**Prediction of Hourly Energy Consumption using RNN and LSTM**

Namita Pradhan

SID: 862099741

**Introduction:**

In recent years, advances in sensor technologies and expansion of smart meters have resulted in massive growth of energy data sets. These big data have created new opportunities for energy prediction, but at the same time, they impose new challenges for traditional technologies. That is the reason I have attempted to solve this problem of prediction using deep learning, like recurrent neural networks.

Prediction of energy consumption is important for the following reasons:

1. Saving energy just in case it is depleted in the future
2. Estimation of the quantity of fuel required to produce energy
3. Planning for the future

A Recurrent Neural Network deals with sequence problems because their connections form a directed cycle. They can retain state from one iteration to the next by using their own output as input for the next step. The hourly energy consumption data has a sequential pattern, because a particular hour’s data depends on its previous hour’s data. Thus, RNNs are used for this prediction.

**Datasets:**

I have used three different datasets to test the models. Those datasets are; The Dayton Power and Light Company, American Electric Power (AEP) and PJM Load. This hourly power consumption data is taken from Kaggle.

The datasets have the datetime in the first column and the second column has the energy consumption in megawatts (MW) range. I have split the data into training and testing data and used only the training examples for training the model. This division is given in the subsections for each dataset below.

**Data Pre-processing:**

The sklearn package MinMaxScaler is used for normalization of the data before using it. I converted the data into the range -1 to 1. Then, the data is divided into training and testing sets where maximum possible data is taken for training. The data is properly shaped into a matrix form.

**RNN:**

Recurrent Neural Networks are artificial neural networks where connections between nodes form a directed graph along a temporal sequence. Each hidden layer is provided with an input and the output from the previous layer. RNNs can use their internal state (memory) to process sequences of inputs. This makes them suitable for time series data like hourly energy consumption.

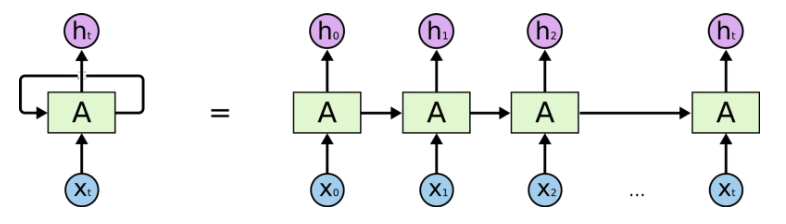
****

Figure 1: RNN architecture

**Vanishing gradient problem:**

During the gradient descent using back-propagation, to update weights, the gradients become so small that the weights are no more updated. This is the vanishing gradient problem. In RNNs, the gradient of the loss function decays exponentially with time (called the vanishing gradient problem). This hinders them from using long term information. They are good for storing memory 3-4 instances of past iterations but larger number of instances don't provide good results.

**LSTM:**

LSTM networks are a type of RNN that uses special units in addition to standard units. LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it is output, and when it is forgotten. This architecture lets them learn longer-term dependencies. Thus, LSTMs overcome the vanishing gradient problem. A single LSTM unit is as shown below:

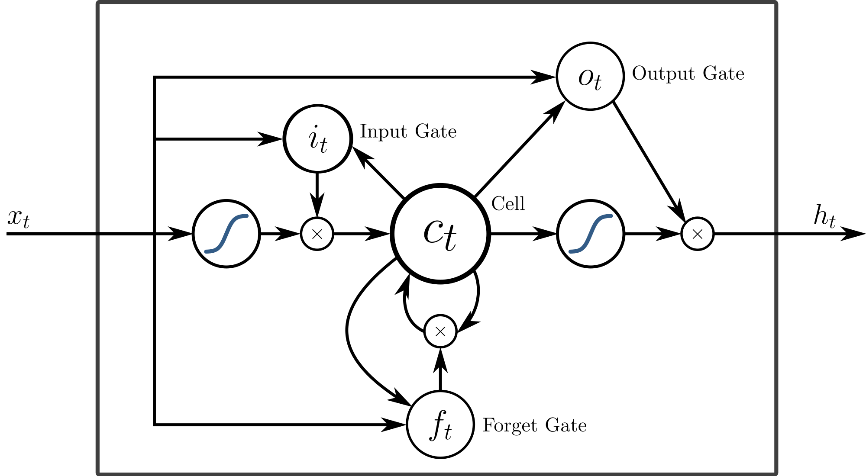


Figure 2: LSTM unit

**Results:**

1. **PJM\_Load\_hourly**

Before normalization:

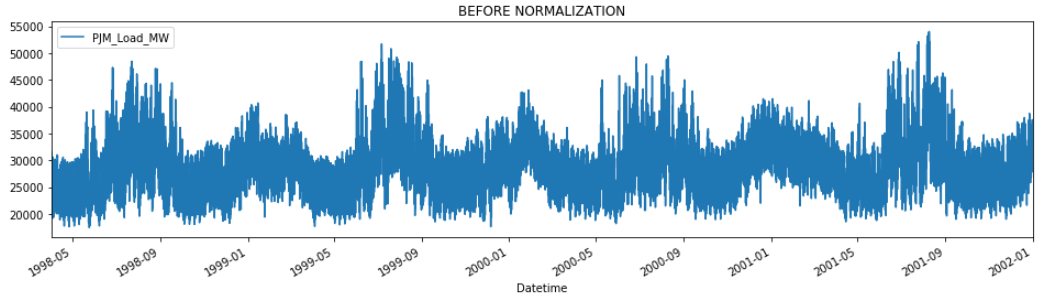


Figure 3: PJM load before normalization

After normalization:

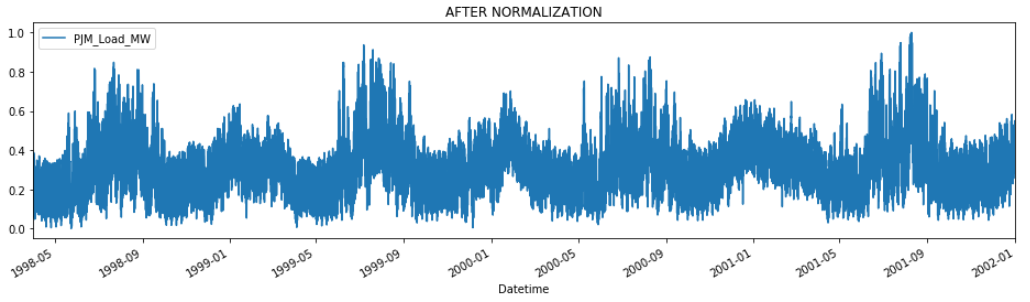


Figure 4: PJM load after normalization

The division of data is:

Training examples: 30000 and Testing examples: 2876

Activation = **tanh**, Optimizer = Adam, Loss = MSE and Epochs=**10** with batch size=1000

**RNN model:**

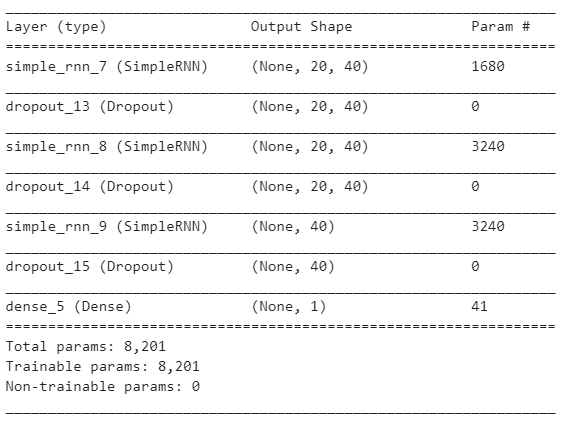


Figure 5: Summary of RNN model

R2 Score of RNN model = 0.9076334510655906

After changing the number of epochs to **30**, R2 Score of RNN model = 0.9503870719182248

**This shows that by increasing number of epochs, the performance of the RNN model increases.**

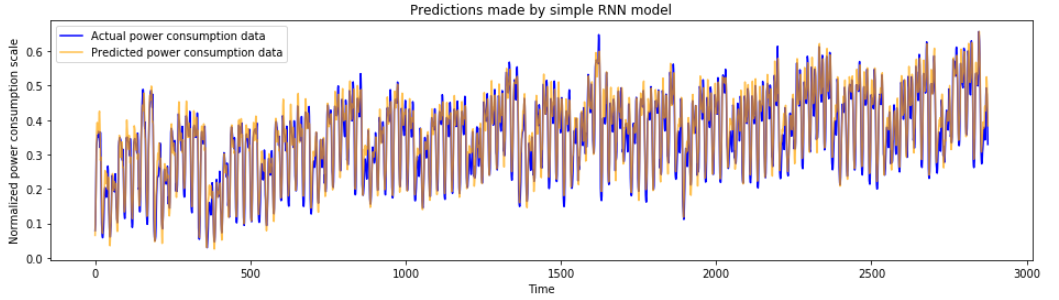


Figure 6: Predictions made by RNN model

With activation = **relu** and epochs = 10, R2 Score of RNN model = 0.9441115701013573

But, activation = **relu** and epochs = 30, R2 Score of RNN model = 0.9081806742125689

**LSTM model:**

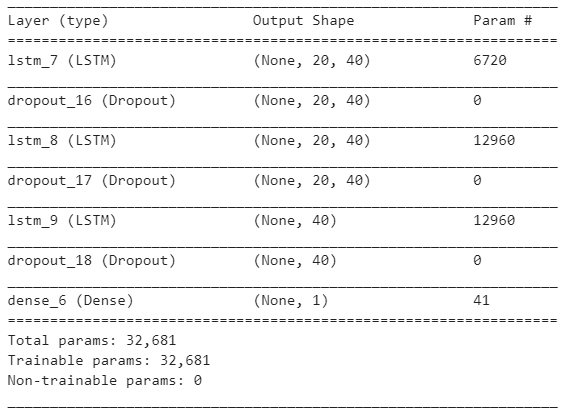


Figure 7: Summary of LSTM model

R2 Score of LSTM model = 0.6462276849673716

After changing the number of epochs to 30, R2 Score of LSTM model = 0.9505655910721739

**This shows that by increasing number of epochs, the performance of the LSTM model increases.**

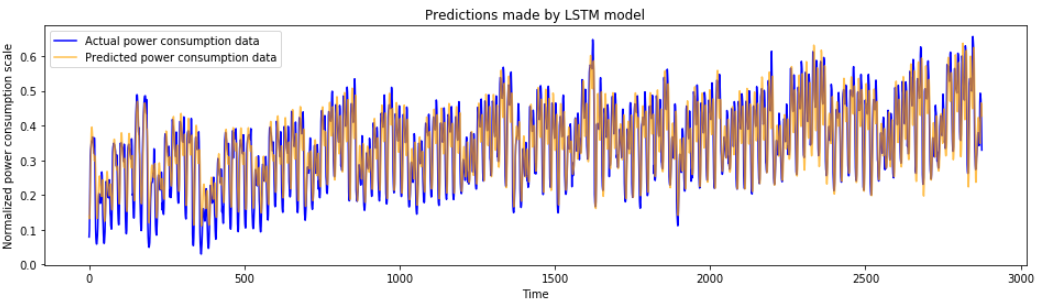


Figure 8: Predictions made by LSTM model

With activation = relu and epochs = 10, R2 Score of LSTM model = 0.4954403129415924

But, activation = relu and epochs = 30, R2 Score of LSTM model = 0.9275921346088842

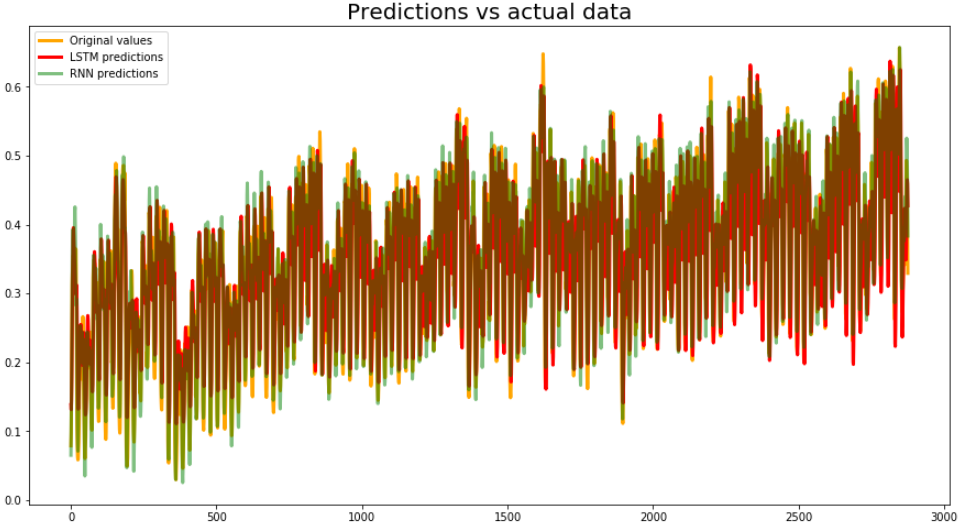


Figure 9: Predictions from both models vs actual data

1. **DAYTON\_hourly**

Before normalization:

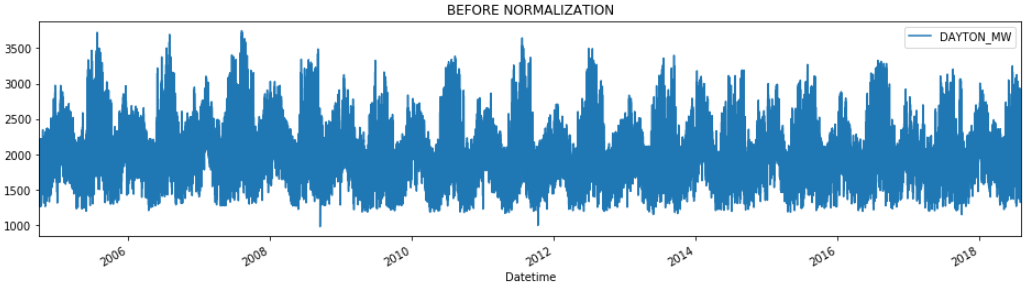


Figure 10: DAYTON before normalization

After normalization:

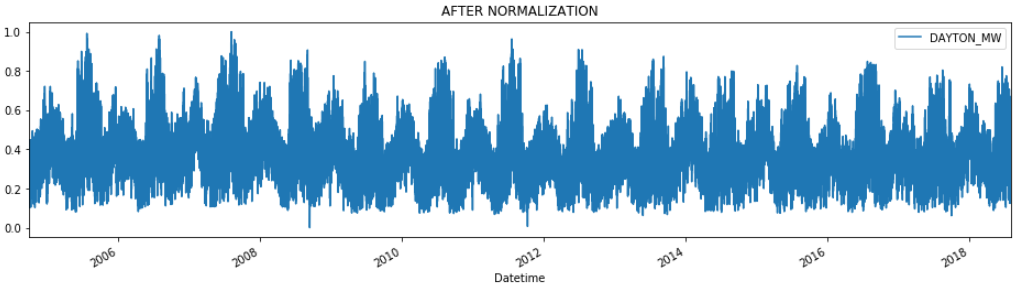


Figure 11: DAYTON after normalization

The division of data is:

Training examples: 120000 and Testing examples: 1275

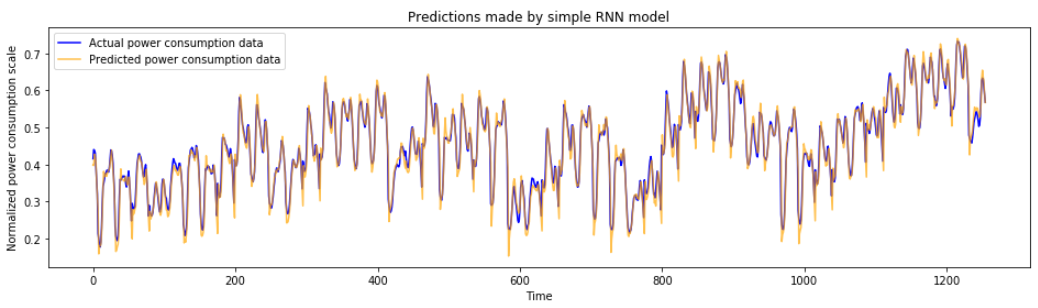
Activation = tanh, Optimizer = **RMSProp**, Loss = MSE and Epochs=30 with batch size=1000

**RNN model:**

R2 Score of RNN model = 0.9466349804119162

With activation = **relu,** R2 Score of RNN model = 0.8835349597538597

**It can be confirmed that, when we change the activation from tanh to relu, the performance decreases.**

Figure 12: RNN predictions

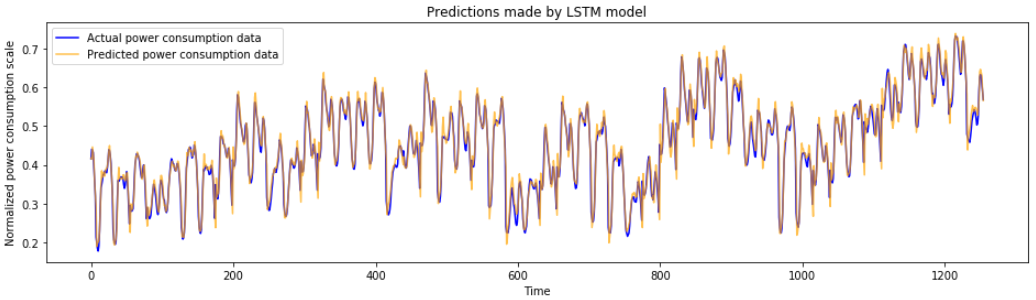
**LSTM model:**

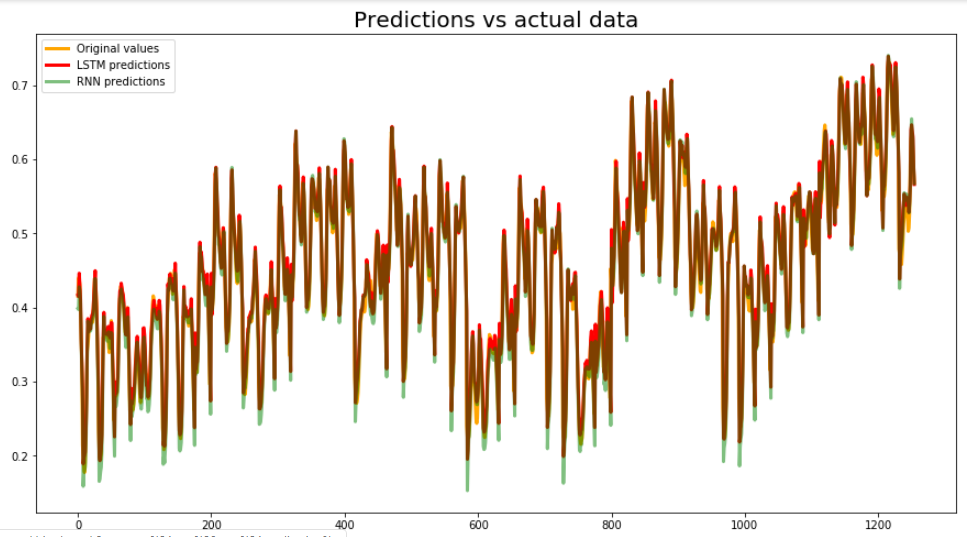
R2 Score of LSTM model = 0.9544953814243844

With activation = relu and optimizer = **RMSProp**, R2 Score of LSTM model = 0.9078728150720207

With activation = relu and optimizer = **Adam**, R2 Score of LSTM model = 0.8528993171555665

**Thus, it can be inferred that when the number of epochs is less, Adam works better than RMSProp and when it is increased, RMSProp works better.**

Figure 13: LSTM predictions

Figure 14: Predictions from both models vs actual data

1. **AEP\_hourly**

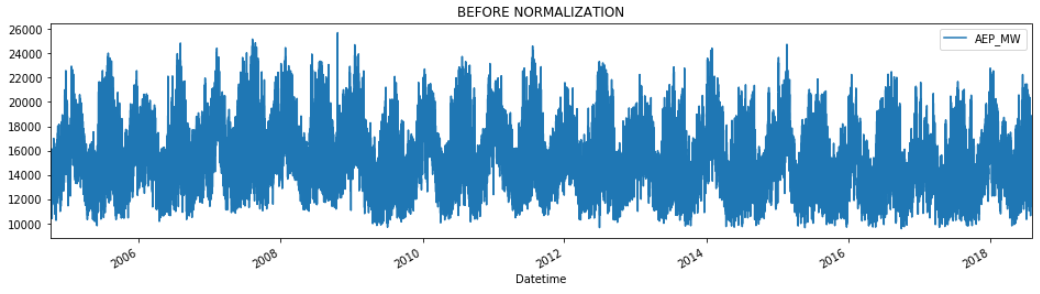


Figure 15: AEP before normalization

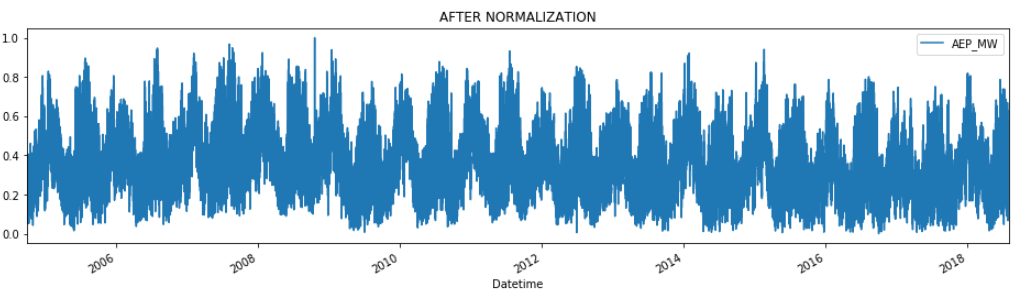


Figure 16: AEP after normalization

The division of data is:

Training examples: 120000 and Testing examples: 1253

Activation = tanh, Optimizer = RMSProp, Loss = MSE and Epochs=30 with batch size=1000

**RNN model:**

R2 Score of RNN model = 0.9495450247441322

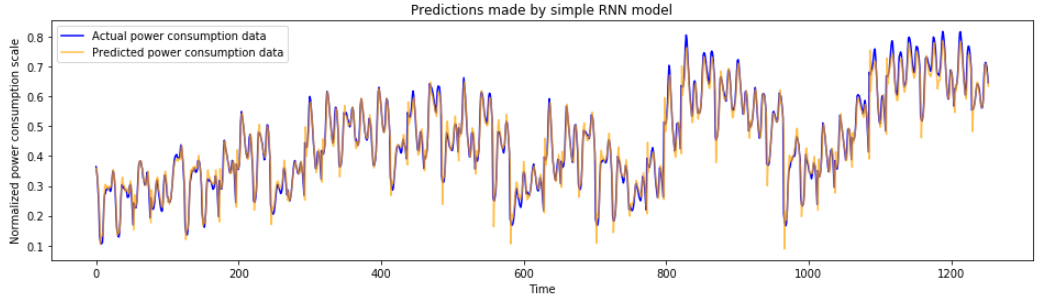


Figure 17: RNN predictions

**LSTM model:**

R2 Score of LSTM model = 0.9500549360337367

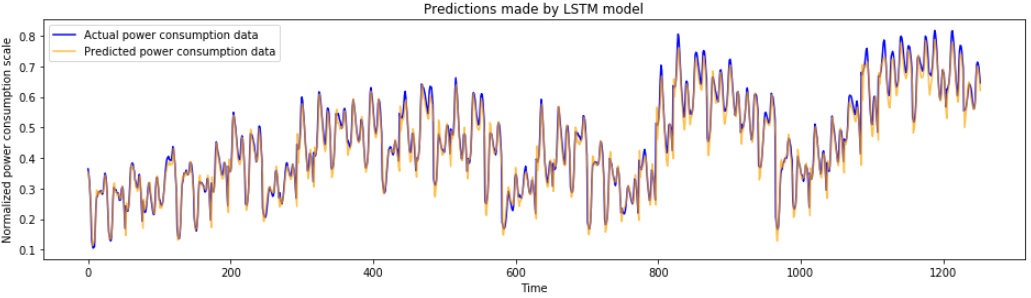


Figure 18: LSTM predictions

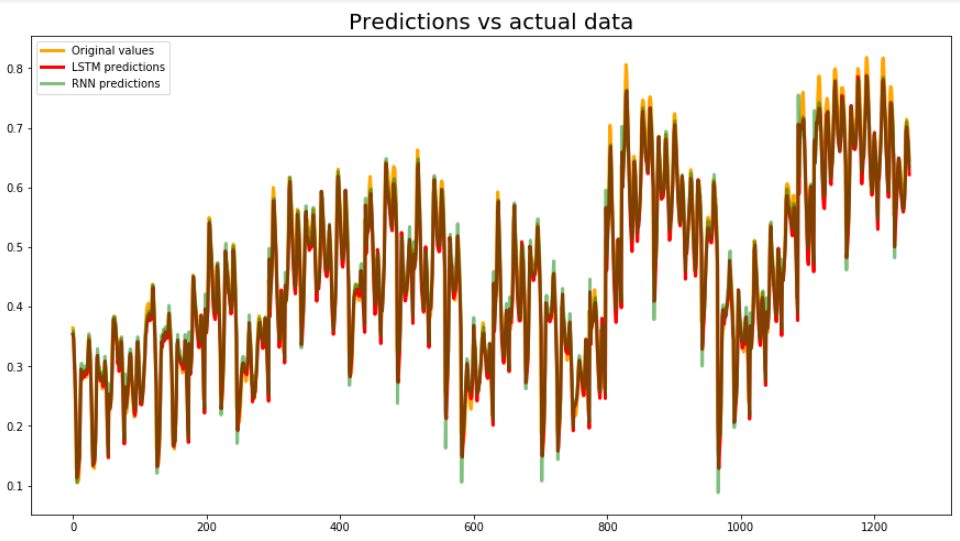


Figure 19: Predictions from both models vs actual data

**Evaluation of results:**

R2 score is used to evaluate the results obtained. The best R2 score is 1 and it can be negative as well for worst models. The R2 score is calculated as follows:



which is the total sum of squares and,



which is the residual sum of squares, where yi are the true data labels, y bar is the average of all the data labels and fi is the prediction from the model.

Therefore,



**Conclusions:**

* As the number of epochs are increased, the performance becomes better.
* Adam optimizer gives a good result during the beginning of the training, but as the number of epochs are increased, RMSProp gives a better result.
* Tanh activation function works best for time series data.
* LSTMs successfully eliminate the vanishing gradient problem.

**References:**

1. <https://www.kaggle.com/thebrownviking20/everything-you-can-do-with-a-time-series>
2. <https://www.kaggle.com/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru>
3. <https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f>