Want: Let’s try to forecast is tomorrow’s weather will be Rain, Snow, or Sunny using the last 2 days of data. (Note: FJF uses the last 90 days od data to predict tomorrow stock price)

Given: Let’s assume we have data for the past 10 days (Note: FJF has data from the last 10 years):

1	2	3	4	5	6	7	8	9	10
❄	❄	☀	🌧	❄	🌧	☀	☀	🌧	❄

PART 1: CREATING OUR MODEL

Since we use an 80:20 training to testing model in FJF, let’s apply the same concept here.

Our training data set will be first 80% of our entire value set. Our training set will be:

1	2	3	4	5	6	7	8
❄	❄	☀	🌧	❄	🌧	☀	☀

Our testing data set will be the last 20% of our entire value set. Our testing set will be:

9	10
🌧	❄

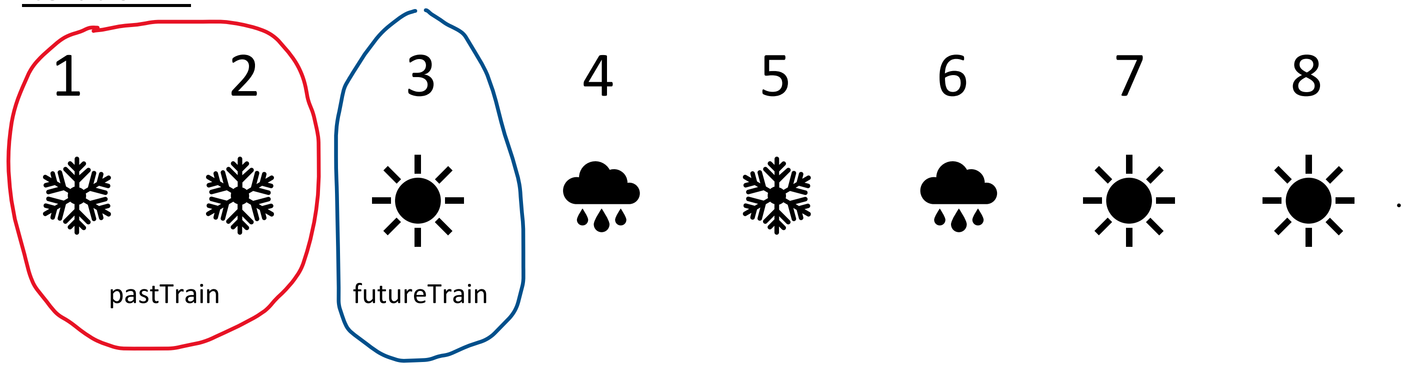
PART 2: TRAINING

Now, let's begin our training. FJF uses pastTrain and futureTrain value sets to pull samples from our training set to give to the LSTM to train.

In this example, pastTrain will be used to store sets of 2 days starting from the beginning of our training data set. futureTrain will be used to store the following value after pastTrain until the end of the training set.

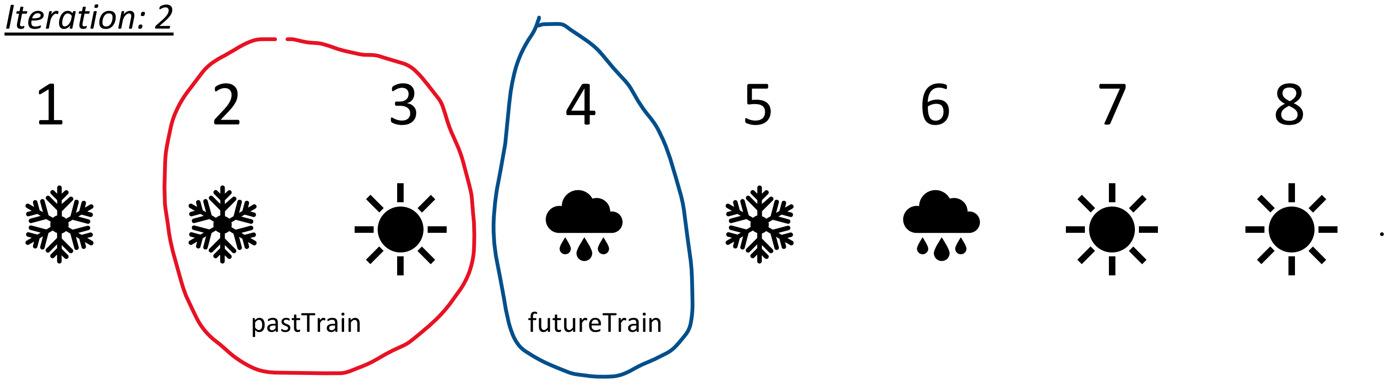
Here's how pastTrain and futureTrain are populated:

Iteration: 1



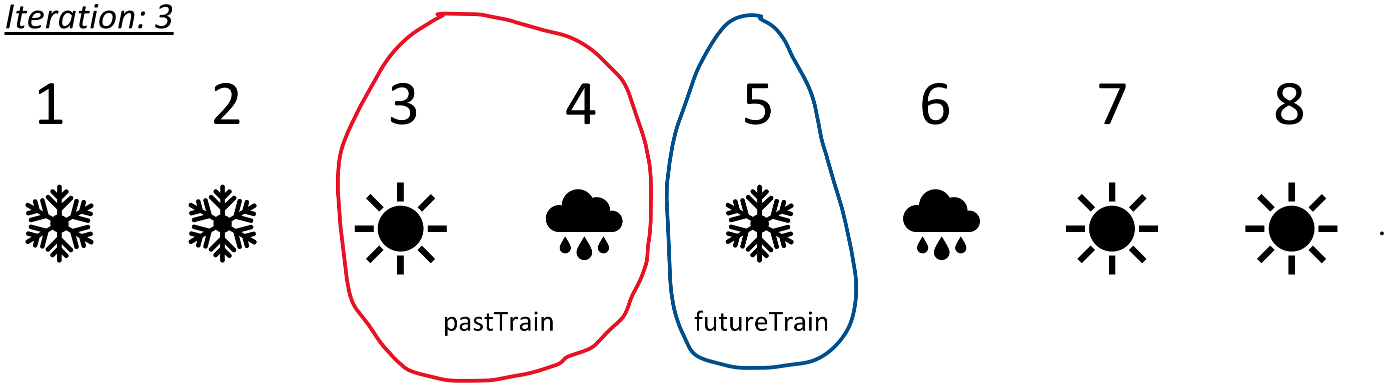
- pastTrain now contains: { (Snow, Snow) }
- futureTrain now contains: { (Sunny) }

Iteration: 2



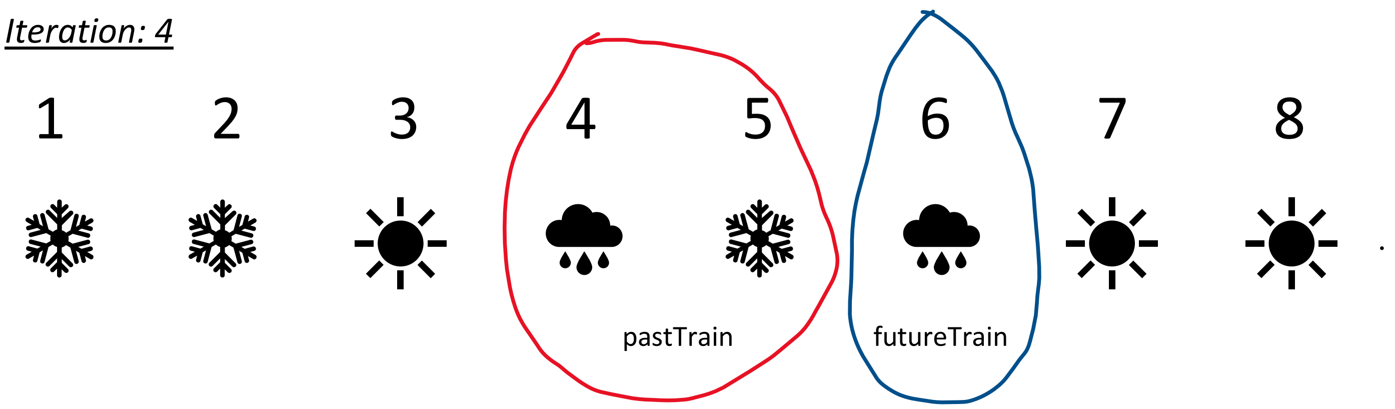
- pastTrain now contains: { (Snow, Snow) , (Snow, Sunny) }
- futureTrain now contains: { (Sunny) , (Rain) }

Iteration: 3



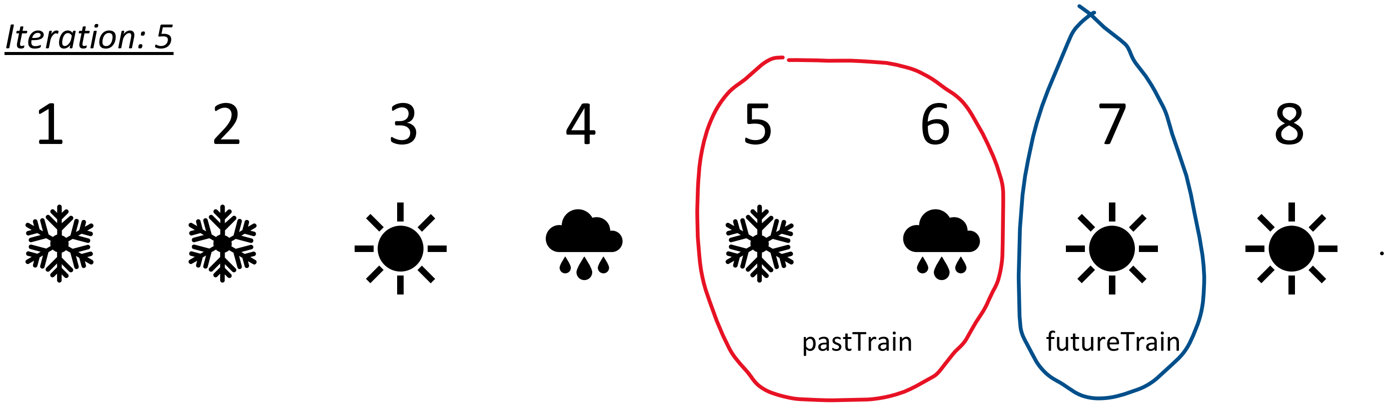
- pastTrain now contains: { (Snow, Snow) , (Snow, Sunny) , (Sunny, Rain) }
- futureTrain now contains: { (Sunny) , (Rain) , (Snow) }

Iteration: 4



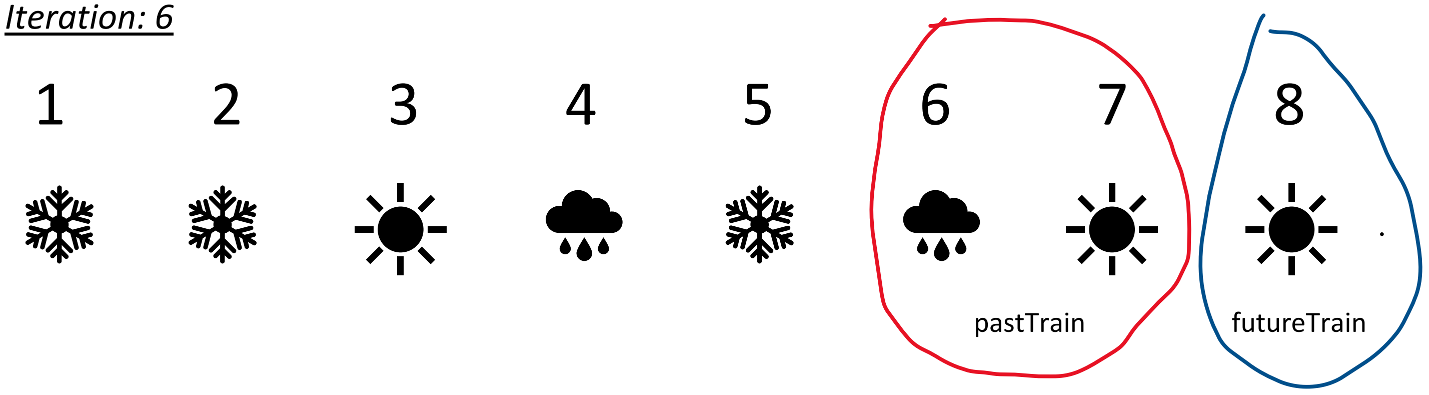
- pastTrain now contains: { (Snow, Snow) , (Snow, Sunny) , (Sunny, Rain) , (Rain, Snow) }
- futureTrain now contains: { (Sunny) , (Rain) , (Snow) , (Rain) }

Iteration: 5



- pastTrain now contains: { (Snow, Snow) , (Snow, Sunny) , (Sunny, Rain) , (Rain, Snow) , (Snow, Rain) }
- futureTrain now contains: { (Sunny) , (Rain) , (Snow) , (Rain) , (Sunny) }

Iteration: 6



- pastTrain now contains: { (Snow, Snow) , (Snow, Sunny) , (Sunny, Rain) , (Rain, Snow) , (Snow, Rain) , (Rain, Sunny) }
- futureTrain now contains: { (Sunny) , (Rain) , (Snow) , (Rain) , (Sunny) , (Sunny) }

We created these pastTrain and futuretrain sets so that we can feed them to the LSTM to learn trends that have occurred in training data set.

Now we can assume the following trends that follow the general format, pastTrain[i] correlates to futureTrain[i]:

Index [0]: Snow followed by Snow results in a Sunny day

Index [1]: Snow followed by a Sunny day results in Rain

Index [2]: A Sunny Day followed by Rain results in Snow

Index [3]: Rain followed by Snow results in Rain

Index [4]: Snow followed by Rain results in a Sunny day

Index [5]: Rain followed by a Sunny day results in a Sunny day

PART 3: LEARNING

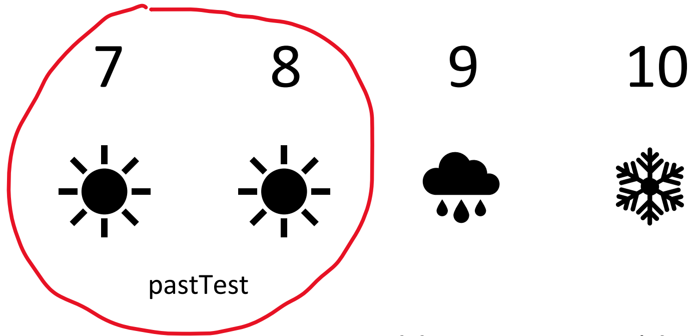
Now we will feed our pastTrain and futureTrain value sets to our LSTM to teach our program trends from the training data set to forecast into the future.

PART 4: TESTING

Now, let's begin our testing. During the testing phase, FJF will use pastTest to be used as controlled data to forecast the value for the next day. Since in this example we are using the past 2 days to predict the weather for tomorrow, we will start our pastTest as the last 2 days before the testing set begins. pastTest will iterate through the set until the end of the entire set is reached.

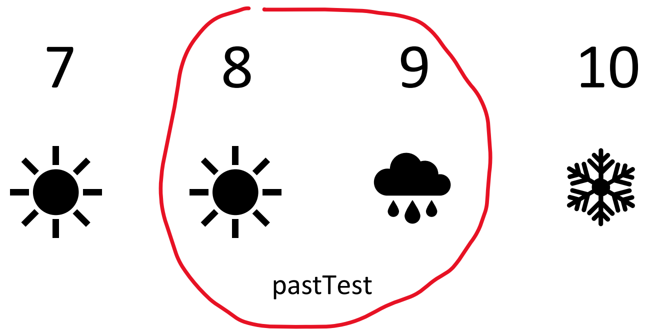
(Remember: Our test set only includes values 9 and 10, we are just using values 7 and 8 to predict into our test set so that we can compare the LSTM's outputs to values 9 and 10)

Iteration: 1



pastTest now contains: { {Sunny, Sunny,) }

Iteration: 2



pastTest now contains: { (Sunny, Sunny,) , (Sunny, Rain) }

Now pastTest has reached the end of the data set and now we can feed pastTest into our LSTM to see what forecasts it will output. The LSTM will give a prediction for each pastTest set.

For example,

The LSTM will read (Sunny, Sunny) and give an output.

- LSTM Predictions now contain { (forecast1) }

The LSTM will read (Sunny, Rain) and give an output.

- LSTM Predictions now contain { (forecast1) , (forecast2) }

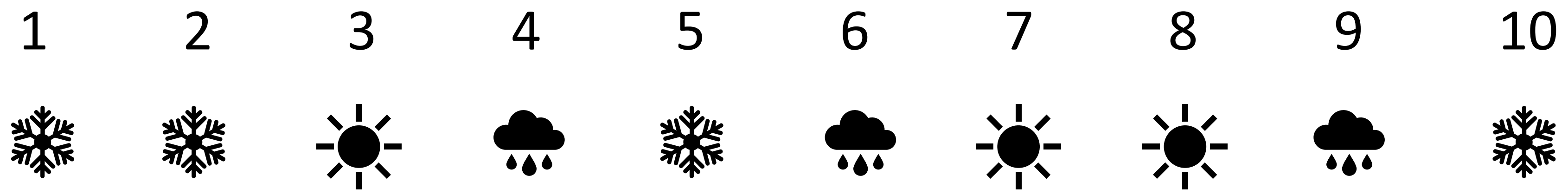
Now we are able to compare the LSTM's forecasted values { (forecast1), (forecast2) } with our actual values { (rain), (snow) } and determine accuracy.

PART 5: FUTURE FORECAST

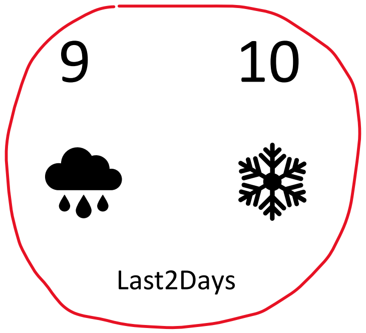
Now that we have trained our LSTM and tested it against stored values, we can now forecast into the future. This will follow a similar method to the testing routine, however we will not be able to determine accuracy in this case because we have nothing to compare it to.

Remember, our program uses the last 2 days to forecast into the future. The future forecast will look something like this:

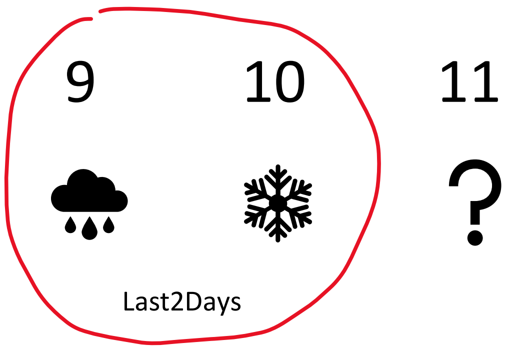
1. Let’s refresh ourselves on the entire data set.



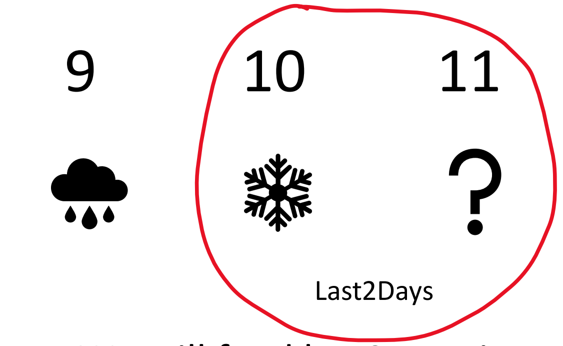
2. To predict into the future, we’ll use the last 2 days, which in our case, will be values 9 and 10. We will create an array to hold the last 2 days at any given time.



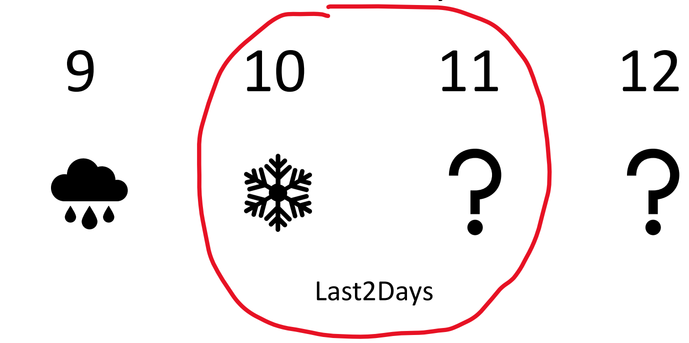
3. We will feed last2Days into our LSTM to generate a forecast value for tomorrow.



4. We will then increment last2Days by 1 so that we can continue this process.



5. We will feed last2Days into our LSTM to generate a forecast value for tomorrow.



6. This process will continue until the desired amount of forecasted days are reached.