Air Pollution Forecasting using LSTM

DATS 6203 Machine Learning II

Group 3

Xiaoyang Chen, Julia Jin

Outline

1. Introduction & Background
2. Dataset Description & Preprocessing
3. Model Description
4. Results
5. Summary & Conclusion

1.1 Introduction

This project predicts air pollution using the Air Quality dataset from Kaggle.

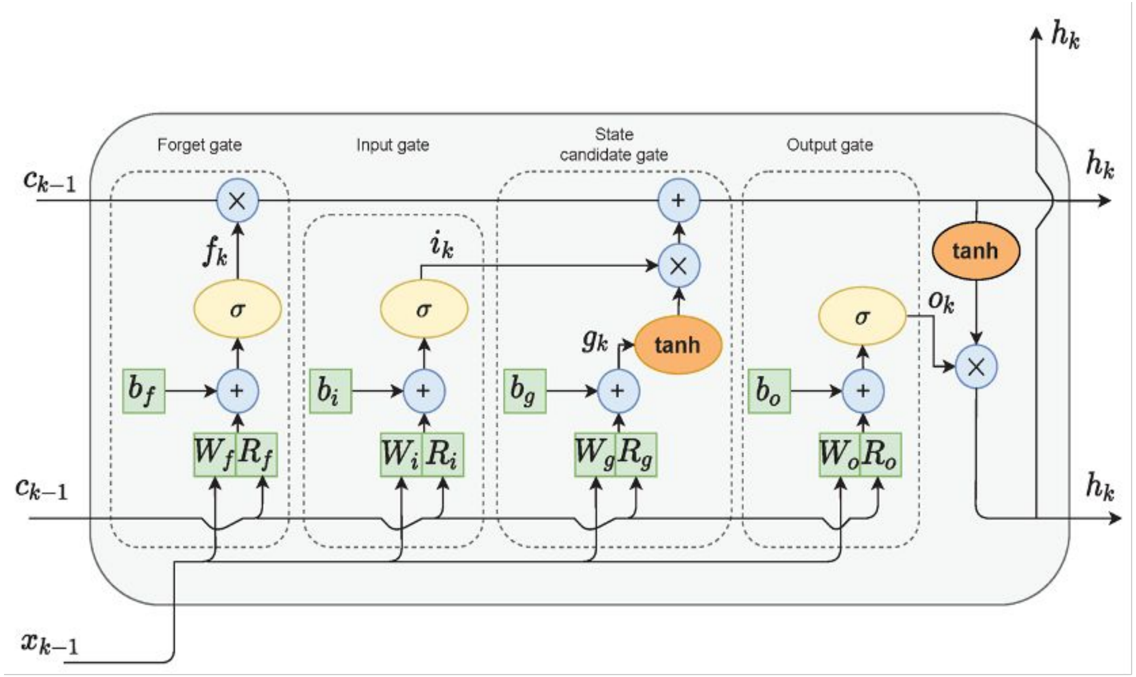
This is a dataset that reports on the weather and the level of pollution each hour for five years at the US embassy in Beijing, China.

The data includes the date-time, the pollution called PM2.5 concentration, and the weather information as our features.

- To predict the PM2.5 concentration using our dataset,
- We choose a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) model for its successful track record with time-series data.
- We will establish a multivariate input model (multivariate) and a single variable input model (univariate).
- We will compare the validation loss and results on both models and pick the best one.

LSTM Network

1) Forget Gate: handles what information to throw away from the block
2) Input Gate: decides which values from the input to update the memory state
3) Output Gate: finally handles what to be in output based on input and the memory gate



2.1 Dataset Description

pm2.5: PM2.5 concentration (ug/m^3)

dew: Dew point

temperature: Temperature around the embassy (F)

pressure: Air pressure

wind_dir: Combined wind direction

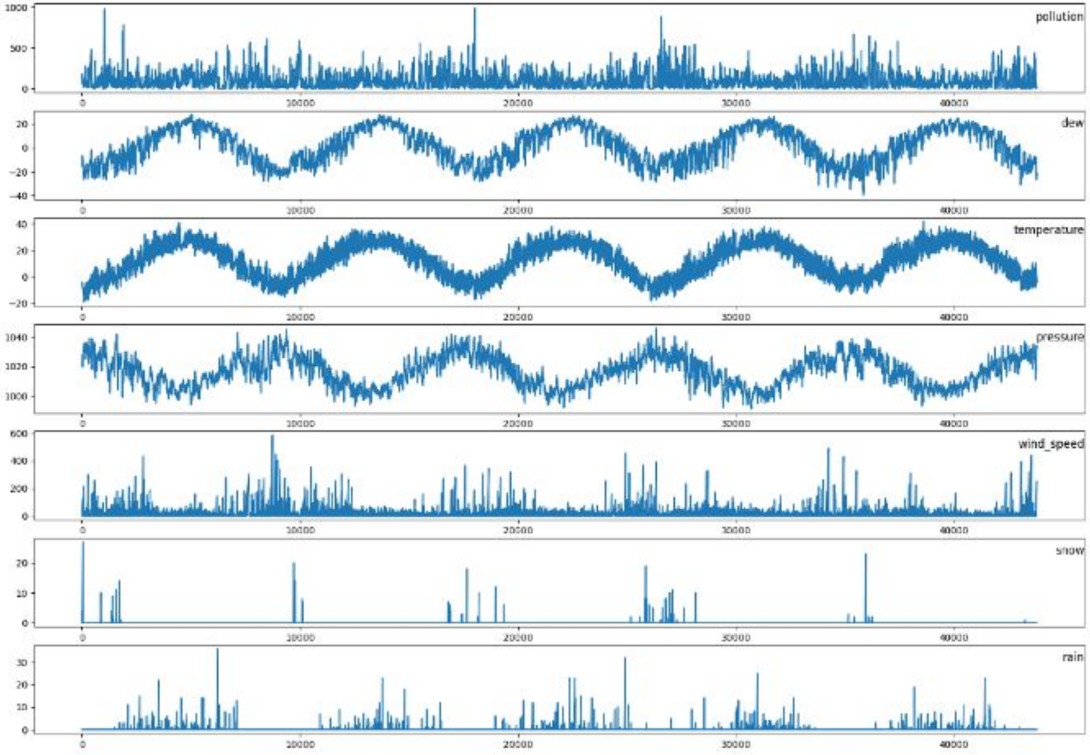
wind_speed: Cumulated wind speed (m/s)

snow: Cumulated hours of snow

rain: Cumulated hours of rain

2.2 Visualization

- High seasonality
- Dew, temperature & pressure show high cyclicity



2.3 Correlation Matrix

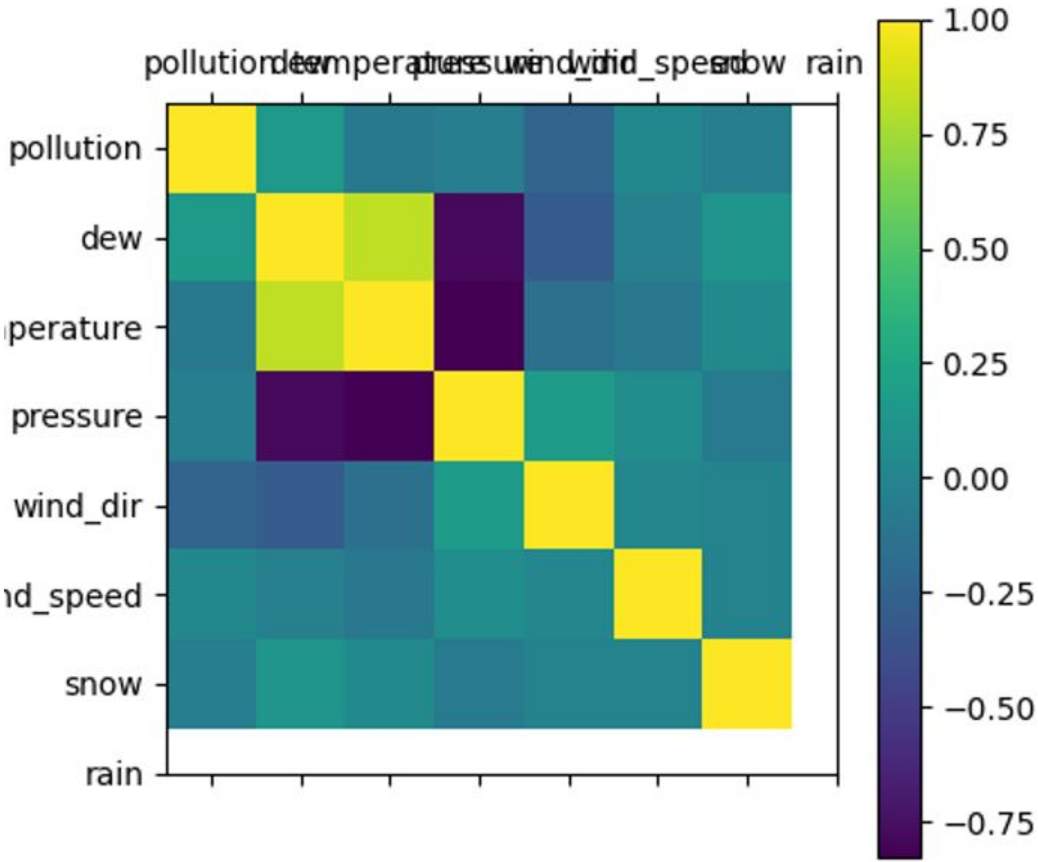
- Highly negative correlated:

  dew & pressure, temperature

  & pressure

- Highly positive correlated:

  dew & temperature

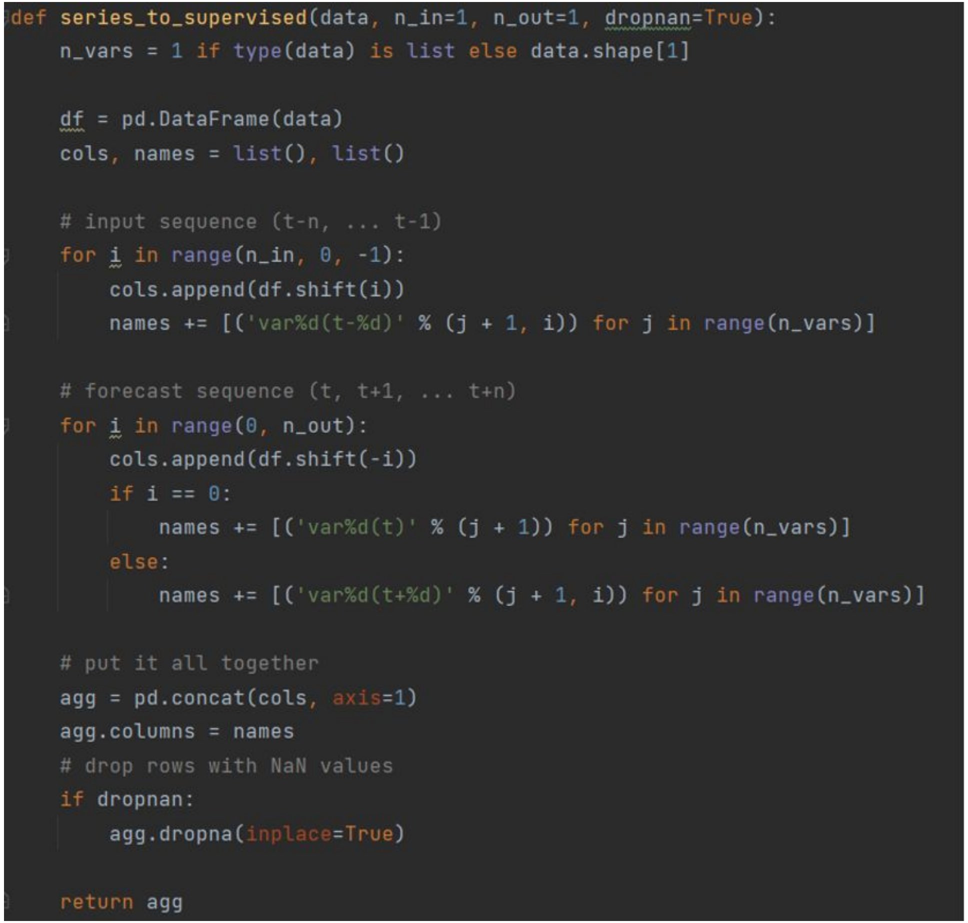


2.4 Preprocessing

- Use “LabelEncoder” from sklearn.preprocessing to change “wind_direction”, a string type, to float.
- Use “MinMaxScaler” to normalize the data.
- Reframe a time series problem as a supervised learning problem - from one sequence to pairs of input and output sequences. Use “series_to_supervised”.

series_to_supervised

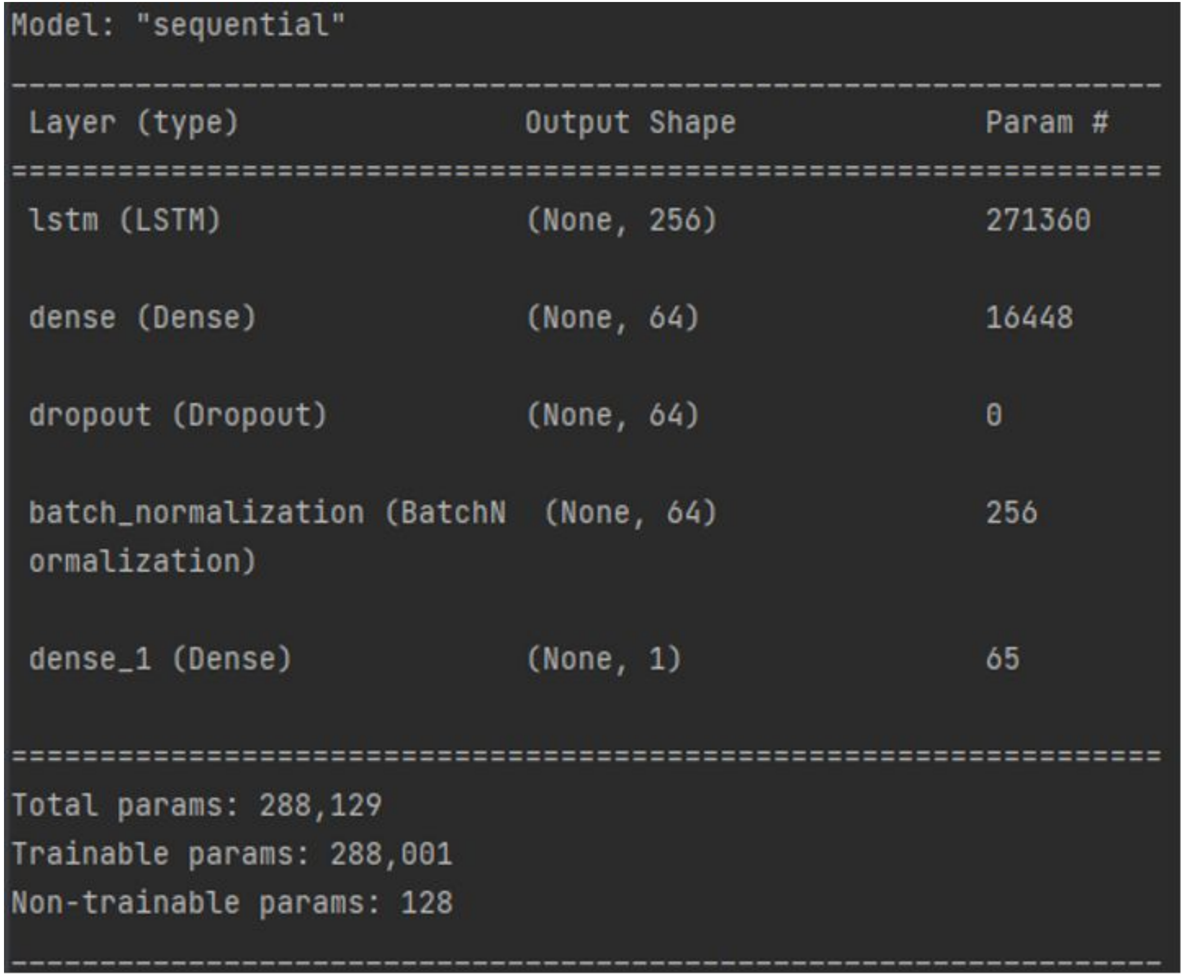
- Takes a univariate or multivariate time series and frames it as a supervised learning dataset.
- n_in: number of lag observations as input
- n_out: number of observations as output



```python
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]

    df = pd.DataFrame(data)
    cols, names = list(), list()

    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j + 1, i)) for j in range(n_vars)]

    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j + 1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j + 1, i)) for j in range(n_vars)]

    # put it all together
    agg = pd.concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)

    return agg
```

3.1 Multivariate Models

- Divide our dataset into two parts. The dataset from 2010 to 2013 is used as the training part, and the dataset from 2014 is used as the validation part.
- Use batch normalization to improve the convergence rate and overall performance of the network
- Use dropout layer to prevent overfitting in our networks
- Use built-in LSTM model in tensorflow.keras library



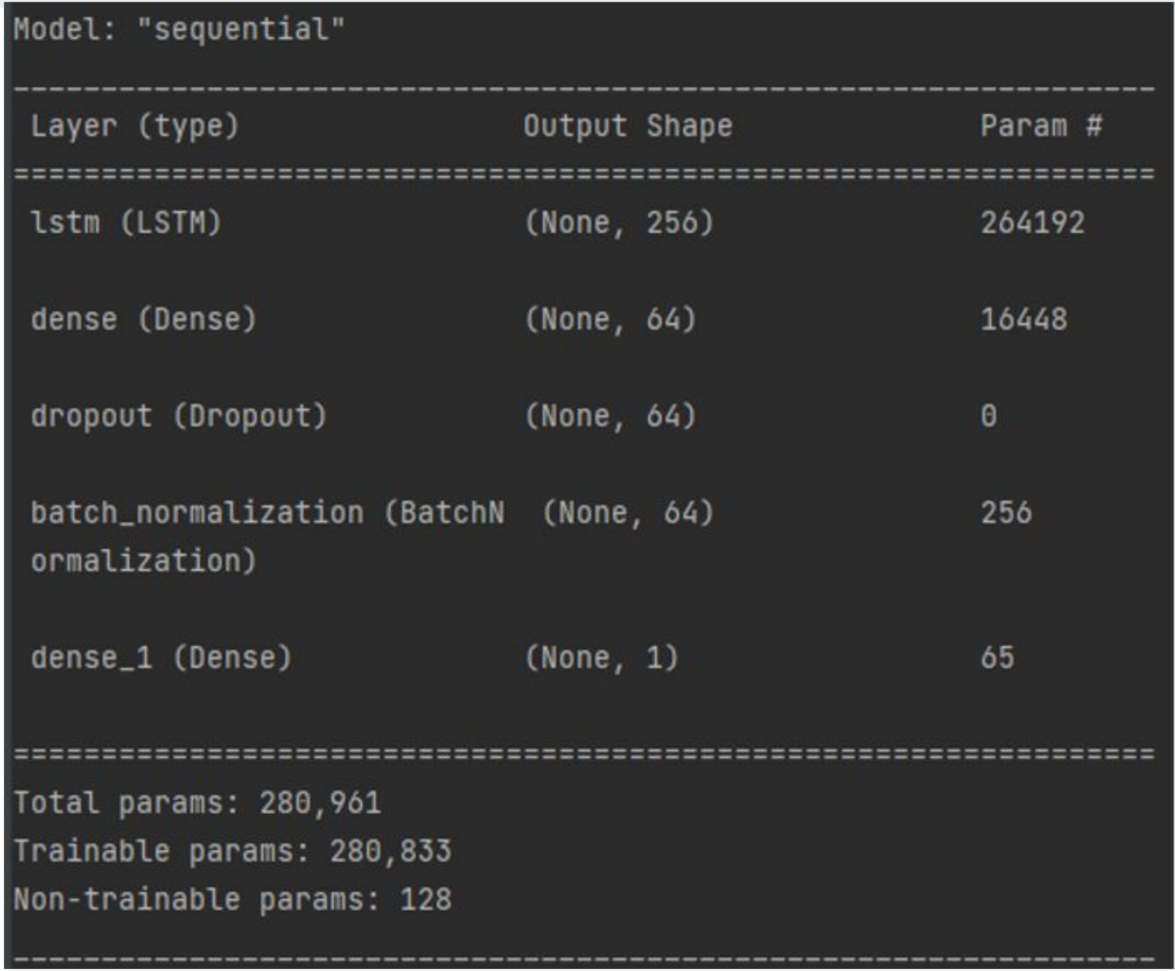
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 256)               271360

 dense (Dense)               (None, 64)                16448

 dropout (Dropout)           (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 288,129
Trainable params: 288,001
Non-trainable params: 128
_________________________________________________________________
```

3.2 Univariate Model

- The univariate model differs from the multivariate model in that multivariate model reads in all the features. In the data preparation for the univariate model, we only read the data of the pollution feature and preprocess it.
- Again, we divide the dataset into two parts as we did for multivariate model.

Parameters in LSTM layer are not the same as the ones in the multivariate model.

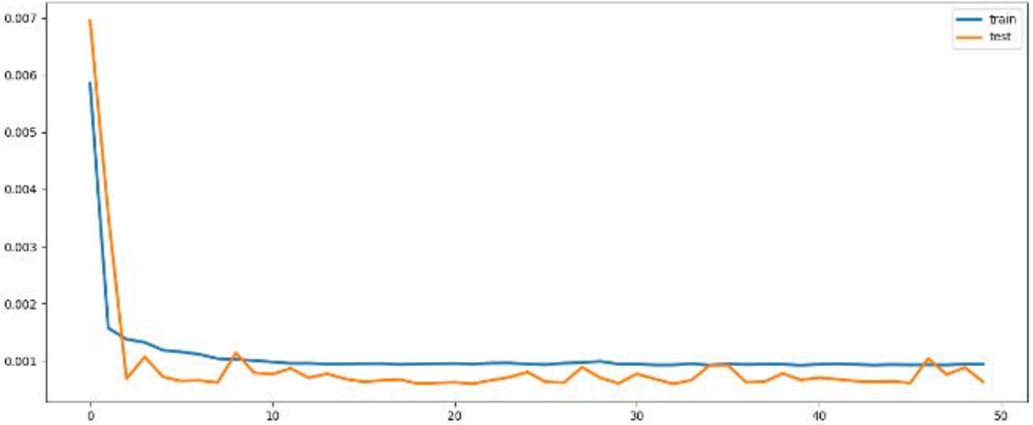
Parameters = ((embedding size + hidden size) * hidden size + hidden size) * 4



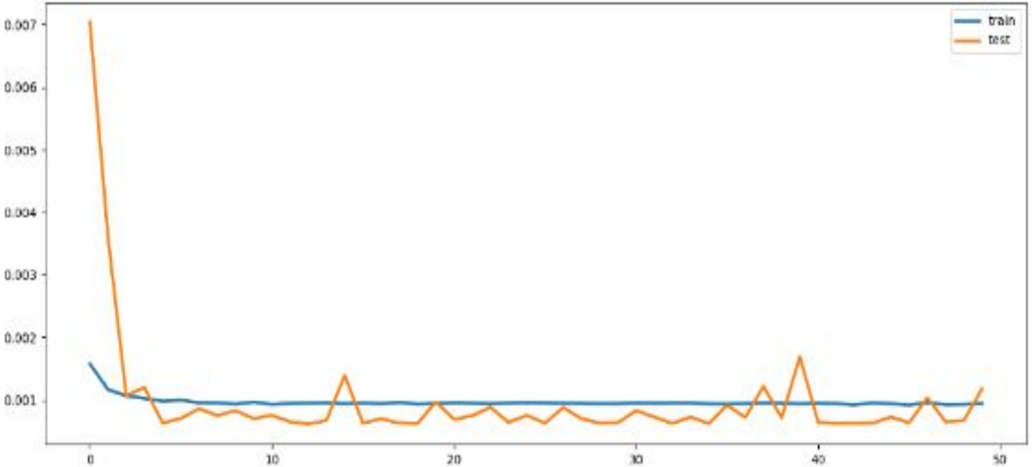
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 256)               264192

 dense (Dense)               (None, 64)                16448

 dropout (Dropout)           (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 280,961
Trainable params: 280,833
Non-trainable params: 128
_________________________________________________________________
```

4.1 Validation Loss

Multivariate Model

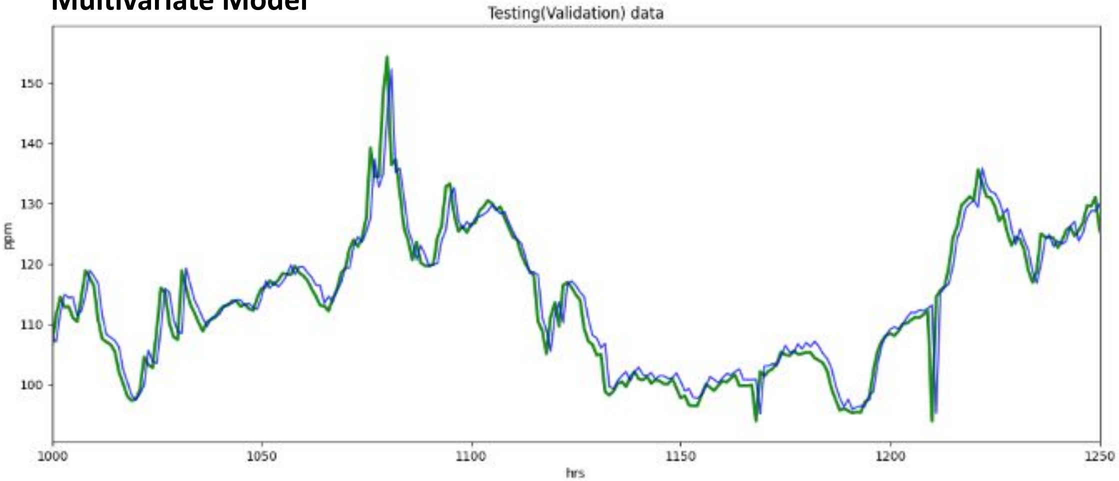


Univariate Model

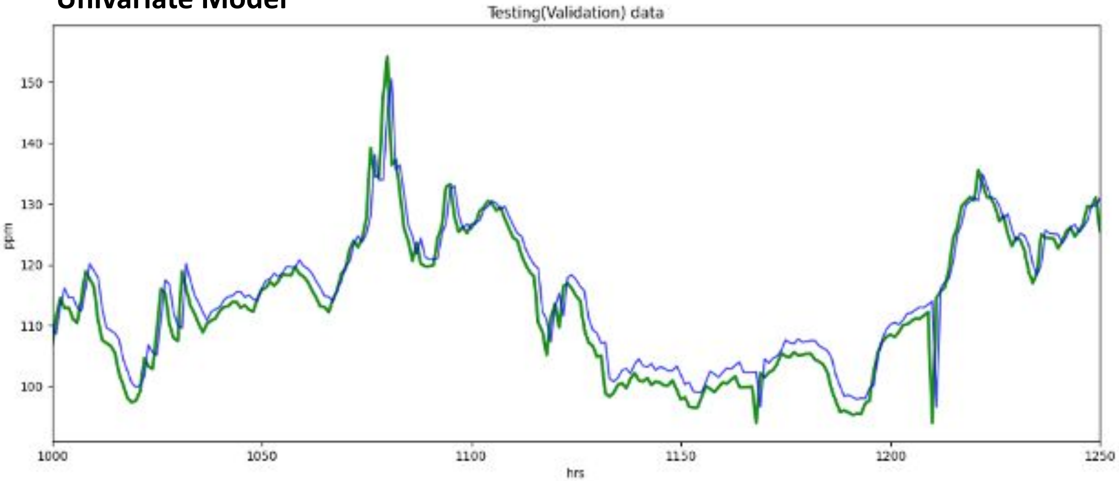


4.2 Output

Multivariate Model



Univariate Model



5.1 Conclusion

- Best model: multivariate model
- Problem: validation loss ups & downs
- Overfitting?
- Small dataset?

5.2 Improvement

- Investigate further into the reasons behind the difference between actual data and model outputs at some certain interval
- Deeper and wider LSTM network
- Learning rate
- Different algorithm
- GRU

Reference

- Tsai, Zeng, Y.-R., & Chang, Y.-S. (2018). Air Pollution Forecasting Using RNN with LSTM. 2018 16TH IEEE INT CONF ON DEPENDABLE, AUTONOM AND SECURE COMP, 16TH IEEE INT CONF ON PERVAS INTELLIGENCE AND COMP, 4TH IEEE INT CONF ON BIG DATA INTELLIGENCE AND COMP, 3RD IEEE CYBER SCI AND TECHNOL CONGRESS (DASC/PICOM/DATACOM/CYBERSCITECH), 1074–1079. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00178
- Wikipedia, “Long short-term memory,” 2022. [Online] Available at: https://en.wikipedia.org/wiki/Long_short-term_memory, Accessed: 2022.
- “A Complete Guide to LSTM Architecture and its Use in Text Classification,” 2022. [Online] Available at: https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture-and-its-use-in-text-classification/, Accessed: 2022.
- “sklearn.preprocessing.LabelEncoder.” [Online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html, Accessed: 2022.
- “How to Convert a Time Series to a Supervised Learning Problem in Python,” 2022. [Online] Available at: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/, Accessed: 2022.
- “tf.keras.layers.LSTM” [Online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM, Accessed: 2022.