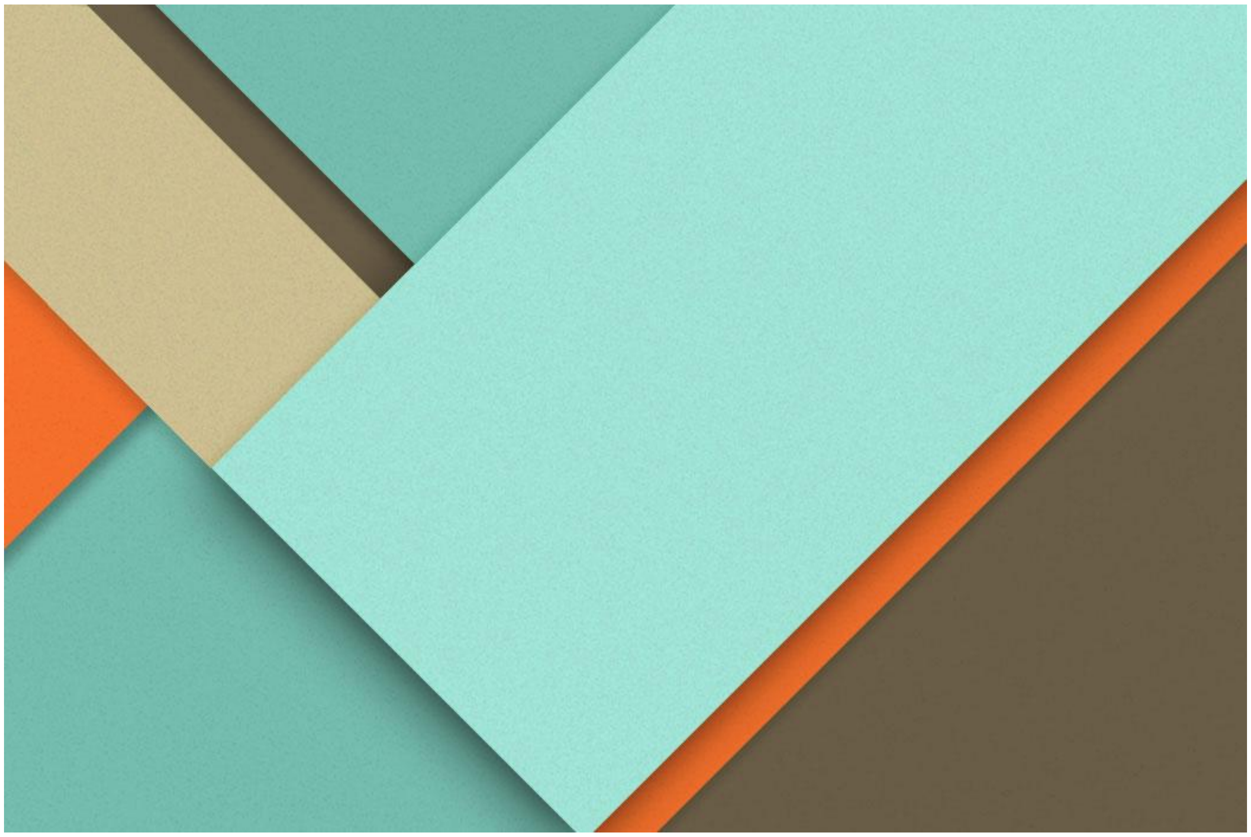


Ocean Wave Prediction

01.07.2023

Amit Samui
Kalyani Government Engineering College
Kalyani, Nadia
Report for Ocean Wave Prediction using LSTM

Overview

Predicting wave behaviors is an important tool for the safety of ship navigation and offshore operations.

In order to predict wave behaviors we need to collect wave data. Wave monitoring buoys are used for collecting the data. Wave monitoring buoys continuously measure the wave height, wave period and wave direction.

As the wave monitoring buoy floats up and down each passing wave, its motion is measured and electronically processed. Data from the wave monitoring buoys are transmitted to a nearby receiver station as a radio signal.

Measured and derived 2021 wave data in this study is collected by oceanographic wave measuring buoys anchored at from the Townsville wave monitoring site.

My aim is to create a Long Short Time Memory (LSTM) deep learning model to predict future wave behaviors. In order to achieve this I will divide the data set into two parts (train and test). Every 30 samples from the training data will be investigated and the 31st sample features will be predicted by the model. Once the model is properly trained, test data will be used as real time wave data. And lastly I will compare these real data samples and my model's predictions to evaluate the model's success.

The wave data used in this study contains following features:

- Date/Time: Date and 30 minute wave record
- Hs : The significant wave height (in meters), defined as the average of the highest one-third of wave heights in a 30 minute wave record.
- Hmax: The height (in meters) of the highest single wave in a wave record.
- Tz: The average of the zero up-crossing wave periods (in seconds) in a wave record.
- Tp: This is the wave period (in seconds) of those waves that are producing the most energy in a wave record.
- Peak Direction: The direction that peak waves are coming from, shown in degrees from true north.
- SST: The sea surface temperature at the wave monitoring buoy, in degrees Celsius.

What is LSTM (RNN)

In recent years, the field of deep learning has witnessed significant advancements, leading to breakthroughs in various domains such as computer vision, natural language processing, and time series analysis. Among the many deep learning architectures, the Long Short-Term Memory (LSTM) network, a type of Recurrent Neural Network (RNN), has emerged as a powerful tool for sequential data analysis, particularly in the realm of time series mining.

Time series data, characterized by its sequential nature and temporal dependencies, can be found in numerous domains, including finance, weather forecasting, signal processing, and healthcare. The ability to accurately model and predict future values or patterns in time series data holds immense value for decision-making, anomaly detection, and forecasting applications.

The LSTM network has garnered attention due to its unique memory cell structure, which enables it to capture long-term dependencies and mitigate the limitations of traditional RNNs, such as the vanishing gradient problem. By selectively retaining or forgetting information over extended sequences, LSTMs excel at capturing temporal dynamics, making them ideal for modeling and analyzing time series data.

This report explores the application of LSTM networks for time series mining and analysis. We delve into the architecture of LSTMs, dissecting their key components, including input gates, forget gates, output gates, and memory cells. Furthermore, we investigate the benefits of LSTM networks over conventional RNNs, highlighting their ability to handle sequences of varying lengths and effectively capture intricate dependencies between past and future observations.

We also examine the practical applications of LSTM networks in time series mining, encompassing tasks such as time series forecasting, anomaly detection, sequence classification, and natural language processing. Through concrete examples and case studies, we showcase the remarkable performance and versatility of LSTM networks in diverse time series analysis scenarios.

Overall, this report aims to provide a comprehensive understanding of LSTM networks in the context of time series mining. By elucidating their architecture, advantages, and real-world applications, we hope to equip readers with the knowledge and insights necessary to leverage the power of LSTM networks for effective analysis and prediction of time series data.

Forecasting on real dataset

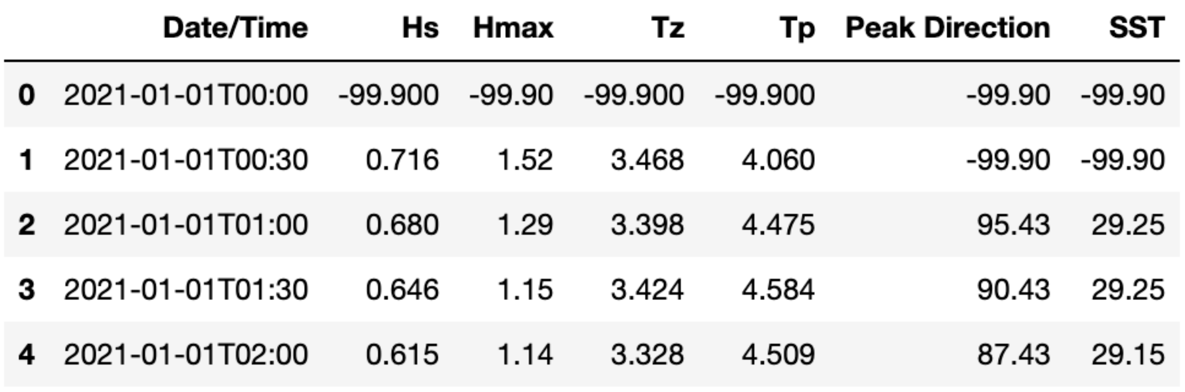
Import necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

Importing Dataset

```
df = pd.read_csv('./wavesData.csv')
df.head()
```

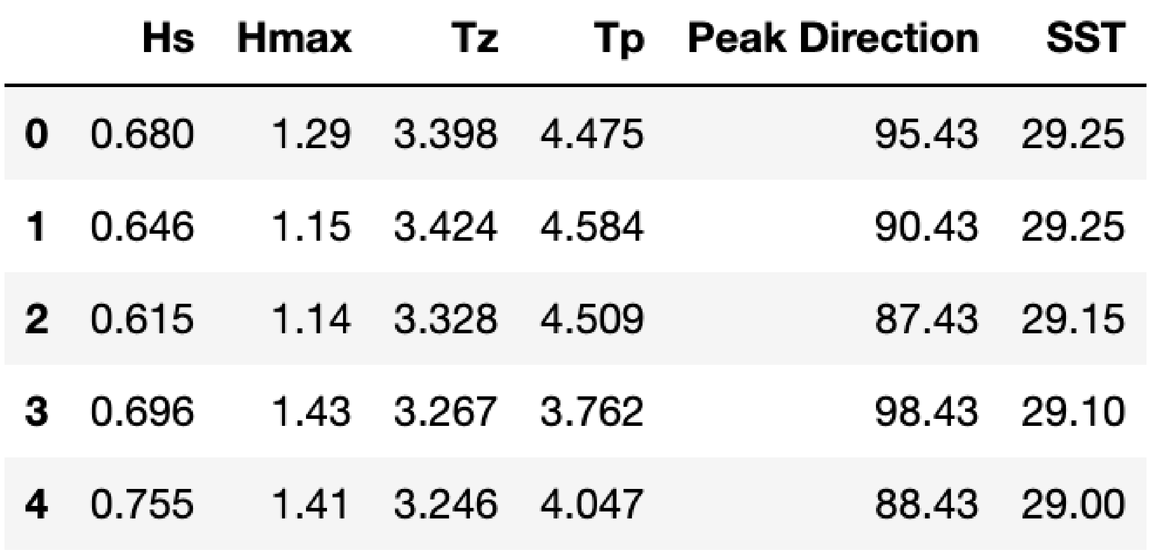
|:



	Date/Time	Hs	Hmax	Tz	Tp	Peak Direction	SST
0	2021-01-01T00:00	-99.900	-99.90	-99.900	-99.900	-99.90	-99.90
1	2021-01-01T00:30	0.716	1.52	3.468	4.060	-99.90	-99.90
2	2021-01-01T01:00	0.680	1.29	3.398	4.475	95.43	29.25
3	2021-01-01T01:30	0.646	1.15	3.424	4.584	90.43	29.25
4	2021-01-01T02:00	0.615	1.14	3.328	4.509	87.43	29.15

Data Cleaning

```
df.replace(-99.90, np.nan, inplace=True)
df.drop('Date/Time', axis=1, inplace=True)
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
df.head()
```



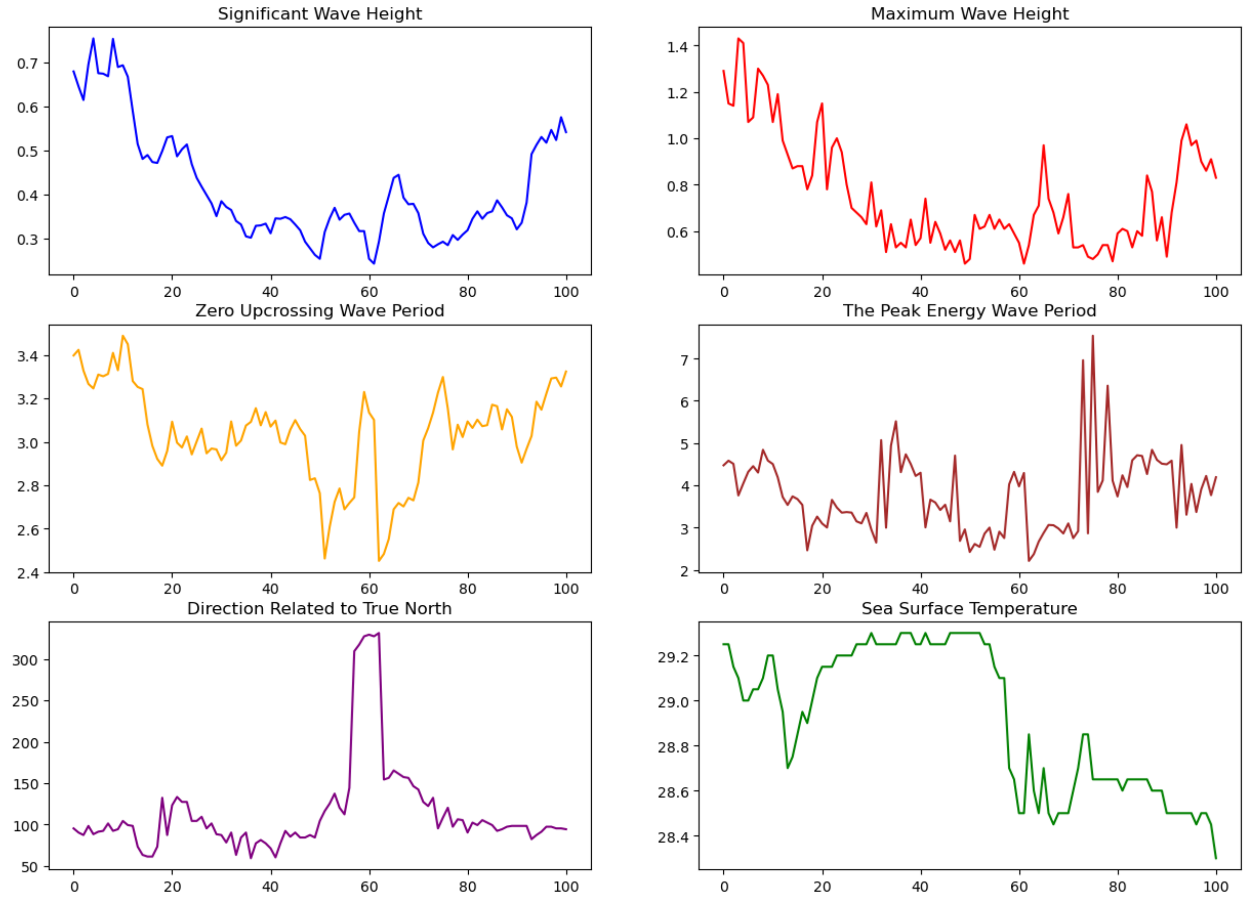
	Hs	Hmax	Tz	Tp	Peak Direction	SST
0	0.680	1.29	3.398	4.475	95.43	29.25
1	0.646	1.15	3.424	4.584	90.43	29.25
2	0.615	1.14	3.328	4.509	87.43	29.15
3	0.696	1.43	3.267	3.762	98.43	29.10
4	0.755	1.41	3.246	4.047	88.43	29.00

```
df.shape
```

```
(11813, 6)
```

Visualization of the Features

```
df_graph = df.loc[0:100]

plt.figure(figsize=(15,22))
plt.subplot(6,2,1)
plt.plot(df_graph['Hs'], color='blue')
plt.title('Significant Wave Height')

plt.subplot(6,2,2)
plt.plot(df_graph['Hmax'], color='red')
plt.title('Maximum Wave Height')

plt.subplot(6,2,3)
plt.plot(df_graph['Tz'], color='orange')
plt.title('Zero Upcrossing Wave Period')

plt.subplot(6,2,4)
plt.plot(df_graph['Tp'], color='brown')
plt.title('The Peak Energy Wave Period')

plt.subplot(6,2,5)
plt.plot(df_graph['Peak Direction'], color='purple')
plt.title('Direction Related to True North')

plt.subplot(6,2,6)
plt.plot(df_graph['SST'], color='green')
plt.title('Sea Surface Temperature')
plt.show();
```



Dataset Info

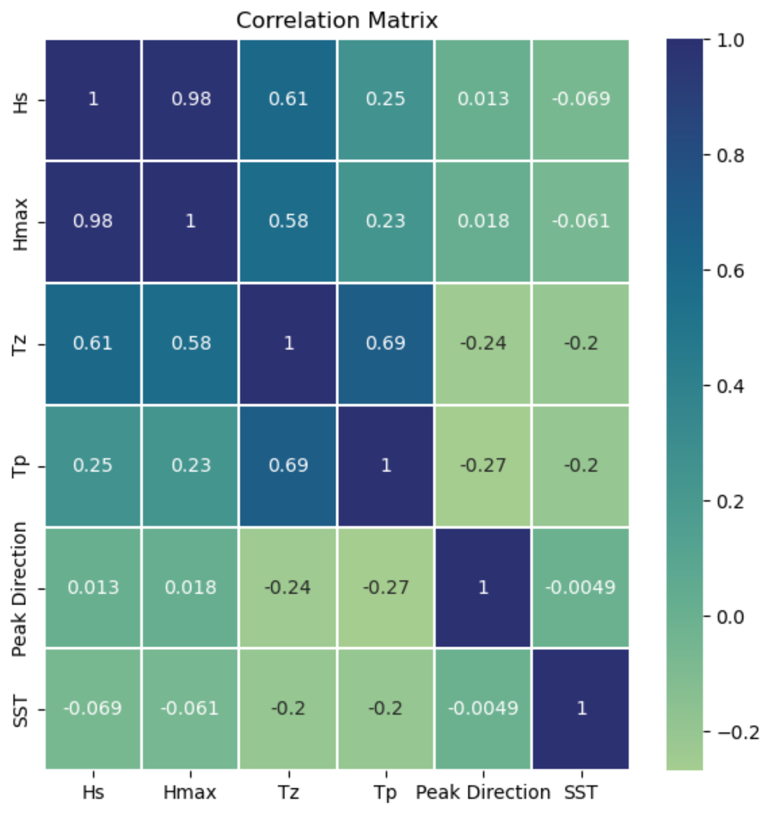
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11813 entries, 0 to 11812
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Hs              11813 non-null  float64
 1   Hmax            11813 non-null  float64
 2   Tz              11813 non-null  float64
 3   Tp              11813 non-null  float64
 4   Peak Direction  11813 non-null  float64
 5   SST             11813 non-null  float64
dtypes: float64(6)
memory usage: 553.9 KB
```

```
df.describe()
```

	Hs	Hmax	Tz	Tp	Peak Direction	SST
count	11813.000000	11813.000000	11813.000000	11813.000000	11813.000000	11813.000000
mean	0.605250	1.063846	3.556560	4.845144	91.268099	26.210188
std	0.309347	0.543180	0.480091	1.541205	56.453256	2.442190
min	0.108000	0.160000	2.296000	1.702000	0.410000	22.000000
25%	0.357000	0.630000	3.196000	3.619000	67.410000	23.550000
50%	0.535000	0.950000	3.540000	4.686000	87.430000	27.000000
75%	0.801000	1.400000	3.876000	5.806000	94.430000	28.350000
max	2.327000	5.110000	5.593000	12.623000	358.430000	31.750000

```
plt.figure(figsize=(7,7))
sns.heatmap(df.corr(), linewidth=.1, annot=True, cmap='crest')
plt.title('Correlation Matrix')
plt.show();
```



	Hs	Hmax	Tz	Tp	Peak Direction	SST
Hs	1	0.98	0.61	0.25	0.013	-0.069
Hmax	0.98	1	0.58	0.23	0.018	-0.061
Tz	0.61	0.58	1	0.69	-0.24	-0.2
Tp	0.25	0.23	0.69	1	-0.27	-0.2
Peak Direction	0.013	0.018	-0.24	-0.27	1	-0.0049
SST	-0.069	-0.061	-0.2	-0.2	-0.0049	1

FEATURE SCALING

```
from sklearn.preprocessing import MinMaxScaler

# Scaling all the values between 0 and 1
scaler = MinMaxScaler(feature_range=(0,1))
data = scaler.fit_transform(df)
print('Shape of the scaled data matrix: ', data.shape)
```

```
Shape of the scaled data matrix:  (11813, 6)
```

```
# Separate data into 2 groups for train and test
import math

partition = math.floor(len(data) * 0.8)
train = data[:partition,]
test = data[partition: ,]

# Shapes of our datasets
print('Shape of train data: ', train.shape)
print('Shape of test data: ', test.shape)
```

```
Shape of train data:  (9450, 6)
Shape of test data:  (2363, 6)
```

```
# Separate every 30 samples as the input and get the 31st sample as the
output.
def prepare_data(data):
    databatch = 30
    x_list = []
    y_list = []

    for i in range(len(data)-databatch-1):
        x_list.append(data[i:i+databatch])
        y_list.append(data[i+databatch+1])

    X_data = np.array(x_list)
    X_data = np.reshape(X_data, (X_data.shape[0], X_data.shape[2],
X_data.shape[1]))
```

```
    y_data = np.array(y_list)

    return X_data, y_data
```

```
# Executing the separation
X_train, y_train = prepare_data(train)
X_test, y_test = prepare_data(test)
print('X_train Shape : ', X_train.shape, 'y_train shape :', y_train.shape)
print('X_test Shape  : ', X_test.shape, ' y_test shape  :', y_test.shape)
```

```
X_train Shape :  (9419, 6, 30) y_train shape : (9419, 6)
X_test Shape  :  (2332, 6, 30)  y_test shape  : (2332, 6)
```

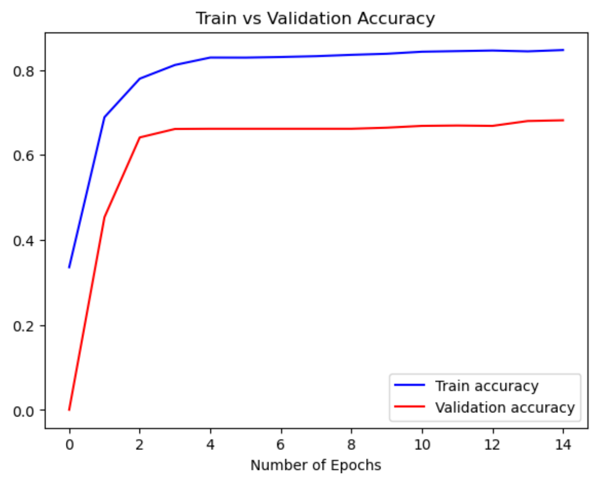
CREATING LSTM MODEL

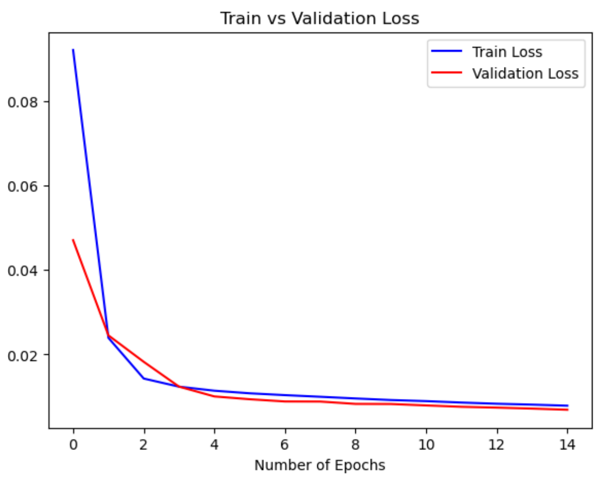
```
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

def lstm_model(x_data, y_data, num_epochs, batch_size, learning_rate):
    # Creating the model
    model = Sequential()
    # Adding the first layer
    model.add(LSTM(32, input_shape=(x_data.shape[1], x_data.shape[2]),
return_sequences=True))
    # Adding the second layer
    model.add(LSTM(16, return_sequences=True))
    # Adding a dropout value in order to prevent overfiting
    model.add(Dropout(0.2))
     # Adding the third layer
    model.add(LSTM(10))
    # Adding the output layer. 6 nodes are selected because the data has 6
features
    model.add(Dense(6))

    # Choosing the optimizer
    optimizer = Adam(lr=learning_rate)

    # Compiling the model
    model.compile(loss='mean_squared_error', optimizer=optimizer,
metrics=['accuracy'])

    # Fitting the model
    history = model.fit(x_data, y_data, validation_split=0.25,
epochs=num_epochs, batch_size=batch_size)

    return model, history
```

```
model,history = lstm_model(X_train, y_train, num_epochs=15, batch_size=200,
learning_rate=.001)
```

Visualization of the Learning

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], color='blue', label='Train accuracy')
plt.plot(history.history['val_accuracy'], color='red', label='Validation
accuracy')
plt.title('Train vs Validation Accuracy')
plt.xlabel('Number of Epochs')
plt.legend()
plt.subplot(1,2,2)
plt.plot(history.history['loss'], color='blue', label='Train Loss')
plt.plot(history.history['val_loss'], color='red', label='Validation Loss')
plt.title('Train vs Validation Loss')
plt.xlabel('Number of Epochs')
plt.legend()
plt.show();
```





This means that the loss is stable after 4 epochs.

Predictions

```
# Defining function to predict data
def predicting(data, y_real):
    predicted_data = model.predict(data)
    # Invert scaling process to get the normal values range for the
features
    predicted_data = scaler.inverse_transform(predicted_data)
    y_real = scaler.inverse_transform(y_real)

    return predicted_data, y_real
```

```
# Executing predictions
train_prediction, y_train = predicting(X_train, y_train)
test_prediction, y_test = predicting(X_test, y_test)
```

```
295/295 [==============================] - 1s 1ms/step
73/73 [==============================] - 0s 1ms/step
```

```
# Defining function to investigate the root of mean squared errors (RMSE)
between predicted and real data

import math
from sklearn.metrics import mean_squared_error
```

```
def examine_rmse(y_data, predicted_data):
    Score_Hs = math.sqrt(mean_squared_error(y_data[:,0],
predicted_data[:,0]))
    Score_Hmax = math.sqrt(mean_squared_error(y_data[:,1],
predicted_data[:,1]))
    Score_Tz = math.sqrt(mean_squared_error(y_data[:,2],
predicted_data[:,2]))
    Score_Tp = math.sqrt(mean_squared_error(y_data[:,3],
predicted_data[:,3]))
    Score_Dir = math.sqrt(mean_squared_error(y_data[:,4],
predicted_data[:,4]))
    Score_SST = math.sqrt(mean_squared_error(y_data[:,5],
predicted_data[:,5]))

    print('RMSE_Hs       : ', Score_Hs)
    print('RMSE_Hmax     : ', Score_Hmax)
    print('RMSE_Tz       : ', Score_Tz)
    print('RMSE_Tp       : ', Score_Tp)
    print('RMSE_Direction: ', Score_Dir)
    print('RMSE_SST      : ', Score_SST)
```

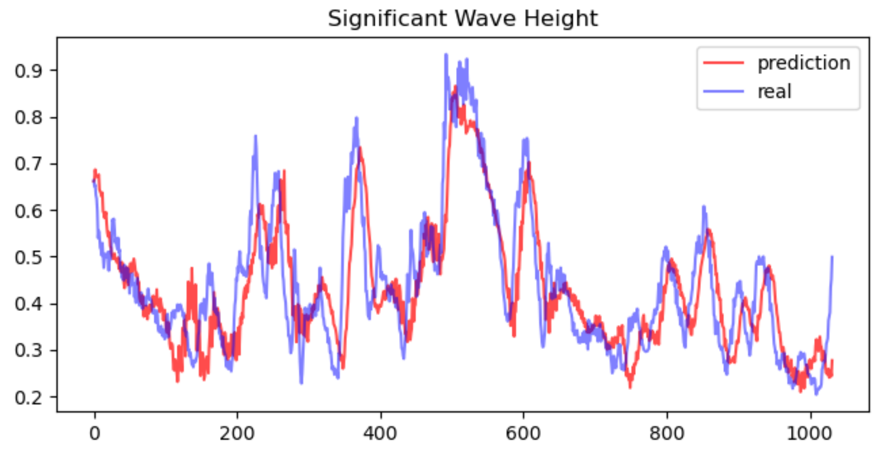
```
# Executing the RMSE comparison
print('Trainin Data Errors')
print(examine_rmse(y_train, train_prediction),'\n')
print('Test Data Errors')
print(examine_rmse(y_test, test_prediction))
```

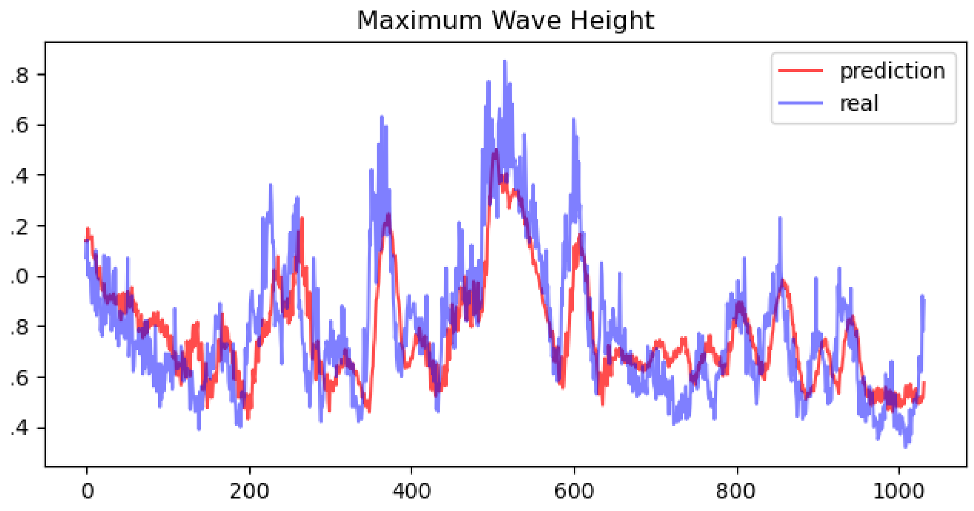
```
Training Data Errors
RMSE_Hs       :  0.12657461309084536
RMSE_Hmax     :  0.23269163194634063
RMSE_Tz       :  0.2714532914368115
RMSE_Tp       :  1.2541744450710208
RMSE_Direction:  44.28924857394634
RMSE_SST      :  0.4013841449429407
None

Test Data Errors
RMSE_Hs       :  0.1411486310868869
RMSE_Hmax     :  0.22614091098549188
RMSE_Tz       :  0.3654937499137811
RMSE_Tp       :  1.2844894828837756
```

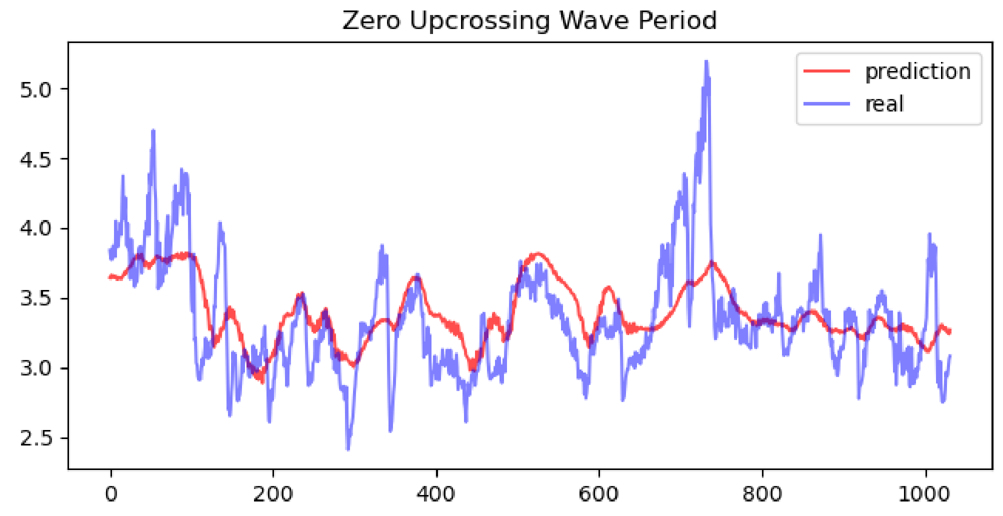
```
RMSE_Direction:  72.16962384897157
RMSE_SST       :  0.5565758039705817
None
```

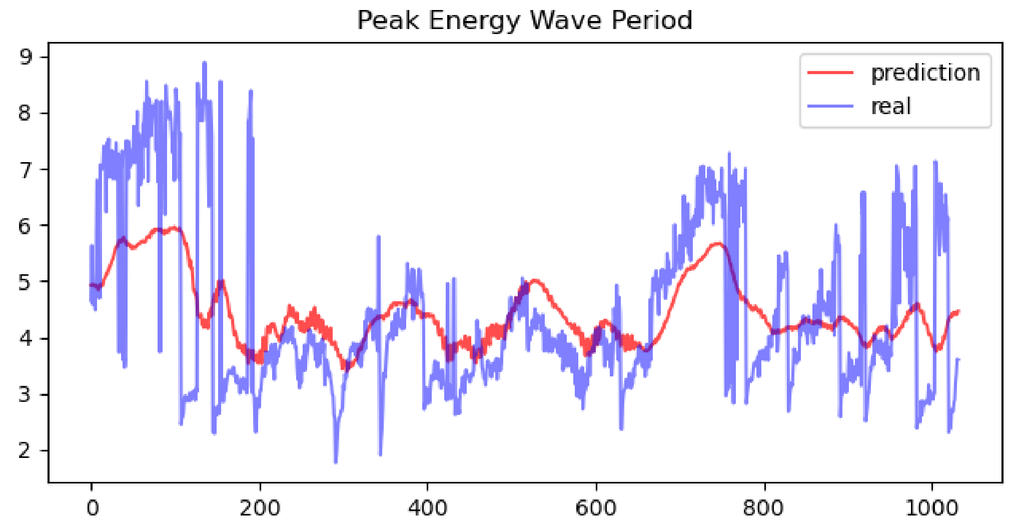
Visualization of the Real and Predicted Values

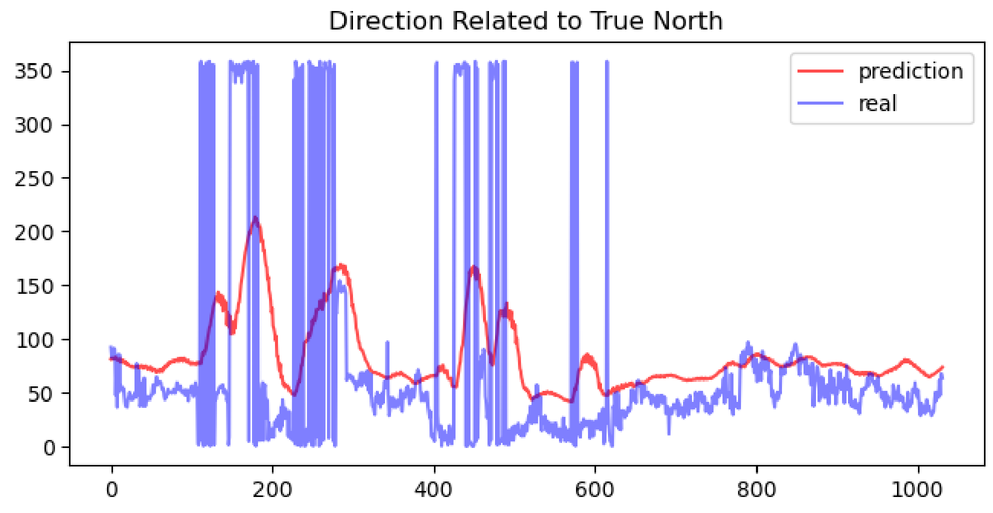
```
plt.figure(figsize=(17,25))


plt.subplot(6,2,1)
plt.plot(test_prediction[1300:,0], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,0], color='blue', alpha=0.5, label='real')
plt.title('Significant Wave Height')
plt.legend()


plt.subplot(6,2,2)
plt.plot(test_prediction[1300:,1], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,1], color='blue', alpha=0.5, label='real')
plt.title('Maximum Wave Height')
plt.legend()


plt.subplot(6,2,3)
plt.plot(test_prediction[1300:,2], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,2], color='blue', alpha=0.5, label='real')
plt.title('Zero Upcrossing Wave Period')
plt.legend()


plt.subplot(6,2,4)
plt.plot(test_prediction[1300:,3], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,3], color='blue', alpha=0.5, label='real')
plt.title('Peak Energy Wave Period')
plt.legend()
plt.subplot(6,2,5)
plt.plot(test_prediction[1300:,4], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,4], color='blue', alpha=0.5, label='real')
plt.title('Direction Related to True North')
plt.legend()


plt.subplot(6,2,6)
plt.plot(test_prediction[1300:,5], color='red', alpha=0.7,
label='prediction')
plt.plot(y_test[1300:,5], color='blue', alpha=0.5, label='real')
plt.title('Sea Surface Temperature')
plt.legend()

plt.show();
```

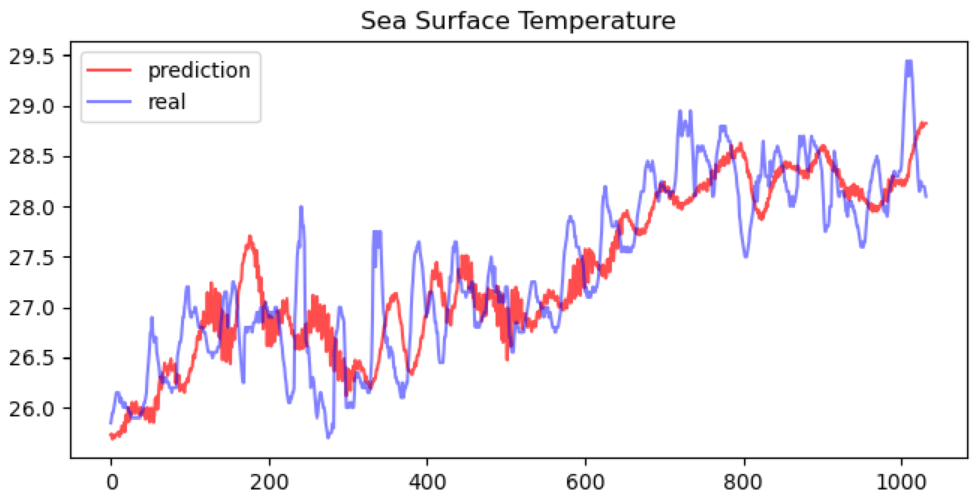





Zero Upcrossing Wave Period
prediction
real
5.0
4.5
4.0
3.5
3.0
2.5
0
200
400
600
800
1000


Peak Energy Wave Period
prediction
real
9
8
7
6
5
4
3
2
0
200
400
600
800
1000


Direction Related to True North
prediction
real
350
300
250
200
150
100
50
0
0
200
400
600
800
1000



Conclusion

The LSTM model is capable of predicting future values by looking at a defined batch of samples. 30 samples are used as a batch for the model in order to predict the 31st sample.

After the training process LSTM model reached:

a. 0.85 training model accuracy and 0.93 validation model accuracy

b. 0.0023 training loss value and 0.0015 validation loss value

c. 0.09 - 0.26 meters of error for predicting wave height (RMSE Hs and RMSE Hmax)

d. 0.3 - 1.96 seconds of error for predicting wave period (RMSE Tz and RMSE Tp)

e. Approx. 15 degrees of error for predicting the wave direction (RMSE Direction)

f. 0.2 degrees celsius of error for predicting the sea surface temperature (RMSE SST)

Webliography

Dataset : 2021 wave data of Townsville

https://www.data.qld.gov.au/dataset/coastal-data-system-waves-townsville/resource/8c83714a-316d-4ea1-a5c7-a25001cd8242

References:

https://www.qld.gov.au/environment/coasts-waterways/beach/monitoring/waves

https://www.data.qld.gov.au/dataset/coastal-data-system-waves-mooloolaba