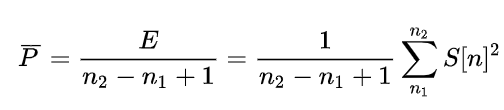
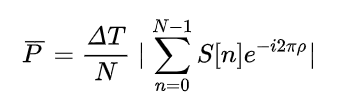
**Machine Learning**

The early mortality prediction systems may be faced with an imbalanced dataset. To handle this issue, we can utilize a resampling method called adaptive semi-unsupervised weighted oversampling (A-SUWO) to balance the dataset. After that, we can build Neural networks and traditional ML models to predict short-term and long-term outcomes.

**Predicting long-term outcomes: To predict the mortality at end of hospitalization, we will extract** **statistical and signal-based features and then** **implement several traditional machine learning approaches.**

In this study, two categories of traditional machine learning classifiers will be implemented: transparent and non-transparent models. Transparent classifiers such as decision tree, logistic regression, and support vector machine (SVM) using the linear kernel will explain hidden clinical implications and integrate background knowledge into the analysis. Also, they are not only easy to interpret and fast but also need small memory in practice. On the other hand, non-transparent classifiers like random forest, K-NN, boosted tree, and Gaussian SVM always provide adequate classification results.

**~~Problem to be discussed:~~** ~~How to extract valid features like~~ **~~averaged power~~** ~~and~~ **~~energy spectral density~~** ~~with the formula as follows:~~

**Predicting** **short-term outcomes: To predict changes in ICP, we will start out using LSTM models which are based on recurrent neural network structures, such as simple vanilla LSTM, stacked LSTM, and CNN-LSTM.**

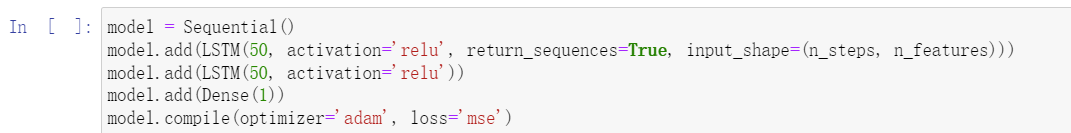
First, we have to split the time series: we can divide the sequence into multiple input/output patterns called samples, where n time steps are used as input and one time step is used as output for the one-step prediction that is being learned.

**Problem to be discussed:** How to divide the sequence reasonably? How to choose a proper n\_steps value?

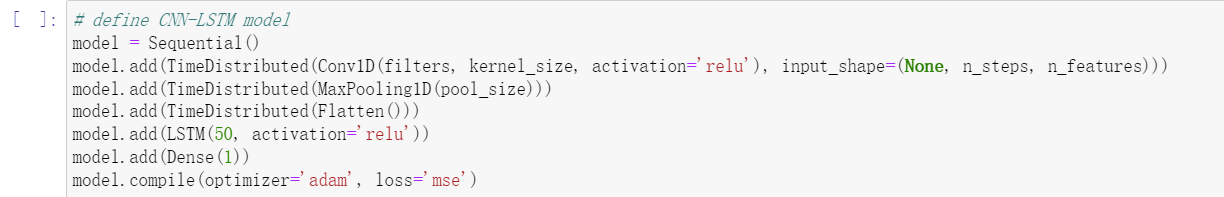
The model **Vanilla LSTM** has a single hidden layer of LSTM units, and an output layer used to make a prediction. 

In this case, we will define a model with 50 LSTM units in the hidden layer and an output layer that predicts a single numerical value. The model is fit using the efficient Adam version of stochastic gradient descent and optimized using the mean squared error, or ‘mse’ loss function.

On this basis, we can define a new **Stacked LSTM** model according to the following example:



When we implement **CNN-LSTM**, we want to reuse the same CNN model when reading each sub-sequence of data separately. This can be achieved by wrapping the entire CNN model in a [TimeDistributed wrapper](https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/) that will apply the entire model once per input, in this case, once per input subsequence.



The CNN model first has a convolutional layer for reading across the subsequence that requires a number of filters and a kernel size to be specified. The number of filters is the number of reads or interpretations of the input sequence. The kernel size is the number of time steps included of each ‘read’ operation of the input sequence. The convolution layer is followed by a max pooling layer that distills the filter maps down to 1/2 of their size that includes the most salient features. These structures are then flattened down to a single one-dimensional vector to be used as a single input time step to the LSTM layer.

**Problem to be discussed:** How to split the input sequences into subsequences that can be processed by the CNN model?

**Problem to be discussed**: We have several different kinds of signals like ECG ABP, etc. Should I do multivariant time series forecasting with NN? Or just univariant time series forecasting?