Model for One day forecast of the total number of daily occurrences

# Data

* Data/daily.csv

This is a processed data that is included in the scripts in beginning section of *naive\_forecast.ipynb*

# Notebook & Code

Codes/Notebook for the data processing can be found in

* *naive\_forecast.ipynb (Jupyter notebook)*
* *tf\_forecast.ipynb* (Google Colab link)

For the Tensorflow model, I used Google Colab notebook instead of a local environment because of the ease of environmental setup. The Local jupyter notebook for the naïve\_forecast is instead just using standard python libraries and packages.

# Methods

Since the focus is for 1-day forecast, and that we have daily data that is just about 13 years.

I am opting for Naïve/ *tf* type of methods instead of more traditional forecast methods such as Arima or Prophet. Since the focus is for 1-day forecast, whereby naïve/ *tf* method would typically perform better with short forecast horizon.

# Forecasting models

**Naïve Methods** - *naive\_forecast.ipynb*

*Please refer to the notebook for the better description, actual codes/methods*

* Last known – Last known value just before the forecast
* Moving Average (3 days) – Past 21 days windows Moving Average
* Moving Average (7 days) – Past 21 days windows Moving Average
* Moving Average (21 days) – Past 21 days windows Moving Average
* Last Valid (7 days) – By taking the last non 0 value over the past 7 days windows
* Last Valid (21 days) – By taking the last non 0 value over the past 21 days windows
* Simple Exp Smoothing
* Holt
* Additive

***tf* Methods** - tf\_forecast.ipynb

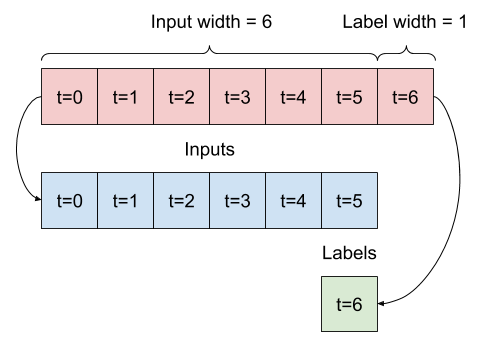
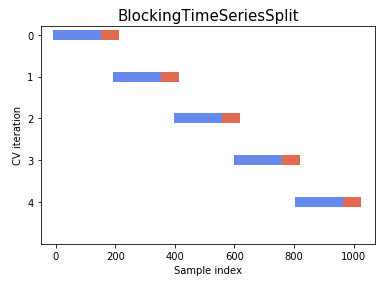
*Please refer to the notebook for the better description, actual codes/methods*. This borrows heavily from <https://www.tensorflow.org/tutorials/structured_data/time_series>.

* Base Line – similar in nature to last known
* Linear
* Dense
* Multistep Dense (30 days)
* CNN (30 days)
* LSTM (30 days)
* Residual LSTM (30 days)

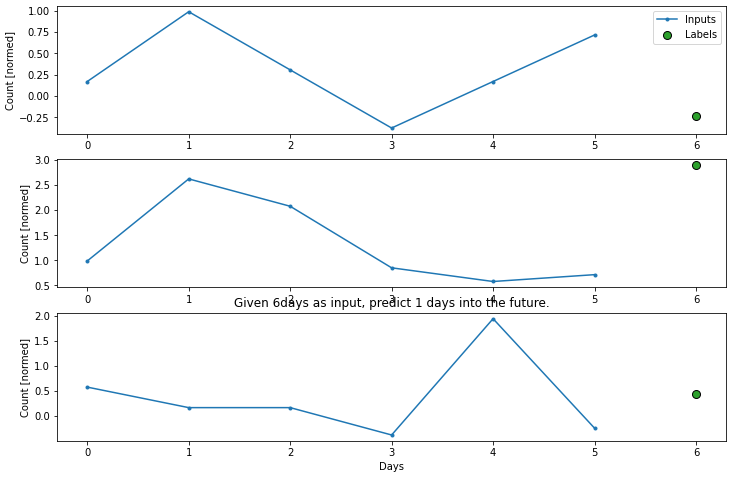
Instead of predicting the next value, predict the how the value will change in the next timestep with RNN/LSTM.

# Splitting & Data windows

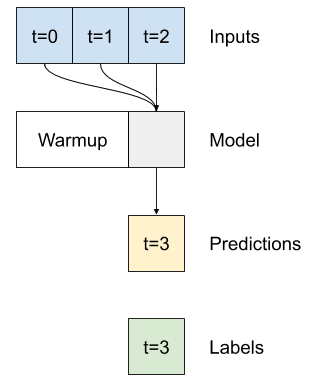
The image below depicts the data used to train individual models with 6 days of data on Event counts. The amount of days used can be tweak/optimized to acquire a better 1-day forecast.



The splitting window is then ‘walk forward’ over the data as such, note that since we are focusing on 1-day forecast, we will always only be taking a single step (day) forward before model and forecast, then rinse and repeat throughout the dataset to measure the effectiveness.

Hence, given 6 days as input and predict 1 days into the future, the splitting windows yields.

Whereby the Inputs are what used to train the model, while the Label is forecasted value.

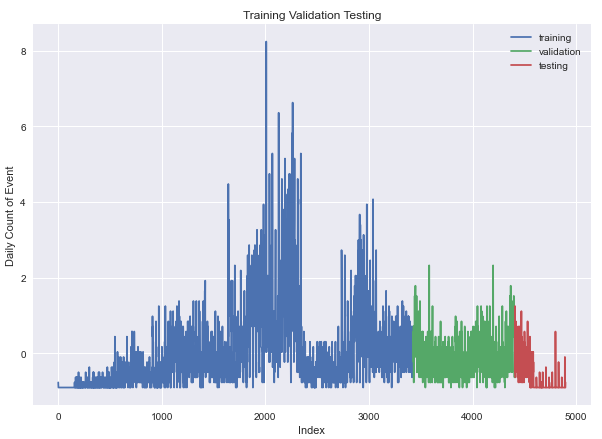


In effect, the forecasting Model - F for different methods is applied on the input to yield prediction that would be compared (in the case of Naïve method) and optimized (For more complex Keras method) with the label data.

# Training, Validation, Testing

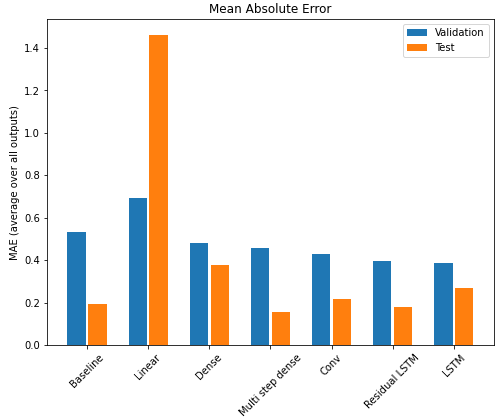
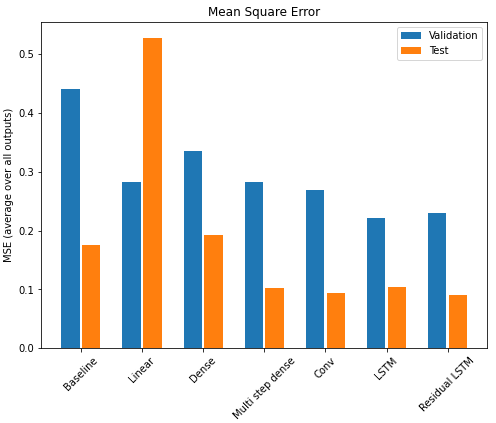
To ensure that the model performance is generalisable to both train data as well with unseen data, the historical data is separated into.

* Training [70%] - Where the model parameters are optimized
* Validating [20%] – Validating the forecast
* Testing [10%] – Where the built model is applied to

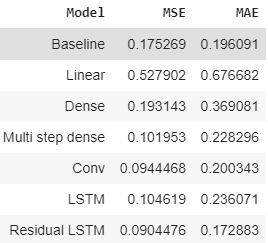


The separation also allows us to train the models with earlier data, while validating and testing the data on recent data points (where it should hold more value).

# Performance Comparison between models

Only **tf Models** are included here, as **Naïve models** performance is inferior from the get go..

**Testing Metrics**

Thus, from the tables and charts tallying data from validation and test, it appears that **Residual LSTM** is the best performing model in both **Mean Square Error (MSE)** and **Mean Absolute Error (MAE)**.