IST 718 – Final Project  
Lucas Daniel Zarzeczny  
Ian Morris

David Doman

**Predicting Total Gross Ads for Telecommunications**

**Summary**

The Telecommunications industry’s success is largely due to promotional campaigns and/or marketing campaigns that are released during the year. These campaigns drive large spikes in gross ads which provides more revenue for the firm. Gross ads are new lines that are added for the firm. The new lines then become a part of the company’s base the next month. It is difficult for firms to predict which campaigns to release and when to release them if historical data is not analyzed and future trends are not predicted out. The problem is the absence of any prediction model that can predict out daily gross ads and revenue. Firms can predict “generally” for financial reporting purposes and earnings releases. However, these tend to be a one-step prediction out one year. The firms do not have daily predictions which are more useful for targeted promotional campaigns. The objective is to build a deep learning model that can predict out the daily trends for Gross Ads which will then provide more insight into the occurrence of large spikes which are primarily due to marketing campaigns. The marketing team can then use this data to create promotions for the following year

**Data and Preprocessing**

The data was obtained from multiple ODBC data sources from T-Mobile. The data sources were primarily views within Teradata and SQL Server. Promotional data from one view was merged with Gross Ad data from another table. Also, monthly data from XLCubed was converted to daily data and also merged with the Gross Ads data.

The final data set was multiplied by a random factor to protect the confidentiality of the firm. Since the dataframe was multiplied by a factor, the trend/seasonality/cyclicality stayed intact.

There were negative values within the dataset which does not make sense. This was due to accounting adjustments. Thus, the data was manipulated to replace anything less than 0 with 0.

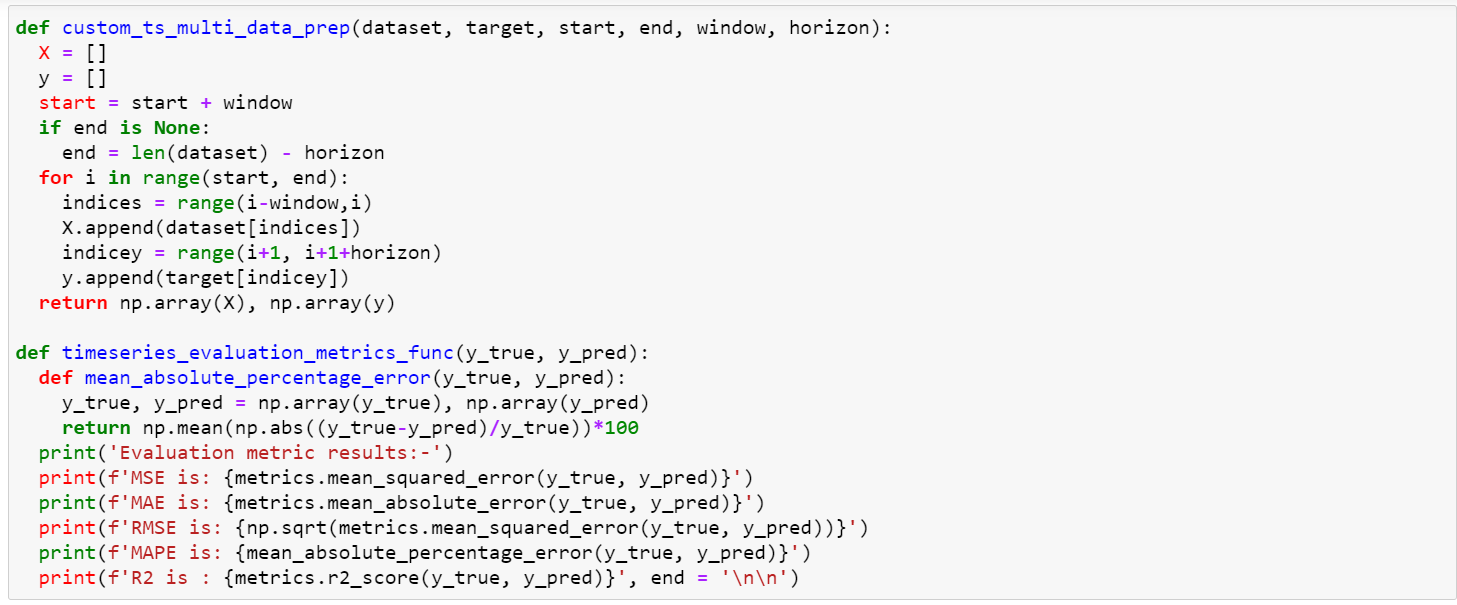
The following preprocessing steps were completed:

* Set date time as an index
* Filled in missing values with the mean value
* Added Month Feature, Day of Week Feature as one hot encoded features
* Added a weekend indicator
* Added a Leap Year indicator
* Added a Quarter Indicator which was one hot encoded
* Added a “Is Holiday” indicator
* Created a separate Jupyter notebook to resample the monthly data to daily data because the data came from difference sources

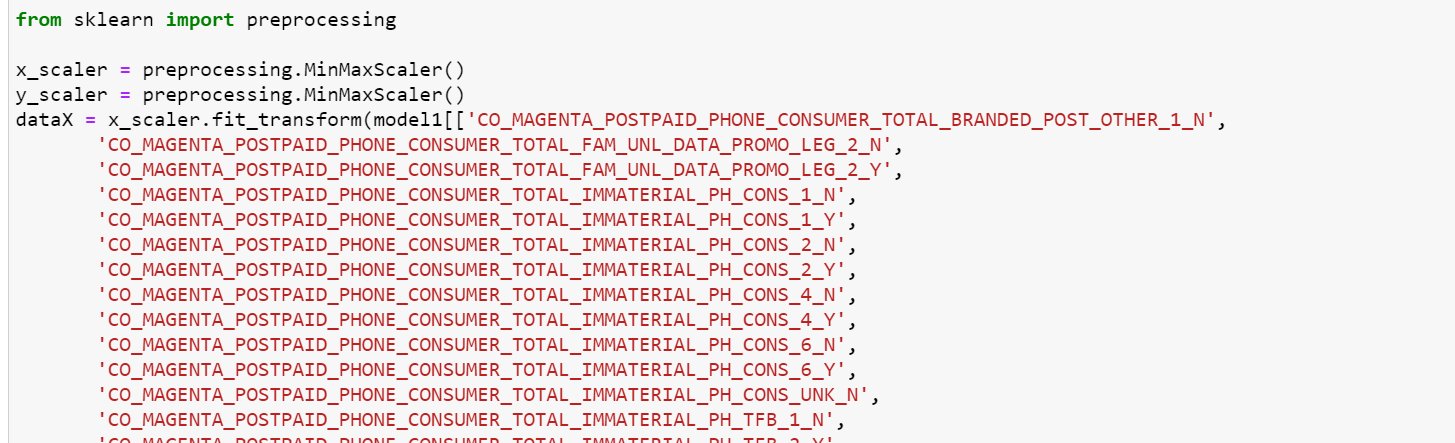
Furthermore, additional steps were taken to perform preprocessing steps for the purposes of machine learning:

* Created a custom data prep function which separated the dataset into an X array and Y array accounting for the windowing time frames and prediction horizon periods
* Created a custom metrics function which returns the Mean\_Squared\_Error. This function provides the validation error during the training process.
* The data was split into X and Y (independent variables and label) and then transformed using the MinMaxScaler() from Sklearn’s preprocessing
* The data was then split into x training and y training sets via the custom data prep function
* The training data was then converted to tensor slices using Tensor Flow’s tf.data.Dataset.from\_tensor\_slices function
* Finally, the data was shuffled and batched

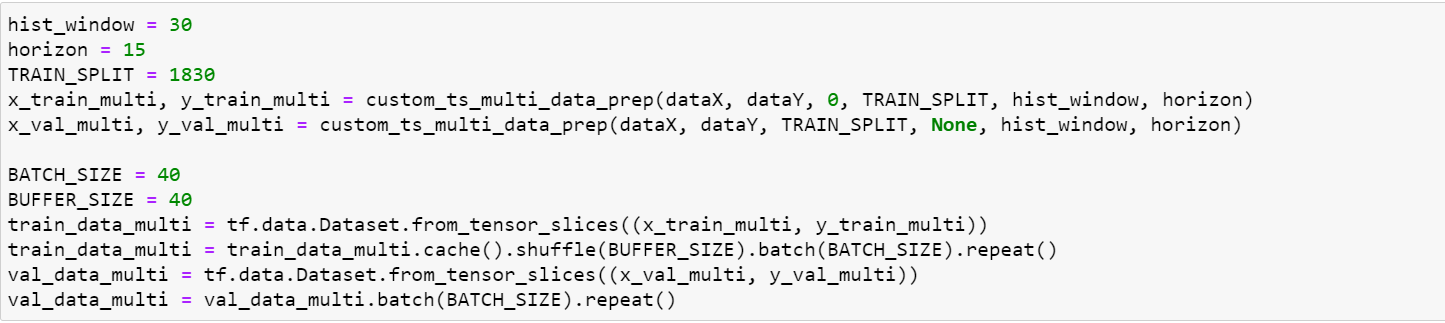
**[Exhibit 1]**



**[Exhibit 2]**



**[Exhibit 3]**



**Observations**

After plotting the Total\_Gross\_Ads, it was clear there was strong Seasonality, Cyclicality, as well as White Noise in the data. The plot below shows major spikes occurring during holiday periods. The most noticeable periods were Thanksgiving, Back Friday, Cyber Monday, and Christmas time. Total\_Gross\_Ads spiked to 80K from an average of 20K, a 300% spike.

These are the periods that would be most difficult for machine learning models to predict, thus most of the architecture will be geared towards predicting these short-term bursts with accuracy.

**[Exhibit 4]**

A picture containing chart

Description automatically generated

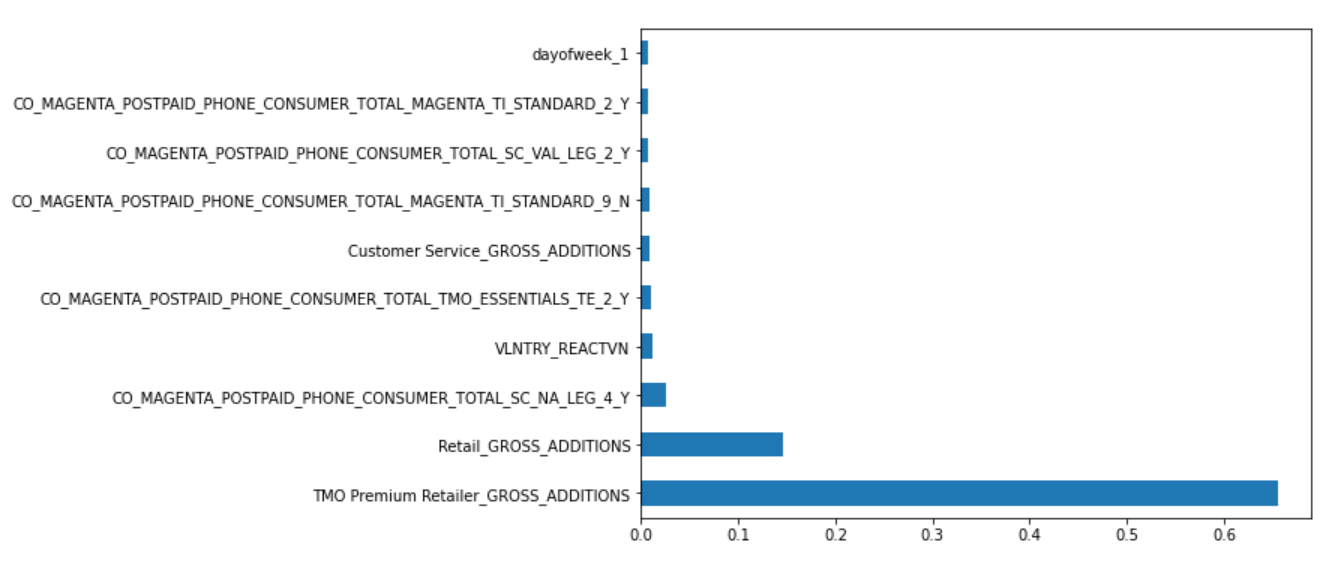
After cleaning the data, a weekly pattern emerged as well. Monday-Thursday was relatively flat. A spike occurred between Friday and Saturday with the peak being on Saturday. Finally, a drastic drop-off occurred on Sunday. The reason for this weekly trend is not quite clear, however; it may be promotion based.

**[Exhibit 5]**

Feature engineering is a critical aspect to any machine learning model. Too many features can be detrimental to the performance and accuracy of the model. This is also known as the “Curse of Dimensionality.” Currently, the data has over 1,500 features some of which need to be dropped because of high correlation with each other. Highly correlated features should be dropped because one of the features is not adding any value but could actually be hurting the model. Its an issue of repetitiveness and duplication.

XGBoost is one of the main algorithms used to determine the top features which impact the dependent variable the most. XGBoost is based on an ensemble method using decision trees. XGBoost can be imported via the XGBRegressor class. The algorithm provided the following results:

**[Exhibit 6]**



The top feature is TMO Premium Retailer Gross Ads. This is basically a sector of T-Mobile. The algorithm is saying that the Retail Sector provides the strongest correlation with the dependent variable – Total\_Gross\_Ads. This is reinforced by the second top feature which is Retail\_Gross\_Ads.

After XGBoost, the other independent variables were dropped to ensure higher accuracy and more efficiency.

**Model Analysis**

Looking at the historical data, it made sense to create three models (similar in nature but having a different architecture). One model was built to handle small fluctuations and flat trends. Another model was built to handle extreme outliers and volatility during holiday periods. The final model was created to handle unexpected events such as COVID 19.

The first model will be referenced as the Nonvolatile Model. The second model will be referenced as the COVID/Recession Model. The final model will be referenced as the Spike Deeptector Model.

**[Nonvolatile Model – Results]**

The nonvolatile model performed extremely well during the nonvolatile periods with **94%** accuracy during the test set. The prediction horizon was 15 days meaning the model predicted the next 15 days of Total\_Gross\_Ads. The validation loss did well against the training loss and there seemed to be very little signs of over overfitting. The model architecture had 408,458 parameters and consisted for Conv1D, Batch Normalization, PreLu, Dropout, MaxPooling1D, Simple RNN, and Dense layers. The optimization function was Adam, and the learning rate was .001.

**[Exhibit 7,8, & 9]**

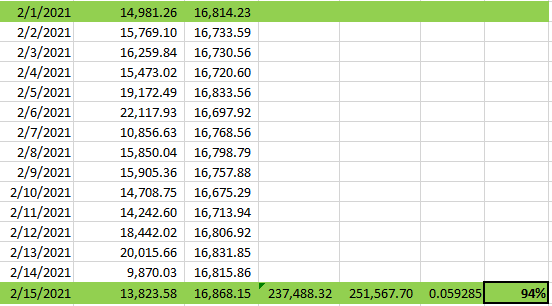
**Actuals**

**Predictions**

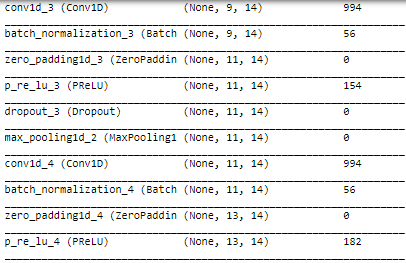
**Actuals**

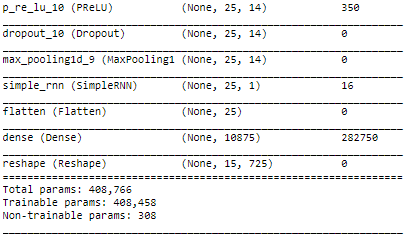
The only downside is the results were relatively flat and the model had a difficult time predicting the fluctuations. However, the key is to be able to predict the sum of the 15 days accurately. The end user will only be seeing the summarized sum of the next 15 days. Thus, it is imperative that the sum of 15 days prediction is as close to the sum of the 15-day actuals. This is how the **94%** accuracy was obtained.

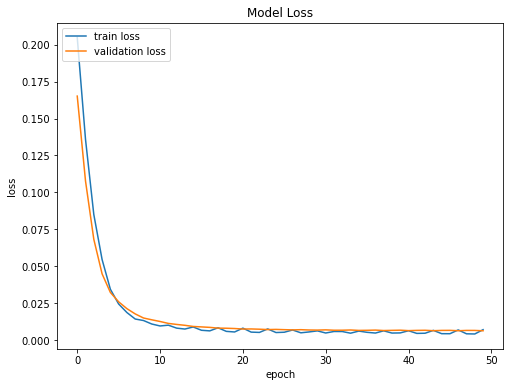
Column A is the date. Column B are the actuals. Column C are the predictions.



**[Exhibit 10 – Model Architecture and Loss]**







**[Covid/Recession Model – Results]**

The Covid/Recession model performed decent during COVID with **81%** accuracy during the test set. The prediction horizon was 15 days meaning the model predicted the next 15 days of Total\_Gross\_Ads. The validation loss was poor and there seemed to be signs of overfitting. The model architecture had 456,666 parameters and consisted of Conv1D, Batch Normalization, PreLu, Dropout, MaxPooling1D, GRU, and Dense layers. The optimization function was Adam, and the learning rate was .06.

The validation loss was poor probably due to the higher-than-normal learning rate. This learning rate was used to help capture the short-term fluctuations. Total\_Gross\_Ads dropped drastically from 20K (on average) to less than 5K (on average). A higher learning rate helped catch this drop in Total\_Gross\_Ads, but overfitting was a byproduct. Also, a major change in this model architecture was changing the history period to 1 day. This means that every iteration uses the past 1 day to predict the next 15 days. Again, this was done to help capture that instant drop in Total\_Gross\_Ads. Finally, the model architecture consisted of fewer neurons and a higher drop out rate to “relax” the model and allow it flexibility in predicting out larger than expected swings.

**[Exhibit 11,12, &13]**

**Actuals**

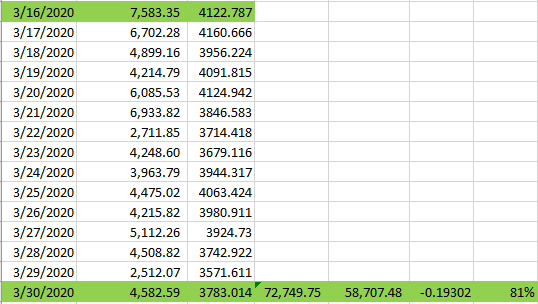
**COVID**

**Predictions**

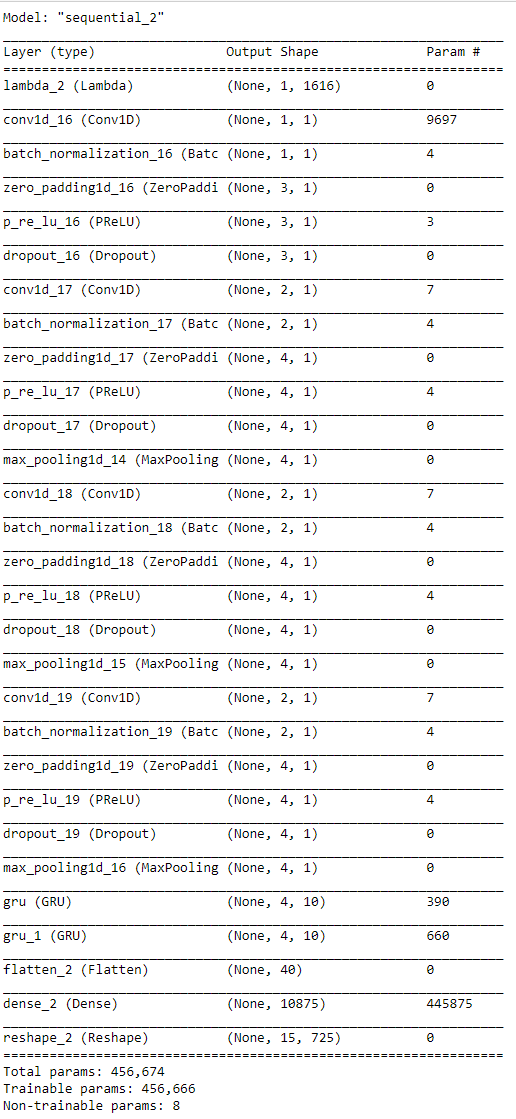
**Actuals**

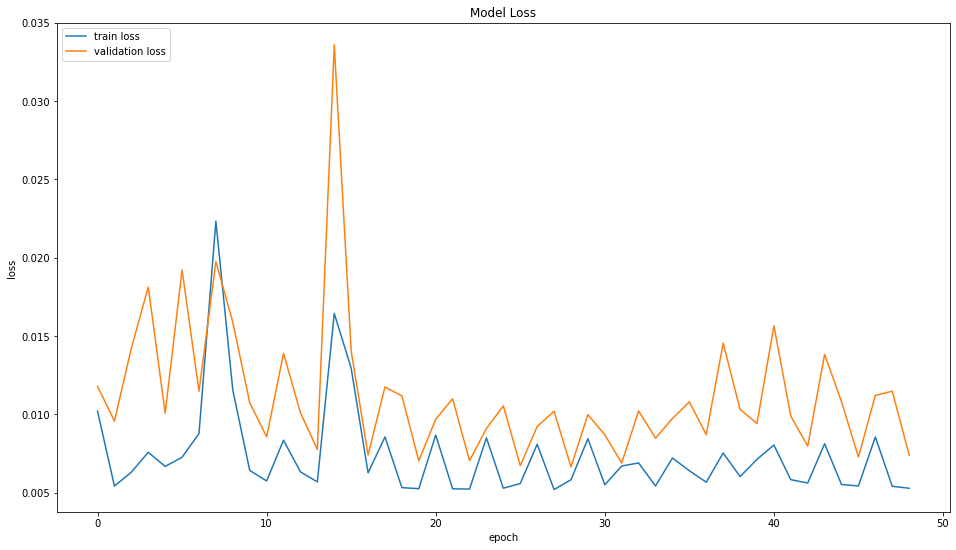
The only downside is the model is relatively flat and still has a difficult time predicting the fluctuations. However, the key is to be able to predict the sum of the 15 days accurately. The end user will only be seeing the summarized sum of the next 15 days. Thus, it is imperative that the sum of 15 days prediction is as close to the sum of the 15-day actuals. This is how the **81**% accuracy was obtained.

Column A is the date. Column B are the actuals. Column C are the predictions.



**[Exhibit 14 – Model Architecture and Loss]**





**[SpikeDeeptector – Results]**

The SpikeDeeptector model performed very well during Christmas time of 2020 which is one of the most volatile periods for Total\_Gross\_Ads with **91%** accuracy. The prediction horizon was 15 days meaning the model predicted the next 15 days of Total\_Gross\_Ads. The validation loss was poor and there seemed to be small signs of overfitting. The model architecture had 6,078,475 parameters and consisted of Conv1D, Batch Normalization, PreLu, Dropout, MaxPooling1D, Simple RNN, Zero Padding, Flatten, Reshape, Leaky Relu, and Dense layers. The optimization function was Adam, and the learning rate was .001.

The validation loss was poor probably due to the number of parameters. This model had a lot of layers and neurons as well as a higher learning rate of .001. A learning rate of .00001 or smaller would have resulted in a smoother loss graph. However, the model would not have been able to predict the volatility. This learning rate was used to help capture the short-term fluctuations. Total\_Gross\_Ads fluctuated between 10K and 70K in a short period of time (between 11/15/2020 and 12/31/2020). Thus, the model architecture’s history period was changed to 1 day from 5 days. This means that every iteration uses the past 1 day to predict the next 15 days. Again, this was done to help capture that instant spikes in Total\_Gross\_Ads. The model does not predict the fluctuations perfectly, but it is able to include the general volatility which is why the model performed well with **91%** accuracy. As a reference, the Nonvolatile model performed with **41%** accuracy during the same period.

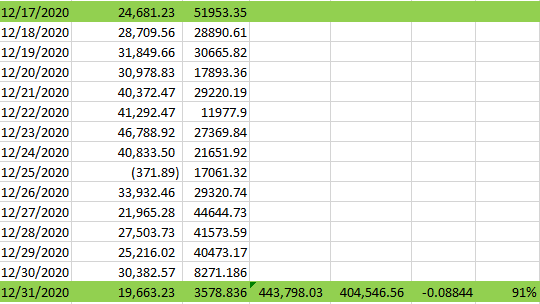
**[Exhibit 15,16 &17]**

**Predictions**

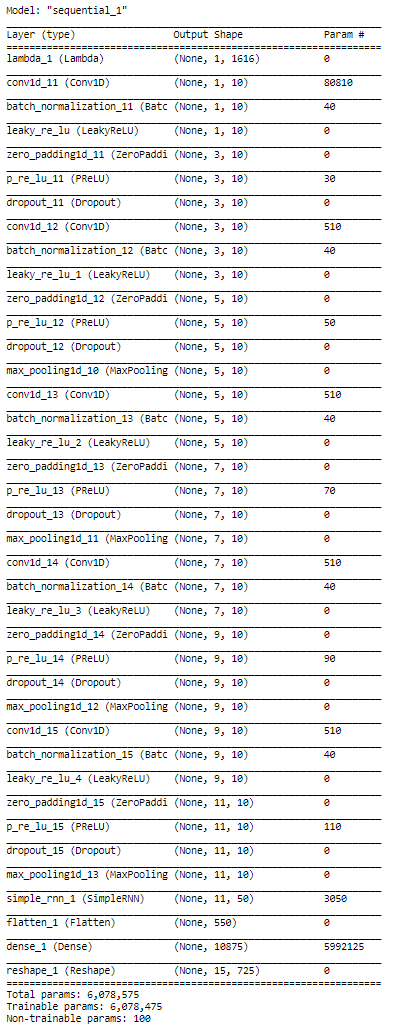
**Actuals**

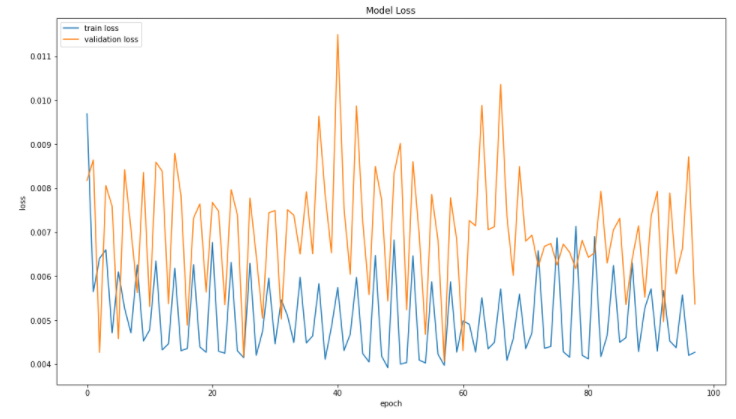
The end user will only be seeing the summarized sum of the next 15 days. Thus, it is imperative that the sum of 15 days prediction is as close to the sum of the 15-day actuals. This is how the **91**% accuracy was obtained.

Column A is the date. Column B are the actuals. Column C are the predictions.



**[Exhibit 18 – Model Architecture and Loss]**



****

**Recommendation**

It is evident that daily time series prediction is a very complex task. This data included non-linearity, multi-dimensionality, interdependency between features, short term trends, long term trends, noise, seasonality, cyclicality, and volatility.

Simple models such as ARIMA and SARIMA are not adequate to deal with such complexity. Even basic machine learning models such as SVM and decision trees cannot deal with this high complexity and dimensionality of the data.

There has been great interest and research into deep learning models, especially those dealing with Convolutional Layers and LSTM (Long Short-Term Memory) networks. In fact, the combination of these two networks are the most accurate to date. Convolutional layers are responsible for learning feature importance and learning short term trends and dependencies (weekly trend). LSTM layers are responsible for learning longer term trends (monthly, yearly).

Deep learning models are also great for multi-variate multi-step prediction cases. Again, simple linear models such as ARIMA and simple machine learning models such as SVM are more accurate for univariate single-step prediction cases.

The complexity of this data also required more than one model. Even today’s networks are not perfect at capturing all trends, noise, and volatility. Finally, this case study reinforces the idea that not one model fits all cases.