- **기존 텀프로젝트 계획** : 24시간 마다 한번씩 측정된 데이터

- **입력 형태** : 시간 간격을 따라 측정된 (829,6)차원의 실수 Tensor

1. Engine RPM (rpm)
2. SLIP (%)
3. Vessel Speed (Knots)
4. WIND SPEED (Knots)
5. WAVE HEIGHT (m)
6. DRAFT (m)

- **출력 형태** : (829,1)차원의 실수 Tensor

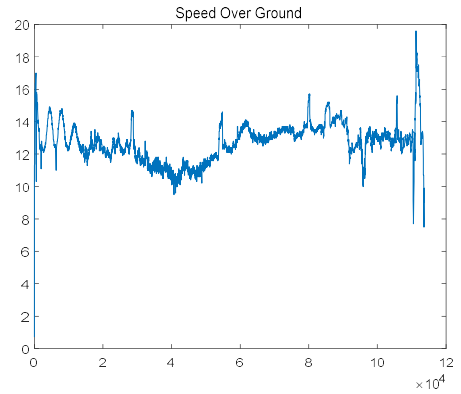
1. FO/ME Consumption(KW)

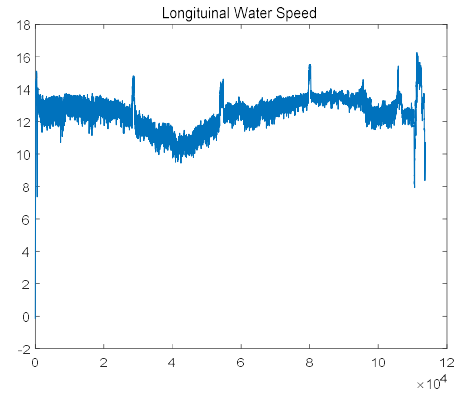
- **계획 변경 : 10초마다 한번씩 측정된 센서 데이터**
- **입력 형태** : 시간 간격을 따라 측정된 (113680,11)차원의 실수 Tensor

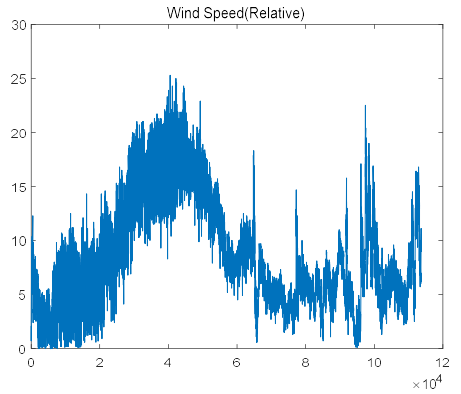
1. Speed Over Ground(Knots)
2. Longitudinal Water(Knots)
3. Wind Speed(Relative)(m/s)
4. Draft Fore(m)
5. Draft Aft(m)
6. Slip(%)
7. Shaft Revolution(rpm)
8. Shaft Power(KW)
9. Shaft Torque(KN)
10. Current Speed(m/s)
11. Significant Height of Swell Waves(m)

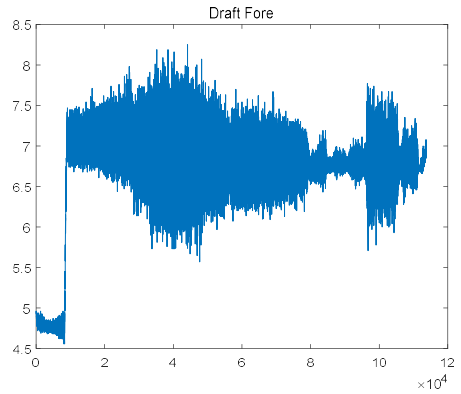
- **출력 형태** : (113680,11)차원의 실수 Tensor

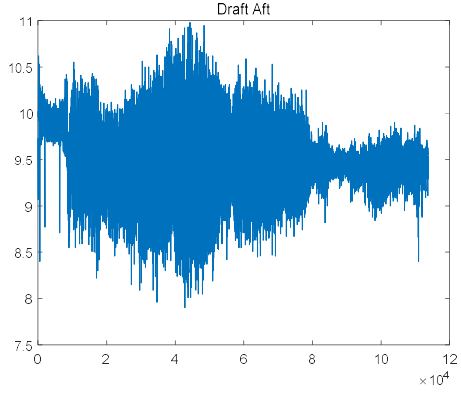
1. FO/ME Consumption(Kg/hour)

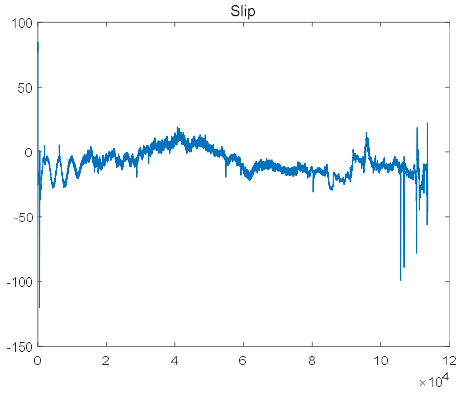

Speed Over Ground
×10^4

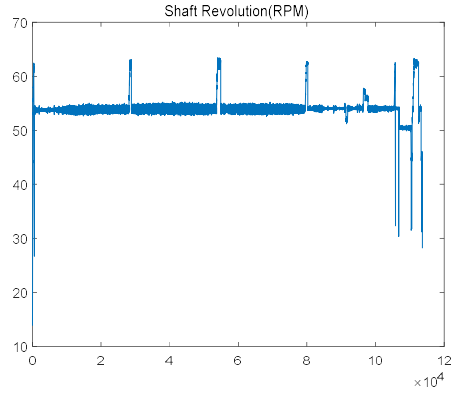

Longituinal Water Speed
×10^4

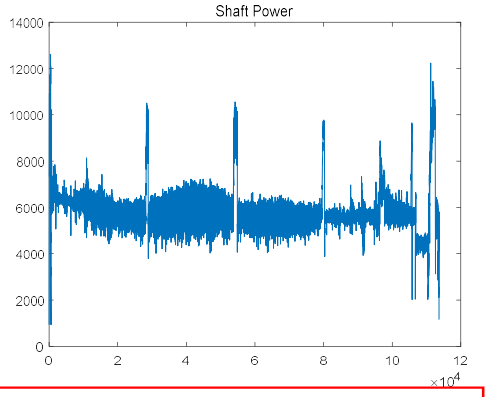

Wind Speed(Relative)
×10^4

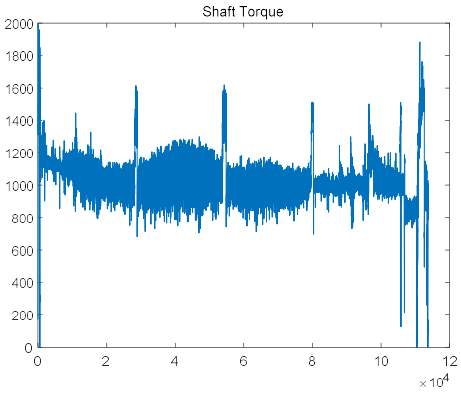

Draft Fore
×10^4

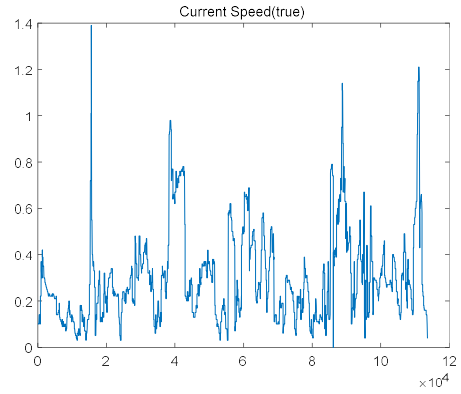

Draft Aft
×10^4

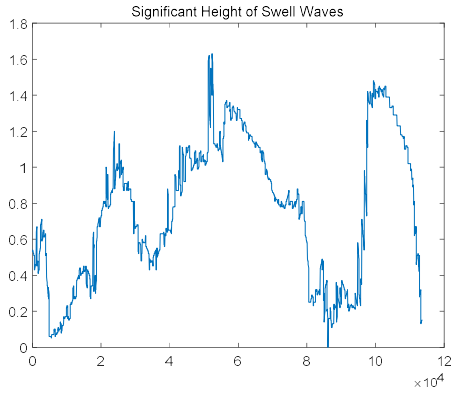

Slip
×10^4

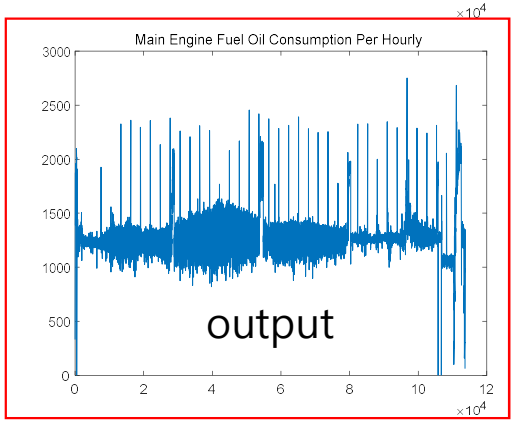

Shaft Revolution(RPM)
×10^4


Shaft Power
×10^4


Shaft Torque
×10^4


Current Speed(true)
×10^4

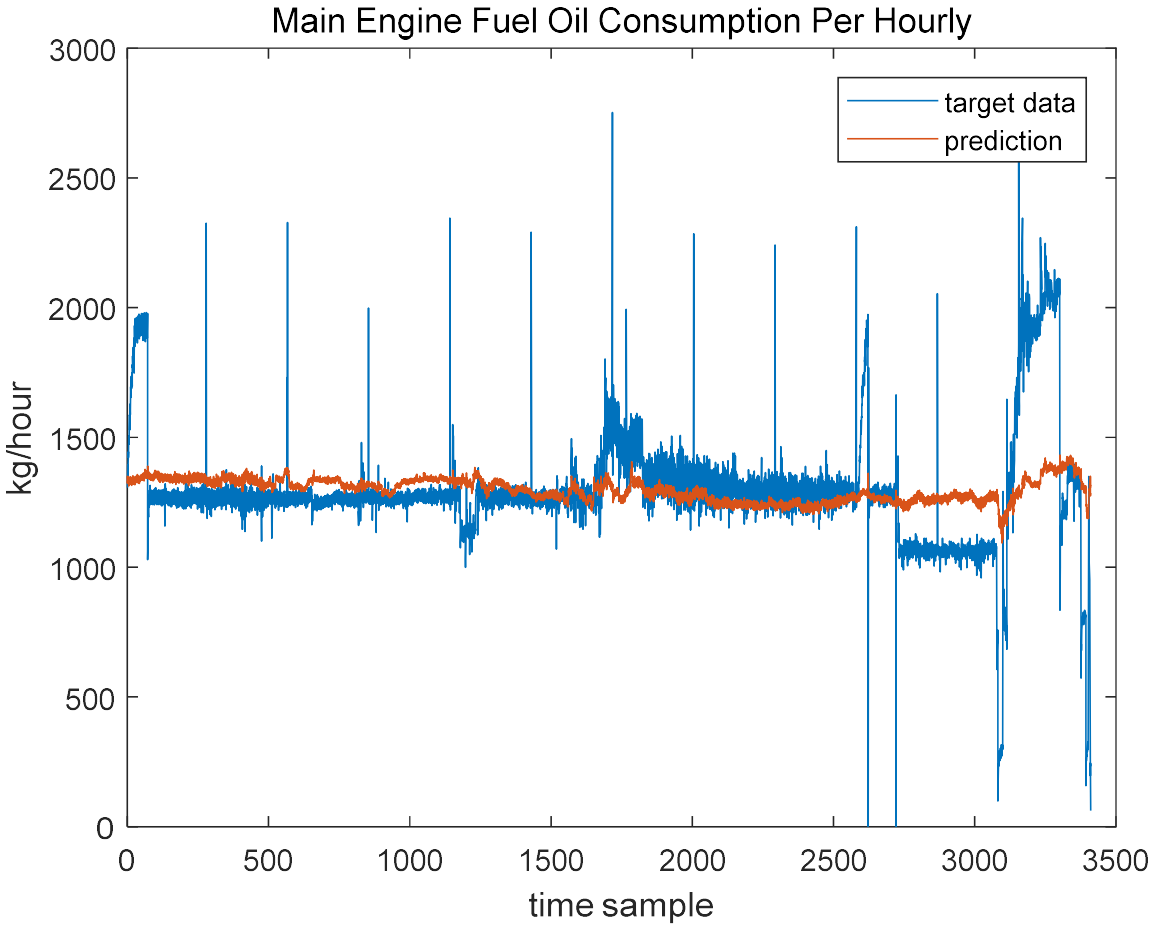

Significant Height of Swell Waves
×10^4


Main Engine Fuel Oil Consumption Per Hourly
output
×10^4

:
```
# train Parameters
seq_length = 5
data_dim = 12
hidden_dim = 5
output_dim = 1
learning_rate = 0.01
iterations = 100
```

```
# build a dataset
dataX = []
dataY = []
for i in range(0, len(y) - seq_length):
    _x = x[i:i + seq_length]
    _y = y[i + seq_length]  # Next close price
    #print(_x, "->", _y)
    dataX.append(_x)
    dataY.append(_y)

# train/test split
train_size = int(len(dataY) * 0.7)
test_size = len(dataY) - train_size
trainX, testX = np.array(dataX[0:train_size]), np.array(
    dataX[train_size:len(dataX)])
trainY, testY = np.array(dataY[0:train_size]), np.array(
    dataY[train_size:len(dataY)])

# input place holders
X = tf.placeholder(tf.float32, [None, seq_length, data_dim], name = 'X-input')
Y = tf.placeholder(tf.float32, [None, 1], name = 'Y-input')
```

```
# build a LSTM network
cell1 = tf.contrib.rnn.BasicLSTMCell(
    num_units=hidden_dim, state_is_tuple=True, activation=tf.tanh)
cell = tf.contrib.rnn.MultiRNNCell([cell1]*3,state_is_tuple=True)
outputs, _states = tf.nn.dynamic_rnn(cell1, X, dtype=tf.float32)
Y_pred = tf.contrib.layers.fully_connected(
    outputs[:, -1], output_dim, activation_fn=None)  # We use the last cell's out

# cost/loss
loss = tf.reduce_mean(tf.square(Y_pred - Y))  # sum of the squares
# optimizer
optimizer = tf.train.AdamOptimizer(learning_rate)
train = optimizer.minimize(loss)

# RMSE
targets = tf.placeholder(tf.float32, [None, 1])
predictions = tf.placeholder(tf.float32, [None, 1])
rmse = tf.sqrt(tf.reduce_mean(tf.square(targets - predictions)))
```


Main Engine Fuel Oil Consumption Per Hourly
target data
prediction
kg/hour
time sample
0
500
1000
1500
2000
2500
3000
3500

추후 계획

- 학습 파라미터 변경 후 학습(sequence length, hidden dimension….)

- 데이터 노이즈 제거 후 학습

질문 사항

- 학습 척도를 정량화하여 나타낼 수 있는 방법은??