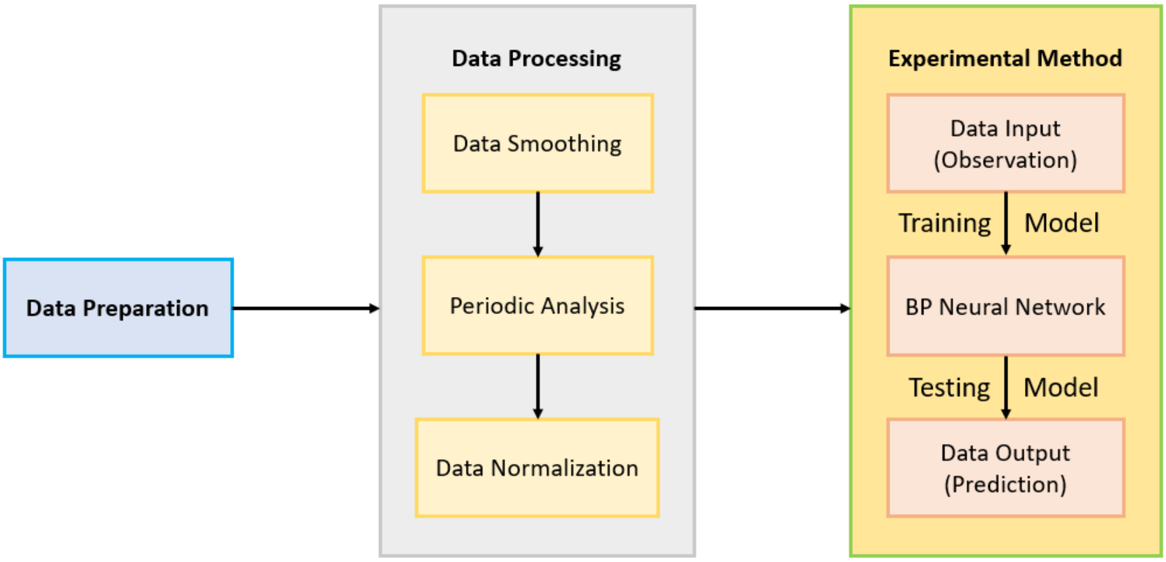
The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing**.

Documentation

Installation Instructions:

* Download Python3
* Download Jupyter Notebook
    1. pip install NumPy
    2. pip install Pandas
    3. pip install Matplotlib

User Guide:



The above shown is the software architecture, which consists of data preparation, data processing and experimental method. Afterwards, open the programme and compile the uploaded code, which will be demonstrated below. Besides, the input and output of the programme will also be mentioned.

Open Git Bash

cd Desktop

git clone https://github.com/acse-2019/irp-acse-cq419.git

Open Jupyter Notebook

/ Desktop / irp-acse-cq419 / Code / Notebook Code

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

1. Data Preparation:

```
def  read_data (filename):
        Read data from text file into lists.
        return list
```

```
def  list_transform_csv (Index, Hour, Elevation, scale, number):
        Transform lists into CSV.
        return CSV
```

```
def  onefigure_plot (start, end, x_axis, y_axis, colour,
                              label_name, title_name, True_or_False):
        Plot one figure.
        return 'one figure plot'
```

```
def  twofigure_plot (start, end, x_axis1, y_axis1, colour1,
                              label_name1, x_axis2, y_axis2, colour2,
                              label_name2, title_name, True_or_False):
        Plot two figure.
        return 'two figure plot'
```

Testing:

```
Index, Hour, Elevation = read_data('../ data / filename')
print(list_transform_csv(Index, Hour, Elevation, scale, number))
```

Comments:

filename means the name of the file, such as *Bournemouth 2020-02.txt*. The sample data is in **/ Desktop / irp-acse-cq419 / Code / data.** *Scale* means taking one data point every few intervals. For example, if *Scale* is selected as 4 or 6, then the timescale is 1 or 1.5 h. *number* is the maximum index in the selected data, such as 2500 or 2700. The input is the sample data file and the output is the ***DataFrame.***

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

```
x_list = list_transform_csv(Index, Hour, Elevation, scale, number)['Hour']

y_list = list_transform_csv(Index, Hour, Elevation, scale, number)['Elevation']

Figure = onefigure_plot(0, height, x_list, y_list, 'colour_name', 'label_name’,
                        'title_name', True)

print(Figure)
```

Comments:

x_list means the value of the time column in the ***DataFrame***, whose unit is hour. *y_list* means the value of the elevation column in the ***DataFrame***, whose unit is meter. *'label_name'* is the figure label and *'title_name'* is the figure title. *height* is the maximum height of vertical axis. *True* means adding grid in the figure. The input is the *height*, *x_list, y_list,* colour name, label name, title name, *True.* The output is the plotting figure whose horizontal axis is hour and vertical axis is elevation.

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

2. Data Preparation:

```
def  data_smooth (input_data):
        Make the data curve smooth.
        return data
```

```
def  data_normalization (input_data):
        Normalize the input data.
        return normalized data
```

Testing:

```
csv_data = list_transform_csv(Index, Hour, Elevation, scale, number)
time_series = data_smooth(csv_data['Elevation'])
data_smooth = onefigure_plot(0, height, csv_data['Hour'], time_series, 'colour_name',
                                        label_name, 'title_name', False)
print(data_smooth)
```

Comments:

csv_data['Elevation'] means the elevation column of the ***DataFrame.*** csv_data['Hour'] means the Hour column of the ***DataFrame.*** time_series is the data after smoothing. *False* means not adding grid in the figure. The input is the *height*, csv_data['Hour'], time_series, colour name, label name, title name, *False.* The output is the plotting figure whose horizontal axis is hour and vertical axis is elevation.

```
x_csv = csv_data['Hour'][: length]
y_csv = time_series[: length]
x_csv1 = csv_data['Hour'][number1: number1 + length]
y_csv1 = time_series[number1: number1 + length]
Periodic_Analysis = twofigure_plot(0, height, x_csv, y_csv, 'colour_name', 'Period 1',
                                        x_csv1, y_csv1, 'colour_name', 'Period 2', 'Tidal
                                        periodic analysis', False)
print(Periodic_Analysis)
```

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

Comments:

length is the selected interval length of the data observation. *x_csv* and *x_csv1* has the same interval length, whose unit is hour. *y_csv* and *y_csv1* are the tidal elevation in different period, whose unit is meter. *'Period 1'* and *'Period 2'* are the figure label and *'Tidal periodic analysis'* is the figure title. *height* is the maximum height of vertical axis. *False* means not adding grid in the figure. The input is the *height*, *x_csv*, *x_csv1*, colour name, label name, *y_csv*, *y_csv1,* colour name, label name, title name, *False.* The output is the plotting figure whose horizontal axis is hour and vertical axis is elevation.

```
x_nor = csv_data['Hour'][: new_number]

y_nor = data_normalization(csv_data['Elevation'])[: new_number]

new_plot = onefigure_plot(0, 1, x_nor, y_nor, 'colour_name', 'label_name', 'title_name',
                                    True)

print(new_plot)
```

Comments:

new_number is the selected interval length of the data observation. Since the data is normalized, then the interval of the vertical axis is chosen as zero to one. *x_nor* is the time column of the ***DataFrame.*** *y_nor* is the normalized elevation column of the ***DataFrame.*** *True* means adding grid in the figure. The input is the *x_nor*, *y_nor,* colour name, label name, title name, *True.* The output is the plotting figure whose horizontal axis is hour and vertical axis is elevation.

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

3. Experimental Method:

```
class BP_Neural_Network:
    Back propagation neural network algorithm.
    def __init__ (self):
    def sigmoid (self):
    def sigmoid_derivative (self):
    def rand_interval (self):
    def weights_matrix (self):
    def initial_setup (self):
    def predict (self):
    def back_propagate (self):
        Call predict function.
    def training_set (self):
        Call back_propagate function.
    def test_set (self):
        Call initial_setup and training_set function.
```

```
def list_simple_transform_csv (Elevation):
    transform list into csv.
    return data_csv
```

```
def create_sample_and_label ( input_data, number, set_index,
                              slice1, slice2, slice3, slice4,
                              choice):
    Create training samples and sample labels.
    return training samples, sample labels
```

```
def prediction_historical_sample (input_data1, input_data2,
                                  number, index):
    Create input historical samples.
    return new_data[: index].tolist()
```

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

```
def iteration_prediction_list (neural_network, historical_sample1,
                                 iterations, steps, number, choice):
    Use historical samples to do prediction.
    return historical_sample, len(historical_sample)
```

```
def data_transfer (input_data, normalized_data):
    Transfer normalized data into actual data.
    return np.array(new_data), len(new_data)
```

Testing:

```
data_A = list_transform_csv(Index, Hour, Elevation, scale, number)
data_A_elevation = data_A['Elevation']
data_B = list_transform_csv(Index1, Hour1, Elevation1, scale, number)
data_B_elevation = data_B['Elevation']
```

Comments:

data_A_elevation and *data_B_elevation* are tidal elevation data, which are selected from different sample data file.

```
new_data = prediction_historical_sample(data_A_elevation, data_B_elevation,
                total_number, index)
training_sample, training_label = create_sample_and_label(data_A_elevation,
                                    total_number, set_index, slice1, slice2, slice3,
                                    slice4, 'true')
```

Comments:

total_number is the maximum index in the selected data. *Index* means that the created input historical sample with first *index* data values. *set_index* is the number of residual values. *Slice1* means the number of initial selected training samples. The values of *slice2, slice3 and slice4* are all the same, which means each sub-sample label is the first value after every *index* continuous data values. *'true'* means adding residual values. The input is the data_A_elevation, total_number, set_index, slice1, slice2, slice3, slice4, 'true'. The output is the training_sample and training_label.

```
BP = BP_Neural_Network()

BP.test_set(training_sample, training_label, input, hidden, output, steps,
            parameter1,parameter2, paramter3)
```

Comments:

BP is the trained neural network. *input, hidden and output* mean the number of input, hidden and output neurons. *steps* mean the maximum iteration steps. *parameter1, parameter2, paramter3* represent learning rate, momentum factor and error convergence respectively. After training BP neural network, the connection weight matrix will be loaded.

```
normolized_prediction_list, normolized_prediction_length =
                          iteration_prediction_list( BP, new_data,
                          len(training_sample), number, index, 'true')

prediction_list, prediction_length = data_transfer(data_A_elevation,
                          normolized_prediction_list)
```

Comments:

number means the number of initial selected training samples. *index* is the number of residual values. *'true'* means adding residual values. *new_data* is the created historical sample. *normolized_prediction_list* is the result of the iterative prediction. The input is the data_A_elevation, *normolized_prediction_list.* The output is the prediction_list and prediction_length.

```
x_new_data = data_A['Hour'][: prediction_length]

y_new_data = prediction_list

x_new_data1 = data_B['Hour'][: prediction_length]

y_new_data1 = data_B_elevation[: prediction_length]

BP_method = twofigure_plot(0, height, x_new_data, y_new_data, 'colour_name',
                    'Prediction', x_new_data1, y_new_data1, 'colour_name',
                    'Observation', 'Tidal elevation analysis', True)

print(BP_method)
```

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

Comments:

y_new_data is the prediction value and *y_new_data1* is the observation value. *x_new_data* and *x_new_data1* are the observation time. *'Prediction'* and *'Observation* are the legend ids and *'Tidal elevation analysis'* is the figure title. *height* is the maximum height of vertical axis. *True* means adding grid in the figure.The input is the *height*, *x_new_data, y_new_data,* colour name, label name, *x_new_data1, y_new_data1,* colour name, label name, title name, *True.* The output is the plotting figure whose horizontal axis is hour and vertical axis is elevation.

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

Performance Analysis:

After finishing compiling the code in the software architecture, which consists of data preparation, data processing and experimental method. To further quantifying performance between prediction and observation, one new measurement indicator is introduced, calling the ***correlation coefficient***.

```
def  correlation_coefficient (prediction, observation):
        Calculate correlation coefficient regarding prediction and observation.
      return correlation_coefficient
```

Testing:

```
method0 = correlation_coefficient(y_new_data1, y_new_data1)
method1 = correlation_coefficient(y_new_data, y_new_data1)

fig, ax1 = plt.subplots(1, figsize=(7, 7))
fig.tight_layout(w_pad=4)

ax1.plot(y_new_data1, y_new_data1, 'colour_name', label = '(Actual Line) correlation
          coefficient: {:.4f}'.format(method0), markersize=5)
ax1.plot(y_new_data1, y_new_data, 'colour_name', label = '(BP Method) correlation
          coefficient: {:.4f}'.format(method1), markersize=5)

ax1.set_ylim([0, height]);
ax1.set_xlim([0, height]);
ax1.set_xlabel('Observation (m)', fontsize=16)
ax1.set_ylabel('Prediction (m)', fontsize=16)
ax1.set_title('Tidal observation site : Name  -  (Timescale : Hour)', fontsize=16)
ax1.legend(loc='best', fontsize=14)
ax1.grid(True)
```

The details of sample data testing can refer to *HTML Testing*, which is in **/ irp-acse-cq419 / Code / HTML Testing.**

Comments:

The input is the prediction and observation. For example, *y_new_data* and *y_new_data1*. The output is the *correlation coefficient*. *height* is the maximum value of the axis defined by user. The plotting figure illustrates the performance between the prediction and observation method0 is the actual line where prediction is always same as the observation, thus the *correlation coefficient* is 1. method0 is the measured line between the prediction and observation. The closer the *correlation coefficient* is to 1, the better the prediction accuracy of the trained neural network will be.