

Network Softwarization: Technologies and Enablers

Lecture 9: Network Data Analytics

Lab

Due date: March 25 – 23:59 pm

Instructions

In this Lab session, you will learn to work with time-series data using Python packages for data preparation, analysis, and forecasting. The Lab includes 3 steps: (1) data preparation, (2) time-series modeling and prediction using 3 different models, and (3) evaluation and comparison. First, you need to install and configure Python environment on your PC.

Configuration of the Python Environment

1. Download and install [Anaconda Navigator](#). It consists of different Python tools and packages.
2. Open a command-line interface (CMD in Windows, or terminal in Linux) and run the following command:

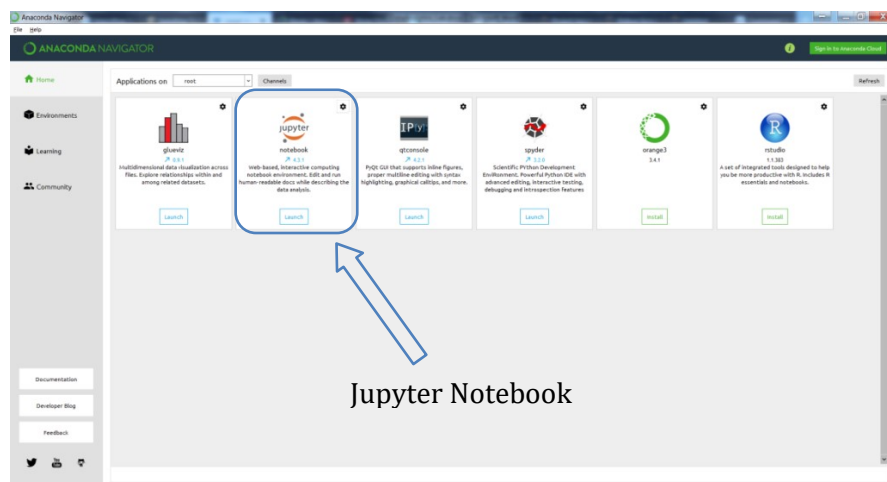
```
conda install -c conda-forge keras tensorflow
```

This command installs Deep Learning packages in Python which includes TensorFlow, and Keras. If there are errors and the command is NOT successful, try the following command:

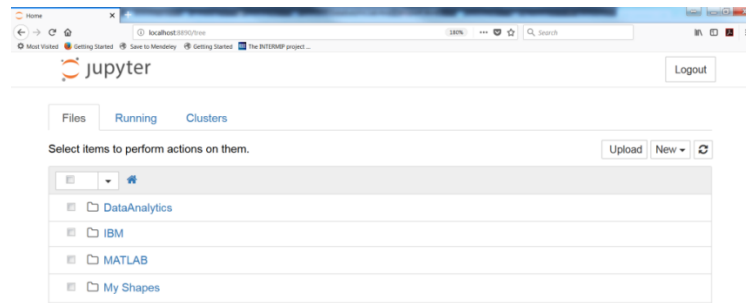
```
pip install keras tensorflow
```

If these commands are not appropriate for your system, search the Internet for a solution. Make sure that the three tools mentioned above are installed on your system. Without those tools, you cannot complete the Lab and its assignment.

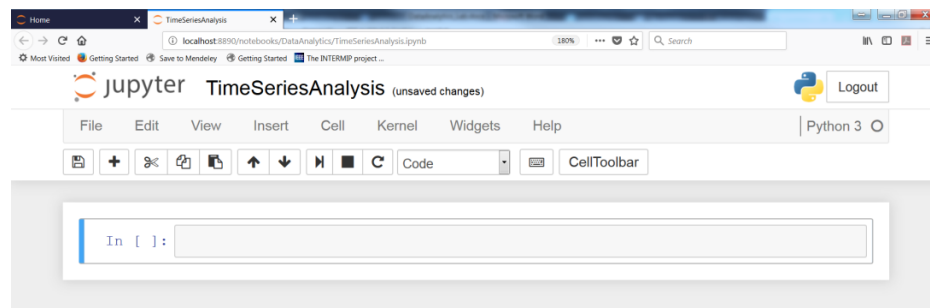
3. Run the [Anaconda Navigator](#). Then, in the navigator windows, launch the [Jupyter Notebook](#).



The Jupyter Notebook is an open-source web application that allows you to create and share your Python code. The interactive computing environment of Jupyter Notebook enables you to author notebook documents that include: Live codes, Interactive widgets, Plots, Narrative text, Equations, Images, and Video. It is a web application which means you find the environment as a new tab in your browser.



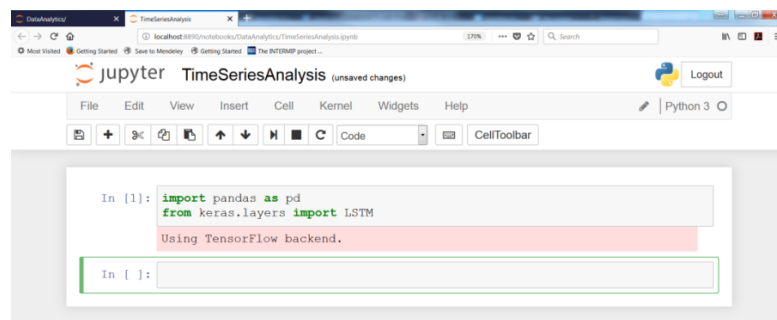
4. Use the button New on the top right of the page to create a new folder and rename the folder to DataAnalytics.
5. Open the DataAnalytics folder (click on its name in the browser), then use the button New to create a Python file. Change the name of the file to TimeSeriesAnalysis.



6. Type the following code in the first **cell**:

```
import pandas as pd
from keras.layers import LSTM
```

Then run the cell (Click on the **Cell > Run Cells**, or use the keyboard **Shift+Enter**). If there is no error message, it means that the environment has been successfully configured.



Quick Review of the Lab

In this Lab, you will learn how to model and forecast time-series. We will use Python packages for machine learning and data visualizations. The time-series data that we use in this Lab has been collected from Abilene Network. The data shows the traffic bit-rate on one of the links in the network. The time interval of sampling is 5 minutes.

We apply two types of transformations on the time-series data. First, we change the range of time-series values, and second, we use the difference operator. The goal of these changes is to remove trend and variance violation in the data (i.e., to have stationary time-series).

We use three models: Multi-Layer Perceptron (MLP or Feedforward Neural Network), Support Vector Regressor (SVR), and Long Short-Term Memory (LSTM) Networks. MLP and SVR are examples of traditional machine learning algorithms while LSTM is a well-known deep learning algorithm.

We perform the modeling and prediction on the transformed time-series (instead of the original time-series). We use the five prior (residual) values of a sample as the feature set. Then we fit a model on the training samples and evaluate the model on the test samples. Finally, we compare results of the three models.

Briefly, the process of the time-series modeling and prediction in this Lab consists of the following steps:

- Reading the [original time-series](#) from file (`time_series.csv`).
- Transforming the data: map values into the range of `[-1, 1]`.
- Transforming the data: apply the difference operator to obtain the [residuals](#).
- Create the dataset (train and test sets).
- Fit a model on the training samples.
- Predict the test samples.
- Evaluate and compare the models

You can find the code and results in the file `DataAnalytics_Lab.ipynb`. There are two versions of this file (in the format of pdf and HTML) which can help you to see the results. Please note that you can only use IPYNB format in Jupyter Notebook.

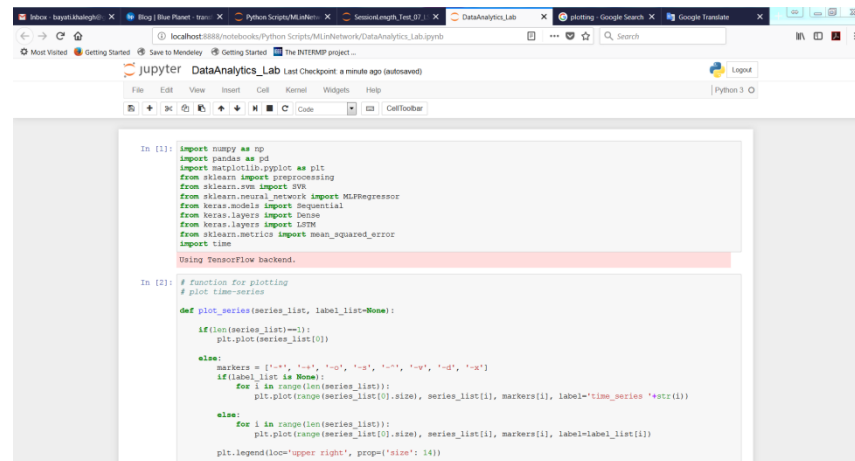
Data and Code

You can find a folder named Code in the Lab materials. You can find the following files in the Code folder:

- `time_series.csv`
- `DataAnalytics_Lab.ipynb`

The first file (csv file) contains the time-series data that you will use in this Lab. The second file contains the Python code. You can open and run this file format (ipynb) in the Jupyter Notebook. First, you need to put a copy of these files in the folder DataAnalytics (which you have created during the part of installation and configuration). Then, open the file in the Jupyter Notebook. As you see, this Notebook includes many cells, and each cell contains: code, images, and texts.

The code in the first cell imports the required packages. Main packages are: Pandas, sklearn (for traditional machine learning algorithms), and keras (for new deep learning algorithms). In the second and third cell, we have defined two functions which are used in the rest of the code. The first function is called `plot_series` and will be used to plot time-series. The second function is named `reconstruct` and will be used to create time-series from residuals. The remaining cells are discussed in the following parts.



```

In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
import time

Using TensorFlow backend.

In [2]: # function for plotting
# plot time-series

def plot_series(series_list, label_list=None):
    if len(series_list) == 1:
        plt.plot(series_list[0])
    else:
        markers = ['+', 'x', 'o', 'v', 'g', 'u', 'd', 's', 'p', 'q', 'h', 'k', 'f', 'n', 'm', 'b', 'i', 'c', 'r', 'w', 'p', 'q', 'h', 'k', 'f', 'n', 'm', 'b', 'i', 'c', 'r', 'w']
        if label_list is None:
            for i in range(len(series_list)):
                plt.plot(range(series_list[0].size), series_list[i], markers[i], label="time_series {}".format(i))
        else:
            for i in range(len(series_list)):
                plt.plot(range(series_list[0].size), series_list[i], markers[i], label=label_list[i])
    plt.legend(loc='upper right', prop={'size': 14})

```

You can clear the results (graphs and outputs) using the menu on the top of the Jupyter Notebook:

Kernel > Restart & Clear Output

Then you can run cell one by one by selecting a cell and using Shift+Enter

Loading data

We use the package **pandas** to read time-series data. The time-series data is stored in a **DataFrame** which is a 2-dimensional labeled data structure with columns of potentially different types. The original time-series is assigned to a column of DataFrame called **ORIGINAL**.

Data transformation: change the range of values

The range of the values of the time-series is $[\sim 0, \sim 1000]$. Many machine learning algorithm assume the range of values are in $[-1, +1]$. Therefore, they have better performance when this assumption is satisfied. We map the values from their original range into the range $[-1, +1]$. There are different tools in the preprocessing module (of package **sklearn**) that allow us to do this transformation. The result of this step is assigned to a new column of DataFrame called **SCALED**.

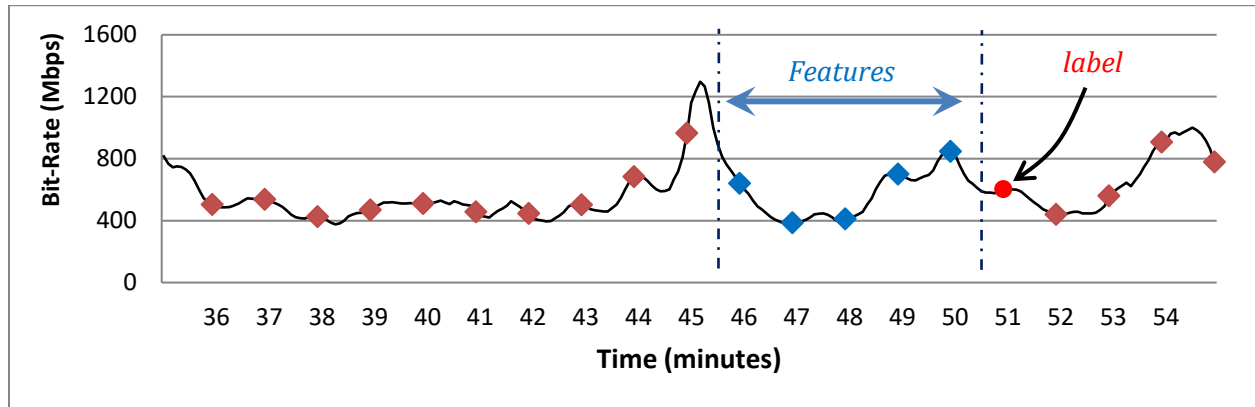
Data Transformation: apply the difference operator

The original data usually is not appropriate for the tasks of modeling and prediction. Actually, we need to perform preprocessing steps to remove undesirable properties of the data (e.g., non-stationarity). In this step, we apply the difference operator which removes trends and reduces variance violations. This step generates the residuals which are assigned to a new column labeled as **DIFF**.

Feature selection

This part is the key step in data preparation. A sample is defined with a set of **features** (or feature vector), and its **label**. Label is the value that we want to predict (i.e., the value of time-series in the next

time interval). Feature vector is set of values that represent the sample. This set is given to the machine, then, machine predicts the label. In our modeling, we use 5 previous residual values as the feature of the next sample as illustrated in the following Figure.



As shown, we want to predict the value at index 51. So, the label is the value of time-series at index 51. In this case, the feature vector consists of the 5 previous values (values at indices: 46, 47, 48, 49, and 50). So, for each sample, we select the 5 previous values as the feature vector. Each feature is assigned to a new column. In the code, we used 5 features (`feature_dimension = 5`). Therefore we have 5 columns for our features (FEATURE_1, FEATURE_2, ..., FEATURE_5). FEATURE_1 include the first prior value, FEATURE_2 contains the second prior value, etc.

We explain this process in an example. Consider the following time-series which include 10 samples:

0.5, 0.7, -0.1, 0.4, 0.6, -0.9, -0.8, 0, 0.1, 0.5

All the values are in the range [-1, +1], so we do not need to change the range of values. We calculate the residuals by applying the difference operator on the data:

0.2, -0.8, 0.5, 0.2, 1.5, 0.1, 0.8, 0.1, 0.4

There are 9 samples in the residual time-series. To create a dataset, we will form a table as follow:

FEATURE_5	FEATURE_4	FEATURE_3	FEATURE_2	FEATURE_1	LABEL
NA	NA	NA	NA	NA	0.2
NA	NA	NA	NA	0.2	-0.8
NA	NA	NA	0.2	-0.8	0.5
NA	NA	0.2	-0.8	0.5	0.2
NA	0.2	-0.8	0.5	0.2	1.5
0.2	-0.8	0.5	0.2	1.5	0.1
-0.8	0.5	0.2	1.5	0.1	0.8
0.5	0.2	1.5	0.1	0.8	0.1
0.2	1.5	0.1	0.8	0.1	0.4

Note the residual time-series in the last column (LABEL). For each residual sample, the features vector consists of the 5 previous residual values. Each row of the table is a sample (with a feature vector, and a label).

Creating training and test sets

Now we can divide the table into two parts: train, and test sets. In the code, 700 time-series values have been assigned to the training set, and 100 samples to the test set. Then we create: `train_x`, and `train_y` which are respectively features and labels in the training set. Also, we have `test_x`, and `test_y` as the features and labels in the test set.

Modeling and Prediction

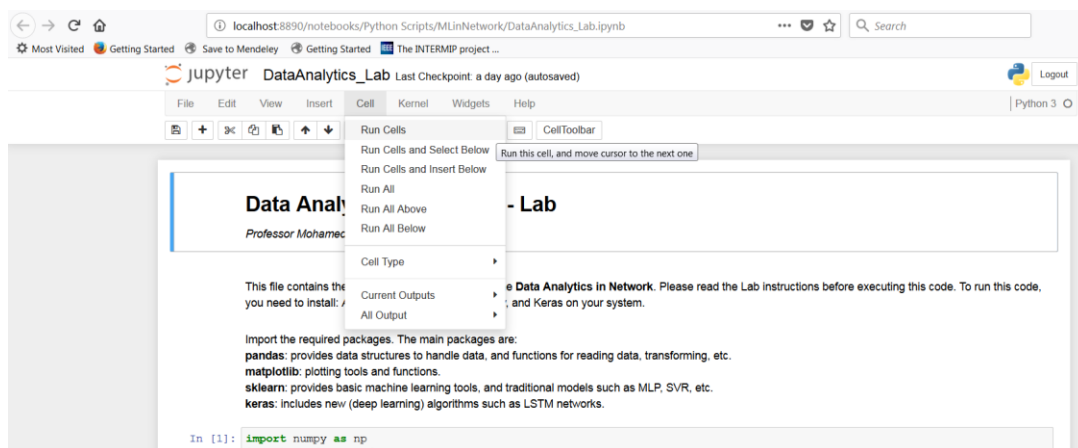
We use the training set (i.e., `train_x`, and `train_y`) to train a model. Then we give the `test_x` to the trained model to predict the test labels. The predicted (residual) values will be compared to the `test_y`. Finally, we create (or reconstruct) the time-series from the predicted (residual) values.

Evaluation and Comparison

We evaluate the accuracy of models using two metrics (i.e., MSE, and NMSE). Then these values are used to compare the models.

How to Run the Code

In Jupyter Notebook, you can run the code in a cell by selecting the cell, and then using `Shift+Enter`. Also you can select the cell, and then click on the `Cell > Run Cells` (the menu on the top of the Jupyter Notebook). These options are to run the cells one by one. There is also an option to run all the cells in the Jupyter Notebook by clicking on menu option `Cell > Run All`. Use the menu option `Kernel > Restart & Clear Output` to clear all the outputs and results.



Assignments

Your answer for the assignments includes:

- Table of results for the questions.
- The Notebook `DataAnalytics_Lab_assignment4.ipynb` (for assignment 4).
- The Notebook `DataAnalytics_Lab_assignment5.ipynb` (for assignment 5).

1. Change number of the features [10 Marks]

- Clear all the results. Then set the value of `feature_dimension` to 2 and run all the cells. Report the error values.
- Repeat the previous section for the following number of features:

[3, 4, 6, 7, 8, 9, 10]

Complete the following tables according to the results.

Table of Results (for **Residuals** time-series)

Number of features	MLP model		SVR model		LSTM model	
	MSE	NMSE	MSE	NMSE	MSE	NMSE
2						
3						
4						
5						
6						
7						
8						
9						
10						

Table of Results (for **Original** time-series)

Number of features	MLP model		SVR model		LSTM model	
	MSE	NMSE	MSE	NMSE	MSE	NMSE
2						
3						
4						
5						
6						
7						
8						
9						
10						

Table of Results (Training Time)

Number of features	MLP model time (sec)	SVR model time (sec)	LSTM model time (sec)
2			
3			
4			
5			
6			
7			
8			
9			
10			

2. Change number of training and test samples [10 Marks]

- In the code, the numbers of training and test samples are determined as 700, and 100 respectively. Clear the output. Change the number of training sample to 300 (number of test samples is 100). Run all the cells and report the error (NMSE and training time). Set the number of features as 5 (`feature_dimension = 5`).
- Repeat the previous section for the following number of training and test samples, and then complete the tables.

Table of Results (Error for **Residuals** time-series)

Number of Training Samples	Number of Test Samples	MLP model NMSE	SVR model NMSE	LSTM model NMSE
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			

Table of Results (Error for **Original** time-series)

Number of Training Samples	Number of Test Samples	MLP model NMSE	SVR model NMSE	LSTM model NMSE
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			

Table of Results (Training Time)

Number of Training Samples	Number of Test Samples	MLP model time (sec)	SVR model time (sec)	LSTM model time (sec)
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			

3. Change parameters of the model [10 Marks]

- a. SVR is a kernel-based model. In this question we will try different kernels in the model.

Open the file `DataAnalytics_Lab.ipynb`. Set the following configuration:

- `feature_dimension = 5`
- `train_num = 1500`
- `test_num = 500`

In the cell of SVR model, find the following command:

```
reg = SVR(kernel='sigmoid', epsilon=0.05)
```

The possible values for kernel are:

```
'rbf', 'linear', 'poly', 'sigmoid'
```

Use other kernels and complete the following table for the original time-series.

Kernel	SVR model NMSE	SVR model time (sec)
rbf		
linear		
poly		
sigmoid		

- b. LSTM network consists of internal elements neurons cells or memory. Number of neuron has been determined in command:

```
neurons = 600
```

Open the file `DataAnalytics_Lab.ipynb`. Set the following configuration:

- `feature_dimension = 5`
- `train_num = 1500`
- `test_num = 500`

Change the number of neurons and complete following table for the original time-series.

Number of neurons	LSTM model NMSE	LSTM model time (sec)
100		
200		
300		
400		
500		
600		
700		
800		
900		

4. Find the optimal number of features [30 Marks]

- a. In the Assignment 1, the different number of features have been tested. There are different approaches to feature selection and feature reduction. In a simple approach (in time-series modeling), the number of features can be selected using autocorrelation function of the data. Change the code and add a part to calculate the ACF (lag=1 to lag=100) for these three versions of data:

- Actual data (in column "DATA")
- Scaled data (in column "SCALED")
- Residual data (in column "DIFF")

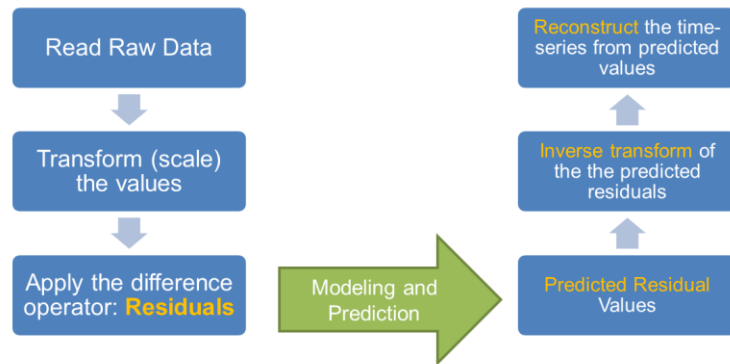
Determine the type of each time-series (LRD, or SRD) based on its ACF and compare the three versions of ACF. For this assignment, you need to create a copy of the DataAnalytics_Lab.ipynb (click on the menu `File > Make a Copy` ...). Rename the new file as the DataAnalytics_Lab_assignment4. You can use the available Python packages for calculating the ACF (or you can implement your own ACF). Add comments to your code and determine the ACF part.

- b. Consider the ACF of the residual signal (the third ACF) and the threshold `thr_acf`. The autocorrelation values which are greater than (or equal to) to `thr_acf` are assumed to be significant. What are the best number of features for the following values of correlation threshold:

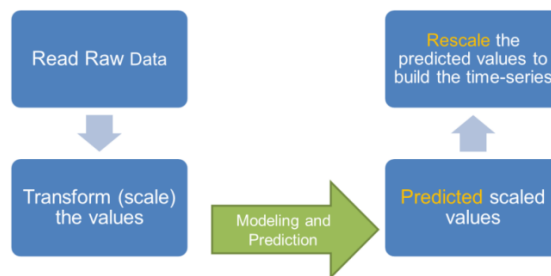
- `thr_acf = 0.5`
- `thr_acf = 0.2`
- `thr_acf = 0.1`
- `thr_acf = 0.05`

5. [40 Marks]

In our modeling, there are three steps before prediction. The process is as follow:



In the last step we apply the difference operator to the time-series data. In this assignment, we avoid that step. That means, we read raw data, apply the scaler (to change the range of values), and then we create our dataset using the scaled version of the time-series. So the new Process is as follow:



The goal of this assignment is to compare the performance of different models when we do not apply preprocessing. First, you need to create a copy of the `DataAnalytics_Lab.ipynb` (click on the menu `File > Make a Copy ...`). Rename the new file as the `DataAnalytics_Lab_assignment5`. Change the code in the new file to remove the difference operation. You also need to change the reconstruction function. Therefore, define and call a new function `reconstruct_2` instead of the function `reconstruct`. Set the number of features as 5 (`feature_dimension = 5`). Then complete following tables.

Table of Results (Error for **Scaled** time-series)

Number of Training Samples	Number of Test Samples	MLP model NMSE	SVR model NMSE	LSTM model NMSE
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			

Table of Results (Error for **Original** time-series)

Number of Training Samples	Number of Test Samples	MLP model NMSE	SVR model NMSE	LSTM model NMSE
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			

Table of Results (Training Time)

Number of Training Samples	Number of Test Samples	MLP model time (sec)	SVR model time (sec)	LSTM model time (sec)
100	100			
300	100			
500	150			
700	100			
800	250			
1000	250			
1500	500			
2000	300			
2000	700			
2000	1000			