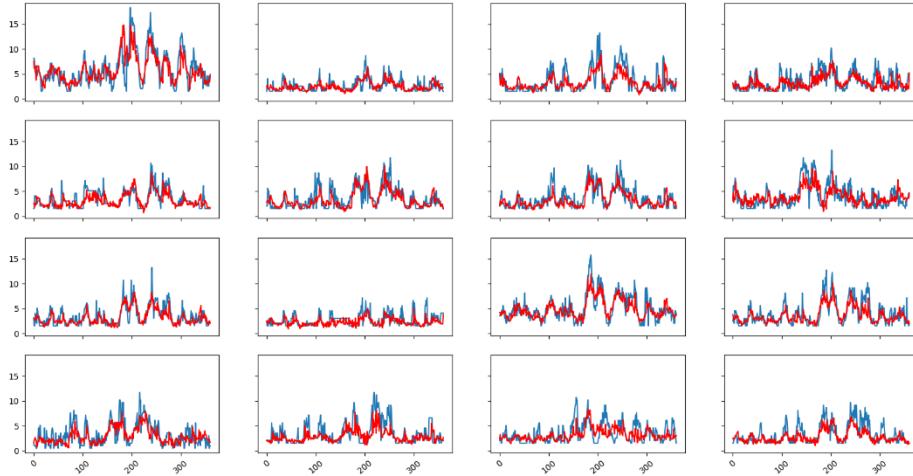


# Experiment Notes

Based on the paper, *Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting*, we analyze the source code and update it to Tensorflow 2.X. The following will show the estimated results of 57 stations with one-step-ahead forecasting. The blue line is ground truth while the red line is estimated.



The errors of the first stations are listed below.

station 1 : MAE = 1.66447	RMSE = 2.25033	nrmse_maxMin = 13.3875	nrmse_mean = 37.4013
station 2 : MAE = 0.73270	RMSE = 0.98935	nrmse_maxMin = 13.8319	nrmse_mean = 37.6449
station 3 : MAE = 1.25285	RMSE = 1.76266	nrmse_maxMin = 15.0499	nrmse_mean = 49.5203
station 4 : MAE = 1.14340	RMSE = 1.53388	nrmse_maxMin = 17.6855	nrmse_mean = 43.6363
station 5 : MAE = 0.95194	RMSE = 1.33948	nrmse_maxMin = 14.6166	nrmse_mean = 38.4311
station 6 : MAE = 1.12947	RMSE = 1.55682	nrmse_maxMin = 15.2748	nrmse_mean = 43.0250
station 7 : MAE = 1.04663	RMSE = 1.43145	nrmse_maxMin = 14.7556	nrmse_mean = 39.3542
station 8 : MAE = 1.41807	RMSE = 1.79599	nrmse_maxMin = 15.3345	nrmse_mean = 45.3454
station 9 : MAE = 0.90828	RMSE = 1.28148	nrmse_maxMin = 10.9415	nrmse_mean = 37.0791
station 10 : MAE = 0.79990	RMSE = 1.14006	nrmse_maxMin = 20.2400	nrmse_mean = 42.6894
station 11 : MAE = 1.19067	RMSE = 1.58106	nrmse_maxMin = 11.0865	nrmse_mean = 30.7594
station 12 : MAE = 1.11359	RMSE = 1.56493	nrmse_maxMin = 13.9463	nrmse_mean = 38.7529
station 13 : MAE = 1.31593	RMSE = 1.69584	nrmse_maxMin = 15.1140	nrmse_mean = 52.3930
station 14 : MAE = 1.26702	RMSE = 1.85482	nrmse_maxMin = 18.1986	nrmse_mean = 53.0058
station 15 : MAE = 1.31790	RMSE = 1.80501	nrmse_maxMin = 19.6965	nrmse_mean = 50.3734
station 16 : MAE = 1.03461	RMSE = 1.46363	nrmse_maxMin = 17.8909	nrmse_mean = 43.8047

We are going to rebuild a library of time series forecasting, *Deep Time Series Forecasting Library* (DTSFL).

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "TSDeepLib". It contains several sub-directories like "DCVL", "DDCL", "DTSFL", "model", and "preprocessor". Inside "preprocessor", there are files: \_\_init\_\_.py, basic\_processor.py, example.py, example\_v2.py, main.py, and readme.md.
- Code Editor:** The main editor window displays the content of basic\_processor.py. The code defines a class BasicPreprocessor that loads data from a CSV file and prepares it for training. It uses numpy and csv modules.
- Toolbars and Status Bar:** The bottom of the screen shows various toolbars and a status bar indicating the current workspace and file paths.

```

1 import numpy as np
2
3 import csv
4
5 class BasicPreprocessor(object):
6
7     def __init__(self, features, look_back, look_ahead):
8
9         self.features = features
10        self.look_back = look_back
11        self.look_ahead = look_ahead
12
13    def load_data_with_filename(self, file_name):
14
15        with open(file_name) as f:
16            raw_data = csv.reader(f, delimiter=",")
17            temp_data = []
18            for line in raw_data:
19                temp_data.append(line)
20            data = (np.array(temp_data)).astype(float)
21
22            data = data[:, :self.features]
23
24            x, y = [], []
25            for index in range(len(data) - self.look_back - self.look_ahead):
26                x.append(data[index:index+self.look_back])
27                y.append(data[index+self.look_back:index+self.look_back+self.look_ahead])
28
29        return np.array(x), np.array(y)
30
31    def load_data(self, data):
32
33        x, y = [], []
34        for index in range(len(data) - self.look_back - self.look_ahead):
35            x.append(data[index:index + self.look_back])
36            y.append(data[index + self.look_back:index + self.look_back + self.look_ahead])
37
38        return np.array(x), np.array(y)
39

```

## Data preprocessor

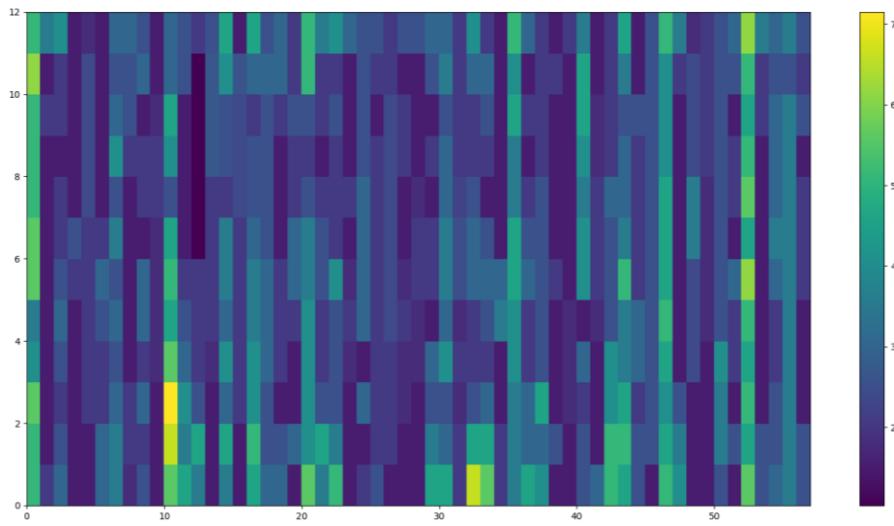
We organize the data in a matrix format. The data shape will be (-1, 12, 57). We randomly pick one matrix, say  $X[0]$ .

$X, Y = \text{BasicPreprocessor}(57, 12, 6).\text{load\_data\_with\_filename}(\text{file\_dataset})$

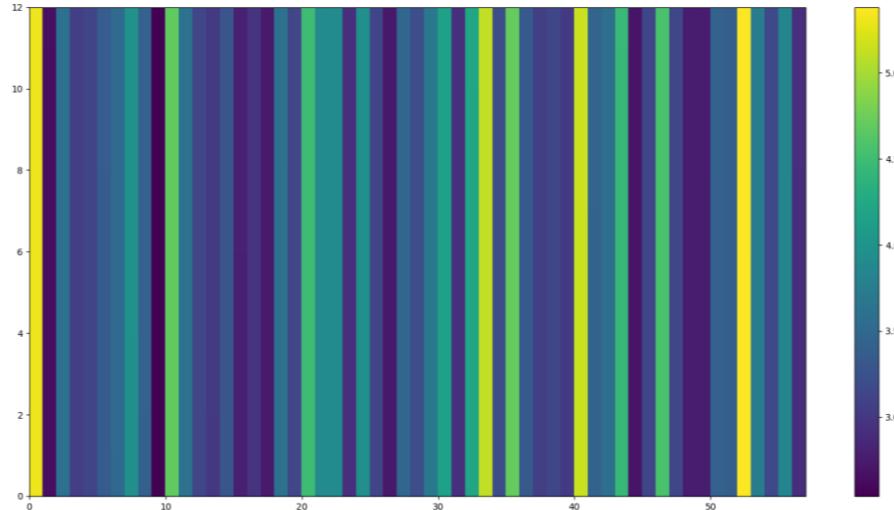
```

plt.figure(1)
plt.pcolormesh(X[0])
plt.colorbar()
plt.show()

```



Then we plot the mean of the data along with axis 0. The data shape will change from  $(-1, 12, 57)$  to  $(12, 57)$ . Interestingly, values in the same column are almost equal. We may think of this as a multi-variate stationary signal.



Based on this paper, we come up with some new ideas such as architecture exploration, sparse representation, and computation acceleration.