Shreyas Puttaraju

CSC 578

Class Project Part A – Kaggle Competition

Kaggle Score – 295 Rank – 20

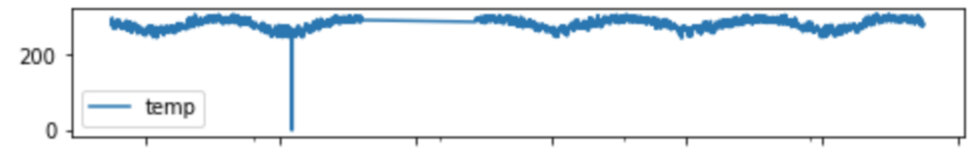
The main goal of the class project was to apply deep learning to do time series forecasting. We were given traffic data at a location in Minnesota.

Journey of Preparing My Best Model:

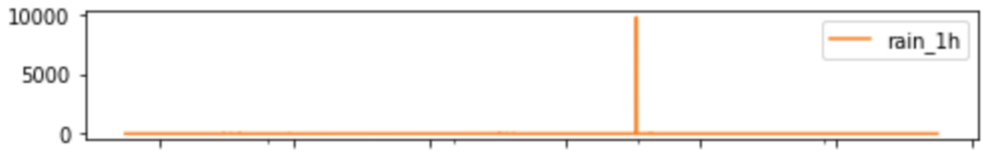
Data Preparation: The data had 5 numeric variables and 3 categorical variables. Traffic Volume was my target variable. We will go through each variable.

Holiday: I did see the number of values. None had majority of the data i.e, more than 40,000 and all other combined was 50. So I combined all the holidays as work day and none as work day. Later this was one hot encoded.

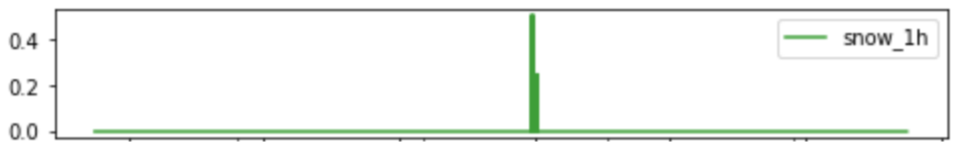
Temp: There was only one outlier in temp. That was replaced with the mean value. Later this was normalized with MinMax scaler before splitting the data.



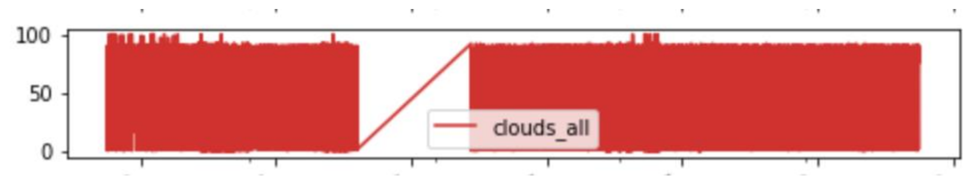
Rain_1h: It had one outlier. I filled it with the very next value. Later this was normalized with MinMax scaler before splitting the data.



Snow_1h: Majority of the vales were zero and very less values of snow record. This was normalized with MinMax scaler before splitting the data.



Clouds_All: Had the values of clouds at that hour.



weather_main: I created dummy variables for this variable.

weather_description: It had a detailed description of the weather main variable. This was dropped and not used in the model since both were highly correlated with each other.

Hour: Created a new variable hour from date time variable.

Date_time: Dropped.

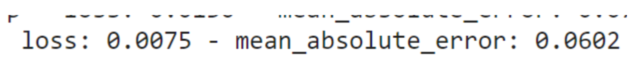
Traffic_volume: Target variable. Was normalized before splitting and denormalized at the end.

After normalization the data was split into train, validation and test.

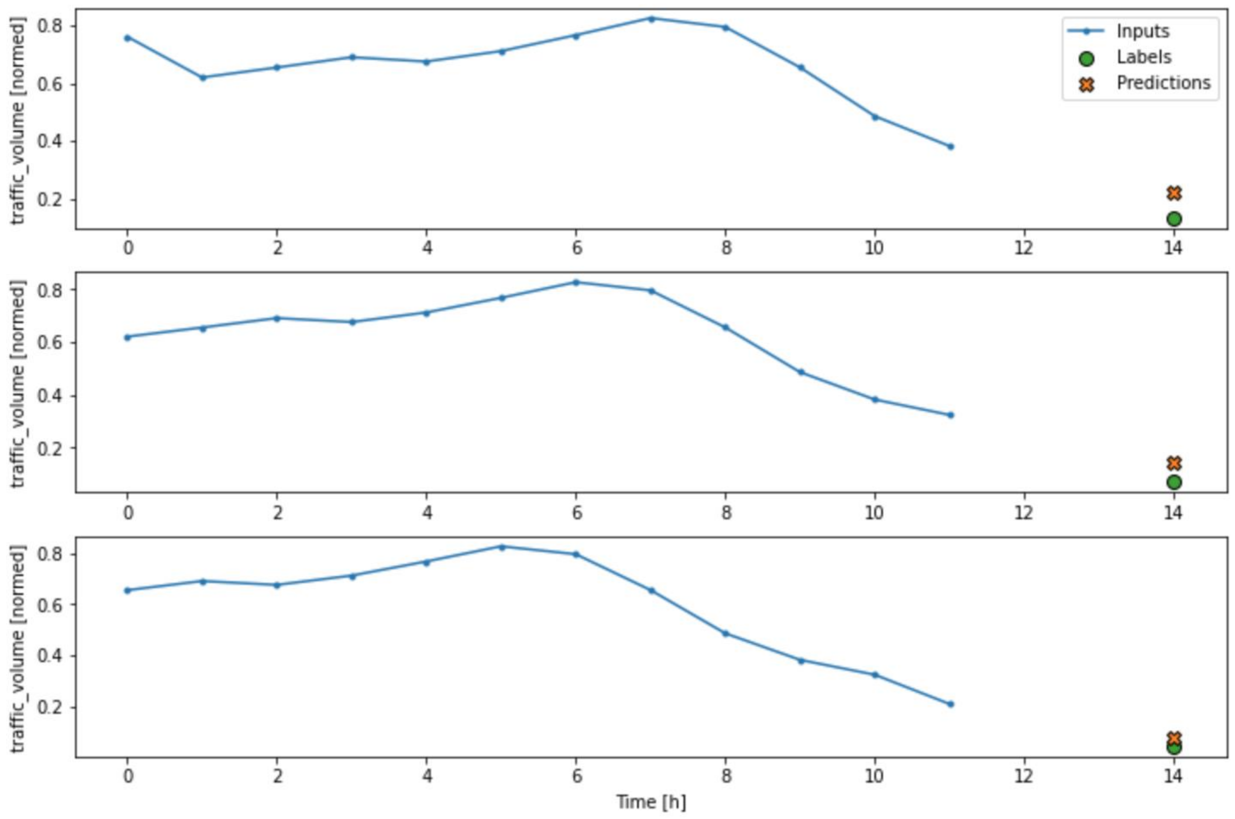
Window was set to 12 inputs, 3 shift and 1 label.

Baseline model:

Basic LSTM Model from the tensorflow tutorial. This had a single LSTM layer of 128 nodes and a single node dense layer. I trained the model for 20 epochs with a batch size of 128. Adam optimizer was used with a learning rate of 0.0001. Early stopping call back was set to monitor validation loss with a patience set to 10. The performance of the model was as follows.



```
loss: 0.0075 - mean_absolute_error: 0.0602
```



After building my baseline model, I did try a number of combinations and different types of models.

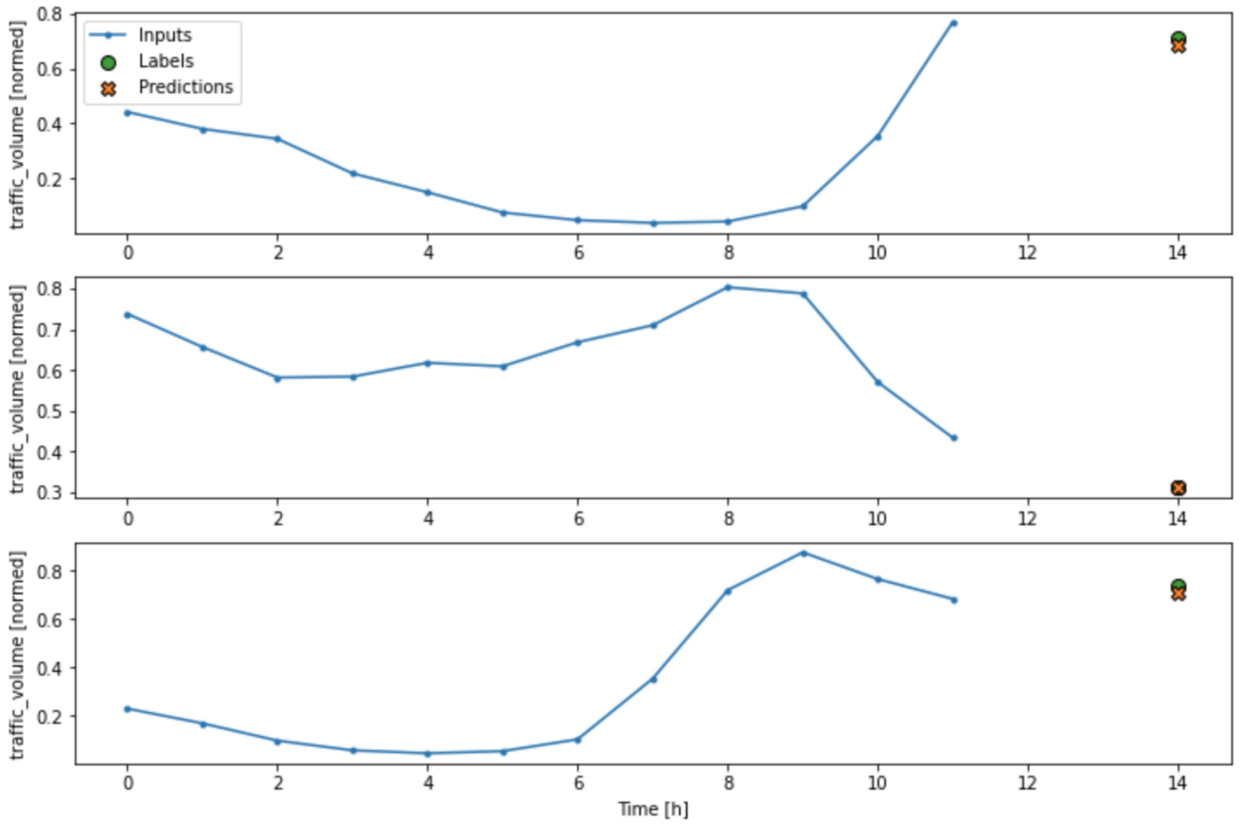
My Best Model:

The model that gave the best predictions was a bidirectional LSTM model. It had bidirectional layer with 512 nodes, two normal LSTM layers with 256 and 128 nodes and a dropout of 0.1. Followed by two fully connected dense layers and a last output dense layer. The structure of the model is shown below.

```
bi_lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(512, return_sequences=True, stateful = False)),
    tf.keras.layers.LSTM(256, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    #tf.keras.layers.BatchNormalization(),
    tf.keras.layers.LSTM(128, return_sequences=False),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(units=128, activation='relu'),
    #tf.keras.layers.Dense(units=128, activation='relu'),
    tf.keras.layers.Dense(units=64, activation='relu'),
    #tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(units=1),
    tf.keras.layers.Reshape([1, -1]),
])
```

The performance of this model is as follows.

```
loss: 0.0066 - mean_absolute_error: 0.0520
```



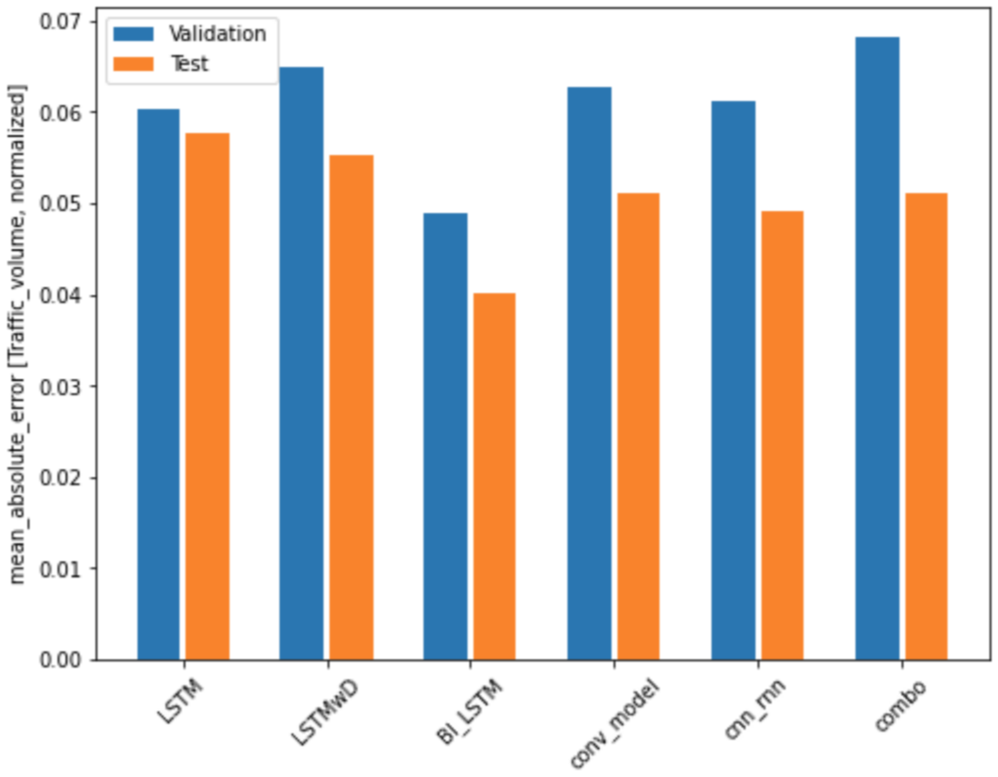
Runner Up Model:

The second-best model was a 1D conv model combined with a Bidirectional with a couple of LSTM layers and Dense layer stacked up. I was expecting this model to perform better than the BI_LSTM model since it was built upon that model. The performance was very close to the BI_LSTM model. There were few epochs where the mae score was lower than the mae scores of the best model. The kaggle score differed from the model by just 10-15 points. The structure and performance is shown below.

```
#1D Conv Model with Bi-LSTM
cnn_rnn_model = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=128,
                           kernel_size=(12,),
                           activation='relu'),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(512, return_sequences=True, stateful = False)),
    tf.keras.layers.LSTM(128, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM(128, return_sequences=False),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=1),
])
```

```
loss: 0.0088 - mean_absolute_error: 0.0611
```

I did try number of models. I did try to tune parameters such as number of epochs, batch size, learning rate, different type of optimizers, with and without early stopping, different number of dense and LSTM layers. I chose the five best models and plotted the performance of those.



Final Conclusion

The difficulty of this project was intermediate level. The TensorFlow tutorial was helpful. I did struggle a lot for getting my Kaggle scores from four digits to three digits. The discussion on d2l helped a lot for this problem. I was a bit confused about the date_time variable, but after several trial and errors finally decided to just keep the hour and dropping year month day minutes and seconds. This was one of the factors that helped me reduce the Kaggle score. I did feel that data preparation and preprocessing had more importance than building complex models with huge number of layers and nodes. At one point I had a common model and tried testing for different types of data transformation techniques(Dropping/Adding columns, normalization techniques, one hot encoding etc.). I still had couple of ideas and combinations to try but the number of submissions was restricted. This was one of the best projects and fun project that I had to work on.