Sebastian Zdarowski
Final Project CSC 578
Neural Networks Final Project
June 8, 2019

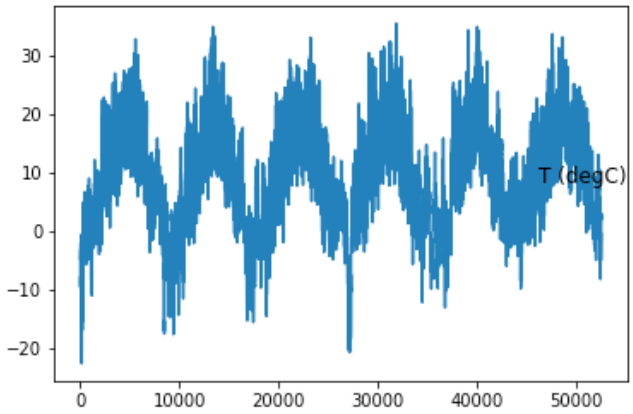
Kaggle username: Sebastian
Private Leaderboard: # 8 score .52544
Public Leaderboard # 6 score .52248
Video Link: https://drive.google.com/open?id=1nqZUQWcjFSpQ4gCvLOQS5tPp_T0bgpMm

Final Project: Kaggle Competition

Introduction: For this final project we were tasked with building a Recurrent Network to predict temperature at the next given hour given the weather conditions within the dataset. We will be using keras, numpy, pandas, sklearn and LSTM to do the final project.

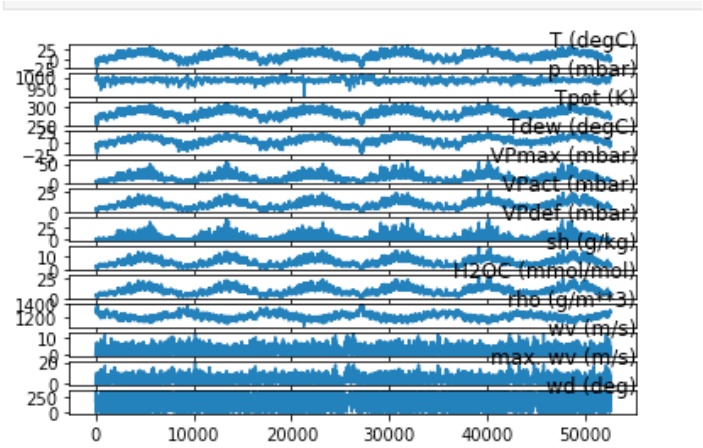
Exploratory Data Analysis: Before we start any processing, like any good data scientist, we need to explore the data that we are working with. The dataset that we started out with was labeled climate hour train, which contained 52,566 data rows and we had an additional test set that we will be using to make our predictions labeled climate_hour_Xtest. The training set held time measurements ranging from Jan 1st 2019 through Dec 31st 2014, our test set contained dates from Jan 2nd 2015 through Jan 1st 2017. The first thing that we checked was what data types we were working with, and it seems that they were all float 64 data types. Now we need to check the descriptive stats of our variables we are working with (See Data Table at Appendix) from our data table we are able to see that our average (mean) temperature (T (Deg C)) is 9.17 with a Std deviation of 8.53. Some other variables that I think are important in predict the weather would be Tdew(deg C) which averaged 4.79, Tpot(K) I feel like was another important measurement in prediction since it was deg c but in kelvin units and that averaged out to be 283.25 which I assume is the average conversion for T deg C. But lets focus on our prediction variable, the highest point for temp c was 35.48 degrees which occurred on July 19th 2012, and the lowest temp point was -22.76 degrees which occurred on Jan 7th 2009.

Temp C Time series plot



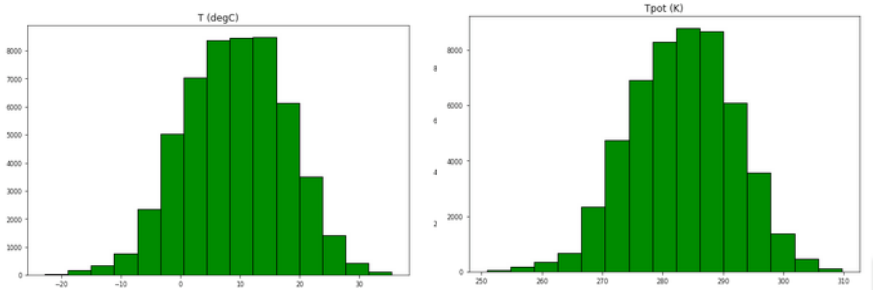
From the Temp C time series plot we are seeing seasonal patterns with the temperature, High periods on this plot most likely are correlated with summer months, while we see that the lowest points are most likely associated with the winter months. Let's see what we have for the other variables as well regarding the time series plots.

Rest of Variables plotted out in time series format

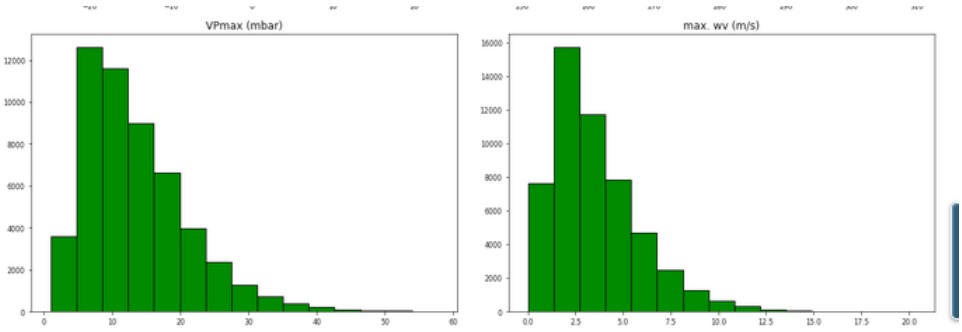


From the chart, we are seeing seasonal trends across all variables, which indicates a good sign since we are detecting weather patterns. The data definitely seems to be following a high and low wave pattern across the time periods.

Lets see the distribution of the data regarding some of the important features for predicting weather patterns.



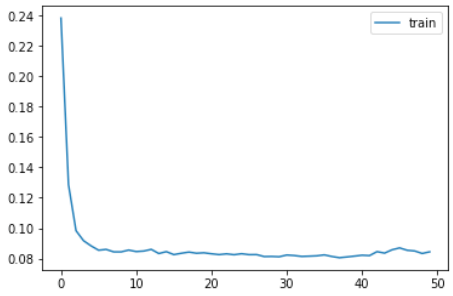
We clearly see the same distribution across T(deg C) and Tpot(K) since they are the same temperature, but are just in different units. Some of the variables are show a skew in the data trend. A few of the variables that are showing this are the following: rh(%), vpdef, max vs wv, wv(m/s) and sh(g/kg)



Preprocessing: After looking at the data, we should begin our pre processing for the dataset, the goal is to set up a matrix for xtrain, ytrain, and use test data xtest and make sure that its keras ready for our neural network. First we needed to drop the date column as its not needed for our model. We will keep the data in the same rows, no shuffling. The first step that needed to be achieved was doing a normalization on the dataset, per instructions we were suggested to do z score normalization on the both training and test data set. So using Sklearn standard scaler, using the formula z = ((x - u) / s) where u is the mean of the samples, s is the standard deviation of the samples and x being the row transformed. We are able to apply normalization to the datasets. Now that we have normalized our dataset we need to setup a function to get our dataset to be keras

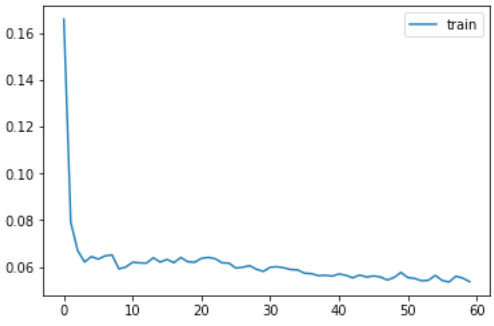
ready, within the notebook the function “setup” was able to transform our columns into a timestep window (timestep window 24) and our features = 14
The shape for the training files should be (Rows, timestep, and data dimensions) So for Xtrain we are striving to hit (52,542 rows, 24 hour columns, 14 features) and for ytrain a shape of (52,542, 1)

Model Building: Now that our dataset is ready for keras we can begin the fun part of building out our neural network to predict the temperature for the next hours. The first model that I built was originally based on 300 nodes for the first layer, the second layer would be an additional 64 nodes, then applying a 25% dropout rating, adding in a dense layer and using a linear activation layer. The model complied the loss based on mean absolute error. In total the model had 5 layers and a total of 471,505 parameters. At first I started out with a batch size of 100, to test out the model and it used 100 epochs. The model wasn’t running as fast as I would of like in the beginning, and the training loss started out at .30 which wasn’t optimal for the task at hand. I tried different batch sizes with this model, 100, 150 and 240. Epochs ranged from 50 to 100.

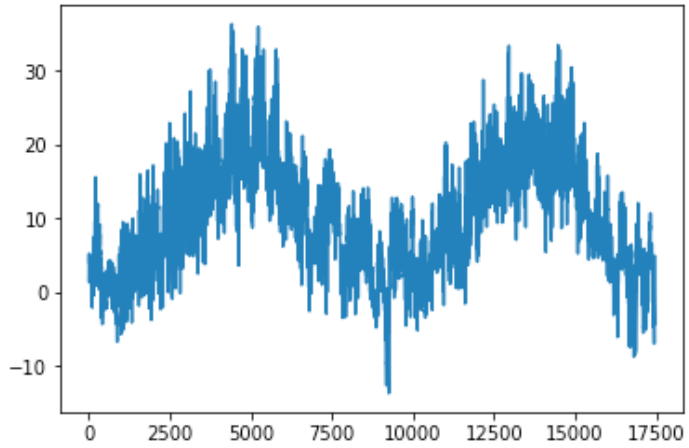


As we see with the chart to the right, the training started off at .24 learned pretty quickly and after 50 epochs the loss values floated around .08. When submitted to Kaggle, these models scored between .90 and 5.1. Which are not the greatest scores.

The second model that was built, had the best performance for me. Building out a simple LTSM model seems to work best with this dataset. I used the original model as a template, but I took away 3 layers from the first model. Ending up with just my first LTSM layer that contained 230 nodes and adding a dense layer. Total Params were 225,631 within this model. Far from simple considering all the nodes that are in work. Still it was less complicated and it seemed to give the best performance. This is the model that I experimented heavily with. I tried at first by making it as simple as possible just have 30 nodes and 50 epochs, that didn’t go so well as the model didn’t really learn. I bumped up the nodes to 240, 300, and settled on 230. I noticed that after running 100 epochs the training set started to jump around with the loss score. So I made the decision to only run 60 epochs as I noticed that values started to converge around that run. Mini batch sizes ranged from 50 to 240, 240 seemed the best, as the model took a while to learn and run when it ran 50 small batches compared to 240.



This chart demonstrates the models learning rate, we see that it started on .16 instead of .24 (first model), and it was able to achieve a loss rate around .056. After submitting my prediction file to Kaggle, I was able to obtain a score of .52248 on the public leaderboard before the competition closed.



In the end our model was able to predict the weather for future dates using our test dataset. This is the output of the prediction file that was created and submitted to Kaggle. Within this chart we are able to see the seasonal patterns being reflected within the chart.

Reflection: Overall the was a pretty challenging Kaggle competition, the pre processing of the dataset was where I ran into the most trouble originally with this project. Setting up the dataset to be keras ready was a challenge and required a lot of python numpy coding to get the file ready to be delivered to the model and a lot of time, I spent most of my time trying to get the data ready. I also ran into a lot of issues when it came to scaling/normalizing the dataset. Using Sklearn to do a zscore normalization wasn't a problem for me, but when it came back to scaling the data using the mean and std the professor provided us that was the challenged. I originally tried using sklearns inv transform function based on the std dev and mean and my prediction file was giving me values around 11 degrees consistently which was giving me very high error scores. Once I wrote a simple numpy code to go into the prediction file and apply the multiplication of the std dev and then adding back in the mean, I was able to obtain better scores. Building the LTSM model was also a challenge, I had to read through a few keras documentations to get the model up and running. The links that the professor provided were helpful guides in setting up this problem. If would do this project again, I would love to see what are the most important features within this dataset, so that we can get a better idea of what really causes the temperature to be predicted accurately and try to avoid the "Black box" of neural networks.

Kaggle leaderboard score:

Sebastian	0.52248	8	3h

Appendix:
Data Table

Variables	count	mean	std	min	25%	50%	75%	max
p (mbar)	52566	988.723002	8.190684	918.5	983.75	989.14	994.07	1012.74
T (degC)	52566	9.172795	8.533081	-22.76	3.11	9.31	15.28	35.48
Tpot (K)	52566	283.254265	8.605048	250.85	277.2425	283.43	289.37	309.69
Tdew (degC)	52566	4.779049	6.922701	-24.8	0.13	5.2	10.03	22.94
rh (%)	52566	76.4443	16.430164	13.06	65.81	79.7	89.8	100
VPmax (mbar)	52566	13.357483	7.572008	0.97	7.64	11.74	17.39	57.8
VPact (mbar)	52566	9.458133	4.201679	0.81	6.17	8.85	12.32	28.04

VPdef (mbar)	52566	3.899249	4.723265	0	0.81	2.09	5.13	41.71
sh (g/kg)	52566	5.977212	2.666892	0.51	3.89	5.595	7.78	17.94
H2OC (mmol/mol)	52566	9.568031	4.253017	0.81	6.24	8.965	12.45	28.53
rho (g/m3)**	52566	1216.71899	40.439912	1066.19	1188.0825	1213.44	1243.05	1392.56
wv (m/s)	52566	2.14217	1.530832	0	1.01	1.79	2.88	12.58
max. wv (m/s)	52566	3.539017	2.313246	0	1.8	3	4.75	20.33
wd (deg)	52566	173.689628	87.251111	0	120.8	197.1	233.8	360