**Name** – Goutham Selvakumar

**Course Name** – Neural Networks and Deep Learning

**Course Number** – CSC 578

**Kaggle Username** – Goutham Selvakumar

**Rank – 10 [246.29115] (Public Score)**

# Final Class Project (A) Time Series Kaggle

## Introduction:

- I created a **Baseline Model** and also **five** other models in order to see which one does better in **performance**. I was able to achieve a better result with the **Model – 4 (Kaggle Best Model)** which will be explained below as we go. So, the other Models were executed and produced results were good and bad at the same time.
- Before conducting the experiments in these models, firstly, I loaded the data and the **preprocessing** steps were done that includes the **Data Cleaning**, **Data Transformation**, and **Feature Extraction**.
- Second step, was to use the method **Data Windowing** as from the **Tensor Flow Tutorial**, this was an important task for the **time-series forecasting**. This **Window Generator class** helps to convert the data into a usable format from which the model might be able to learn from the last 12 hours in order to predict the future data.
- Splitting the Data was done as the next process, it was split into three parts, namely **training**, **validation**, and **testing**. Next, I created **dictionaries** in order to store the performance of the model on the **validation** and **testing** datasets.
- According to the instructions, I created a **Window Generator class** with the respective parameters as the input width – 12, offset – 3, and label width – 1. Using the **Tensor Flow Tutorial**, I was able to create a function that compiles and fits the model and uses the **MSE** as the **loss function** and the **Adam Optimizer** in it.
- An important note is that **I did not shuffle the data before or after the train/test splitting**. So, the next process would be to construct the Baseline Model and the other models in order to see the performance in them.

## Models and Development:

## BASELINE MODEL

### Model – 1:

#### Architecture: -

- Here, in this baseline model I trained a **LSTM-based model** that has 2 **LSTM layers**, 2 **dropout layers**, and 3 **dense layers**. In order to avoid the **overfitting** in the model I also included the dropout layers on **regularizing** the model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 12, 128)           78336

 dropout (Dropout)           (None, 12, 128)           0

 lstm_1 (LSTM)               (None, 12, 128)           131584

 dropout_1 (Dropout)         (None, 12, 128)           0

 dense (Dense)               (None, 12, 64)            8256

 dense_1 (Dense)             (None, 12, 32)            2080

 dense_2 (Dense)             (None, 12, 1)             33

 reshape (Reshape)           (None, 1, 12)             0

=================================================================
Total params: 220,289
Trainable params: 220,289
Non-trainable params: 0
_________________________________________________________________
```

#### Expectation: -

- From the **epoch**, we can see that the model is learning pretty well with each of the epoch, where the **loss** and **mean absolute error** is getting **reduced** on the **training** and **validation**. This will lead to a **better accuracy** of the model.
- Furthermore, from the **epoch**, the performance on the validation seems to be **irregular**, like we can see there is a small improvement in the **mean absolute error** and **loss** that tells about the overfitting present in it.

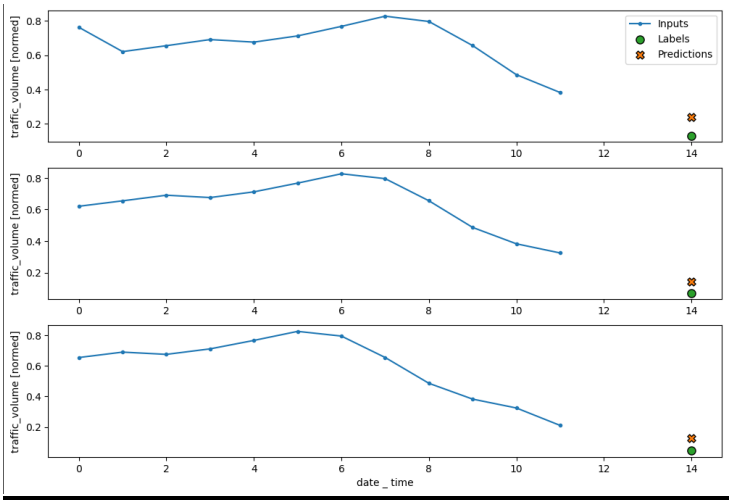
#### Result: -

- In order to further improve the performance of the model and reduce the loss, we are moving onto the other models that helps in improving the performance.
- Among the other models, this model's performance was good with the performance and the value on both the **loss** and **mean** absolute error.

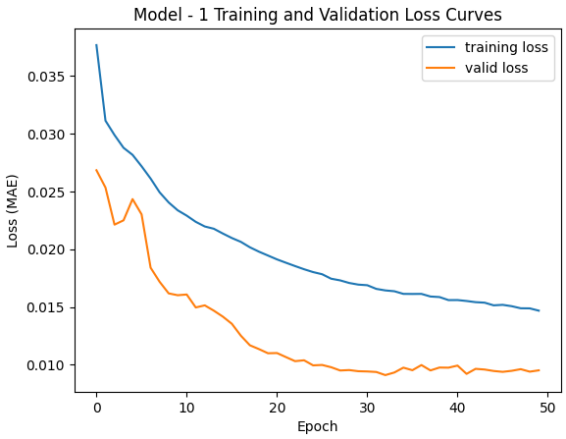
#### Epoch (50/50):

```
Epoch 50/50
955/955 [==============================] - 12s 13ms/step - loss: 0.0147 - mean_absolute_error: 0.0781 - val_loss: 0.0095 - val_mean_absolute_error: 0.0681
156/156 [==============================] - 1s 4ms/step - loss: 0.0095 - mean_absolute_error: 0.0681
```

**Window Plot:**




**Loss Plot:**




**BEST MODEL**

**<u>Model – 4: Bidirectional RNNs:</u> [Kaggle-Best Model]**

- Among all the models that I have experimented, the **loss value** in this **Model -4** seems to be **decreasing** very good consistently which indicates that the model is trained well.

  **Architecture:-**

  - This Model – 4 uses **one Bidirectional LSTM model** with **512 units and** 2 **LSTM layers** of 128 **units**, 2 **additional LSTM layers** with 128 **units** on each of the layer and then 2 **dense layers** that consists of 64 and 32 units respectively. The **output layer** consists of a 1 unit **dense layer**.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 12, 1024)         2199552
 l)

 lstm_13 (LSTM)              (None, 12, 128)           590336

 lstm_14 (LSTM)              (None, 12, 128)           131584

 dense_9 (Dense)             (None, 12, 64)            8256

 dense_10 (Dense)            (None, 12, 32)            2080

 dense_11 (Dense)            (None, 12, 1)             33

 reshape_3 (Reshape)         (None, 1, 12)             0

=================================================================
Total params: 2,931,841
Trainable params: 2,931,841
Non-trainable params: 0
_________________________________________________________________
```

**Expectation**:-

- From the below epochs of 50, we can conclude that the **loss** on the **training** seems to be **reduced increasingly well** and the **validation loss** is also **reduced** by the number of epochs that indicates that the model is **not overfitting**.
- The **MAE (Mean Absolute Error)** is also getting **reduced** overtime by the number of epochs, since MAE tells how the model's prediction are on average. I trained the LSTM on the sequence data, and the **bidirectional RNNs** here is capable of allowing the past and future data in the sequence to be added.

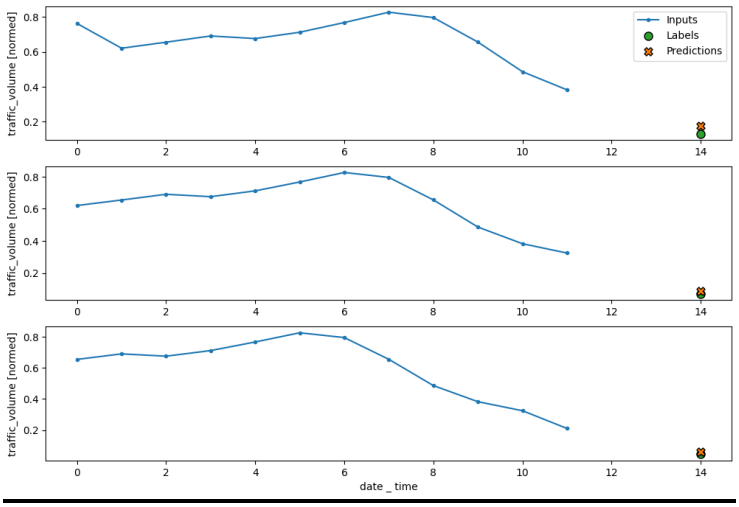
**Result**:-

- With this model, I was able to achieve a **significant result** among the **Kaggle** [**246.29115**] and this was the **Best Model** in comparison with the other models because of **the LSTM Bidirectional LSTM layer**.

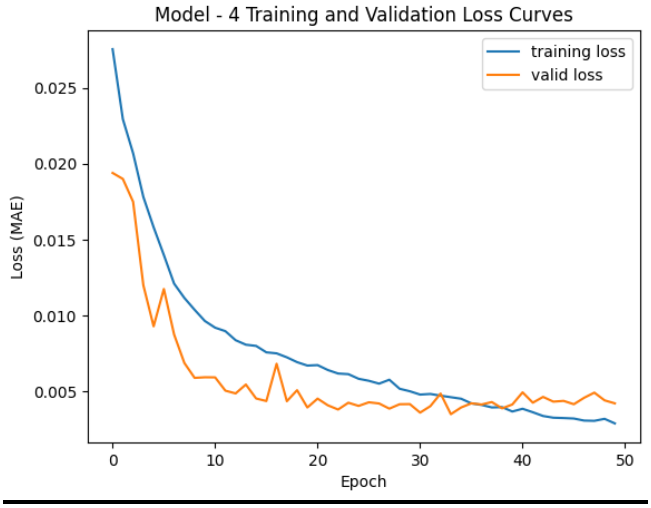
**Epoch (50/50):**

```
Epoch 50/50
955/955 [==============================] - 83s 87ms/step - loss: 0.0029 - mean_absolute_error: 0.0337 - val_loss: 0.0042 - val_mean_absolute_error: 0.0415
156/156 [==============================] - 4s 25ms/step - loss: 0.0042 - mean_absolute_error: 0.0415
```

**Window Plot:**




**Loss Plot:**




**WORST MODEL**

**<u>Model – 2: Adding Layers</u>**

**Architecture:-**

- Due to the addition of 2 **LSTM layers** with 128 units each may have been not good as it's contributing to the **model's worst performance**. This model consists of 2 **dense layers** with 64 and 32 neurons and the **last dense** having **one** unit respectively with 2 **dropout layers** in the model.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 12, 240)           254400

 lstm_3 (LSTM)               (None, 12, 240)           461760

 dropout_2 (Dropout)         (None, 12, 240)           0

 lstm_4 (LSTM)               (None, 12, 128)           188928

 lstm_5 (LSTM)               (None, 12, 128)           131584

 dropout_3 (Dropout)         (None, 12, 128)           0

 dense_3 (Dense)             (None, 12, 64)            8256

 dense_4 (Dense)             (None, 12, 32)            2080

 dense_5 (Dense)             (None, 12, 1)             33

 reshape_1 (Reshape)         (None, 1, 12)             0

=================================================================
Total params: 1,047,041
Trainable params: 1,047,041
Non-trainable params: 0
_________________________________________________________________
```

**Expectation**:-

- The **training** and **validation loss** was way worse in comparison to the other models that I experimented, within each of the epochs I could see the model performing worst.
- This could be due to the **overfitting** of the model, since there are **additional layers** present in the model that could make the model to fit too closely to the training data, which in term causes the performance to be worse.
- Also, the use of **dropouts** which prevents the overfitting by setting a **fraction rate** of units randomly for each training time. These may lead to overfitting of the model if not performing better.

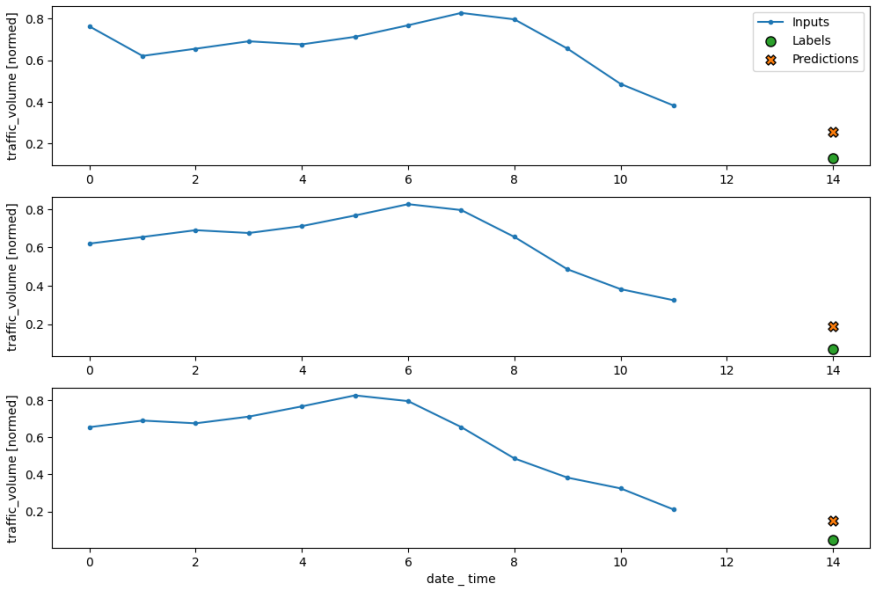
**Result:-**

- With the **additional layers** the model performed the **worst** among the others, and also the **performance** was not that good.

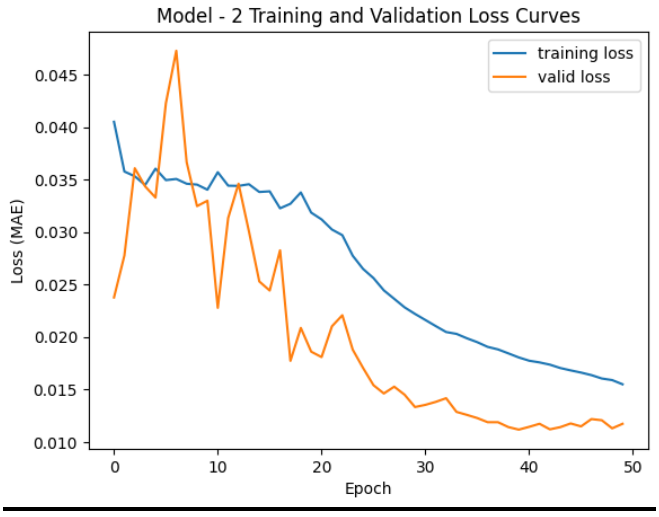
**Epochs (50/50):**

```
Epoch 50/50
955/955 [==============================] - 51s 53ms/step - loss: 0.0155 - mean_absolute_error: 0.0809 - val_loss: 0.0117 - val_mean_absolute_error: 0.0779
156/156 [==============================] - 2s 11ms/step - loss: 0.0117 - mean_absolute_error: 0.0779
```

**Window Plot:**



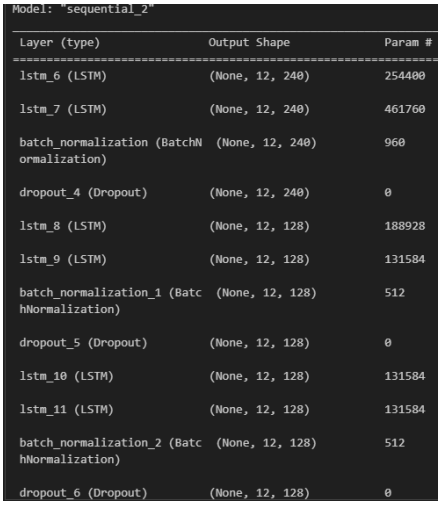

**Loss Plot:**




## Model – 3: Batch Normalization

## Architecture:-

- This model consists of one **LSTM layer** with 240 units and 2 **additional LSTM layers** with, 128 units of respectively. Also, 3 **BatchNormalizatio**n **layers**, and 3 **dropout layers**. This model also uses an additional of 3 **dense layers** with the neurons of 64 and 32 respectively with the last dense layer with **one** unit.




**Expectation:-**

- With the end of the **epochs** 50/50 we can see that the **loss** and **mean absolute error** was 0.0157 and 0.0748 on the **validation data** which explains that the model is learning well and is performing well after the end of epoch.
- I thought the model would **accurately** run the **training** and predicts the **target variable**, and as we know the model's accuracy improves when the there is a **decrease** in the **loss**.

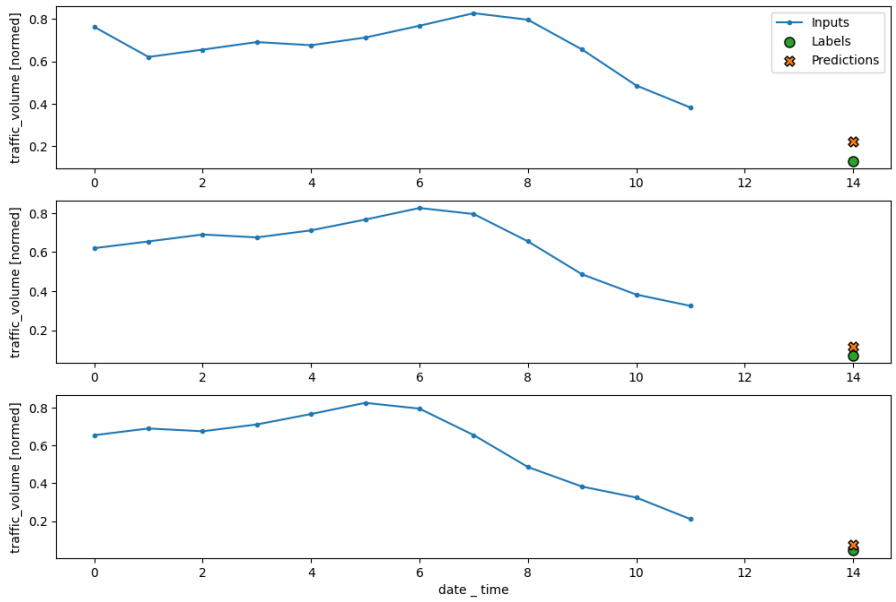
**Results:-**

- This model performed well in comparison to the other models but not as good as the **Model – 4** that was the **best model**. Because of the **additional LSTM layers**, and the **batch normalization**, the model was able to perform better.

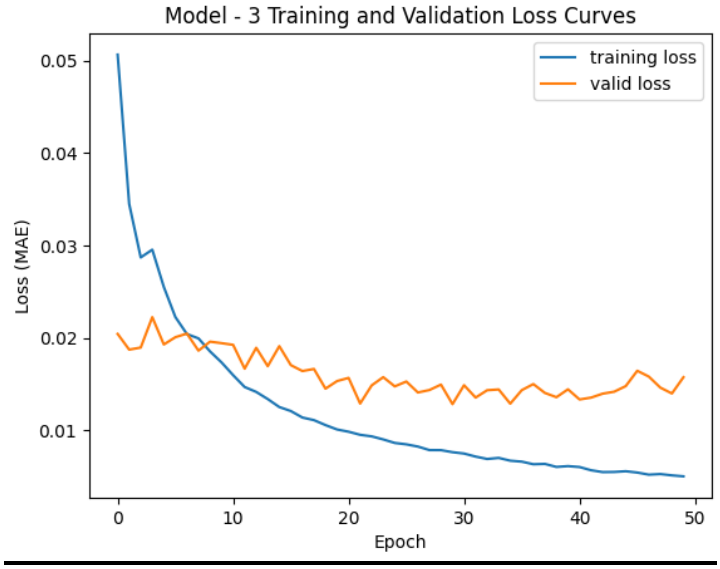
**Epochs (50/50):**

```
Epoch 50/50
955/955 [==============================] - 60s 63ms/step - loss: 0.0050 - mean_absolute_error: 0.0450 - val_loss: 0.0157 - val_mean_absolute_error: 0.0748
156/156 [==============================] - 2s 13ms/step - loss: 0.0157 - mean_absolute_error: 0.0748
```

**Window Plot:**




**Loss Plot:**




## Reaction and Reflection on the results and competition:

- This is the second time I'm joining a kaggle competition. It was challenging and enjoyable at the same time, watching many of my classmates competing each other for a better score. I was able to bring up a good score with my model, but the hardest part was to experiment with different hyperparameters and try to achieve a better score with everyone among the leaderboard was the hardest. My previous scores and the ranking motivated me to bring up with different models and get a better score every time I submit my predictions.

**Reaction and Reflection on the project and course:**

- With this advanced course, I was able to learn more in-depth about the neural networks, deep learning and with various assignments and quiz I was able to learn more methods in python and also the tensorflow. This is my second advanced course, and with every assignment the difficulty was increasing, overall the course was very challenging since it involved new techniques that I have not implemented and tried to learn it from the beginning and getting the results were challenging.