
mean_absolute_error: 0.1062
1642/1642 [==============================] - 1s 526us/step - loss: 0.0182 -
mean_absolute_error: 0.1063
```

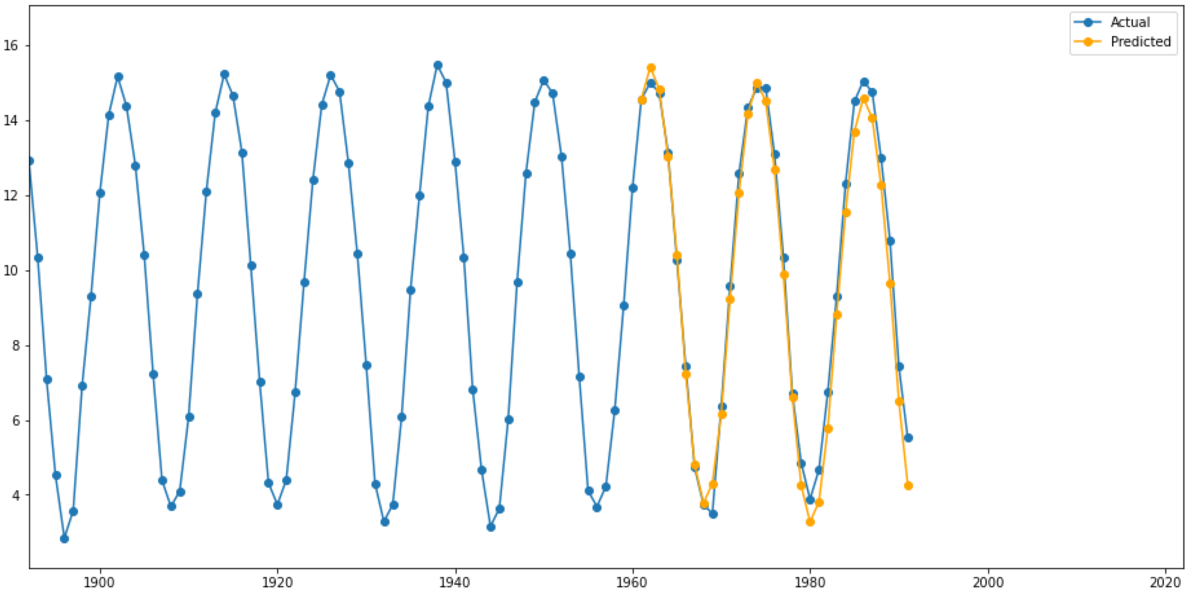
In [868…

```
# evaluate predictions
rmse_avg, msle_avg, mae_avg = evaluate_predictions(actual_noise, preds_noise
print("rmse_avg: ",rmse_avg, "msle_avg: ", msle_avg,"mae_avg: ", mae_avg )
```

```
t+1 RMSE: 0.372933      t+1 MSLE: 0.003467      t+1 MAE: 0.288346
t+2 RMSE: 0.388316      t+2 MSLE: 0.003113      t+2 MAE: 0.304853
t+3 RMSE: 0.452880      t+3 MSLE: 0.004219      t+3 MAE: 0.367058
t+4 RMSE: 0.459978      t+4 MSLE: 0.004084      t+4 MAE: 0.370360
t+5 RMSE: 0.504910      t+5 MSLE: 0.004327      t+5 MAE: 0.412873
t+6 RMSE: 0.550695      t+6 MSLE: 0.004475      t+6 MAE: 0.438308
t+7 RMSE: 0.534835      t+7 MSLE: 0.004343      t+7 MAE: 0.416915
t+8 RMSE: 0.516602      t+8 MSLE: 0.004852      t+8 MAE: 0.402290
t+9 RMSE: 0.556630      t+9 MSLE: 0.007071      t+9 MAE: 0.436670
t+10 RMSE: 0.567380     t+10 MSLE: 0.007947     t+10 MAE: 0.451414
t+11 RMSE: 0.583092     t+11 MSLE: 0.007964     t+11 MAE: 0.452528
t+12 RMSE: 0.587370     t+12 MSLE: 0.006966     t+12 MAE: 0.456067
t+13 RMSE: 0.593897     t+13 MSLE: 0.005818     t+13 MAE: 0.472970
t+14 RMSE: 0.565733     t+14 MSLE: 0.004888     t+14 MAE: 0.456495
t+15 RMSE: 0.579737     t+15 MSLE: 0.004526     t+15 MAE: 0.476051
t+16 RMSE: 0.658694     t+16 MSLE: 0.005350     t+16 MAE: 0.541801
t+17 RMSE: 0.661924     t+17 MSLE: 0.006191     t+17 MAE: 0.533662
t+18 RMSE: 0.659800     t+18 MSLE: 0.008358     t+18 MAE: 0.522438
t+19 RMSE: 0.632205     t+19 MSLE: 0.010700     t+19 MAE: 0.493147
t+20 RMSE: 0.599366     t+20 MSLE: 0.011411     t+20 MAE: 0.462802
t+21 RMSE: 0.631405     t+21 MSLE: 0.012037     t+21 MAE: 0.491923
t+22 RMSE: 0.659616     t+22 MSLE: 0.009481     t+22 MAE: 0.528889
t+23 RMSE: 0.693853     t+23 MSLE: 0.007893     t+23 MAE: 0.577667
t+24 RMSE: 0.669659     t+24 MSLE: 0.005907     t+24 MAE: 0.573254
t+25 RMSE: 0.737767     t+25 MSLE: 0.006975     t+25 MAE: 0.634752
t+26 RMSE: 0.777312     t+26 MSLE: 0.007343     t+26 MAE: 0.675291
t+27 RMSE: 0.751891     t+27 MSLE: 0.007512     t+27 MAE: 0.633869
t+28 RMSE: 0.736867     t+28 MSLE: 0.007945     t+28 MAE: 0.624203
t+29 RMSE: 0.724275     t+29 MSLE: 0.008989     t+29 MAE: 0.595878
t+30 RMSE: 0.731941     t+30 MSLE: 0.010709     t+30 MAE: 0.604940
rmse_avg:  0.6047187341551716 msle_avg:  0.006828717097695854 mae_avg:  0.48
992387559688094
```

In [864…

```
# plot forecasts
plot_predictions(series, preds_noise, n_test+n_seq-1, test_index)
```


Actual
Predicted