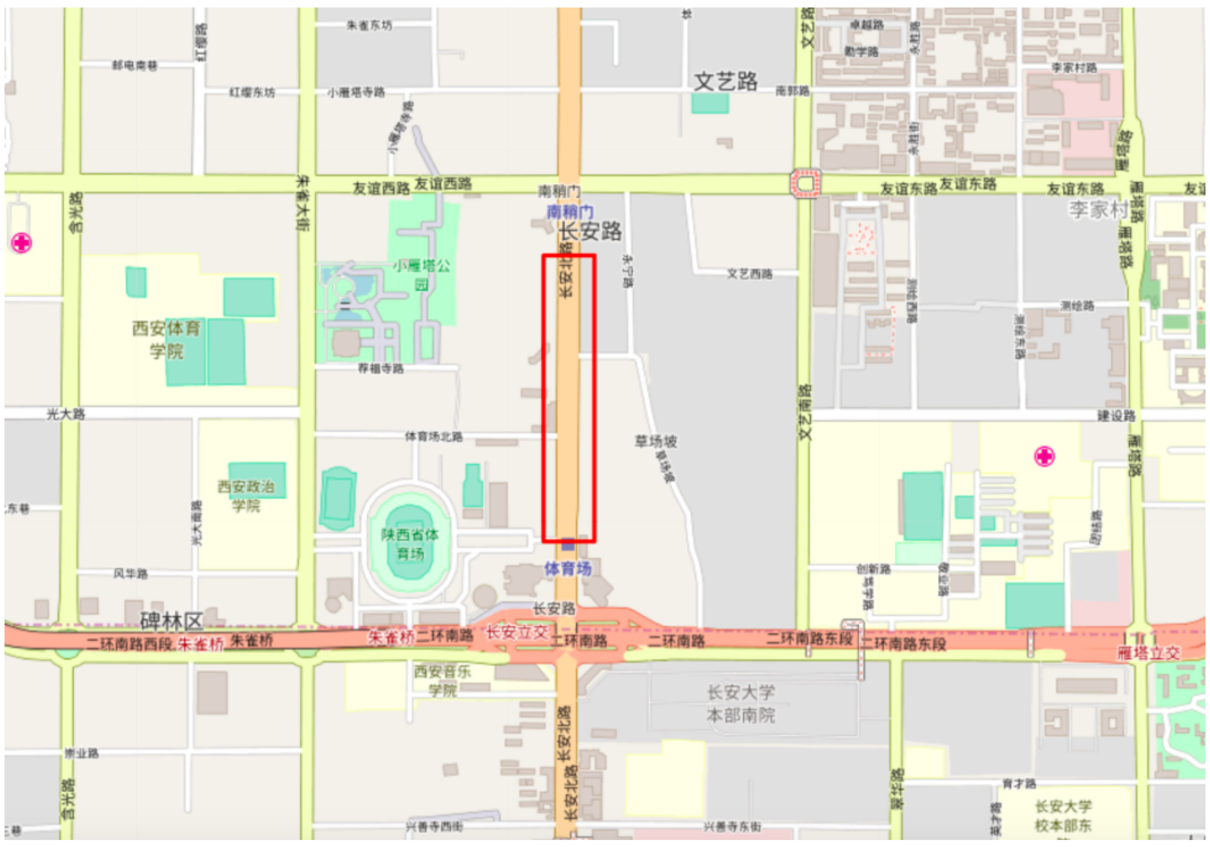
Forecasting Road Speed with DiDi Data: STA 141B Final Project

Project Outline

Introduction
Research Question
Data
Methodology
Results
Conclusion

Introduction

Traditional traffic dynamics models quantify flow through loop flow detectors that are built beneath road pavement or other types of sensors. In California, the PeMS dataset provides access to all traffic data on most highways collected in this manner. However, with the advent of big data, large amounts of transportation data can be collected, which allows researchers to forecast traffic flow, speed, and congestion by manipulating this data. The Transportation Forecasting Competition (TRANSFOR 19), organized through the Transportation Research Board (TRB) challenges participants to use DiDi rideshare data to forecast traffic conditions in the city of Xi'an, China on the Second Ring Road, as shown in the red box below.



DiDi is a ridesharing company based in China, and is similar to the American company Uber. Users hail a DiDi car using their smartphones, and drivers then pick up clients based on their location. Since DiDi drivers share the road with all other vehicles, it is possible that such ridesharing data can be strong proxies for other traffic measurements like speed, congestion, flow, shockwaves, and the existence of bottlenecks.

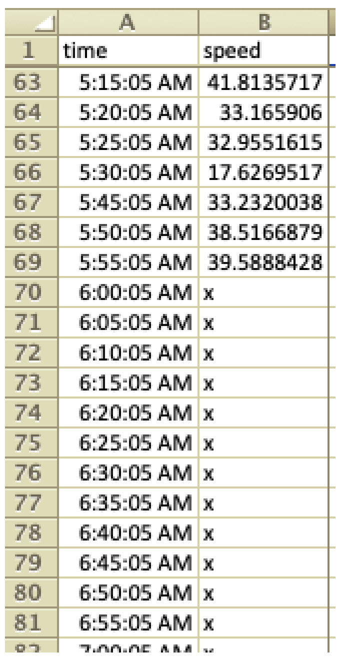
Research Question

The question which we wish to answer is, how can we manipulate DiDi rideshare data in order to predict road speed for certain times of the day? The goal is to estimate speed values that match the actual speed values as close as possible for two missing windows of time: 6am-11am and 4pm-9pm on a given day. This day will be referred to as the "missing day" from here on out. We are given DiDi rideshare trajectory data for 2 months of time, with a "missing" day being the day for which we must calculate the missing speeds. To better understand how the data interfaces with the research question, we describe the data in the next section.

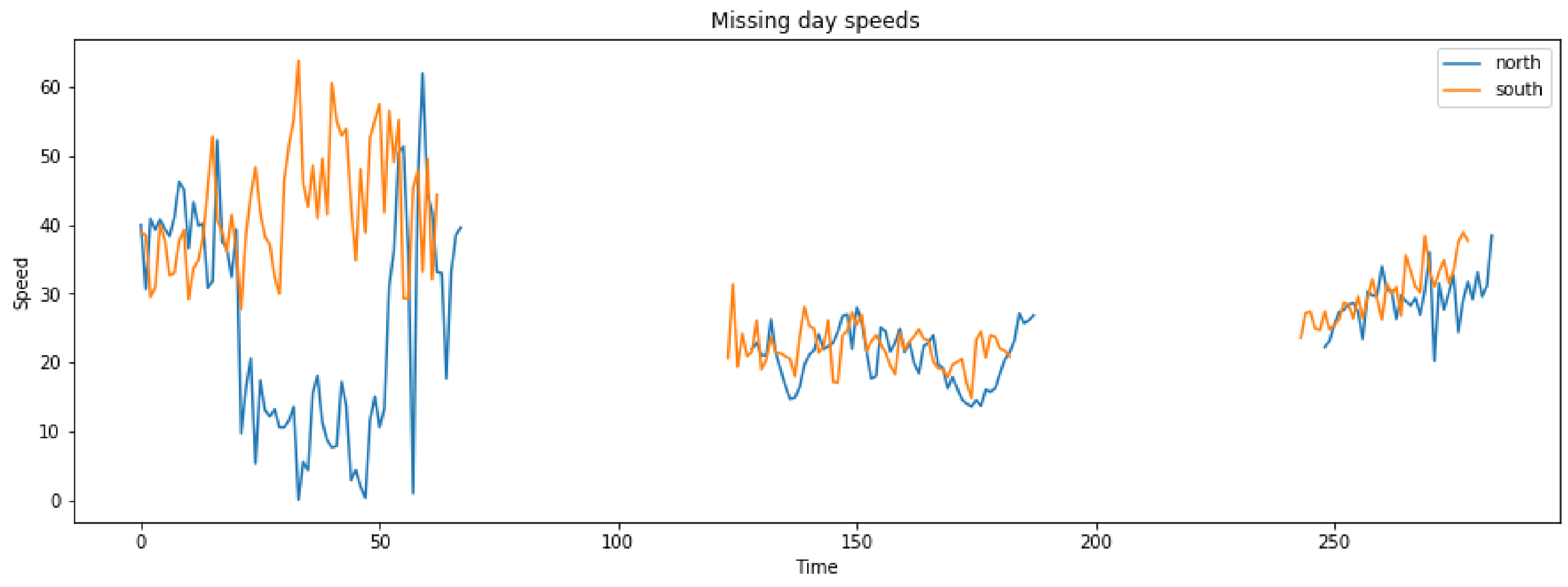
Data

"Missing Day" Speed Data

We were provided two .csv files consisting of speed estimates of the northbound and southbound Second Ring Road, every 5 minutes, for one day (1st December 2016). The speed estimates were missing estimates from 6am-11am and from 4pm-9pm. The times with no speed estimate are the times we are responsible for estimates speeds, which is why they are marked with "x" below.



	A	B
1	time	speed
63	5:15:05 AM	41.8135717
64	5:20:05 AM	33.165906
65	5:25:05 AM	32.9551615
66	5:30:05 AM	17.6269517
67	5:45:05 AM	33.2320038
68	5:50:05 AM	38.5166879
69	5:55:05 AM	39.5888428
70	6:00:05 AM	x
71	6:05:05 AM	x
72	6:10:05 AM	x
73	6:15:05 AM	x
74	6:20:05 AM	x
75	6:25:05 AM	x
76	6:30:05 AM	x
77	6:35:05 AM	x
78	6:40:05 AM	x
79	6:45:05 AM	x
80	6:50:05 AM	x
81	6:55:05 AM	x
82	7:00:05 AM	x



The figure above shows the north bound and south bound speeds for the missing (extra) day file. The missing speeds in the middle of the trend is what is supposed to be predicted.

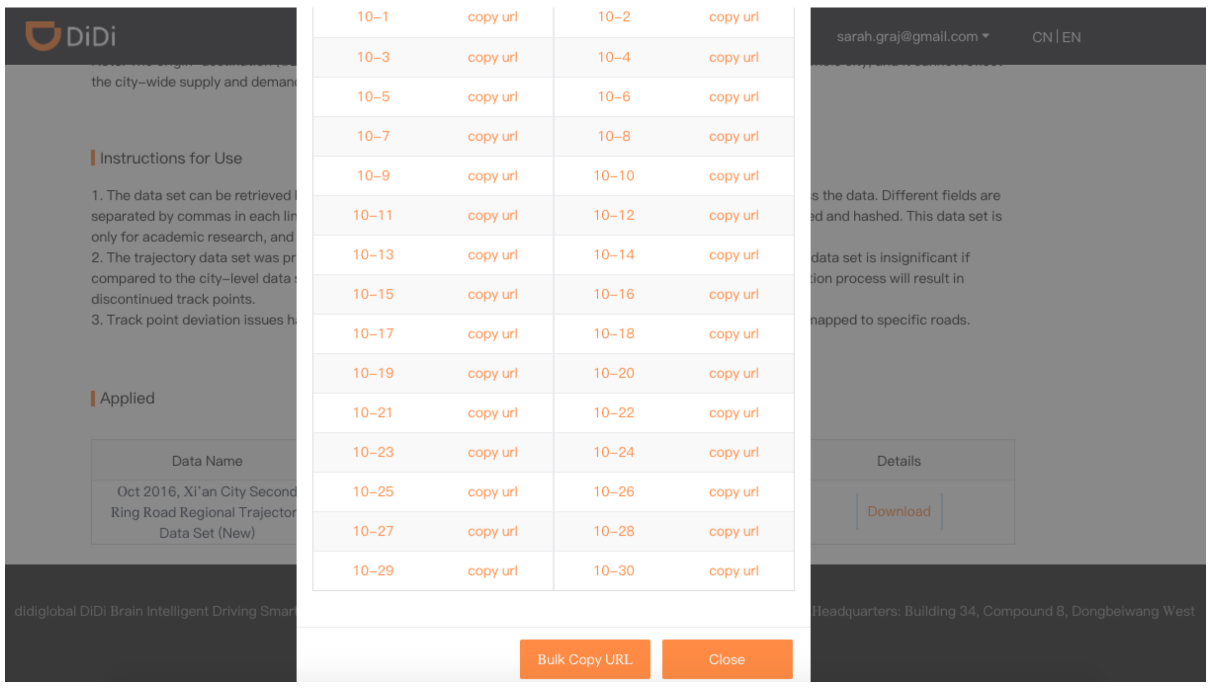
DiDi Data

The data was provided by DiDi Chuxing Technology company. The data consists of all DiDi rideshare trips within a designated bounding box that surrounds the road of interest, the Second Ring Road, in Xian, China. Essentually we have information on all trips within the bounding box for the period October 1st-November 30th 2016 (except 31st october).

The DiDi data consists of the following, every 2-4 seconds:

- Driver identifier (string)
- Order identifier (string)
- Time stamp (string)
- Driver latitude and longitude (string)

Overall, the dataset consisted of 60 days of data, which in total was around 157 GB. Because this dataset is confidential, we first needed to request access, then download each day manually from the DiDi Gaia Initiative website, as can be seen in the image below, for all days in October and November 2016.



DiDi sarah.graj@gmail.com CN | EN

the city-wide supply and deman

Instructions for Use

1. The data set can be retrieved | s the data. Different fields are
separated by commas in each lin | d and hashed. This data set is
only for academic research, and
2. The trajectory data set was pr | data set is insignificant if
compared to the city-level data | ion process will result in
discontinued track points.
3. Track point deviation issues h | napped to specific roads.

Applied

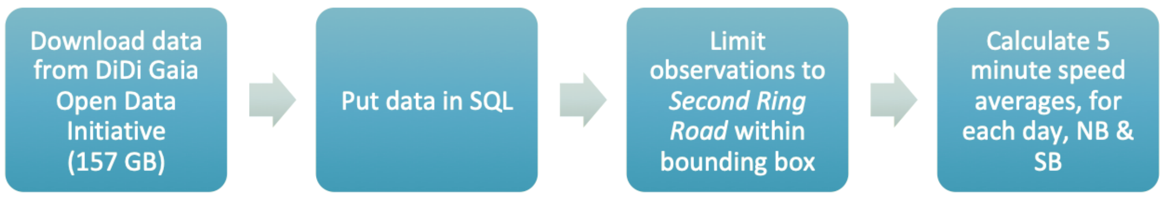
Data Name	Details
Oct 2016, Xi'an City Second Ring Road Regional Trajector Data Set (New)	Download

10-1	copy url	10-2	copy url
10-3	copy url	10-4	copy url
10-5	copy url	10-6	copy url
10-7	copy url	10-8	copy url
10-9	copy url	10-10	copy url
10-11	copy url	10-12	copy url
10-13	copy url	10-14	copy url
10-15	copy url	10-16	copy url
10-17	copy url	10-18	copy url
10-19	copy url	10-20	copy url
10-21	copy url	10-22	copy url
10-23	copy url	10-24	copy url
10-25	copy url	10-26	copy url
10-27	copy url	10-28	copy url
10-29	copy url	10-30	copy url

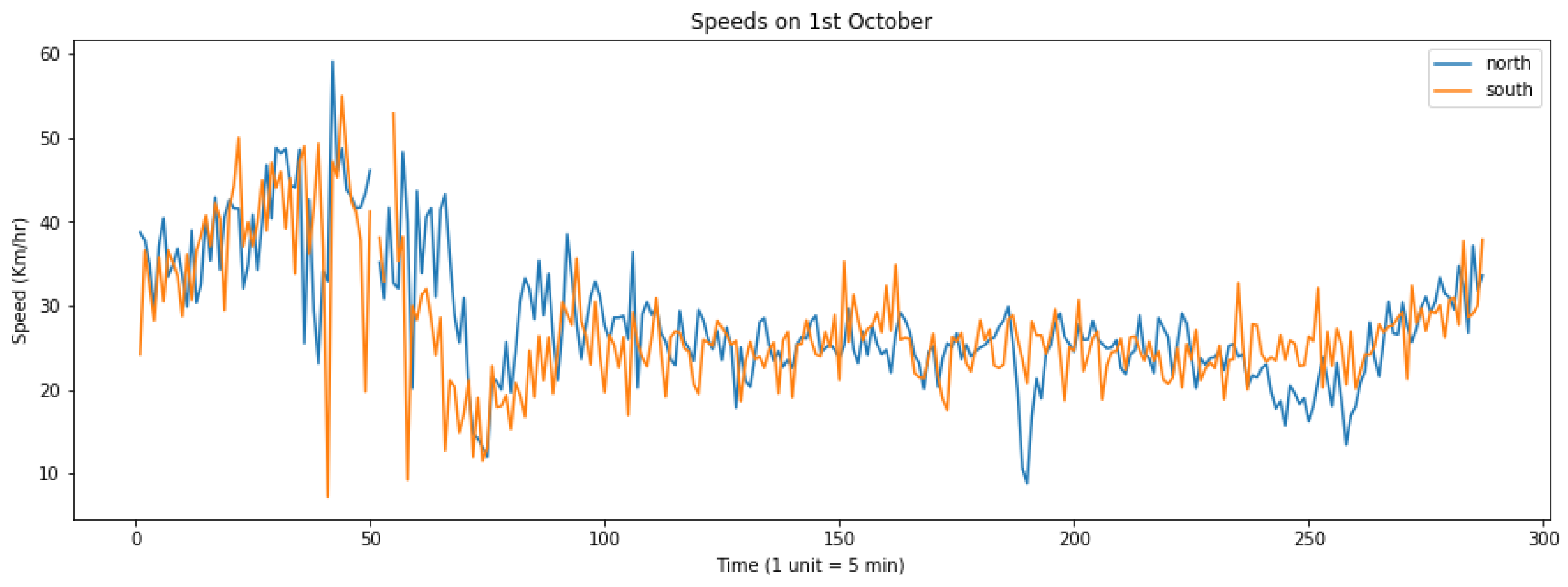
Bulk Copy URL Close

didiglobal DiDi Brain Intelligent Driving Smar | Headquarters: Building 34, Compound 8, Dongbeiwang West

In order to analyze this large amount of data, we decided to put the data into SQL, where it would be easier to process. The general sequence of steps we followed can be seen below:


Download data from DiDi Gaia Open Data Initiative (157 GB)
Put data in SQL
Limit observations to *Second Ring Road* within bounding box
Calculate 5 minute speed averages, for each day, NB & SB

After putting the DiDi data into SQL, we next wanted to look at only drivers on the Second Ring Road. To do this, we created a smaller bounding box surrounding this road, and selected all DiDi trips whose geo-coordinates fell within this box. Lastly, in order to see speed trends for other days, we calculated the speeds for the northbound and southbound segments of the road for all days.



As you can see in the above figure that the 5 min averages are very noisy with some missing values (very few though) in between. Also, there seems to be high speeds during the night time and then speed seems to saturate as the morning strikes.

Methodology

After fetching the data from the source, sending it to sqlite database and calculating 5 min average speeds for drivers on the north/south bound segment, we proceed with the problem with two approaches:

- Approach 1) Short Term Speed Prediction - Use only the past (lagged) observations to predict some future steps. In this case, we would need to tune the number of lagged observations to be used (we will call it Window) and the number of future steps (pred_step) to be predicted. One thing must be kept in mind while using this approach is that, ultimately since we have to predict 60 continuous speeds in the missing day file, we would have to use some predicted speeds in order to predict future steps that could cause the error to accumulate.
- Approach 2) Long Term Speed Prediction - Use both past and future values to predict the intermediate values. The only thing to be kept in mind for this apprach is that we have to predict 60 continuous time steps at once, which seems unreasonable for now.

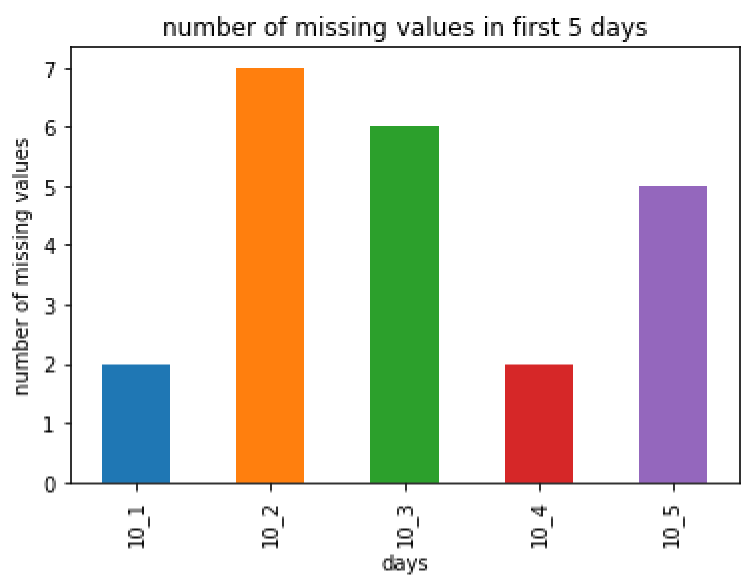
For both of these approaches, we will use state of art deep learning models as the current literature clearly state that these methods have outperformed the conventional time series prediction approaches or machine learning methods for travel speed prediction (Ref. 1, 2, 3). We will explain the architecture of our models in the upcoming sections.

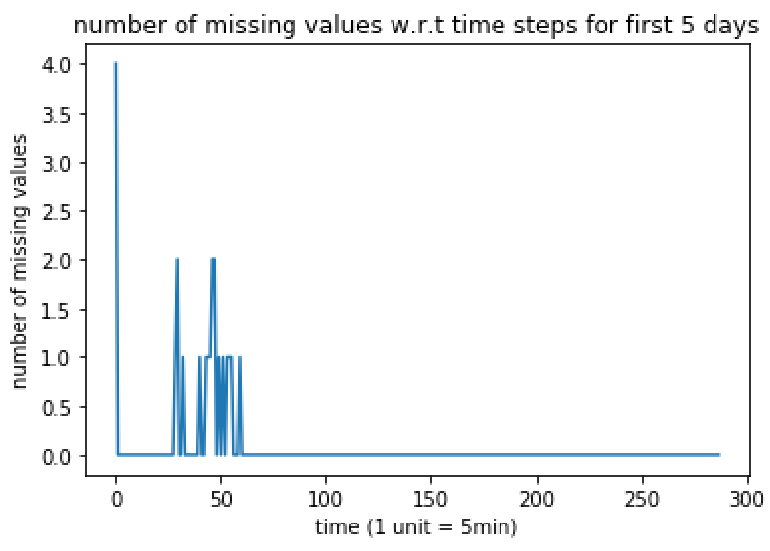
Feature engineering

Dealing with missing values and scale

Let's have a look at the statistics of the first 5 days and the missing values they have.

	10_1	10_2	10_3	10_4	10_5
count	286.000000	281.000000	282.000000	286.000000	283.000000
mean	28.214103	28.707581	28.856404	28.552659	28.709572
std	7.776892	9.496469	8.613030	9.489870	9.594138
min	8.810108	3.291958	0.000000	0.000000	11.154640
25%	23.744643	23.643642	23.722631	23.091210	21.885123
50%	26.264222	26.500049	26.679505	26.234914	25.740473
75%	31.348848	31.710291	32.525729	32.523210	33.139427
max	59.140877	66.846090	60.564791	69.219648	70.376037





As one can see that most of the values are missing at 12 am (4 out of 5).

1) For deep learning models to work, the values of the predictors are normally scaled between [0,1] or standardized. Literature review suggested the former method. Hence, we divide the speeds by 80 to scale most of the values between [0,1] without disturbing the variance structure of the time series. Similarily, the flow features will be scaled to [0,1] for both the directions using appropriate scaling value (for north bound flow; div_n = 63, for south bound flows; div_s = 48)

2) To deal with the missing speed values, which occur very few times (apart from the speeds at 12 am), we decided to impute them by the mean of speeds on that particular day.

Features

This section will be divided w.r.t the 2 approaches defined above.

- ***For approach 1***: The only features we use for this approach are the lagged speeds. One must observe that as we would have to use some of the predicted speeds to infer future speeds for the extra day file (assuming we don't predict all 60 steps at once for this approach). We can not use other features for those predictions as we have no data for the missing hours.
- ***For approach 2***: Here as we use both future and past speeds to predict all the intermediate speeds, we use 2 different types of features, speeds and flow. Flow is defined as the number of vehicles on that particular road segment per 5 minutes. Remember for this approach we will have 2 windows, first for the past (window 1) and second for the future (window 2).

Now to create features, we use the sliding window concept. Let's say we have 100 values in time series. What we mean by sliding window is that, first we use first 50 (1-50) values to predict 51st (assuming approach 1, window = 50), then we slide the window by 1 and next use second 50 (2-51) values to predict 52nd value. As one can visualize the speeds/flow over 2 months as one long time series, we used sliding window method to create different observations.

Also, to distinguish between north and south trajactories, longitude value of the driver is used. That is, if the longitude value of the driver is greater than 108.9468 (gcj coordinate system), then it is assumed to be in the north bound segment, otherwise south bound.

Splitting into training, validation and testing sets

As per the basic machine learning approch, we divide our data into training, validation and testing sets. We use 28th and 29th November as validation set, 30th November as testing set, and rest of the days as training set. Thus, we train our models on the training set, and use validation set for tuning the hyperparametes. Finally, test set will be use as the ultimate set to describe which model is the most superior one. Also, the respective sets (training, validation and test) for north and south bound directions are concatenated to form bigger sets.

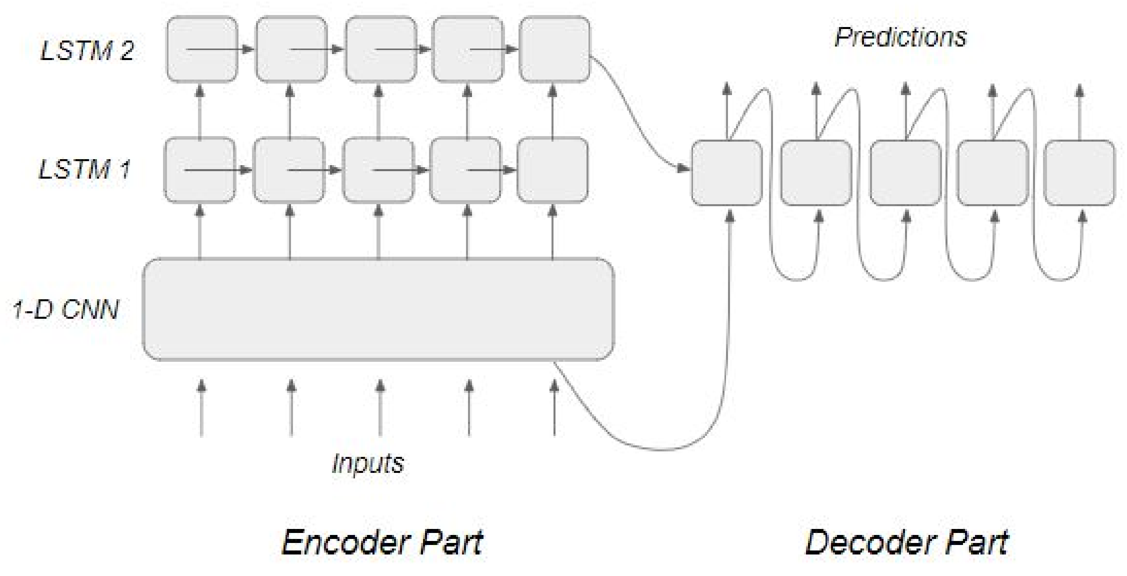
Model architecture

For both the approaches, the starting point is to use 1-D Convolutional Neural Networks (CNN) to extract spatial features from the time series data and then pass these features to the Long Short Term Memory (LSTM) cells for sequence learning. A brief overview of these layers are as follows:

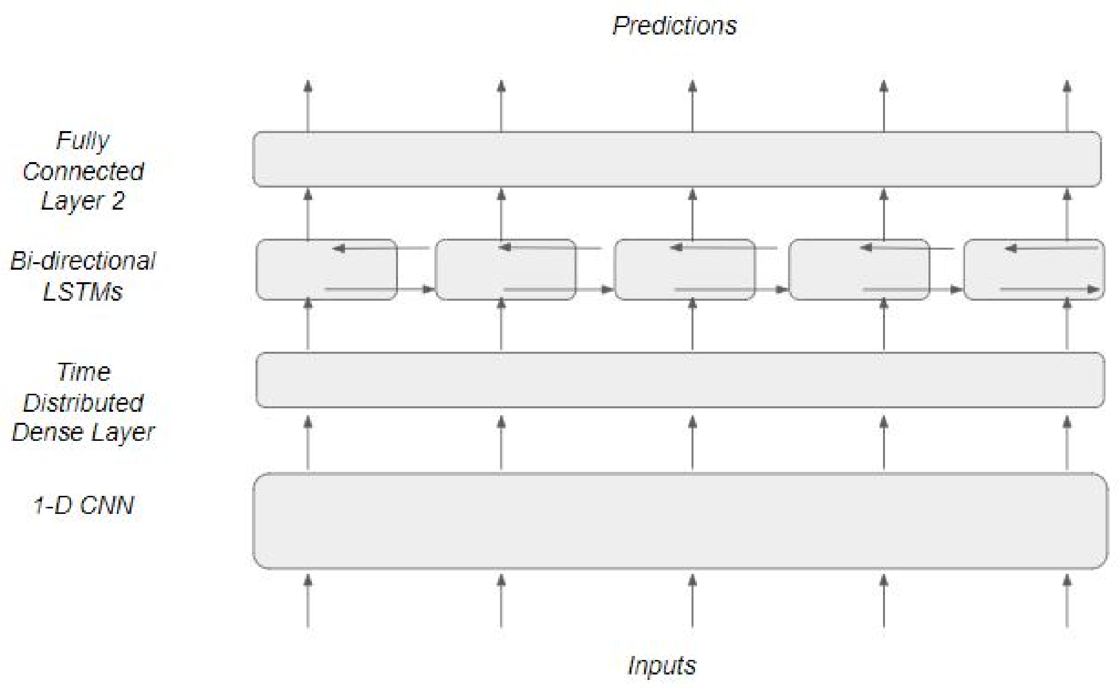
- CNNs - CNN is a feed-forward neural network. Its artificial neurons can respond to the part of the coverage area. CNN has good image processing performance, and some researchers also used it for time series analysis. A classical CNN has three cascaded layers (e.g., convolutional, activation and pooling layers). Due to the time shift and periodicity of traffic flow data, we use one-

dimensional CNN to carry out the sequence local trend learning, which extracts the local trend features by convolution operations of CNN
- LSTMs - LSTM is a special Recurrent Neural Network (RNN) module developed in tackling the knowing gradient vanishing issue for RNN. In contrast to a single layer of a standard RNN, the LSTM is capable of learning long-terms dependencies across samples by employing a number of gates to control the information generated to flow through designed memory cells

Now, for approach 1, After the CNN layer, we employ an encoder-decoder module. Basically, the encoder part of the module, takes the data as input, encodes it and sends the resulting states to the decoder part. The decoder part then predicts the time steps one by one. The figure depicting this structure is shown below. Note that each CNN cell is followed by batch normalization and activations (ReLu) and ultimately a max pool layer is applied with stride and pool size 2. A Dropout layer (0.2) is followed after the LSTM 2 layer in the encoder part.



For approach 2, After the CNN layer, we employ a bi-directional layer that works on the information from the past as well as the future and produces inferences of the intermediate steps. Again, the figure below depicts this architecture. Again, each CNN cell is followed by batch normalization and activations (ReLu) and ultimately a max pool layer is applied with stride and pool size 2. A Dropout layer (0.2) is followed after the time distributed dense layer and bi-directional LSTM.



Model Training

*The deep learning library, **Keras** that works on tensor flow backend was used to code the above mentioned architectures. As deep learning models have a lot of trainable parameters, the use of GPU becomes essential to save time. In this project, we use the GPUs provided by the **Google Colabs** to train our deep learning models.*

From here, we will discuss the specifics of training for both the approaches separately.

Approach 1

Here are the specifics of the parameters used:

- Window, pred_step: Window and pred_step length used for this approach are 72 and 2, respectively, i.e., we use the past 72 values to predict the next 2 values. These values are chosen keeping in mind the amount of information captured in the lagged values (more lags, more information till certain extent) and amount of time for training (more lags, more features, more training time).
- Optimizer, learning rate, batch size: Adam optimizer with 0.0001 learning rate and 128 batch size was chosen.
- LSTM layer had 256 cells, and 1-D CNN consisted of 6 dilated layers, each with 128 filters and 2 kernel size.
- The training, testing and validation sets were reshaped to 3 dimensions. First dimension represents number of samples, second represents number of time steps and last dimension depicts number of features. Thus, Training, validation and test sets have (32686, 72, 1), (504, 72, 1), and (216, 72, 1) dimensions, respectively
- Similarly, training, validation and test output values have (32686, 2, 1), (504, 2, 1), and (216, 2, 1) dimensions, respectively

Approach 2

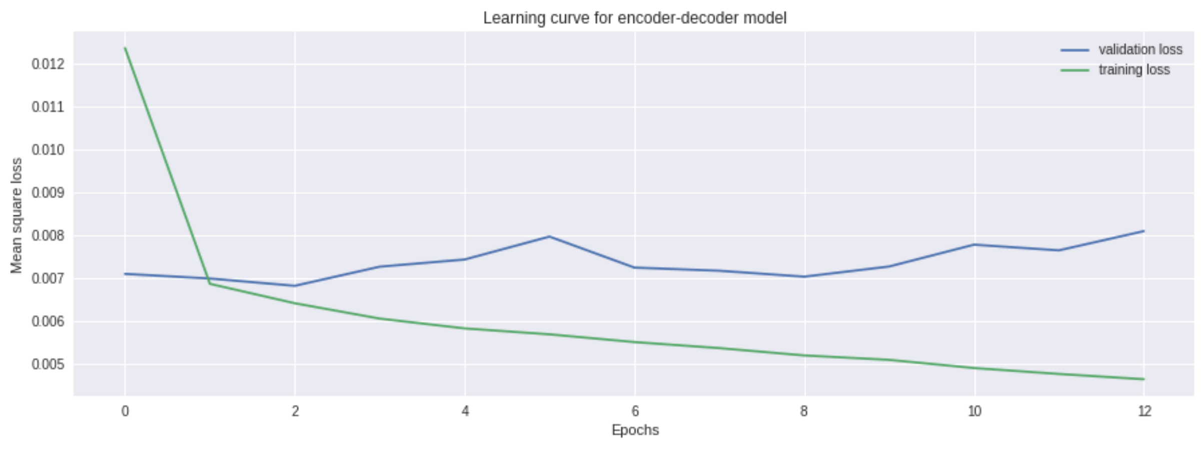
Here are the specifics of the parameters used:

- Window1, Window2 and pred_step: As we use both past and future values to predict speeds, the length of these parameters is decided by the extra day file. Both the missing segments have a size of 60 (pred_step). The maximum length of the past window could be 60 (window1) and for the future window could be 36 (window2). Hence, we don't tune these parameters here.
- Optimizer, learning rate and batch size: Adam optimizer with 0.0001 learning rate and 128 batch size if used
- LSTM layer has 256 cells, 1-D CNN consisted of 6 dilated layers, each with 128 filters and 3 kernel size, time distributed dense layer had 128 neurons and fully connected dense layer had 60 (pred_step) neurons.
- Training, validation and test sets have (32522, 96, 2), (16, 96, 2) and (6, 96, 2) dimensions, respectively. Note that now there are 2 features, speeds and flow. Similarly, training, validation and test output values have (32522, 60, 2), (16, 60, 2) and (6, 60, 2) dimensions, respectively.

Results

Approach 1

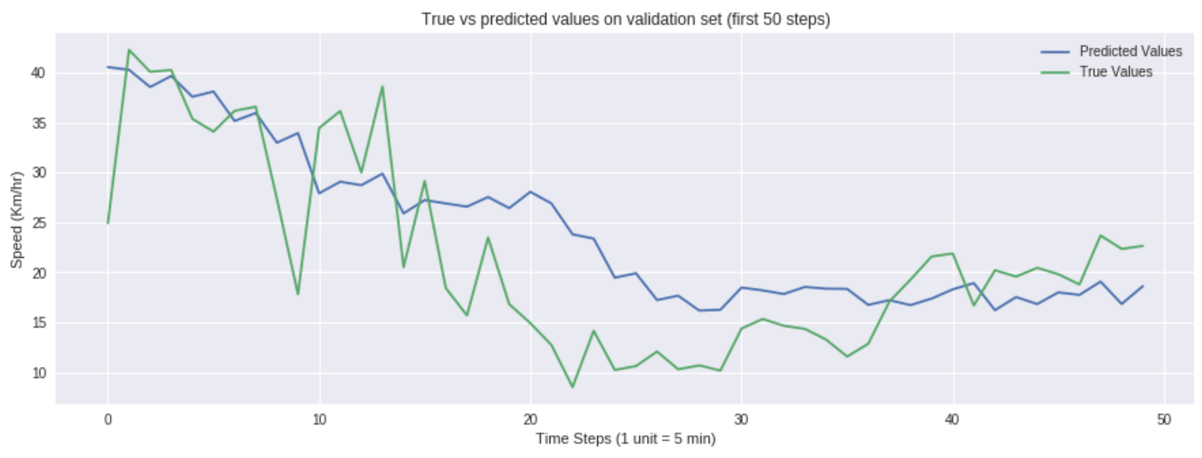
Early stopping was used with patience 10. The following is the learning curve for the encoder decoder model



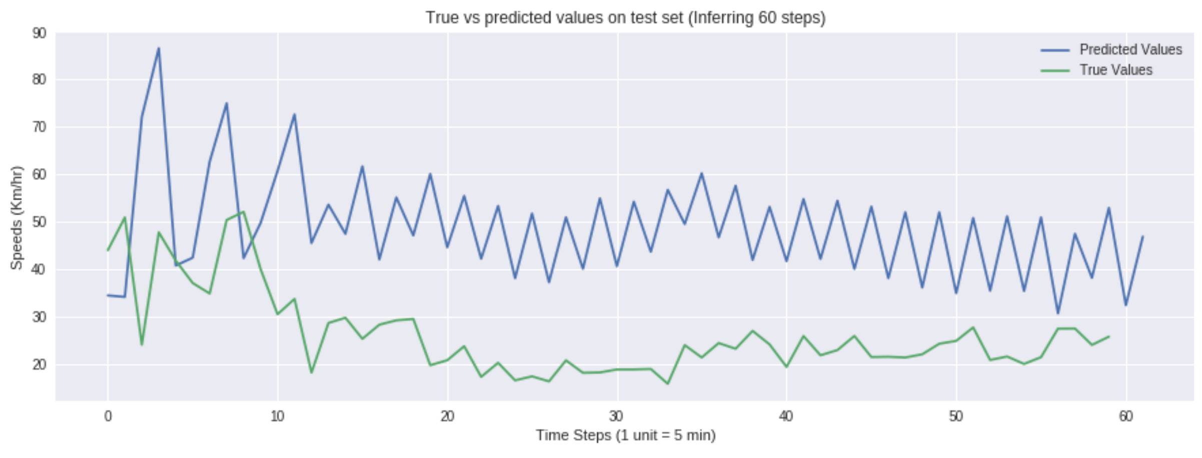
Note that the values on the y axis are of this magnitude because we have scaled our speeds by 80. The validation and test errors are as follows:

Validation loss	Test loss
32.640	19.700

Let's plot the true vs predicted values on the validation set for first 50 time step predictions.



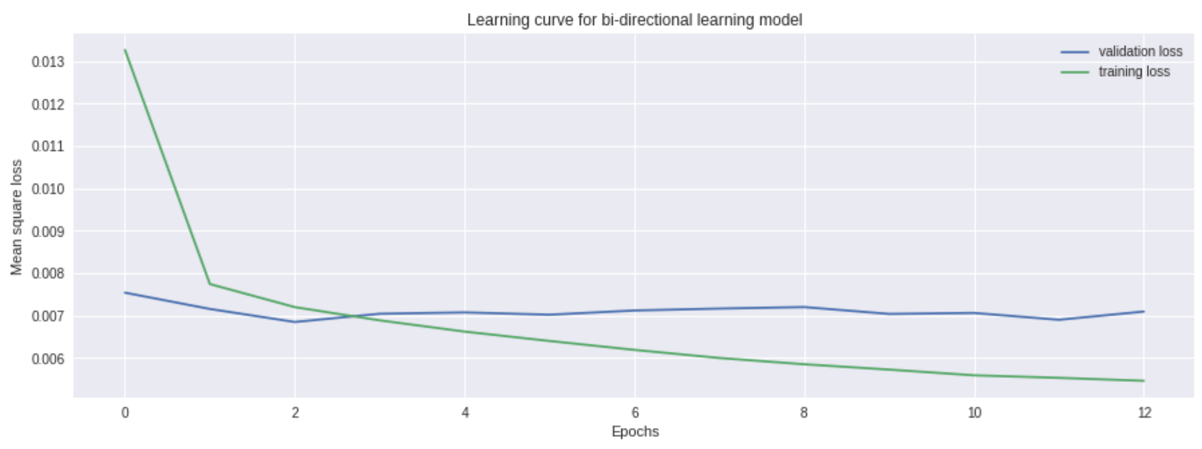
Since, ultimately we want to predict 60 continuous speeds and this model predicts just 2 steps at once, we would need to use the predicted values for further predictions. Let's try inferring 60 values in the test set using this approach. Below is the plot.



As one can see, how this approach fails. Using previous predictions to find the future values result in accumulation of errors and hence, future values are predicted with high degree of error. Thus, in practice, using approach 1 for predicting long sequences is not recommended

Approach 2

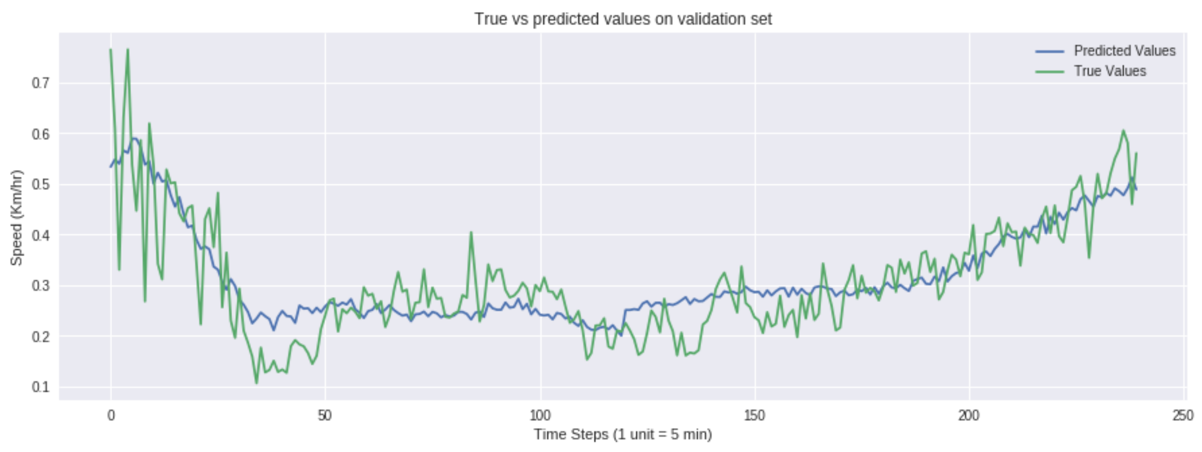
The learning curve for this model is as follows:



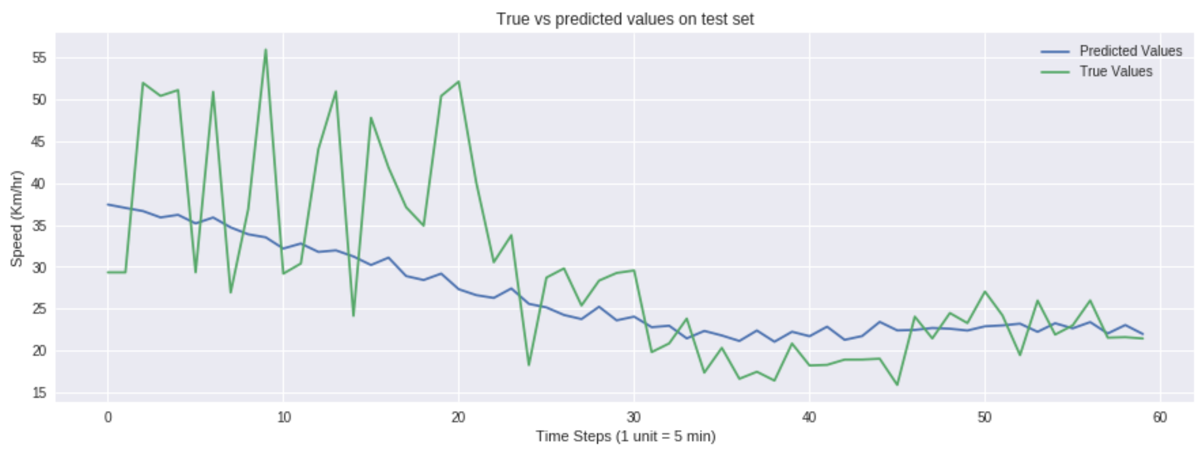
Note that the values on the y axis are of this magnitude because we have scaled our speeds by 80. The validation and test errors are as follows:

Validation loss	Test loss
36.124	27.178

Let's plot the true vs predicted values on the validation set.



Again, the aim is to predict 60 continuous steps, and that is excatly what we are doing here. Thus, plotting the true vs predicted valus on test set:

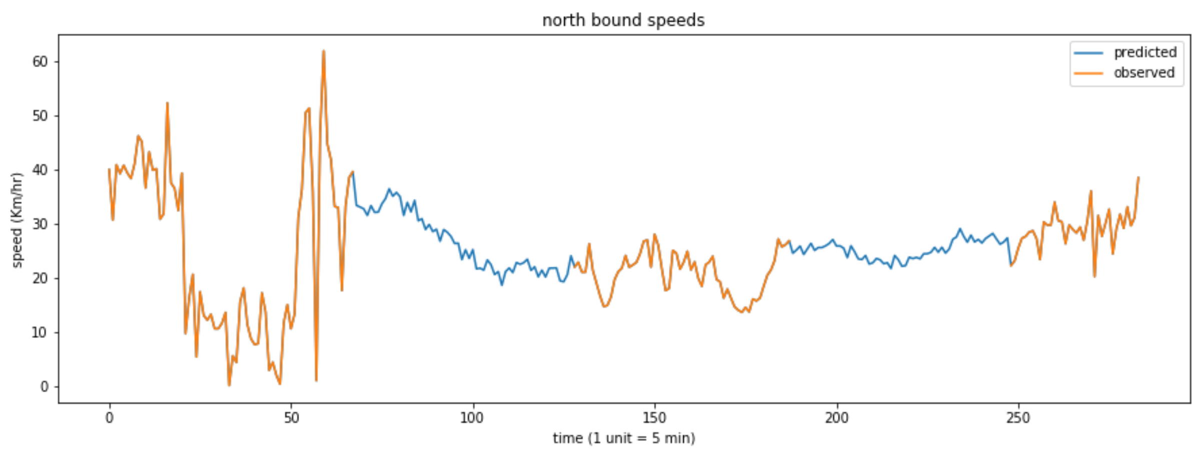


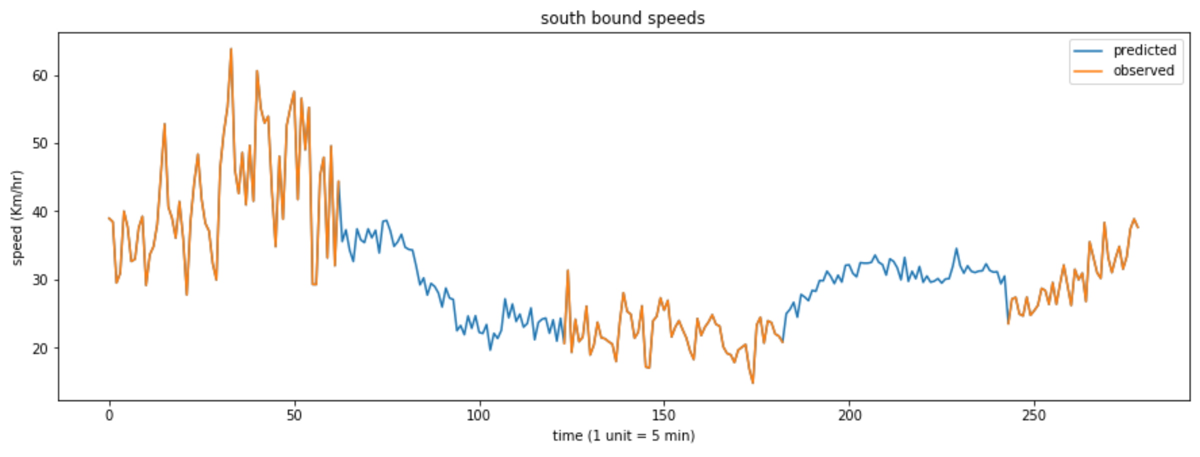
Clearly, the inference on the test set is much better for this approach as compared to the approach 1.

Note that the test sets are diffrent for both the approaches as the lengths of windows are different.

Conclusion

As the results suggest, approach 2 where bi-directional learning was adopted to predict all the intermediate values outperforms the approach1 where only lagged features were used for predicting few future values. Now that we have the model, let us predict the missing extra day speeds by using the approach 2 model.





References

1) https://arxiv.org/ftp/arxiv/papers/1803/1803.02099.pdf (https://arxiv.org/ftp/arxiv/papers/1803/1803.02099.pdf)

2) https://arxiv.org/ftp/arxiv/papers/1809/1809.01887.pdf (https://arxiv.org/ftp/arxiv/papers/1809/1809.01887.pdf)

3) https://arxiv.org/pdf/1604.04527.pdf (https://arxiv.org/pdf/1604.04527.pdf)