Forecasting of Temperature with Various Neural Networks

Project Technical Report

By

Dhaval Delvadia

CSC 578, Section 710, Date: 2019-06-10

Kaggle Leaderboard: Dhaval Delvadia – Best Score Public: 0.53692 and Private: 0.54441

Video Link: https://youtu.be/LLGuBjxyhv0

INTRODUCTION:

Given a historical data of climate conditions, we were to predict the temperature of the next hour based on the climate conditions and temperature over the prior 24 hours. To achieve the goal, we explored various neural networks (i.e. RNN, LSTM, GRU, Conv1D, etc). This technical report provided the overview of various neural networks which were tested as well as one that gave lowest Mean Squared Error (MSE) values. This report concludes with my final thoughts and reflection on the project.

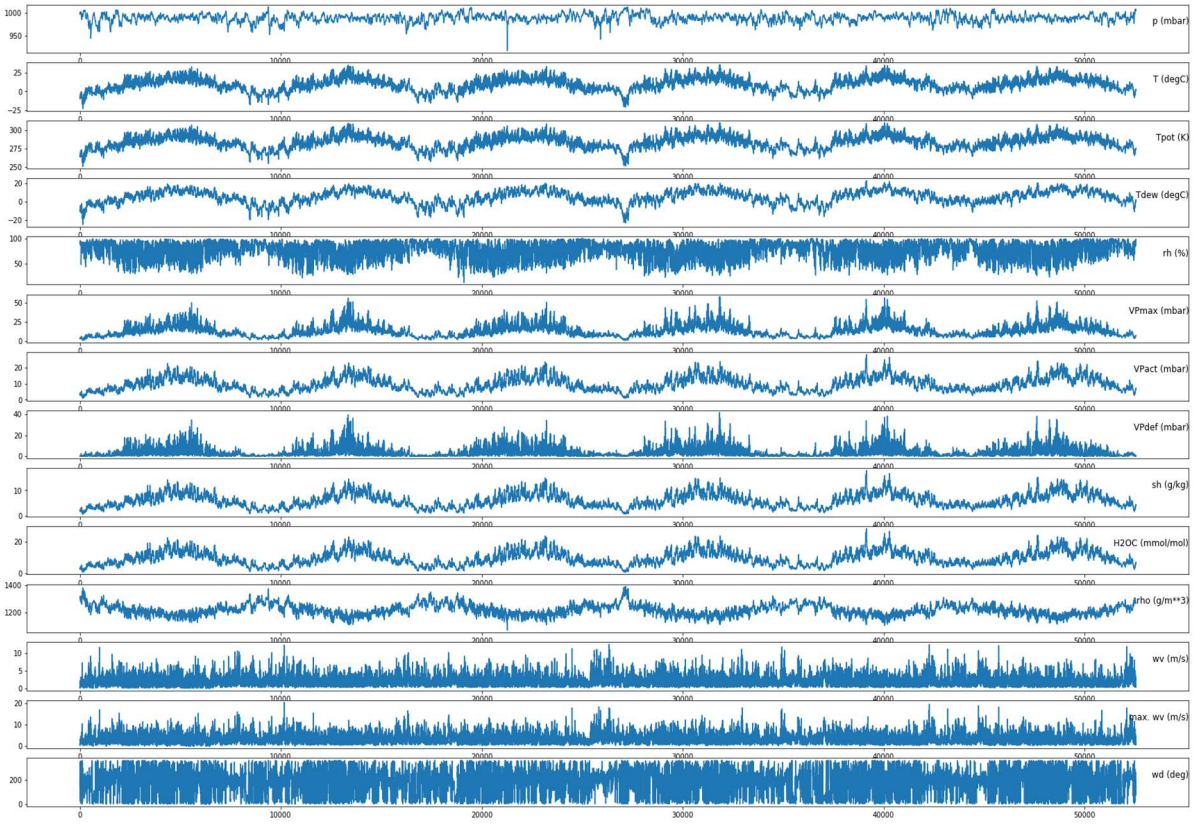
EXPLORATORY DATA ANALYSIS (EDA)

Before we model neural nets, we explored the dataset to get familiar with it. We were provided with two data files. One named “climate_hour_train.csv” (the training data) and “climate_hour_Xtest.csv” (the test data). The data in each of this dataset files consisted of the 14 different quantities (including air temperature, atmospheric pressure and humidity, etc.) recorded every 10 minutes. The original data went back to 2003, but we used a subset for this competition, from 2009 to 2016 (both inclusive). This subset was further reduced in which recordings were kept for every hour. The training set (but excluding the "Date time" column) to build a model and apply the model to the test set to generate predictions for the test instances. The training data file contained climate measurements from "01.01.2009 01:00:00" (Jan 1, 2009 1:00 am) to "31.12.2014 23:00:00" (Dec 31, 2014 11:00 pm). There were 52566 data rows plus one header row at the top, and 15 columns where the first column was "Date time" and the remaining 14 columns were climate measurements (of the date-time). The test data file contains climate measurements, to predict, from "02.01.2015 00:00:00" (Jan 2, 2015 0:00 am) to "01.01.2017 00:00:00" (Jan 1, 2017 0:00 am) and has 17447 data rows with NO HEADER ROW, and 336 columns where 14 climate measurements over 24 hours are concatenated (i.e., 336 = 14 * 24). There was no date-time column (nor header row) in this file.

The statistical summary of the training dataset was noted below:

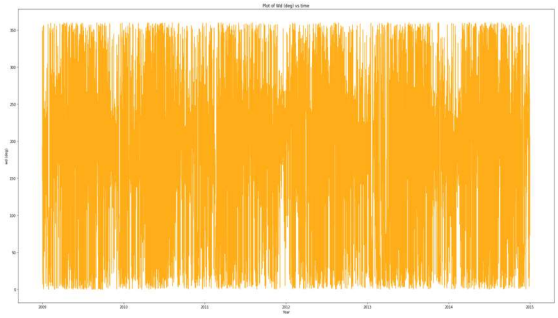
climate_hour_train.csv								
Total 52566 records of 14 variables noted in this table as columns								
	P (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh(%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)
Mean	988.72	9.17	283.25	4.78	76.44	13.36	9.46	3.89
Std	8.19	8.53	8.61	6.92	16.43	7.57	4.20	4.72
	sh (g/kg)	H2OC (mmol/mol)	rho(g/m**3)	wv(m/s)	max wv(m/s)	wd (deg)		
Mean	5.98	9.56	1216.7	2.14	3.54	173.7		
Std	2.66	4.25	40.4	1.53	2.31	87.25		

Next, we plotted each variable against time.

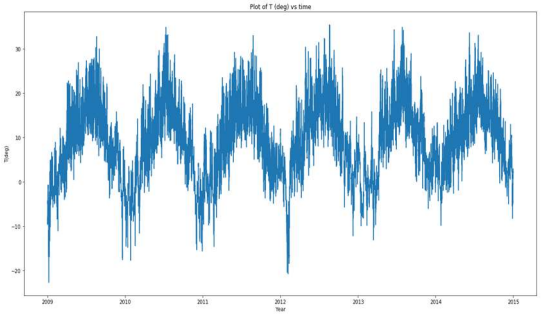


Based on the above graph, it appears most of the variables have seasonal pattern whereas others have no patterns. No pattern plots seem to be the last three plots in the above graph. It did not look like there was any seasonality in the last three variables. We picked two variables to see the seasonality little closer. One is Wd(deg) and another Temperature. Based on the seasonal nature of the target variable, Long-Short Time Memory (LSTM) neural network would be best.

Plot of the Wd(deg) vs date

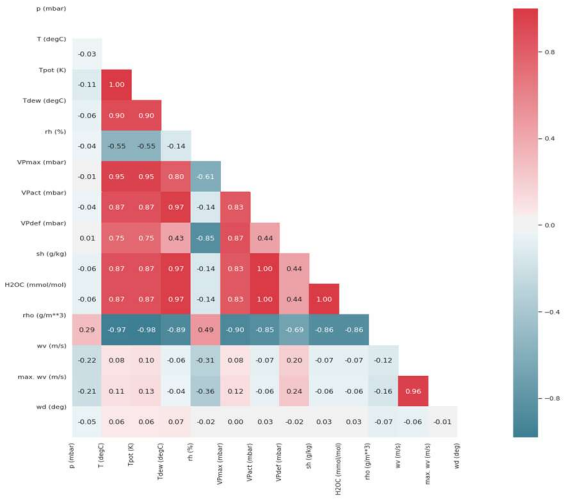


Temperature vs Date



It appears that there's a pattern in these variables.

Next, we reviewed the correlation plot with the correlation values for each variable:



MODELING:

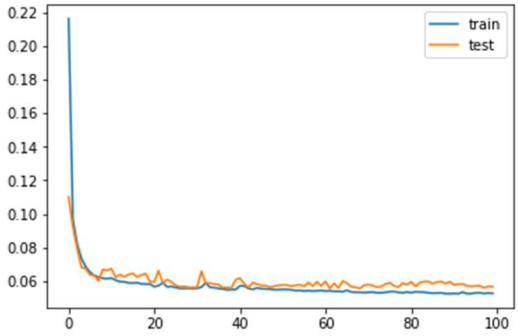
Based on the above correlation plot and the correlation matrix, we observed seven variables that were highly correlated with the temperature. These variables were Tpot(K), Tdew(degC), Vpmax(mbar), Vpact(mbar), Vpdef(mbar), sh(g/kg), and H2OC since their correlation values are higher than 0.7. Also, we noticed that other independent variables are highly correlated with each other. This created the problem of multicollinearity. Even though these values were highly correlated and had multicollinearity, we left these variables as-is as neural networks are not affected by the multicollinearity.

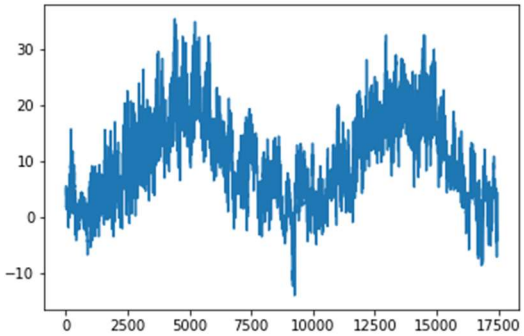
Also, the input was converted into z-score normalization since some of the data were various scale. This was accomplished by subtracting mean and dividing it by standard distribution. The inverse z-score normalization function was also written by utilizing the provided mean and standard deviation to rescale the prediction to actual scaled prediction.

Next, the table below describes which neural networks were tested. We also described how they were different it's training and testing score and MAE score on Kaggle.

Type of neural network	Training loss	Testing loss	epochs	Layers	Dropout Layer	Batch Size	Kaggle Score
LSTM	0.0527	0.0567	100	1	0	100	0.53692
LSTM with Dropout layers	0.0189	0.0088	25	4	4	32	0.7775
Bidirectional LSTM	0.0065	0.0067	25	1	0	32	0.59537
Bidirectional with dropout layers	0.0205	0.0085	15	4	4	32	0.74058
Bidirectional GRU	0.0068	0.0110	20	1	0	16	0.84165
Conv1D + GRU	0.1516	0.2402	20	3	0	16	2.32182
Conv1D + RNN	0.1428	0.2596	100	2	0	16	2.70749

As the table states, few different neural networks were tested. The simplest LSTM model performed really well. In the first attempt, the first LSTM Neural Network had batch size 32 and 50 epochs. The Kaggle MAE score was more than 1 so batch size and epochs were changed to 72 and 100, respectively. The Kaggle MAE score was recorded as 0.588. To explore further, the batch size and epochs were changed to 100 and 100, respectively and the Kaggle MAE score was recorded as 0.53692. The train and test loss and predicted value plot for the final run are shown below. We observed that train and test values follows each other and the predicted values has similar seasonal pattern as the plot explored earlier under EDA in this report.





Next, we tested the same LSTM model except it was added with stack of 4 layers with 3 layers dropout network was tested. After testing this model with batch size 32 and 25 epochs, Kaggle MAE score was recorded as 0.7775. If there was more time to train, this model also had great potential and would have performed well. Next, a simple single layer bidirectional neural network was tested. In this neural network the LSTM model learns the input sequences both forward and backwards and concatenate both interpretations. For this model, batch size and epochs were 32 and 25, respectively. In the single try the Kaggle MAE score was recorded as 0.59537. This network performed well. Again, if there was more time to train with fine tuning of parameter maybe it would have come down in MAE. It would have learn all the intricate details of data and would have reduced the loss. Next, bidirectional network with dropout layers and stacking were tested. There were 4 layers of bidirectional and dropout layers in this network. The Kaggle score for this network was recorded as 0.74058. Next Bidirectional with Gated Recurrent Unit (GRU) neural network was tested. This network only had one layer and employed batch size of 16 and epoch size of 20. The Kaggle score for this model was 0.84165. The last two neural networks tested were Convolution 1D with GRU and Convolution 1D with RNN. Both networks were set to batch size of 16 and 20 epochs and they both performed poorly in predicting correct temperature. The Kaggle score for both were 2.32 and 2.71, respectively.

Conclusions

In conclusion, clearly the basic LSTM neural network recorded the lowest MAE after adjusting batch sizes few times. It seems there was a sweet spot. If too high or too low batch size, the MAE errors are high. Also, due to the limited runtime and timeouts by Kaggle, it was difficult to train different deep neural network models for many epochs with various batch sizes. If the Kaggle limitation was not there or I had machine with good GPU, the MAE could have come down even further. Also, if I had more time, I would have study neural nets further so that I could understand which tuning parameters are important to adjust in specific neural network and I would have also experimented with totally new neural networks and fine tune them. Overall, I liked this Kaggle competition. It was fun.