The files are categorized into four parts:

1. Data Processing

--- Test\_data

--- Train\_data

--- Vaid\_data

In this file, there are 3 sub files for preprocessing input data like delete blank variables and imputation. In each file, there is a step instruction about how to preprocess files.

2 Carry-forward

--- input.xls

--- targetvalid. xls

--- targettest. xls

--- targettrain. xls

--- test.m

The first four xls files are the same as the given TADPOLE data renamed. The test.m implements the carry-forward algorithm.

How to run the CarryForward model:

To run the code, you need to set up Matlab and run the test.m in it. Details of codes can be found in the comments in test.m.

3. SVM

--- SVR

--- SVR.py

--- dataset3

---target\_train.csv

---input\_train.csv

---input\_test.csv

---input\_valid.csv

---target\_test.csv

---target\_valid.csv

--- SVC

--- SVC.py

--- target\_test.csv

--- target\_test.csv

--- target\_test.csv

In SVM file, there are 2 sub files: SVR and SVC. In SVR, SVR.py can be used for regression prediction and preprocessed datasets are in dataset3. In SVC, SVC.py can be used for classification prediction and it has 3 preprocessed datasets. The details of data processing can be found in our report.

How to run the SVM models:

All parameters are set inside the SVR.py and SVC.py. To change to parameters, you will need to modify the parameters inside the code. Details of the SVM can be found in the comments of the code.

Example: call python SVR..py (SVC.py)

3. Data-modified LSTM

--- LSTM\_regression.py

--- LSTM\_classification.py

--- dataset5

--- dataset5copy

The LSTM\_regression.py and dataset5 are used to predict 3 continuous variables while LSTM\_classification.py and dataset5copy are used to make classification. The datasets in dataset5 and dataset5copy have been preprocessed.

How to run the D-LSTM::

All parameters are set inside the D-LSTM.py. To change to parameters, you will need to modify the parameters inside the code. Details of the data-modified LSTM can be found in the comments of the code.

Example: call python D-LSTM.py

4. Time-aware LSTM[1]

--- T-LSTM-master

--- main.py

--- result\_AD.mat

--- result\_ADAS13.mat

--- result\_Vent.mat

--- result\_MMSE.mat

---T-LSTM-master\_Classification

--- main.py

--- result\_AD.mat

--- result\_test\_AD.mat

--- result\_valid\_AD.mat

There are many similar datasets in T-LSTM-master because we have training, testing and validation datasets so we just list the sub flies of training datasets here. The original data needs to be transformed from CSV to the .mat format of MATLAB. The original information is decomposed into three .mat file. (i) Result\_patient.mat: 1\*12 cell, each cell yields[ number of patient \* visit times+1\* features] (ii)Result\_types.mat 1\*12 cell, if classification task, each cell yields[ number of patient \* one-hot encoded output]. If regression task, each cell yields [ number of patient \*1 ](iii)Result\_time.mat: 1\*12 cell, each cell yields[ number of patient \* 1\*time elapse of visits].

How to run the T-LSTM:  
Data is a list where each element is a 3 dimensional matrix which contains same length sequences.  
Instead of zero padding, same length sequences are put in the same batch.  
Example: L is the list containing all the batches with a length of N.  
 L[0].shape gives [number of samples x sequence length x dimensionality]  
Call python main.py 50 1028 512 2 0.6  
For instance; number\_epochs: 50, hidden\_dim:1028, fully\_connected dim: 512, output\_dim: 2, training dropout prob: 0.6

Our code is refereed from [1]. It is used for classification and we modified this code to make it can be used for regression problem.

Reference:

[1] https://github.com/illidanlab/T-LSTM