Capstone project

Abhishek Pandey

Machine learning engineer nanodegree

November 4, 2019

I. Definition

a. Project overview

Climate change is not a new term to common man anymore. It is an issue, which if not addressed soon enough, may lead to unprecedented devastations. Weather pattern has started to change. The extremes of seasons have risen to a whole new level and its catching a new one every subsequent year.

Even if we take all the necessary steps to reverse the effects of climate change, it would take few years for the steps to have substantial impact and to have clear visualization. Till then, the only option we are left with is to prepare ourselves enough to counter the devastations caused by natural calamities. Two of these natural calamities are flood and extreme temperatures. Every year thousands of lives are lost due to severe weather conditions and heavy rainfall. If we can accurately predict the severity of weather well in advance, then appropriate precautionary measures could be taken to save lives.

Here, with this project, I am making an attempt to predict weather based on past weather conditions. With the help of this model, one would be able to predict temperature conditions and amount of rainfall in any given month. For this model, dataset used will consist of temperature and rainfall patterns at Indian subcontinent which can be found here: Jun-Sept rainfall in central India data, Annual Minimum/Maximum temperature data, Monthly average temperature .

b. Problem statement

The current method used by metrological department, although produces good results, but are not robust enough to handle vast complexities involved in atmospheric phenomenon and are also prone to error given perturbations in input data.

Machine learning on the other hand are more robust to handle data perturbations and doesn't require prior knowledge of complexities involved in natural phenomenon. The task here is to predict temperature and rainfall in coming years given previous year's dataset. This is a regression problem involving time series analysis.

The dataset is mainly composed of two variables- temperature and rainfall. The temperature variable is further divided into annual minimum, monthly minimum, annual maximum, monthly maximum, annual mean and monthly mean. Rainfall is further divided into June rainfall, July rainfall, August rainfall, September rainfall. There are data for 116 years, from year 1901 to year 2016. Temperature and rainfall for any given month may depend of temperature and rainfall of previous years and/or temperature and rainfall of previous months. Formulating this problem as time series analysis, this project intends to forecast weather (temperature and rainfall) for coming years.
Temperature of an area is one of the factors that are directly proportional to total rainfall in an area. Similarly, amount of rainfall also decides temperature range in the area. Hence study of temperature (min, max and mean) and amount of rainfall for past years can help us to predict future temperature and rainfall.
As time series analysis will be required to solve this kind of problem, Naïve approaches like Single Exponential smoothing, Holt's linear trend method can be used to initial study and then more advanced methods like Holt's Winter seasonal method, ARIMA, multivariate time series prediction using LSTMs RNN can also be applied if required.

Tasks involved in making this model are as follows:

1. Gathering the required dataset
2. Preprocessing the dataset according to the requirements of the model.
3. Defining and evaluating the dataset on benchmark model
4. Defining a better model with appropriate parameters
5. Training the model and monitoring the losses
6. Deciding on the best hyper-parameters via loss monitoring.
7. Evaluating test dataset on the improved model
8. Comparing the errors of benchmark model and improved model.

The output of the model would give us temperate conditions and amount of rainfall for subsequent years on monthly basis.

c. Metrics

Since this is a regression problem and every input feature will have a specific output value, the best method for evaluating performance of the model is RMSE (root mean square error).

$$\text{rmse} = \frac{\sqrt{\sum(\text{actual value} - \text{predicted value})^2}}{length\ of\ dataset}$$

RMSE will also be used for comparing the improved model with the benchmark model.

II. Analysis

1. Data Exploration

The dataset used was taken form data.gov.in. Three datasets (Jun-Sept rainfall in central India data, Annual Minimum/Maximum temperature data, Monthly average temperature) were merged to form a single dataset and the final assembly looked as follows.

	year	annual-min	jan-min	feb-min	mar-min	apr-min	may-min	jun-min	jul-min	aug-min	...	jul-mean	aug-mean	sep-mean	oct-mean	nov-mean	dec-mean	jun-rain	jul-rain	aug-rain	sept-rain
0	1901	19.51	13.58	14.72	17.91	20.93	23.18	24.05	23.82	23.58	...	27.49	26.98	26.26	25.08	21.73	18.95	99.3	295.4	354.8	113.8
1	1902	19.44	13.08	14.20	18.44	21.30	23.63	23.97	23.68	23.34	...	27.29	27.05	25.95	24.37	21.33	18.78	62.3	334.2	237.6	216.0
2	1903	19.25	13.20	14.55	17.12	20.67	22.95	23.85	23.53	23.29	...	28.04	26.63	26.34	24.57	20.96	18.29	96.2	392.7	286.6	211.5
3	1904	19.22	13.04	14.07	17.70	21.42	23.02	23.64	23.35	23.03	...	26.84	26.73	25.84	24.36	21.07	18.84	180.5	259.1	200.9	165.8
4	1905	19.03	12.83	12.74	16.88	19.62	23.32	24.13	23.93	24.31	...	27.67	27.47	26.29	26.16	22.07	18.71	59.8	340.6	178.9	204.1

	year	annual-min	jan-min	feb-min	mar-min	apr-min	may-min	jun-min	jul-min	aug-min	...	jul-mean	aug-mean	sep-mean	oct-mean	nov-mean	dec-mean	jun-rain	jul-rain	aug-rain	sept-rain
111	2012	19.54	12.91	14.45	17.79	21.30	23.15	24.22	24.12	23.69	...	27.98	27.31	26.65	24.85	22.26	19.91	99.3	311.9	305.5	218.2
112	2013	19.83	13.22	15.54	18.45	21.29	23.68	24.08	23.94	23.70	...	27.50	27.22	26.87	25.63	22.18	19.69	275.9	437.6	299.2	180.7
113	2014	19.77	13.79	14.72	17.73	21.07	23.16	24.68	24.37	23.58	...	28.07	27.42	26.61	25.38	22.53	19.50	64.6	358.5	256.4	201.1
114	2015	19.96	13.51	15.55	17.99	21.18	23.54	23.82	24.28	23.82	...	28.03	27.64	27.04	25.82	22.95	20.21	204.5	267.9	204.8	141.8
115	2016	21.28	14.91	17.44	20.59	23.75	25.10	25.36	24.72	24.56	...	28.18	28.17	27.72	26.81	23.90	21.89	137.3	382.9	320.9	194.6

The dataset is of size 116 x 44, where 116 represent year from 1901 to 2016 and 43 represents total number of features. All the temperature value presented are upto two decimal places and the rainfall values are upto one decimal place. There are no categorical variables and also there are no missing values in the dataset.

Following features are involved in this dataset.

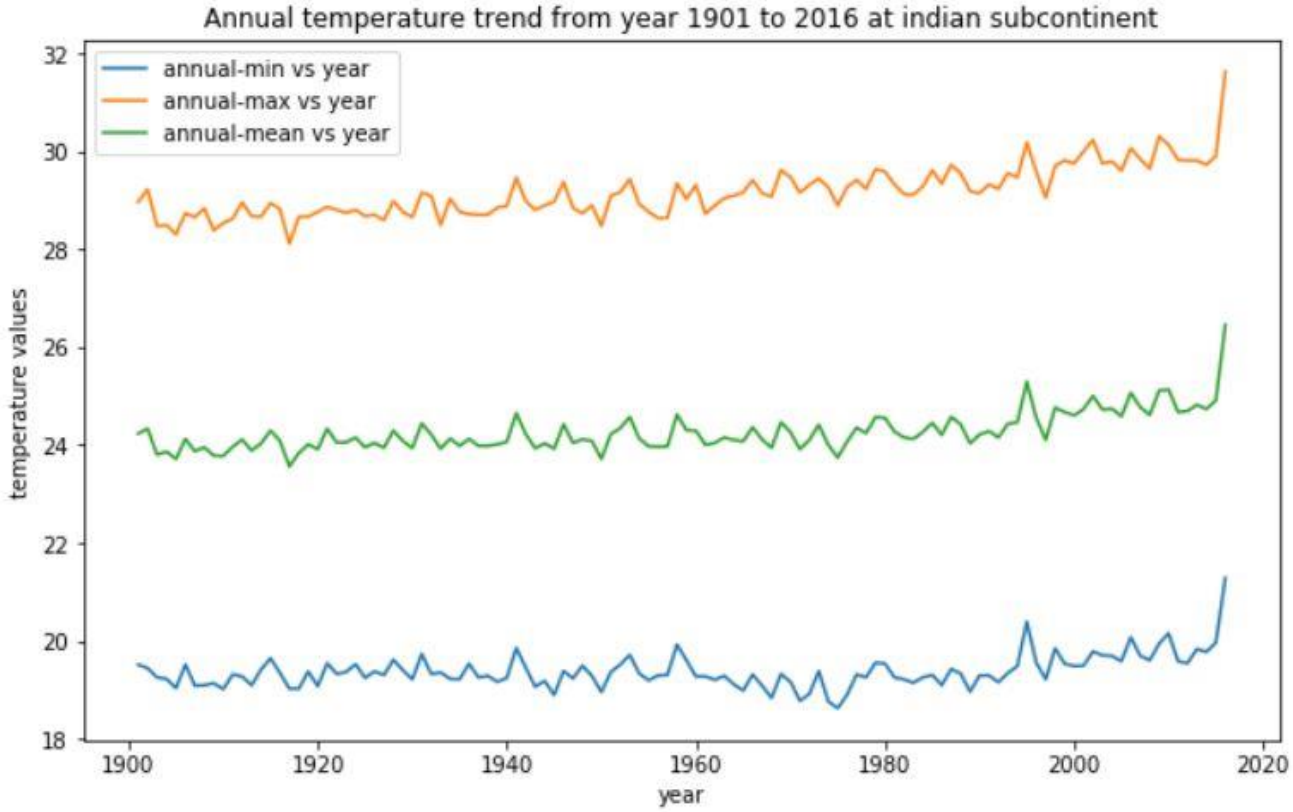
```
['year' 'annual-min' 'jan-min' 'feb-min' 'mar-min' 'apr-min' 'may-min'
 'jun-min' 'jul-min' 'aug-min' 'sep-min' 'oct-min' 'nov-min' 'dec-min'
 'annual-max' 'jan-max' 'feb-max' 'mar-max' 'apr-max' 'may-max' 'jun-max'
 'jul-max' 'aug-max' 'sep-max' 'oct-max' 'nov-max' 'dec-max' 'annual-mean'
 'jan-mean' 'feb-mean' 'mar-mean' 'apr-mean' 'may-mean' 'jun-mean'
 'jul-mean' 'aug-mean' 'sep-mean' 'oct-mean' 'nov-mean' 'dec-mean'
 'jun-rain' 'jul-rain' 'aug-rain' 'sept-rain']
```

As it can be seen, the dataset contains annual minimum, monthly minimum, annual maximum, monthly maximum, annual mean, monthly mean and rainfall for the months of June, July, August and September.

2. Exploratory Visualization

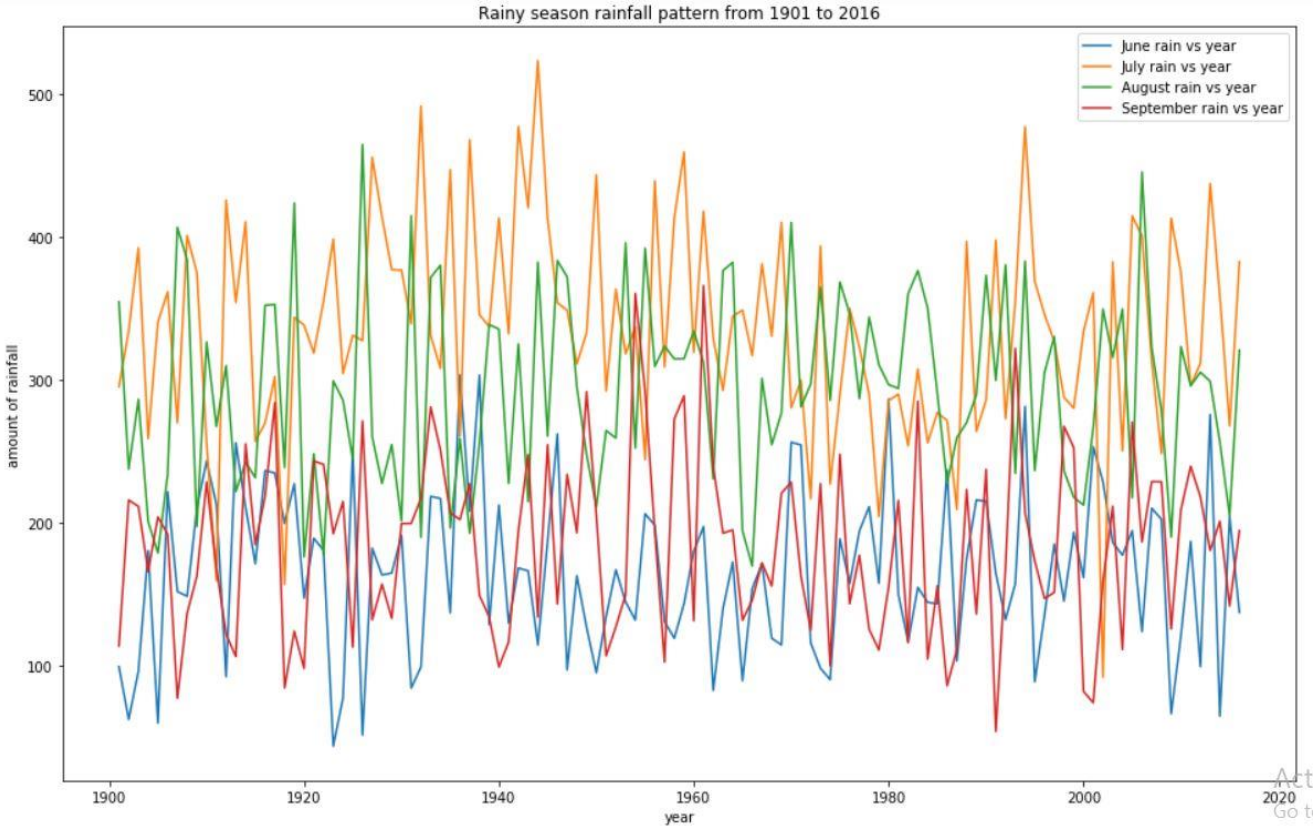
Both temperature and rainfall trends have been visualized for their study.

On visualization, temperatures showed an increasing trend which can be understood as an effect of global warming. Also, on careful visualization, one can see sharper rise in later years than early years as seen below.



Here annual minimum, annual mean and annual maximum are plotted across years as a general visualization for temperature pattern as monthly minimum, monthly mean, monthly maximum are bound to follow similar trend respectively.

Study of rainfall pattern showed a different kind of behavior. Month of June and September which are beginning month and end month for monsoon in India receive similar amount of rainfall while month of July received maximum amount of rainfall followed by August which is evident in the below figure.
An interesting behavior is also displayed by rainfall pattern. Amount of rainfall increased from 1901 to near about 1940 and then decreased till around 1990 and then again is in increasing trend from 1991 onwards.



3. Algorithms and Techniques

Three different time series analysis models have been used. One of them is a bench mark model and other two are used as advanced models for improved prediction.

Simple Exponential Method: This method is also used a bench mark model. Simple exponential model takes into account all the data while weighing the data points differently. This method attaches larger weights to more recent observations than to observations from the distant past. The algorithm forecasts are calculated using weighted averages where the weights decrease

exponentially as observations come from further in the past; the smallest weights are associated with the oldest observations. The equation is as follows:

$$\hat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots$$

Where 0≤ α ≤1 is the **smoothing** parameter. The rate at which the weights decrease is controlled by the parameter α. **For this project, α = 0.5, 0.3, 0.1, 0.8 were used to study this dataset and α = 0.3 produced minimum rmse.**

Holt's Linear Trend Method: Holt extended simple exponential smoothing to allow forecasting of data with a trend. It is nothing more than exponential smoothing applied to both level(the average value in the series) and trend. To express this in mathematical notation we now need three equations: one for level, one for the trend and one to combine the level and trend to get the expected forecast ŷ. The equations are as follows:

$$Forecast\ equation: \hat{y}_{t+h|t} = \zeta_t + h\, b_t$$
$$Level\ equation: \zeta_t = \alpha y_t + (1-\alpha)(\zeta_{t-1} + b_{t-1})$$
$$Trend\ equation: b_t = \beta * (\zeta_t - \zeta_{t-1}) + (1-\beta) b_{t-1}$$

As with simple exponential smoothing, the level equation here shows that it is a weighted average of observation and the within-sample one-step-ahead forecast The trend equation shows that it is a weighted average of the estimated trend at time t based on $l_{(t)} - l_{(t-1)}$ and $b_{(t-1)}$, the previous estimate of the trend.

We will add these equations to generate Forecast equation. We can also generate a multiplicative forecast equation by multiplying trend and level instead of adding it. When the trend increases or decreases linearly, additive equation is used whereas when the trend increases of decreases exponentially, multiplicative equation is used.

Long Short Term Memory Recurrent Neural Networks (LSTM RNN): Advanced deep learning models such as Long Short Term Memory Networks (LSTM), are capable of capturing patterns in the time series data, and therefore can be used to make predictions regarding the future trend of the data.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell. Intuitively, the *cell* is responsible for keeping track of the dependencies between the elements in the input sequence. The *input gate* controls the extent to which a new value flows into the cell, the *forget gate* controls the extent to which a value remains in the cell and the *output gate* controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM *gates* is often the logistic sigmoid function. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

Cell structure can be seen below.

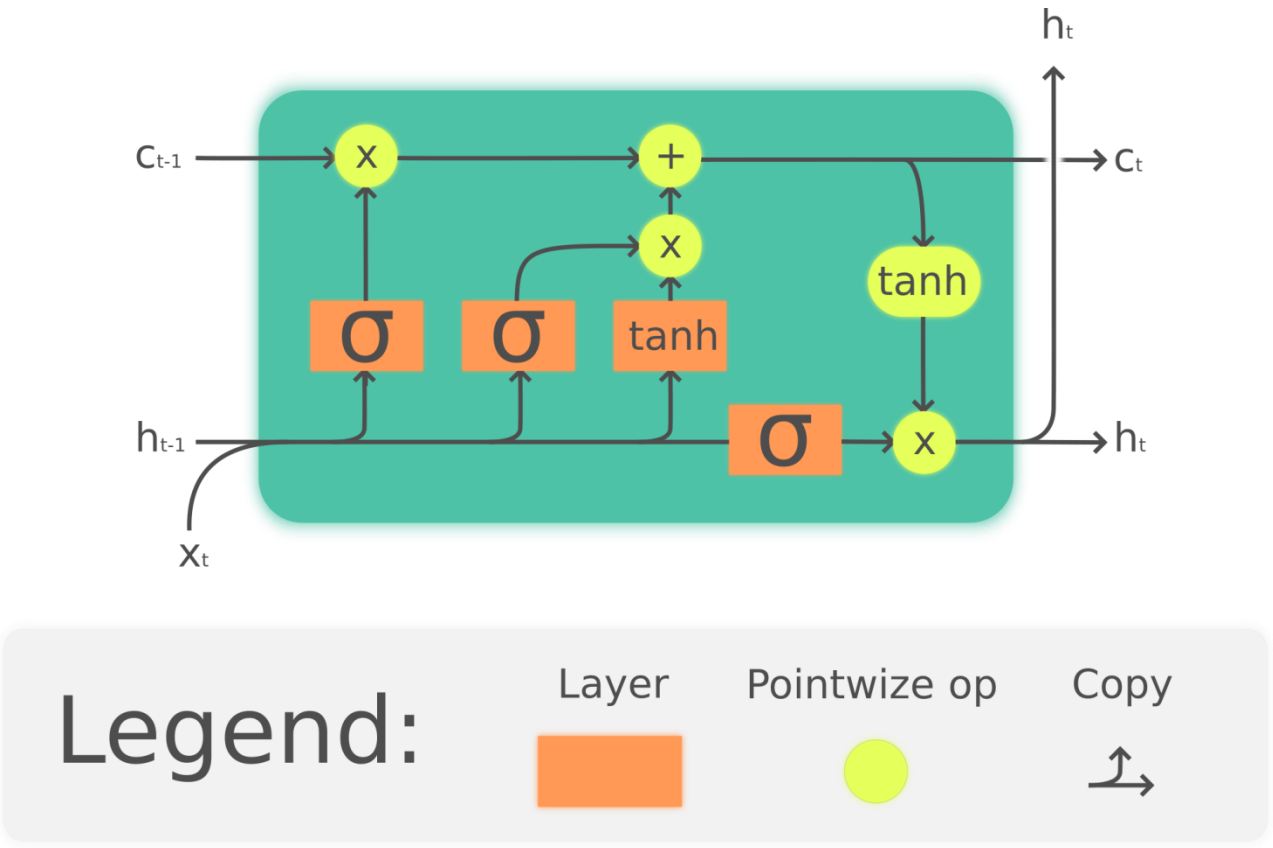


Image Source: Wikipedia

The gates in this cell can be described as follows:

LSTM with a forget gate

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

Variables

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in \mathbb{R}^h$: forget gate's activation vector
- $i_t \in \mathbb{R}^h$: input/update gate's activation vector
- $o_t \in \mathbb{R}^h$: output gate's activation vector
- $h_t \in \mathbb{R}^h$: hidden state vector also known as output vector of the LSTM unit
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Activation functions

- σ_g: sigmoid function.
- σ_c: hyperbolic tangent function.
- σ_h: hyperbolic tangent function

Image Source: Wikipedia

For Simple Exponential method and holt linear trend method implementation, **statsmodels** library was used. fit() method of these respective algorithms were used fit train these models for the dataset. Then rmse was calculated against the predicted results and test values.
There was no specific data preprocessing step required for these two methods. Entire column for a specific feature was used as input.
For simple exponential method value of smoothing level = 0.3 was used and for holt linear trend method value of smoothing slope=0.1 and smoothing level = 0.3 was used.
LSTMs required data preprocessing. The data was normalized using MinMaxScaler() method of sklearn.preprocessing to convert all the data into range (-1, 1). Further, the input data was transformed into batch of batch_size (in this case =1) and specific sequence length (here, 24). The normalized and sequenced data was fed into defined LSTM network with hidden dimension = 100 and number of layers =2. Apart from these, a smaller learning rate of 0.0001 was used. For calculating error, mean square error loss was used and optimizer used was Adam optimizer. And finally the model was run for 300 epochs to arrive at loss less than 0.1.
To decide on the values of these parameters, the network was run over various parameters and above mentioned parameters lead to minimum loss.
The preprocessed data was first passed through LSTM layers. The output of first LSTM layer is passed on as input of second hidden layer. The final output is flattened and then it is trained through a feed forward network and then required number of output values is generated.

4. Benchmark

Simple Exponential method is used as benchmark method as it is the simplest of all methods applied and doesn't consider trend in its method. The predicted values had same result for all the years as evident form figure below

year	annual-min	jan-min	feb-min	mar-min	apr-min	may-min	jun-min	jul-min	aug-min	...	jul-mean	aug-mean	sep-mean	oct-mean	nov-mean
2001	19.547747	13.334009	14.986686	17.763504	21.182447	23.34734	23.813917	23.82222	23.395397	...	27.54922	27.105341	26.681902	25.130567	22.405087
2002	19.547747	13.334009	14.986686	17.763504	21.182447	23.34734	23.813917	23.82222	23.395397	...	27.54922	27.105341	26.681902	25.130567	22.405087
2003	19.547747	13.334009	14.986686	17.763504	21.182447	23.34734	23.813917	23.82222	23.395397	...	27.54922	27.105341	26.681902	25.130567	22.405087
2004	19.547747	13.334009	14.986686	17.763504	21.182447	23.34734	23.813917	23.82222	23.395397	...	27.54922	27.105341	26.681902	25.130567	22.405087
2005	19.547747	13.334009	14.986686	17.763504	21.182447	23.34734	23.813917	23.82222	23.395397	...	27.54922	27.105341	26.681902	25.130567	22.405087

It achieved rmse values for temperature profiles, in the range of (-1, 2). But in case of rainfall predictions the result wasn't appreciable. There was large variation in rmse values for each month rainfall prediction. The given below image dictates the calculated rmse values for all input features.

```
rmse_exp
```

```
{'annual-min': 0.5132513590975231,
 'jan-min': 0.6665157046452513,
 'feb-min': 0.8806026708421039,
 'mar-min': 1.015602222782082,
 'apr-min': 0.8328656215104412,
 'may-min': 0.5684815367159812,
 'jun-min': 0.5039300206208394,
 'jul-min': 0.3459972223268492,
 'aug-min': 0.41679828566889154,
 'sep-min': 0.41923865374130037,
 'oct-min': 0.7289138717350783,
 'nov-min': 0.5869387599421291,
 'dec-min': 0.9010448856240822,
 'annual-max': 0.5807570993892286,
 'jan-max': 0.97691944182958,
 'feb-max': 1.43349528337312,
 'mar-max': 1.3539781306641319,
 'apr-max': 0.8877859013655024,
 'may-max': 0.6909737180277318,
 'jun-max': 0.7346333131789662,
 'jul-max': 0.48054961389241996,
 'aug-max': 0.4553696493250594,
 'sep-max': 0.3835067127283721,
 'oct-max': 0.6287592398944386,
 'nov-max': 0.6256138646394915,
 'dec-max': 0.7187415985188413,
 'annual-mean': 0.5432060461454536,
 'jan-mean': 0.748311289601709,
 'feb-mean': 1.117403224451074,
 'mar-mean': 1.3135687713563002,
 'apr-mean': 0.8484687905322217,
 'may-mean': 0.622718544552888,
 'jun-mean': 0.5777800244133491,
 'jul-mean': 0.3735186248044951,
 'aug-mean': 0.40920657071106825,
 'sep-mean': 0.3262343166473149,
 'oct-mean': 0.5730022292348036,
 'nov-mean': 0.5184522914427212,
 'dec-mean': 0.7449788554004532,
 'jun-rain': 60.917211442572395,
 'jul-rain': 85.53031188283303,
 'aug-rain': 76.23888311275189,
 'sept-rain': 51.541648493899096}
```

III. Methodology

1. Data preprocessing

The gathered dataset had no missing values and no outliers as well. Hence, Simple Exponential method and Holt linear trend method required no preprocessing as for these methods inputs are plain numerical values as in dataset.

However, LSTMs required feature transformation as preprocessing step. The dataset was split into test and train dataset with test data size =10. Further, for train dataset there were two

steps of preprocessing: normalization and converting the normalized data into fragments of given batch size and sequence length.
For normalizing data sklearn.preprocessing.MinMaxScaler() method was used and the data was converted into feature range of (-1, 1). The last two values of normalized training dataset looked as follows:

```
[[ 0.00564957  0.27196598  0.         -0.0851078   0.02745056 -0.35092354
  -0.06493568  0.44444466  0.15607071 -0.22891617 -0.01754284 -0.14195538
  -0.10163879 -0.1981144  -0.7814207  -0.5969238  -0.15398216  0.01063919
  -0.2429676   0.23077011 -0.03333473  0.0093441   0.08108139  0.24698639
   0.01898575 -0.07836914 -0.22543526 -0.4538746  -0.372849   -0.22449017
   0.11347389 -0.31233597 -0.00392151  0.01507568  0.09219742 -0.05952263
   0.18181801 -0.1162796  -0.27340698 -0.57170314 -0.05740738  0.25338745
  -0.6165173 ]
 [-0.07344627 -0.14644432 -0.26262617  0.29078007  0.31764603 -0.11345768
  -0.13419914  0.27160645 -0.12138557 -0.20481873 -0.41754436 -0.02208138
  -0.02295017  0.04717064  0.6010933   0.15384579  0.09380531 -0.05850983
  -0.07928467  0.07100677 -0.2266674   0.32710075 -0.43629456 -0.26506042
  -0.61392593 -0.36050034 -0.10982704  0.29151344 -0.00573635  0.05215359
   0.20567513 -0.10236263 -0.1294117  -0.18592834  0.19148636 -0.4285698
  -0.35606003 -0.42635727 -0.40074825 -0.8562091   0.1222223  -0.54065025
   0.03777206]]
[[ 0.08474541  0.5983267   0.28282785  0.5390072  -0.02745056 -0.5092354
  -0.01298714  0.3580246  -0.05202293  0.3734932  -0.22105217 -0.28075695
  -0.08852386  0.40566063  0.19125748 -0.10461617 -0.00177002  0.2553196
  -0.16112709  0.64497185 -0.20000076  1.          0.52895737  0.13253021
   0.2025299   0.31034565  0.17918968  0.34317398  0.04015303  0.07029343
   0.26241112 -0.34383297  0.3019619  -0.13567924  0.70212555  0.6071434
   0.         -0.1162796  -0.06367016  0.16109204  0.4962964  -0.6768291
   0.3866837 ]
 [ 0.6384182   0.933054    1.          0.16312027  0.12156868 -0.15039635
  -0.4285717   1.          0.14451027 -0.01204872  0.10877132  0.75394344
   0.75082016  0.8396225   1.          0.99999905  0.01946831  0.37234116
   0.1048584  -0.08283806  0.22000122  0.35513878  0.1583004   0.54216766
   0.2088604   0.41065884  0.73410225  1.          1.0000005  -0.0340147
   0.41134644 -0.02887154 -0.37254906  0.4472351   0.33333206  0.13095284
   0.431818    0.55038834  0.5056181  -0.383314    0.4319445   0.8692415
  -0.1504482 ]]
```

Also, since pytorch was to be used for implementing LSTMs, these normalized values were converted into tensors. Moving on, these normalized training datasets were split into batch size =1 and a specific sequence length (final parameter was 24).

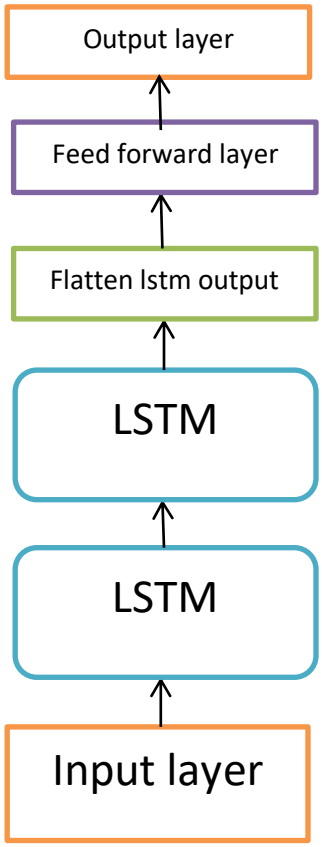
2. Implementation

The implementation for simple exponential method and holt linear exponential method were relatively simple. The model was trained on dataset by passing data into **predefined fit() method of respective algorithms from statsmodels** library. Then **forecast() method of respective algorithms form statsmodels library** was used to predict the values for test dataset. Finally to calculate rmse for predicted values against test values, math.sqrt() and numpy.mean_squared_error() functions were used.

Things were much complicated in case of LSTMs. The method of creating sequence requires a special mention here. The following create_sequence() function was used to create sequences.

```
seq_len = 24
def create_sequence(input_data, seq_len):
    sequence = []
    l = len(input_data)
    for i in range(l-seq_len):
        train_seq = input_data[i:i+seq_len]
        try:
            train_label = input_data[i+seq_len: i+seq_len+1]
        except IndexError:
            train_label = input_data[0]
        sequence.append((train_seq, train_label))
    return sequence
```

The sequence created was list of tuples – (train_seq, train_label). As seen in the code, train_seq contains values from index “i to i+seq_len” and train_labels contains just the next value in the dataset, i.e., the value corresponding to index” i+seq_len+1”. Hence this way we create and input and target values for a time series problem.
These preprocessed data were fed into model for training. The simple model architecture is as follows.



The detailed architecture of 2 layers LSTM network is as follows:

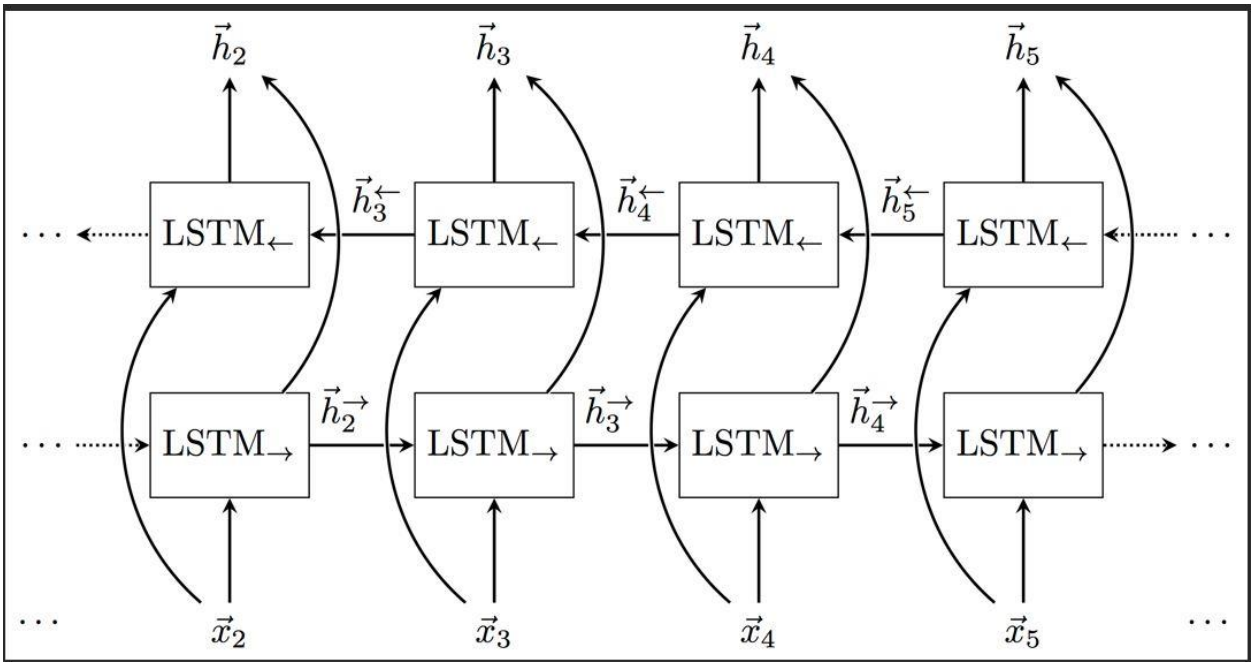


Image source: GitHub

The implementation steps are as follows:

- Preprocessing the data as described in data preprocessing and creating sequence of data as described in this section
- Passing the processed data into 100 hidden layer, 2 layers lstm network. Note that bias was set true for this layer.
- Flattening the lstm output to process it into feed forward layer network.
- Fetching the last value of output of feed forward layer as network output.
- Calculating training loss using torch.nn.MSELoss() function , calculating gradients using backpropagation and optimizing the weights values using torch.optim.Adam() optimizer.
- Monitoring losses for finding out best set of hyperparameters.
- After training model for best set of hyperparameters, feature values from year 2007 to 2016 was generated.
- RMSE was calculated for generated values against test values.

Given below is the table of predicted values from our trained model. Following it is the image of rmse values of lstm trained model.

year	annual-min	jan-min	feb-min	mar-min	apr-min	may-min	jun-min	jul-min	aug-min	...	jul-mean	aug-mean	sep-mean	oct-mean	nov-n
2007	20.069792	14.528858	16.779573	18.450363	20.998057	23.777682	23.811584	24.134981	23.613248	...	27.732755	27.218238	26.518523	25.405107	22.29
2008	20.033244	14.372534	16.627826	18.442538	20.956751	23.718674	23.703837	24.121476	23.607414	...	27.746987	27.227437	26.576965	25.364215	22.27
2009	20.017523	14.192190	16.463240	18.374727	20.982690	23.651766	23.676260	24.088888	23.594540	...	27.757179	27.232305	26.589230	25.318347	22.27
2010	19.995715	14.023429	16.282796	18.322766	21.038330	23.561846	23.636722	24.013812	23.559266	...	27.718846	27.211404	26.562330	25.277462	22.28
2011	19.984668	13.922332	16.148652	18.247531	21.096845	23.596612	23.596240	23.952731	23.532750	...	27.688691	27.194708	26.525331	25.291987	22.36
2012	19.991681	13.882951	16.101007	18.228155	21.196474	23.750411	23.580234	23.900292	23.520541	...	27.665890	27.194081	26.491531	25.325472	22.46
2013	20.015787	13.889683	16.140272	18.238729	21.235733	23.878120	23.608947	23.902726	23.537154	...	27.684007	27.212095	26.490918	25.363938	22.54
2014	20.035524	13.898924	16.169484	18.290799	21.249605	23.911103	23.651785	23.924012	23.551220	...	27.704788	27.227654	26.508406	25.360597	22.56
2015	20.035557	13.922519	16.198621	18.337182	21.219136	23.867208	23.667498	23.938635	23.548984	...	27.699225	27.228066	26.531557	25.331538	22.51
2016	20.025099	13.930638	16.199875	18.336217	21.176454	23.830808	23.664962	23.943393	23.539927	...	27.682413	27.221199	26.541184	25.305490	22.46

```
rmse_lstm
```

```
{'annual-min': 0.4914297464716202,
 'jan-min': 0.8923853821199306,
 'feb-min': 1.3039211737236787,
 'mar-min': 0.896117960036379,
 'apr-min': 1.0370411286949084,
 'may-min': 0.5937288697887277,
 'jun-min': 0.7227594779693627,
 'jul-min': 0.33253452112003457,
 'aug-min': 0.3858588190592551,
 'sep-min': 0.4246842380326448,
 'oct-min': 0.7367605141260767,
 'nov-min': 0.5622321400554741,
 'dec-min': 0.7831636479292476,
 'annual-max': 0.5815378157573945,
 'jan-max': 0.9415617184631693,
 'feb-max': 1.9135257745977032,
 'mar-max': 1.29893589643481,
 'apr-max': 1.1218977898963294,
 'may-max': 0.714828561249209,
 'jun-max': 0.8887489716619672,
 'jul-max': 0.2975629209756805,
 'aug-max': 0.5474151466929394,
 'sep-max': 0.5252646886797476,
 'oct-max': 0.7015421093062395,
 'nov-max': 1.007166023625734,
 'dec-max': 1.025935208925279,
 'annual-mean': 0.5271656553098051,
 'jan-mean': 0.8505625760473516,
 'feb-mean': 1.7986768820692742,
 'mar-mean': 1.2691444678621495,
 'apr-mean': 1.1178954044556408,
 'may-mean': 0.6627250987668207,
 'jun-mean': 0.816370133053054,
 'jul-mean': 0.2703483892775622,
 'aug-mean': 0.3988494547231046,
 'sep-mean': 0.4644907120807294,
 'oct-mean': 0.5655941452647089,
 'nov-mean': 0.5664272777680319,
 'dec-mean': 0.7430871616512513,
 'jun-rain': 71.02655610708626,
 'jul-rain': 74.0402090020612,
 'aug-rain': 70.98978716425856,
 'sept-rain': 34.72393841169928}
```

3. Refinement

Model was trained on lot of sets of parameters to find out best set of hyperparameters. With initial hyper-parameter values of hidden_dim=100, n_layers=2, lr=0.0001, sequence_length=1 and n_epochs = 100, the training loss was 0.16984613. keeping sequence length=1, every parameter was changed but there was no significant improvement in training loss as evident from image below.

Loss data with parameters ¶

loss = 0.16984613; parameters: hidden_dim=100, n_layers=2, lr=0.0001, sequence_length=1, n_epochs=100

loss = 0.17003609; parameters: hidden_dim=100, n_layers=1, lr=0.0001, sequence_length=1, n_epochs=100

loss = 0.17011546; parameters: hidden_dim=150, n_layers=2, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.16984379; parameters: hidden_dim=150, n_layers=1, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.16991393; parameters: hidden_dim=250, n_layers=2, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.16984379; parameters: hidden_dim=250, n_layers=1, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.16984379; parameters: hidden_dim=250, n_layers=1, lr=0.0001, sequence_length=1,n_epochs=200

loss = 0.16986375; parameters: hidden_dim=50, n_layers=2, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.17167109; parameters: hidden_dim=50, n_layers=1, lr=0.0001, sequence_length=1,n_epochs=100

loss = 0.14154065; parameters: hidden_dim=100, n_layers=2, lr=0.0001, sequence_length=3,n_epochs=100

loss = 0.03827548; parameters: hidden_dim=100, n_layers=2, lr=0.0001, sequence_length=12,n_epochs=300

loss = 0.03016614; parameters: hidden_dim=100, n_layers=2, lr=0.0001, sequence_length=24,n_epochs=300(final parameters)

Finally, sequence length was changed to first 3, then 12 and then to 24 and result was significantly better. The final training loss decreased to as low as 0.03.
Since learning rate was originally kept very low, it wasn't changed during the hyper-parameter tuning process.

IV. Result

1. Model evaluation and validation

During model development, values generated from model were tested against true test values and rmse was calculated for each features. The rmse values were found to be significantly better than our benchmark model.
The final architecture and hyper-parameters were chosen because they produced best results as seen in the figure in refinement section.
The Holt linear method is very simple and doesn't require any additional information apart from already provided in previous sections.

For architecture of LSTM network and LSTM layer, please ref figures in implementation part of previous section and for complete description, the following list:-

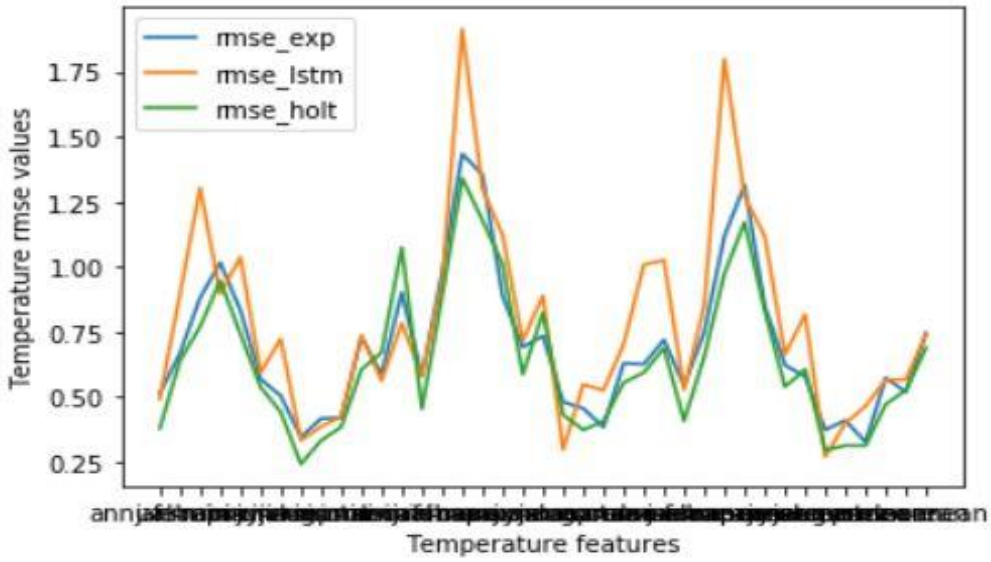
- The input layer had 43 features consisting of various temperature and rainfall features
- Input features are fed to LSTM network which has two layers along with hidden layer size of 100. Bias was set equal to its default value, True.
- The outputs of LSTM is a tuple consisting of hidden value and output for input features. Hidden value is passed onto next set of LSTM layers and output is sent to feed forward layer
- Before sending it to feed forward layer, LSTM output is flattened. The size of feed forward layer is 100 x 43. 100 represents hidden layer size and 43 represents number of features in our dataset.
- Final output is output of feed forward layer. Note: there are no activation function applied to output of feed forward layer as this is a time series regression problem whose output is expected to be any numerical value generated by trained model.
- Sequence length is kept at 24. It was observed that higher the value of sequence length, better is the result. But its value was restricted to 24 so that our network doesn't learn training dataset and doesn't overfit.
- Learning rate was kept low at 0.0001. Smaller learning rates are preferred as they tend to converge better.
- Number of training epochs was kept to 300. Generally higher the value, better is the training but it was limited to value of 300 to prevent overfitting.

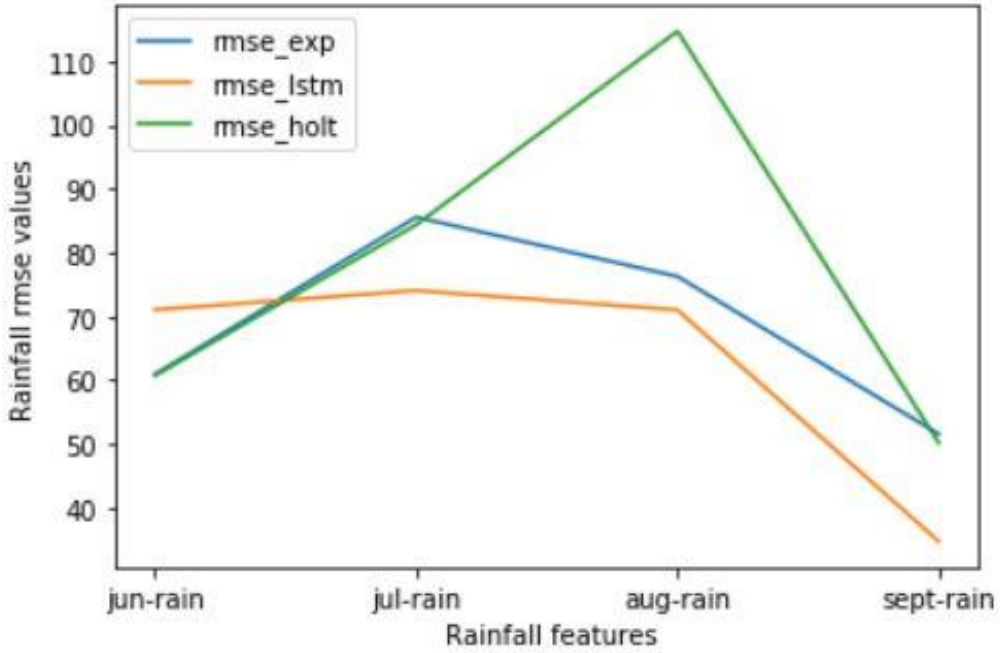
To verify the robustness of LSTM model, the trained model was tested on test dataset. Following are the observations:-

- The model was greatly able to learn the non-linearity present in the rainfall features and at the same time did not severely overfit on temperature features.
- RMSE for temperature features were in comparable range to benchmark model but rmse for rainfall features were greatly improved which can be easily seen in LSTM rmse values in figure implementation part of previous section.

2. Justification

The following figures compare the rmse values for all the three models. There are two plots, one for temperature and other for rainfall. Huge difference in rmse values of temperature and rainfall is genesis for two plots.





It can be seen that temperature rmse are nearly similar for all the three methods with two spikes in case of lstm network. The increase in rmse value for LSTM network at these two points can be attributed to overfitting. There were two variables in the training dataset with different kind of time series relationship. Temperature had more degree of linearity as opposed to rainfall which was evident from rmse values of simple exponential method and holt linear trend method. Hence, to accommodate for high non-linearity of rainfall features, complex LSTM networks were used which lead to slight overfitting for temperature features. From rainfall rmse plot, it can be clearly seen that LSMT network has outperformed the other two methods

by a huge margin. Holt linear trend method performs worse than even benchmark model. This signifies that there is high degree of non-linearity involved in rainfall features data. For more details, please visit improvement section.

V. Conclusion

1. Free-form visualization

As evident from rmse comparison figures in the last section, predictions on both temperature and rainfall features are quite close to original values.

may-min	jun-min	jul-min	aug-min	...	jul-mean	aug-mean	sep-mean	oct-mean	nov-mean	dec-mean	jun-rain	jul-rain	aug-rain	sept-rain
3.777682	23.811584	24.134981	23.613248	...	27.732755	27.218238	26.518523	25.405107	22.291269	19.892777	148.767050	409.503379	378.635200	209.399512
3.718674	23.703837	24.121476	23.607414	...	27.746987	27.227437	26.576965	25.364215	22.272593	19.880729	158.000355	393.225207	361.431707	192.791748
3.651766	23.676260	24.088888	23.594540	...	27.757179	27.232305	26.589230	25.318347	22.271761	19.828104	160.979979	369.632715	343.616167	194.310845
3.561846	23.636722	24.013812	23.559266	...	27.718846	27.211404	26.562330	25.277462	22.281788	19.749046	167.579252	347.424189	325.823927	198.239778
3.596612	23.596240	23.952731	23.532750	...	27.688691	27.194708	26.525331	25.291987	22.360414	19.721651	178.556065	334.978049	313.300484	198.244589

In the above figure first row corresponds to predicted value for year 2007 and last row for year 2011.

As seen from above figure, rainfall prediction for July month in year 2007 is pretty high. This heavy rainfall can transform into severe flood. Due to available quantification of total amount of precipitation, better disaster management could be done. Appropriate Quantification of precipitation can lead to improvement in mobilization of materials, in terms of quality and quantity, required to fight a natural disaster, be it flood or drought.

2. Reflection

The domain of weather prediction with deep learning is an area I always wanted to explore. But there were several challenges. The biggest challenge was to find an appropriate dataset. Generally, data was available but it was in bits and pieces and it was very difficult to find a complete set of data. The dataset used in this model is made by merging three datasets. Fortunately, the data I collected had not missing values and/or outliers which could have added more difficulties in the process.

The next difficulty was to find a bench mark model. I am new to the field of machine learning and haven't explored all of material available for our usage. Finding a benchmark model that is appropriate to the dataset and at the same time is naïve in its approach was a tedious task initially. But few look over internet and devoting some time in learning those models helped in overcoming this difficulty.

The interesting aspect of the project was to how to implement LSTM for this kind of dataset as this is my first project related to time series analysis. Normalizing the data was easy but the difficult part was creating the sequence. What would be the sequence length was a big

question. Although, I had worked on LSTMs before, but applying LSTMs on multivariate time series problem was first of a kind for me. How many features to keep in sequence length was a tough decision to take. How to keep some features and how ignore some features while creating a sequence length is still a problem to be resolved. Defining LSTM network and training model didn't take much effort but model evaluation required much effort, especially in creating initial test sequence for model evaluation.
Then another major problem was to invert the scaled tensor prediction back to simple tabular dataframe. Inverting requires proper handling of dimensions which took some effort.
The project handles variations in dataset quite good and is become a very a very handy tool in quantification measurement of total amount of precipitation over any area at a given time. This model can even be used by disaster management team for better preparation of amnesties and their mobilization.

3. Improvement

Although the model has shown considerable flexibility to accommodate maximum variability of the dataset, there are some areas, working on which can lead to improvement in existing model A better dataset. Dataset used had rainfall values only for the month June-September. A complete data of January –December could help in better training of model. Also a significant data is missing from dataset –"partial pressure of water" and "wind velocity". Adding more features to dataset can help in better understanding of complexities underlying natural process. Creating a better sequence of inputs to model. In current model, all of the features are used in training but if a sequence is created with selected features for training of different variables, then model could perform better.