Time Series

권지혜 윤빈나 윤재경
전혜민 정재원 최성웅

KU-BIG

Index

1. LSTM

2. Hybrid model

LSTM – hyperparameter tuning

- Set function

```
def tuning(window_size,num_layer,batch_size,epoch):

    window_size=window_size

    # frame as supervised learning
    reframed = series_to_supervised(scaled, window_size, 1)

    # drop columns we don't want to predict
    reframed.drop(reframed.columns[[-6,-5,-4,-3,-2,-1]], axis=1, inplace=True)

    #split train and test
    values = reframed.values
    n_test_hours = 31*24
    train = values[:-n_test_hours, :]
    test = values[-n_test_hours:, :]
    # split into input and outputs
    train_X, train_y = train[:, :-1], train[:, -1]
    test_X, test_y = test[:, :-1], test[:, -1]

    # reshape input to be 3D [samples, timesteps, features]
    train_X = train_X.reshape((train_X.shape[0], window_size, int(train_X.shape[1]/window_size)))
    test_X = test_X.reshape((test_X.shape[0], window_size, int(test_X.shape[1]/window_size)))

    # design model
    model = Sequential()
    for i in range(num_layer-1):
        model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2]),return_sequences=True))
    model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
    model.add(Dense(1))
    model.compile(loss='mae', optimizer='adam')
```

```
    #fit model
    history = model.fit(train_X, train_y, epochs=epoch, batch_size=batch_size,
                        validation_data=(test_X, test_y), verbose=0, shuffle=False)

    # make a prediction
    yhat = model.predict(test_X)
    test_X = test_X.reshape((test_X.shape[0], window_size*test_X.shape[2]))
    test_X[:,1:]

    # invert scaling for forecast
    inv_yhat = np.concatenate((yhat, test_X[:, 1:7]), axis=1)
    inv_yhat
    inv_yhat = scaler.inverse_transform(inv_yhat)
    inv_yhat = inv_yhat[:,0]

    # invert scaling for actual
    test_y = test_y.reshape((len(test_y), 1))
    inv_y = np.concatenate((test_y, test_X[:, 1:7]), axis=1)
    inv_y = scaler.inverse_transform(inv_y)
    inv_y = inv_y[:,0]

    # return RMSE
    rmse = math.sqrt(mean_squared_error(inv_y, inv_yhat))
    return rmse
```

LSTM – hyperparameter tuning

- Find the best hyperparameter

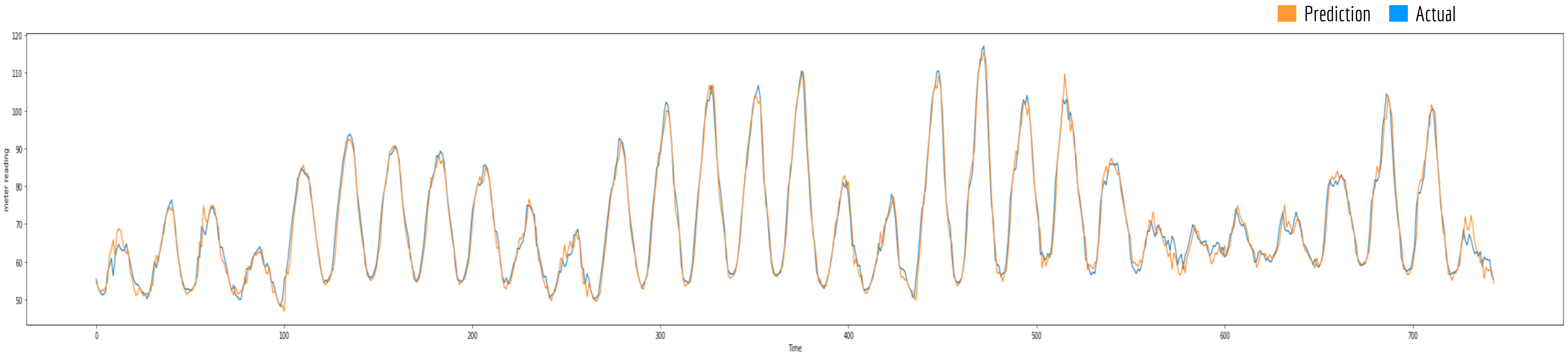
```
random_state = 42
num_loop = 30
hyperparameters_list = []

for loop in range(num_loop):
    window_size=np.random.choice(np.arange(24, 24*8, 24))
    num_layer=np.random.randint(1,4)
    batch_size=np.random.randint(32,129)
    epoch=np.random.randint(50,201)

    parameters = {'loop':loop,
                  'window_size': window_size,
                  'num_layer': num_layer,
                  'batch_size': batch_size,
                  'epoch': epoch}

    score = tuning(window_size,num_layer,batch_size,epoch)

    parameters['score'] = score

    print(f"{loop:2} iteration = {parameters['epoch']}, Score = {parameters['score']:.3f}")

    hyperparameters_list.append(parameters)
    hyperparameters_data = pd.DataFrame(hyperparameters_list)
    hyperparameters_data = hyperparameters_data.sort_values(by="score")

    hyperparameters_data.to_csv("paramsearch.csv")

hyperparameters_data.head(10)
```

loop	window_size	num_layer	batch_size	epoch	score
22	168	1	90	85	2.336529
28	168	1	46	92	2.367447
3	168	1	46	64	2.369685
17	120	1	66	89	2.387569
6	120	3	100	70	2.411416
4	96	1	63	81	2.415116
5	144	1	85	134	2.416007
2	168	1	106	103	2.421331
18	144	3	56	62	2.514466
7	120	3	36	100	2.558349

Best 10

LSTM – hyperparameter tuning

- Window size: **168**, # of hidden layers: **1**, batch size: **90**, epoch: **85**



Test RMSE: 2.360

Hybrid Model

ARIMA + ANN = Hybrid Model

Linear part

Non-linear part

Good!

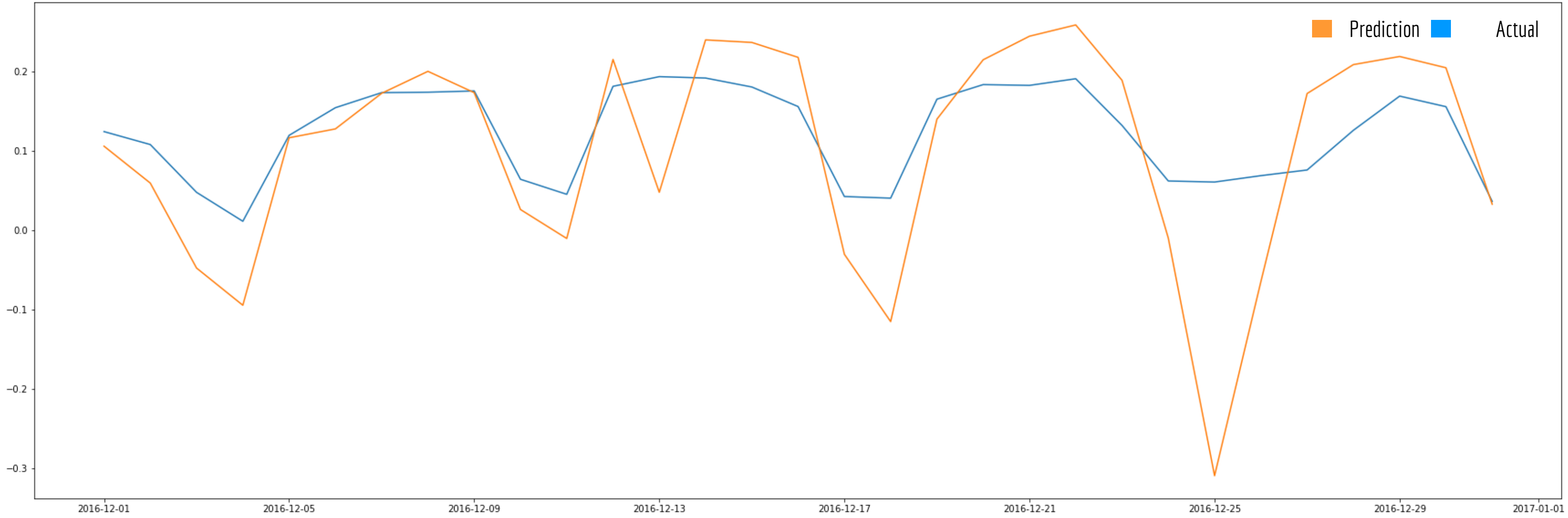
- $y_t = L_t + N_t$
- $e_t = y_t - \hat{L}_t$

Hybrid Model

Step 1	Predict daily consumption using ARIMA
Step 2	Convert daily data into hourly data using LSTM
Step 3	Fit LSTM on the residuals & add to linear model

Hybrid Model

Step 1 Predict daily consumption using ARIMA : SARIMAX(1,1,2)(1,0,0,7)



Hybrid Model

Step 2 Convert daily data into hourly data using LSTM

```
ready_portion['predicted_meter_reading'] = ready_portion['portion_yhat'] * (ready_portion['pred_day_sum']*(maxx-minn)+minn)
```

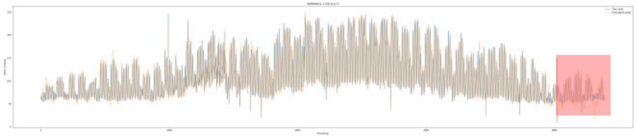
```
ready_portion.head()
```

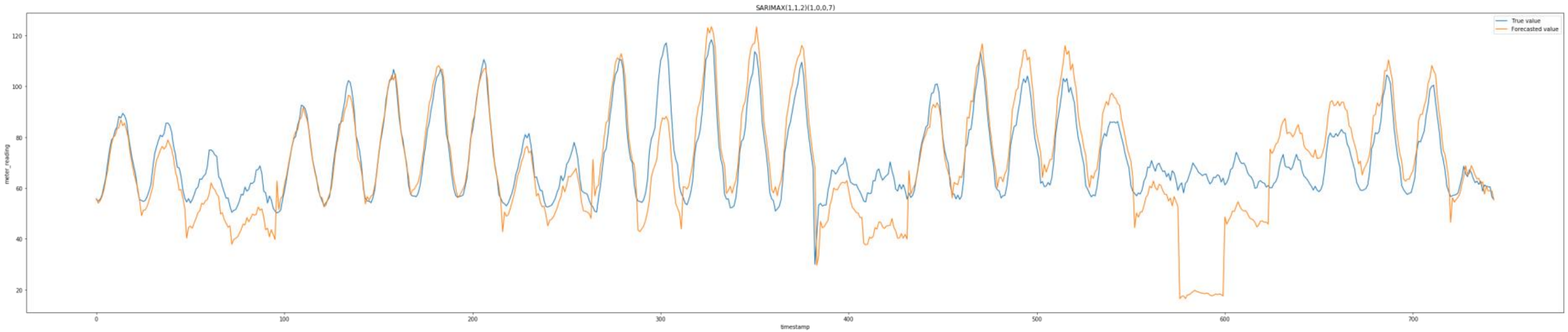
	day	portion_yhat	pred_day_sum	predicted_meter_reading
0	2016-12-01	0.033315	0.10603	55.907080
1	2016-12-01	0.032182	0.10603	54.005716
2	2016-12-01	0.032741	0.10603	54.944029
3	2016-12-01	0.033756	0.10603	56.647213
4	2016-12-01	0.035290	0.10603	59.221912

- portion_yhat : 시간대별 분포
- pred_day_sum : daily prediction (min-max scaling)
- predicted_meter_reading : proportion_yhat × inverse-scaled pred_day_sum

Hybrid Model

Step 2 Convert daily data into hourly data using LSTM





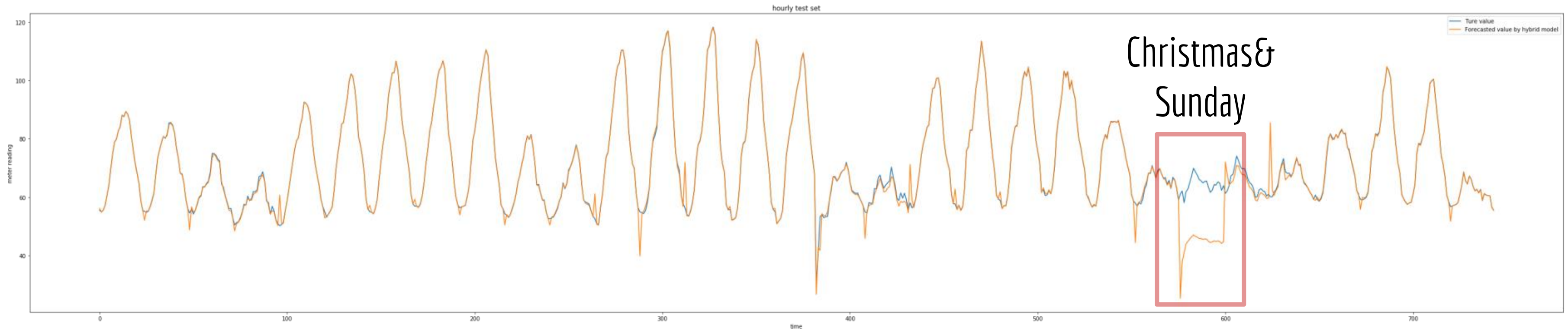
Test RMSE: 12.431

Hybrid Model

Step 3 Fit LSTM on the residuals & add to linear model

```
hybrid_prediction2 = inv_yhat + tabulated_portion.predicted_value[-744:]
```

LSTM ARIMA



Test RMSE: 4.135

Hybrid Model

SARIMA summary

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1182	0.090	-1.319	0.188	-0.294	0.058
air_temperature	0.7159	0.048	14.920	0.000	0.621	0.810
Sunday	-0.2174	0.015	-14.284	0.000	-0.247	-0.187
Saturday	-0.1735	0.015	-11.456	0.000	-0.203	-0.144
holiday	-0.2084	0.032	-6.559	0.000	-0.271	-0.146
summer	0.0835	0.018	4.730	0.000	0.049	0.118
cold	0.0946	0.025	3.778	0.000	0.045	0.144
winter	-0.0866	0.018	-4.765	0.000	-0.122	-0.051
dew_temperature	0.0957	0.030	3.173	0.002	0.036	0.155
hot_temperature^2	0.2878	0.068	4.235	0.000	0.154	0.422
hot	-0.1647	0.053	-3.121	0.002	-0.269	-0.061
wind_speed	0.0899	0.046	1.973	0.049	0.000	0.180
sea_level_pressure	0.2097	0.100	2.089	0.037	0.012	0.407
Friday	-0.0263	0.015	-1.789	0.075	-0.055	0.003
wind_direction	0.0481	0.034	1.433	0.153	-0.018	0.114

$$\hat{\beta}_{Sunday} = -0.2174 < 0$$

$$\hat{\beta}_{holiday} = -0.2084 < 0$$

Much lower value than real data

Thank you