IE7860: Intelligent Engineering Systems

Assignment 5 : Temporal Processing and RNN

Building a RNN with LSTM and Forecasting

Overview

The problem was to forecast/predict the future trend of energy consumptions. Approaching the problem with an available time series dataset. The dataset has 27 features along with a date variable and one output variable. The size of the data was about 19300 and we divided the data into training and testing where the training size was 17500. Each of the features represent a sensor value recorded continuously.

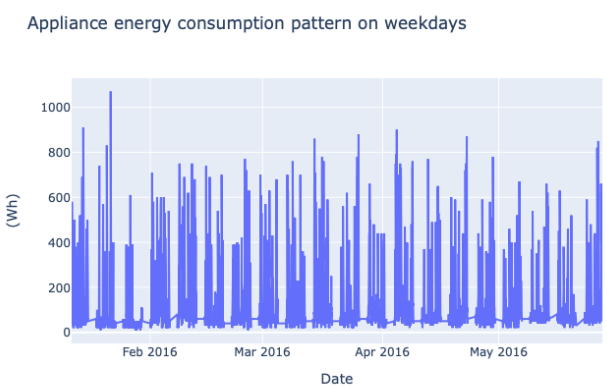
	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	7.000000	63.000000	5.3	13.275433	13.275433
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	6.666667	59.166667	5.2	18.606195	18.606195
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	...	17.000000	45.50	6.366667	733.7	92.0	6.333333	55.333333	5.1	28.642668	28.642668
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	...	17.000000	45.40	6.250000	733.8	92.0	6.000000	51.500000	5.0	45.410389	45.410389
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	5.666667	47.666667	4.9	10.084097	10.084097

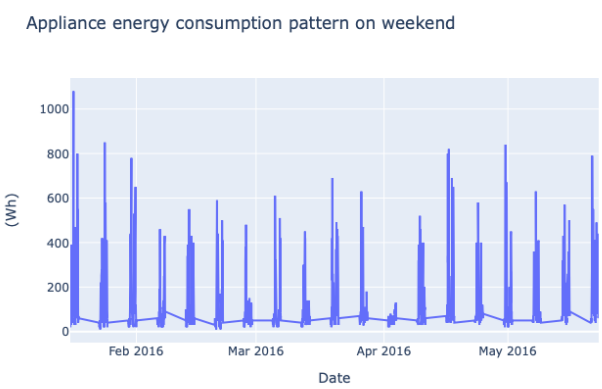
To solve such a problem we implement a RNN using LSTM. The model has been built in python and tensoflow along with callback as tensorboard to further explore the neural network and optimise it.

Data Preprocessing

Then dataset is clean but require certain level of pre-processing. We Start by loading the data from the text file into dataframes. After which we shape the data based on the input and output parameters. This is done so as to shape the data for the model to process.

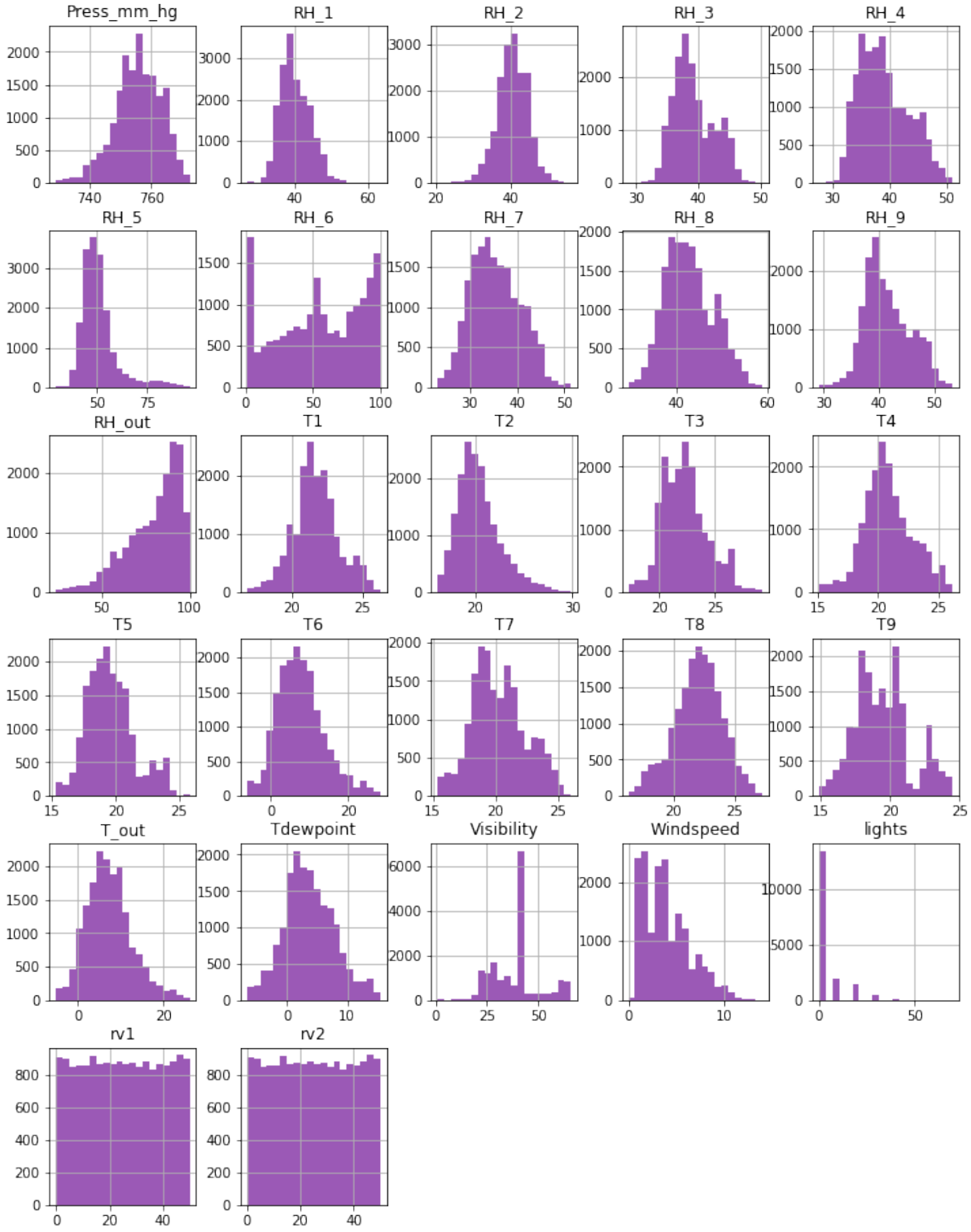
The validation set is used during the model fitting to evaluate the loss and metrics. However, the model is not fit with this data. The test set is completely unused during the training phase and is only used at the end to evaluate how well the model generalizes to new data. This is especially important with imbalanced datasets where overfitting is a significant concern from the lack of training data.



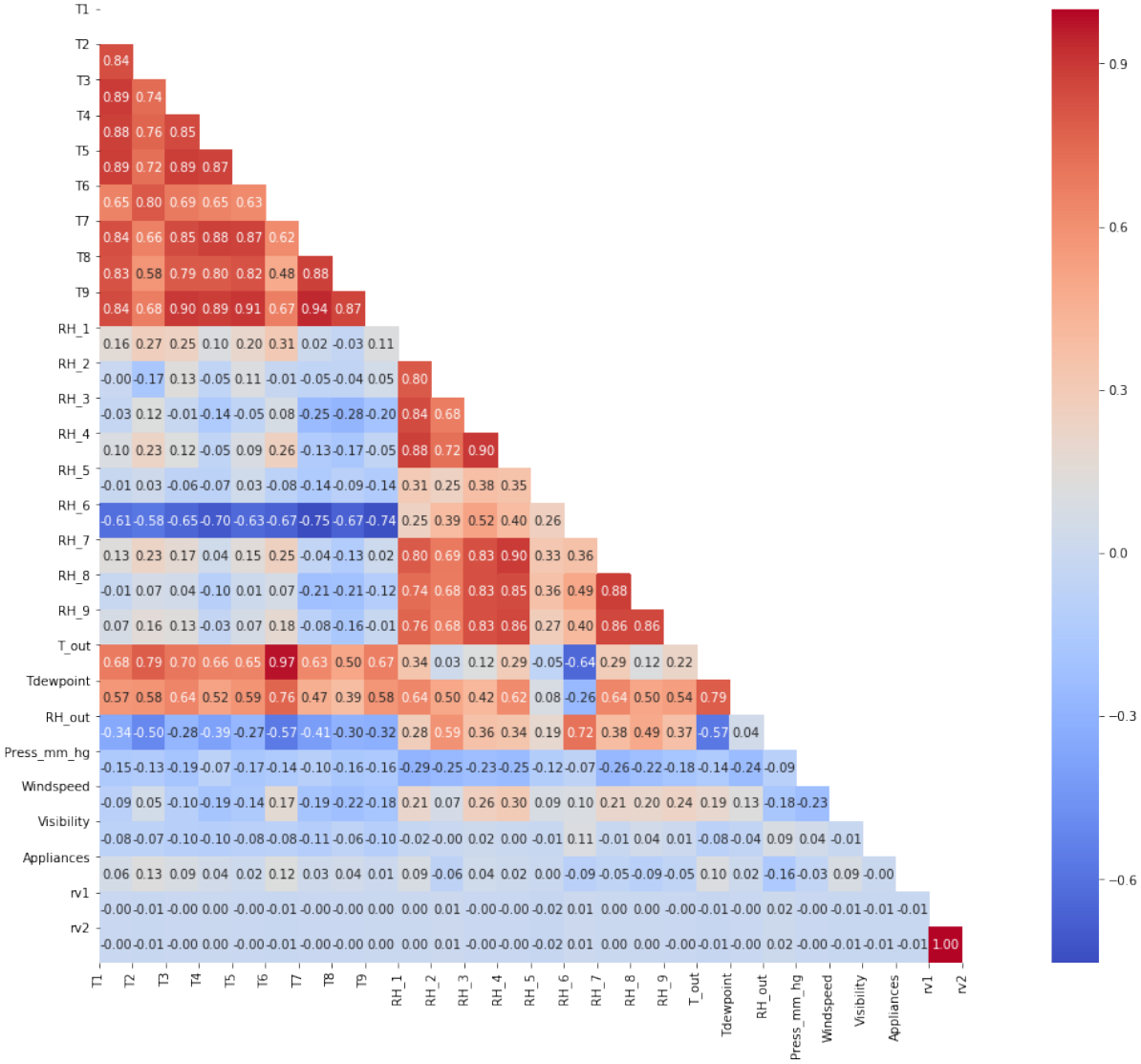


Exploratory Data Analysis

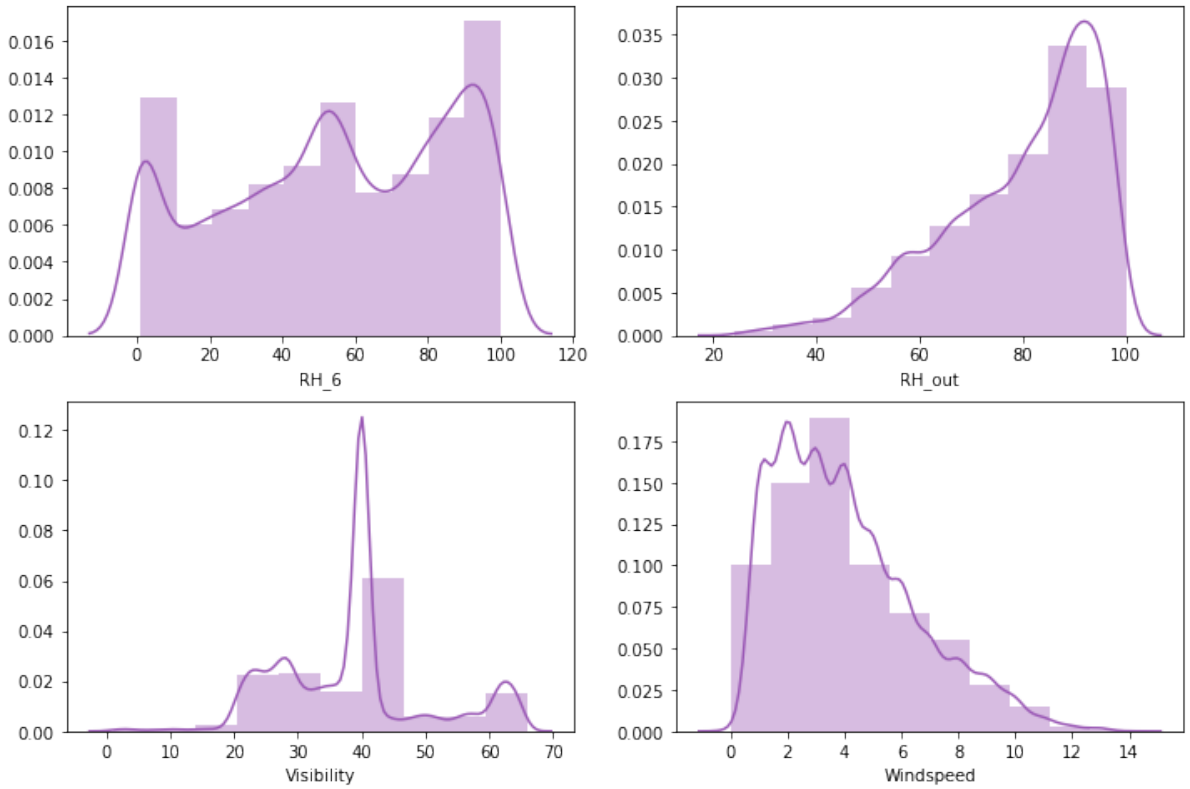
We will begin by exploring the data a bit and highlight the basic features of the data including the bias and distribution. From the distribution plots below we can observe that the data is very consistent and uniform. Features such as rv1 and rv2 which are random variables do not add any significant information since they are there to prevent over training. Most of the features show similar patterns, which is a good sign to train a model.



Next step is to examine the partial correlation between the features. In the plot below we can see that there is high correlation between the T values and the RH values but the rest of the features are not at all corelated since the values are less than 0. One of the anomalies that we can observe clearly is the RH6 feature. Here it is inversely corelated with all the T values, indicating a relation between them. Secondly the features T_out and T7 are also highly corelated which ay be due tp the fact that the sensors where relatively in the same environment. Next we see that the random variables are following the exact same pattern since the score is 1. Which ass mentioned below will not contribute towards the optimisation of the model.



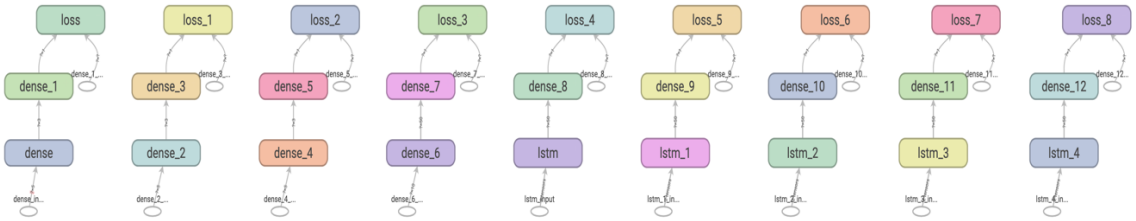
Coming back to the distribution of RH6, RHout, Visibility and Windspeed. The correlation anomalies were since the distributions are irregular and not consistent with the data. Hence we will be removing all of the features are an issue and might mislead the model.



Creating the Neural Network and Optimising

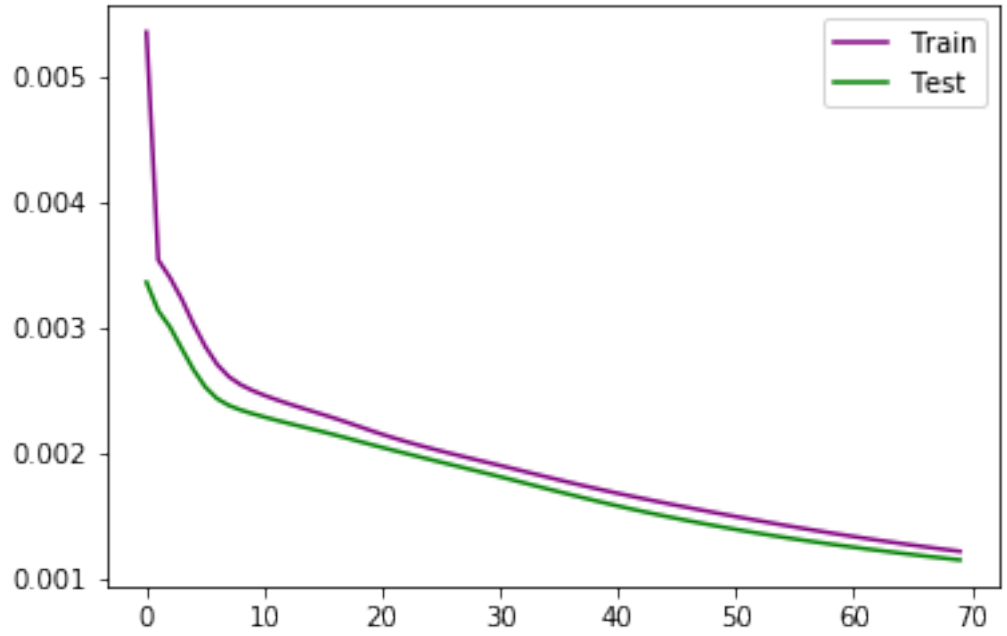
The model definition is just a few lines of code but has all the parameters required. The backend and all the complications are handled by Keras and Tensorflow. The algorithm uses randomness in order to find a good enough set of weights for the specific mapping function from inputs to outputs in your data that is being learned. It means that your specific network on your specific training data will fit a different network with a different model skill each time the training algorithm is run.

Another preparation is to convert the data into a time series format with features, timesteps and outputs. Also it has to be reframed to match the shape for the LSTM layer. The first layer that is created is a LSTM layer with 50 nodes. The second layer is a Dense layer with 1 node, which is the output layer. The model architecture is simple and only the variation to nodes were sufficient enough to get a good result. The model is then compiled with loss function as mse and optimizer as adam optimizer.



I used TensorBard to analyze the model and its activation, initialization bias and consistency. Which was also declared as a callback and then passed in the fit function. Fitting the model over 70 epochs and batch size of 10. 70 epochs was determined based on the early stopping callback. Since two methods of call back was not possible I hardcoded the 70 epochs to use TensorBoard. After which I got the training loss as low as 0.0012 and testing loss as 0.0011.

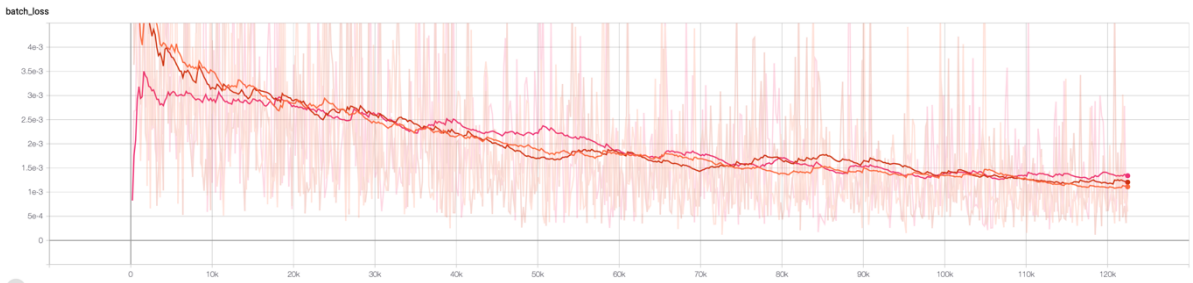
```
loss: 0.0012 - val_loss: 0.0011
```

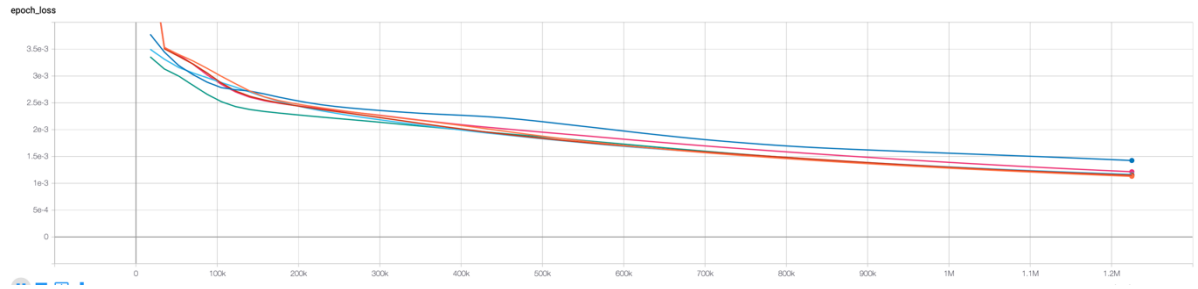


As we can observe from the above plot the major part of the training was complete within 10 epochs and the rest was just reinforcing the knowledge.I then computed the R2 score for both training and testing. Where we can clearly see a good result.

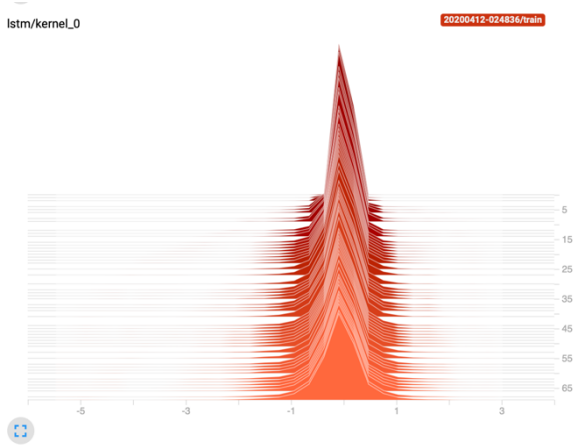
```
The R2 score on the Train set is:     0.963
The R2 score on the Test set is:      0.964
```

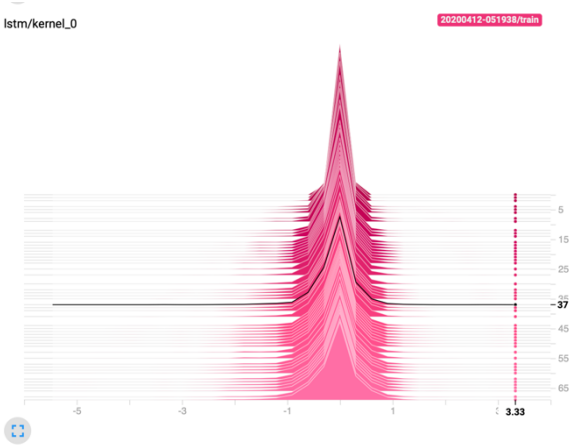
The above result was achieved with the help from TensorBoard. Instead of simply running GridSearch for hours and with very varying result, I analyzed the following biases from the lstm layer until I reached a consistent activation of the layer over the 70 epochs. The results given below are for the final optimised model. The firrt two plots highlight the smoothend batch loss and epoch loss for multiple runs.



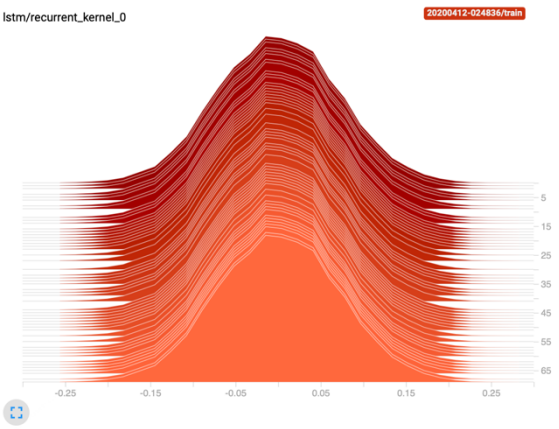


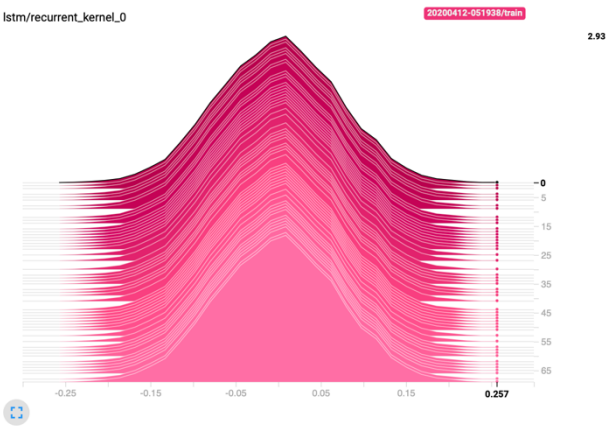
The 3D Histograms below highlight the activation and initialization of the biases and kernals. The first two show the initialization of the kernel of the first layer. Which as mentioned before should be consistent over the third dimension. It represents the robustness of the model over the given data.



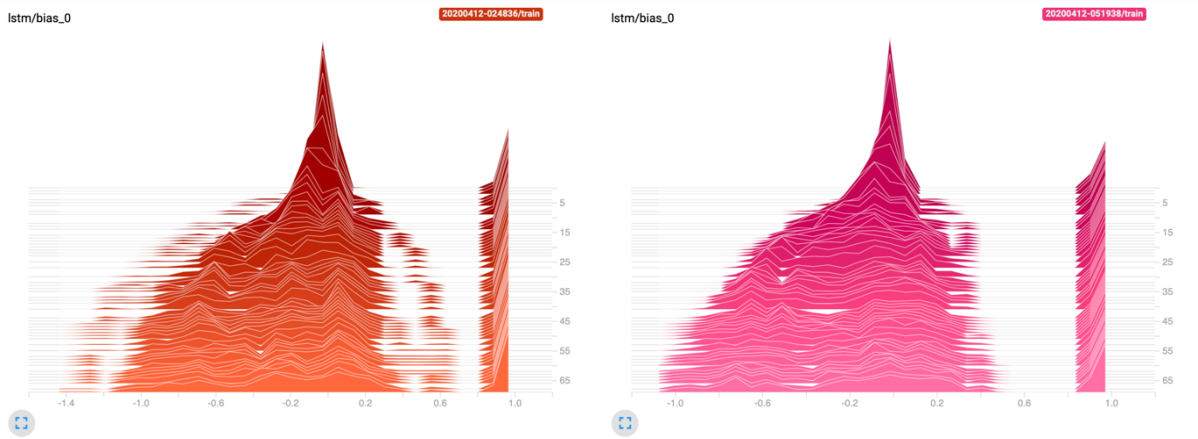


The plots below are the initialization of the recurrent kernel of the first LSTM layer. Which also is very consistent and further validates the initial claim. Here we observe that the initialization is also more distributed. Which signifies that the model was able to isolate the patterns.



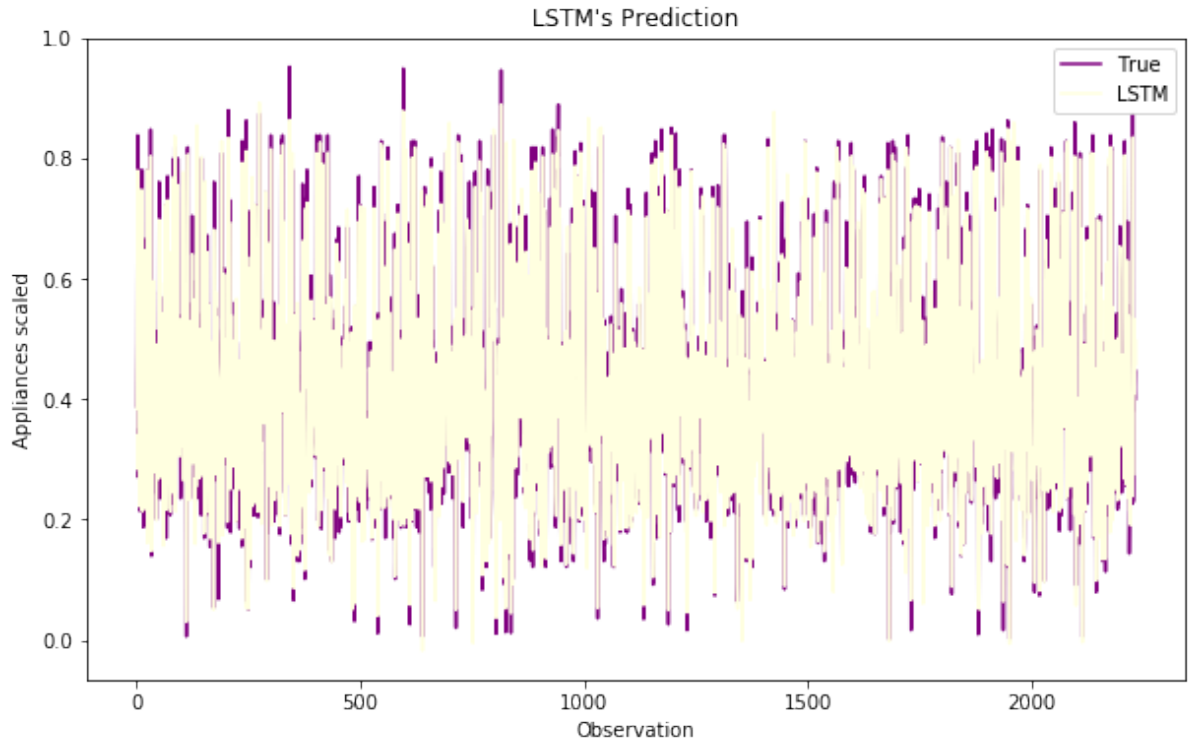


In the below plots we see the activation of the bias of the LSTM layer. The converging over the epochs is a highlight. Its signifies that the model is predicting values more accurately and hence the layer doesn't have to stay active for long.



Evaluating the Model

Starting with looking at the predictions made, we see high overlap of the predictions over the true value. Hence the model is able to predict accurately over the given test data.



After validating the model based on the test data predictions, we move on to looking at the monthly, and daily distribution trend. To further forecast the time lagged trend. The daily trend doesn't have a very significant pattern, but the monthly pattern shows overall decreasing slope.

The following plot highlights the monthly trend in various time steps, and clearly there is trend in seasonal view. This gives us the proper time lag to be used. Which is seasonality. Here the model will be the most efficient.

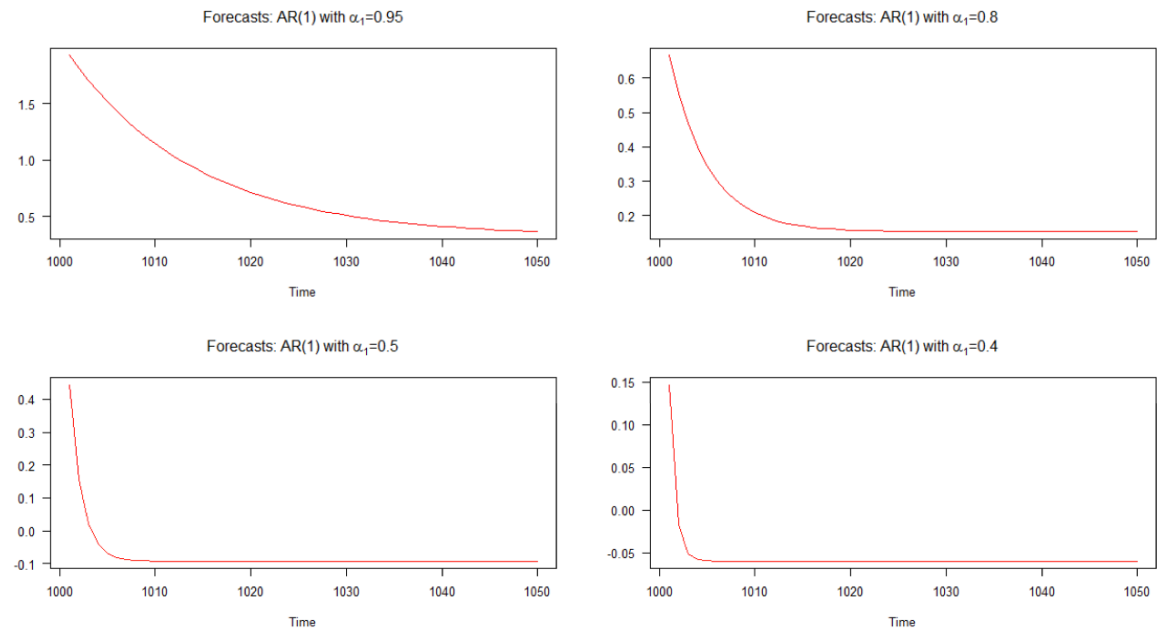
Forecasting

We now begin the forecasting using a TLNN approach using ARIMA. Since the data is stationary it is the same as Focused TLNN. Since we chose LSTM to create the Neural Network. Which by nature is distributed. To get the best order and seasonal order parameters, we iterate through combinations and compute the AIC value. Based on which we automatically chose the best parameters as order=(1, 1, 1) and seasonal order=(1, 1, 1, 12).

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.1213	0.112	1.088	0.277	-0.097	0.340
ma.L1	-1.0000	669.795	-0.001	0.999	-1313.775	1311.775
ar.S.L12	-0.1071	0.137	-0.783	0.434	-0.375	0.161
ma.S.L12	-1.0000	669.786	-0.001	0.999	-1313.756	1311.756
sigma2	921.5928	0.667	1380.778	0.000	920.285	922.901

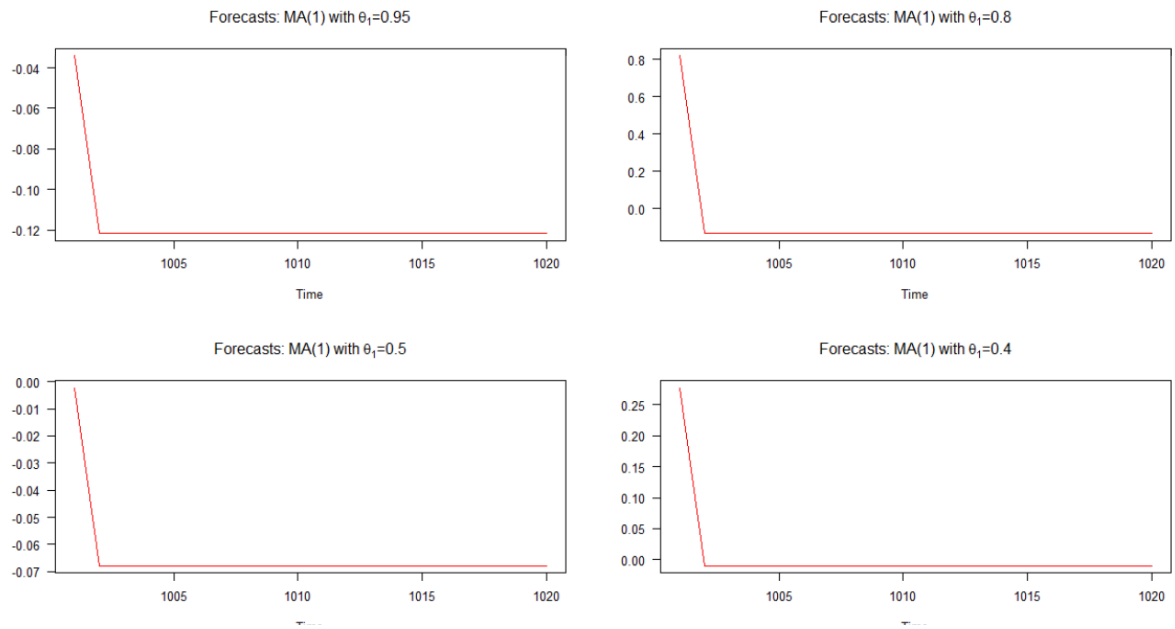
Let's consider an AR(1) model for a moment. In this model, we can say that the lower the value of $\alpha 1$ then the quicker is the rate of convergence (to the mean). We can try to understand this aspect of AR(1) models by investigating the nature of the forecasts for a small set of simulated AR(1) models with different values for $\alpha 1$.

In the graph below, I have plotted out-of-sample forecasts for these four AR(1) models. It can be seen that the forecasts for the AR(1) model with $\alpha 1$=0.95α1=0.95 converges at a slower rate with respect to the other models. The forecasts for the AR(1) model with $\alpha 1$=0.4 α1=0.4 converges at a quicker rate than the others.

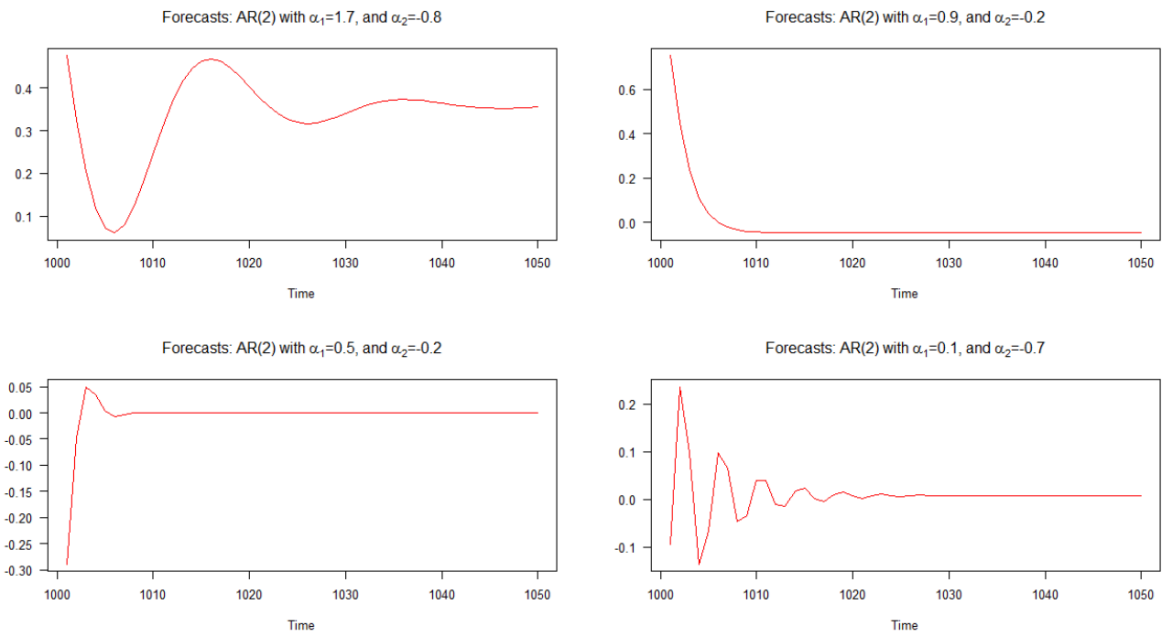


Now let's consider four MA(1) models with different values for $\theta 1$. In the graph below, I have plotted out-of-sample forecasts for these four different MA(1) models. As the graph shows, the behaviour of the forecasts in all four cases are markedly similar; quick (linear) convergence to the

mean. Notice that there is less variety in the dynamics of these forecasts compared to those of the AR(1) models.



Things get a lot more interesting when we start to consider more complex ARIMA models. Take for example AR(2) models. These are just a small step up from the AR(1) model, right? Well, one might like to think that, but the dynamics of AR(2) models are quite rich in variety. The out-of-sample forecasts associated with each of these models is shown in the graph below. It is quite clear that they each differ significantly and they are also quite a varied bunch in comparison to the forecasts that we've seen above - except for model 2's forecasts (top right plot) which behave similar to those for an AR(1) model. When the red line is horizontal, it has reached the mean of the simulated series.



Finally looking at the forecasting, we can see that it is only forecasting the general trend and not the anomalies. From a practical perspective we, electricity consumption trend is only significant to predict the load over time and when the energy department has to be ready. The one step ahead forecast encloses even the anomalies but the general trend dosen't. The later plot shows an extended forecast where there is no validation data available. Such forecasting can provide great insights and help the energy department or any stakeholder.