FIT5147 VISUALIZATION PROJECT – ARRAY OF THINGS

Swaraj Purohit

ID: 29286174

Table of Contents:

1. Introduction

The aim of this data visualization project is to create an output provisioning for 'Array of Things' data. The 'Array of Things' (AOT) is an open data initiative which is a collaboration effort between American government agencies, researchers and some private firms. The project involves setting up of Internet of Things (IOT) sensors which capture a range of parameters ranging from temperature to carbon monoxide levels in various American cities. The objective of the AOT project is to create an infrastructure which will enable 'fitness tracking' of cities. The IOT engine is powered by software and hardware platform known as Waggle which is developed by Argonne Labs. This project uses data for the city of Chicago ranging from 2nd February 2019 to 10th February 2019.

The objective of this visualization project is to create a dashboard which will serve as a 'fitness tracker' for the city of Chicago. The intended audience would be someone who wants this decision support for urban planning or governance. We aim to provide decision support by showing past trends in various parameters, current status and also by providing forecasting. Pollution checking is a key aspect of this dashboard and we want the user to be alerted when pollutant concentrations are higher than the standards set by the American government.

2. Design

The data required consists of two files – a shapefile folder containing spatial data of Chicago community areas and a final csv file which has aggregated the week's data by node and parameter on an hourly basis. This aggregation along with data cleaning and checking such as finding semantic errors has reduced the data (main file) from roughly 28 million rows to 80 thousand rows. The final csv file contains the following required attributes – timestamp, node id, parameter, community area and hrf (human readable form) value.

The dashboard is designed keeping the end user requirements in mind and is intended to provide decision support to an urban planner or government official. Hence, the narrative we are showing is past trends to paint a picture of what has happened, current status and forecasting so that the user can create advisories. Design methodology for each of these facets are explained using Munzner's framework (what, why, how) below:

- **Past:** We are looking to visualize tabular sensor data to present past trends across various community areas in Chicago. To do this, we have aggregated the data by node and parameter on an hourly basis. We allow the user to aggregate data by community area to get the hourly trend for a parameter by area. The data will be encoded in a 2D line chart with the x axis being the timestamp and y axis being the parameter value. The user can manipulate the chart by selecting the parameter and community area. The user can also aggregate the data over all communities to get the overall trend for a parameter.
- **Present:** We are looking to visualize tabular sensor data by linking it to spatial data to present the current status of various parameters across the city of Chicago. To do this we will select only the data pertaining to the latest timestamp. The data is represented on the map obtained from the shapefiles and is encoded in the colour of each community area (choropleth map). The user can hover over a particular community to get the value for that community.
- **Future:** We are looking to derive insights from the tabular data to create a model which can forecast the parameters. This will give the user predictive capabilities so as to generate advisories. For this purpose, the main data file has been aggregated by all locations, and at a 5-minute interval to create a univariate timeseries for that particular parameter. This is used

to train the model which can forecast accordingly. The forecast will be presented in the form of a line graph and the user can select the parameter by using the parameter filter.

5 Sheet Design Summary:

- Tabs have been provided so that the user can easily navigate between the facets mentioned above.
- For the past trend and forecast we have chosen a 2D line graph as it gives continuity which makes it easier for the user to notice trends.
- Initially we were going to plot the node locations on the map and encode data in the colour of the node. However, we have chosen to go by community area as a whole as it makes this more user friendly and is more apt for the user requirement.
- Our focus has been to alert the user when the pollutant levels exceed the limits set in the government standards. Hence, we have used a permissible limit line wherever required. The line has been coloured red as the user will then intuitively consider it to be a maximum threshold.
- For the present/current panel, we have also included a text box which shows the number denoting the current overall average. For pollutants, this number will be red if it exceeds the limits and green otherwise for the same reasons as stated above.
- For the present/current panel, we allow the user to hover on each community to see the current value for that particular community. This is done as we cannot see the precise value by looking at the colour coded scale.
- Initially we were going to go with a green-yellow-red colour scale but have decide to go with a single-coloured scale. This is because the coloured scale might be misleading. For example, the user may think that communities with colour red are hot (when parameter is temperature) however, it is February and the temperature is not more than 12 °C. Also, the green-yellow-red scale does not make sense for parameters like humidity or SO2 concentration since, the colour red need not mean that the user should be alerted since the scale is relative. Hence, we have chosen the Yellow-Orange-Brown scale in colour brewer which is both sequential and colour blind friendly. The scale was inverted to encode a greater number with a darker shade as this is more intuitive since we perceive light shade on map as sparse.
- Note that the graph titles, labels and legends change according to the filter selected by the user.
- Ideally forecasting should have been done for at least two parameters – SO2 and CO. However, we could get a viable forecasting model only for the SO2 time series. Hence, we have forecasted only SO2. This is useful because we have found that Chicago has very high levels of SO2 in our exploratory analysis hence, it would be beneficial to get a forecast for this parameter.
- To summarise, the final design has been created keeping in mind user requirements, ease of navigation, pollution level alerting and maximizing data to ink ratio to keep a minimalistic and effective interactive visualization.

3. Implementation:

Platform Used: R Studio, 64bit.

Requirements: Anaconda 3 5.3.1, 64 bit.

Libraries Used:

- *ggplot2*: For plotting.
- *magrittr:* For piping operation.
- *dplyr:* For data wrangling functions such as *group_by.*
- *maptools:* To load and plot the shapefile.
- *shiny:* To create the dashboard.
- *rgeos:* As a dependency to ggmap library.
- *ggmap:* For the *theme_nothing()* function. API key not required.
- *mapproj*: Used for creating the map.
- *RColorBrewer:* Used to define the palette for the choropleth map.
- *scales:* Used to create the gradient scale for the choropleth map.
- *plotly:* To make the visualizations more interactive.
- *tensorflow*: Used to create forecasting model. This library is installed from the official GitHub repository in an anaconda virtual environment and therefore anaconda software is required.

Implementation Process and Considerations:
The dashboard is implemented with an R shiny application which runs on the Shiny server. The broad implementation methodology can be seen in the diagram below.

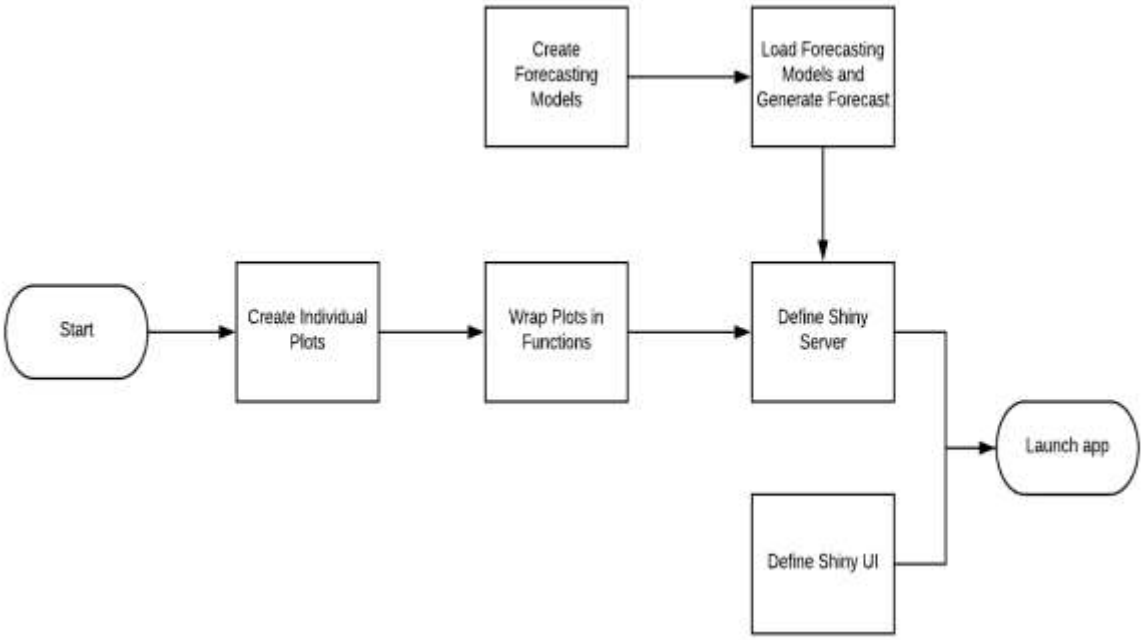


Fig1. Implementation Process

The process ensures that unit testing and integration testing occur so as to ensure robust design and no bugs. Note that there are three R scripts - Firstly, the plots were created by randomly initializing

the parameters. This ensured that the structure of each plot was well defined. The plots were then wrapped into functions where the functions are inputted the parameters which were initialized before. Three functions were created, one for the plot on each tab. The functions take in inputs which the user will feed through the Shiny app and apply appropriate logic to deliver the right plot. For instance, only the pollutant concentration line graphs will have a permissible limit indicator to alert the user of high levels of pollution.

Some key considerations for the Shiny application are that the sidebar input panels are different for each tab panel. Hence, everything is nested in the *tabsetPanel* of the Shiny UI. Also, for the past trend tab, there are two filters - one for the community area and other for the parameter. Note that a parameter will not be recorded in each community. Hence, the drop-down options for the community area filter depend on the parameter chosen. If parameter is changed but it does not have a value for the selected area, then the area defaults to overall.

Creation of the forecasting model was done using *tensorflow* library. The problem was boiled down to a univariate timeseries forecasting problem. Hence, recurrent neural networks (RNN) were the best way to go as they perform sequential modelling where the probability of the next value in the sequence depends on the previous sequence. Initially *keras* library was going to be used due to its ease of use. However, that turned out to be the problem as the high-level API would not deliver the required implementation. The model creation process is as follows:

- Try with two data sets – one with the actual time series and other with the differences between each timestamp. Surprisingly, the former gave better results.
- The values are scaled.
- The last 50 are removed and make the test set.
- Both GRU (gated recurrent unit) and LSTM (long short-term memory) cells were tried with LSTM giving better results. The cell layer had to be added to overcome the vanishing gradient problem associated with RNNs.
- The model essentially takes in a random sequence of length equal to the defined timestep and learns to predict the next value by learning the various weights using *RMSprop* optimizer. This is repeated for 'n' iterations to train the model.
- The model predicts forward by starting with a seed – the latest timestep. It uses this to predict the next value and then appends that to the seed. The new latest timestep is now used and this process is carried out for 50 iterations so we can get the forecast for the next 250 minutes (since, the timeseries is at intervals of 5 mins). This is then compared with the test set to get an idea of model performance.
- The best result found was given by the hyperparameters given in the table below. Reducing the learning rate greatly boosted model performance.

Hyper Parameter	**Value**
Time step	75
Number of Neurons	200
Learning Rate	0.0001
Training Iterations	100,000
Batch Size	1

- The performance of the model on the test set can be seen in the chart below:

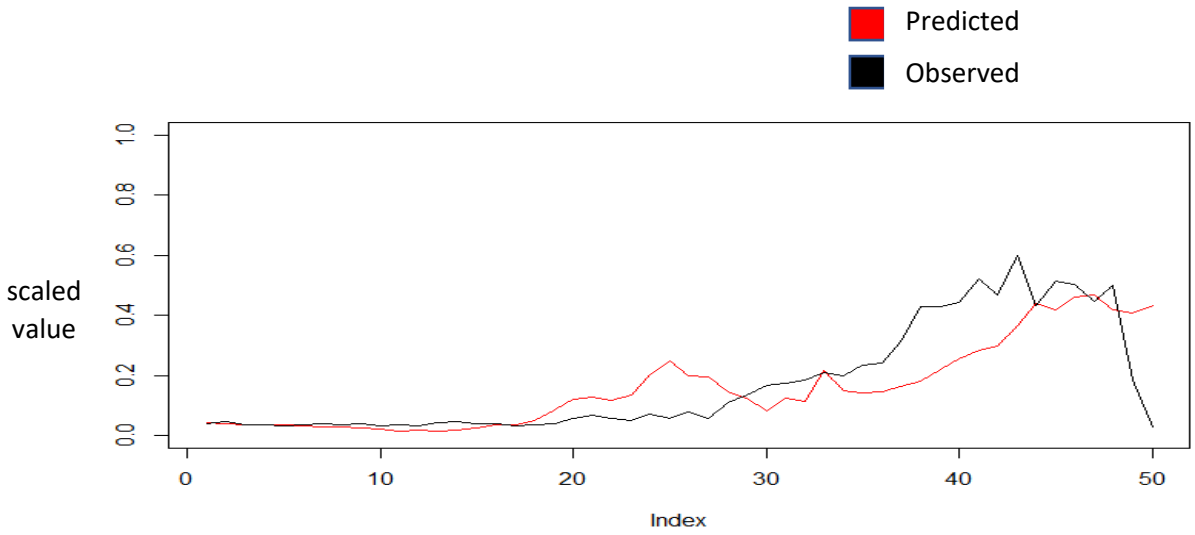


Fig2. Model Performance on test set

- Unfortunately, other pollutant parameters could not give viable results as the data was not rich enough either for training or for testing. For instance, in the time series graph below for O3, we can see that the pattern in indices before 1400 is radically different from that after. Hence, the model won't get trained on the pattern required to predict the coming future. For the CO plot, we can see that there are about three spikes. Our model won't be able to predict the spikes as we have only 3 data points (1 is probably an outlier) and will give mostly constant results. The SO2 time series had enough patterns and a decent prediction compared to the test set.

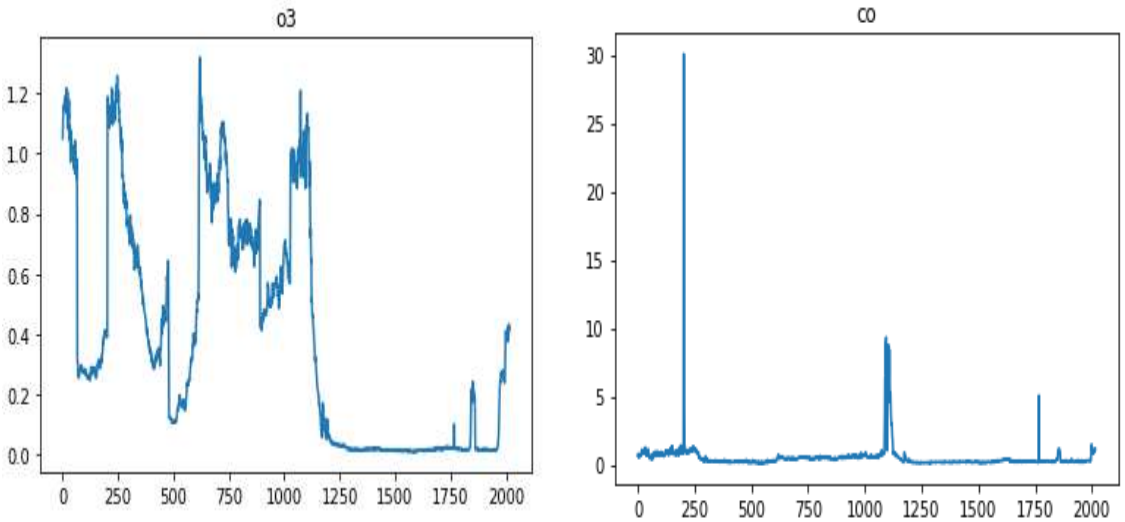


Fig3. Timeseries Distribution for O3(left), CO(right)

- The trained model is saved as checkpoint files so that it can be loaded into the Shiny application.

4. User Guide

This section will discuss the dashboard functionalities, features and give a user walkthrough. Note that there are three tabs which can be easily accessed by clicking on the tab in the top left corner as seen in the figure below. The figure below showcases the past trends tab. The user has two drop down filters as seen in the left side panel – one to select the parameter and other for selecting the community area. In the diagram, the user has selected 'OVERALL' which is giving the overall aggregated result.

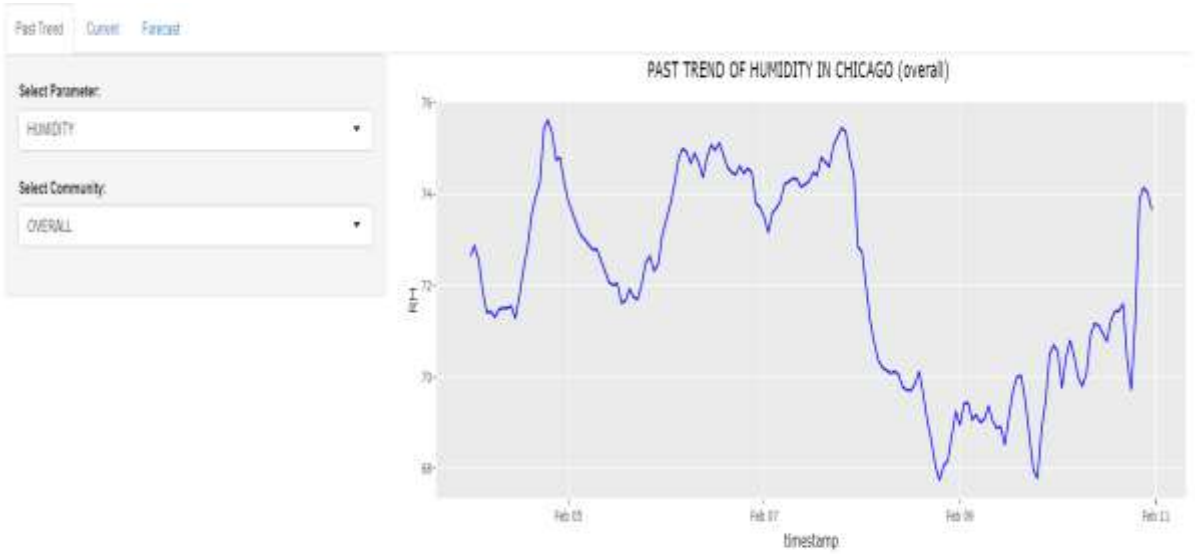


Fig4. Past Trend Tab - View1

The figure below shows the past trend tab when the user has selected 0_3 parameter and community as Uptown. We can see that for pollutant parameters, the plot shows the limit prescribed by the American government standards in the form of a dashed red line. We can also see that the user is able to see the exact timestamp and value where the mouse is hovered.

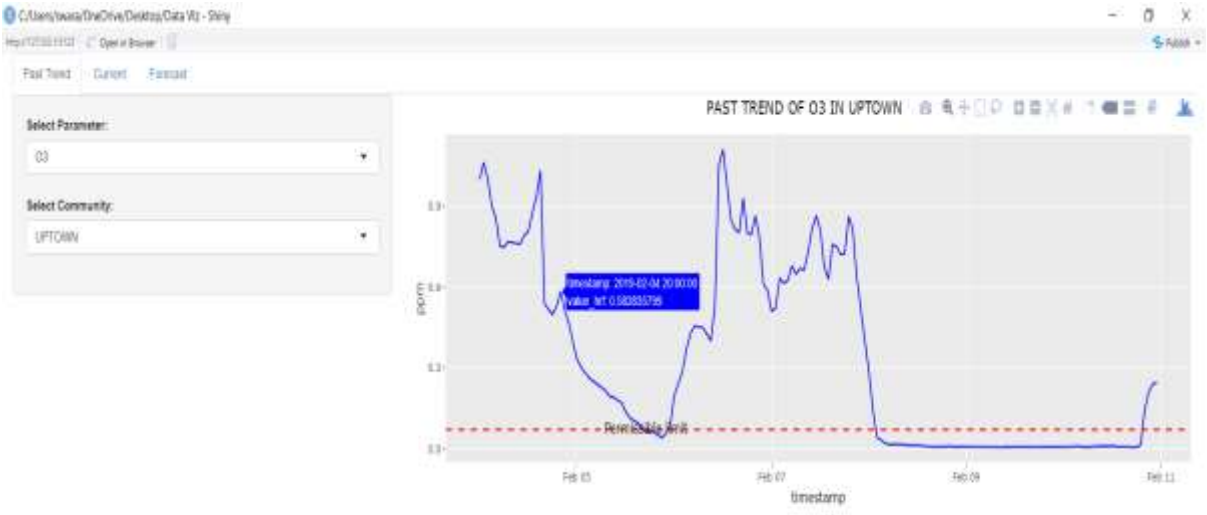


Fig5. Past Trend Tab – View2

The next tab is the current tab which gives the current spatial distribution of the parameter selected via the dropdown filter. Once the parameter is selected, we can see that overall current average just below the filter on the left sidebar. The user can hover on a particular community on the map to see the exact value as shown in figure 6. Note that the name of each community is shown in the map. If the user hovers over the text, he/she will get a text box as shown in figure 7 which will show the coordinates and community name. Hence, to see the value, the user must hover over the community region but not on the text.

Figure 7 also shows that when a pollutant parameter is selected, the average is shown in red indicating that the average exceeds permissible levels. The average would be shown in green if the levels are within the limit. The grey coloured communities indicate that no readings have been taken. For instance, if the parameter is changed to car or person total, there will be no data shown on the map as readings have not been recorded by that sensor at the latest timestamp.

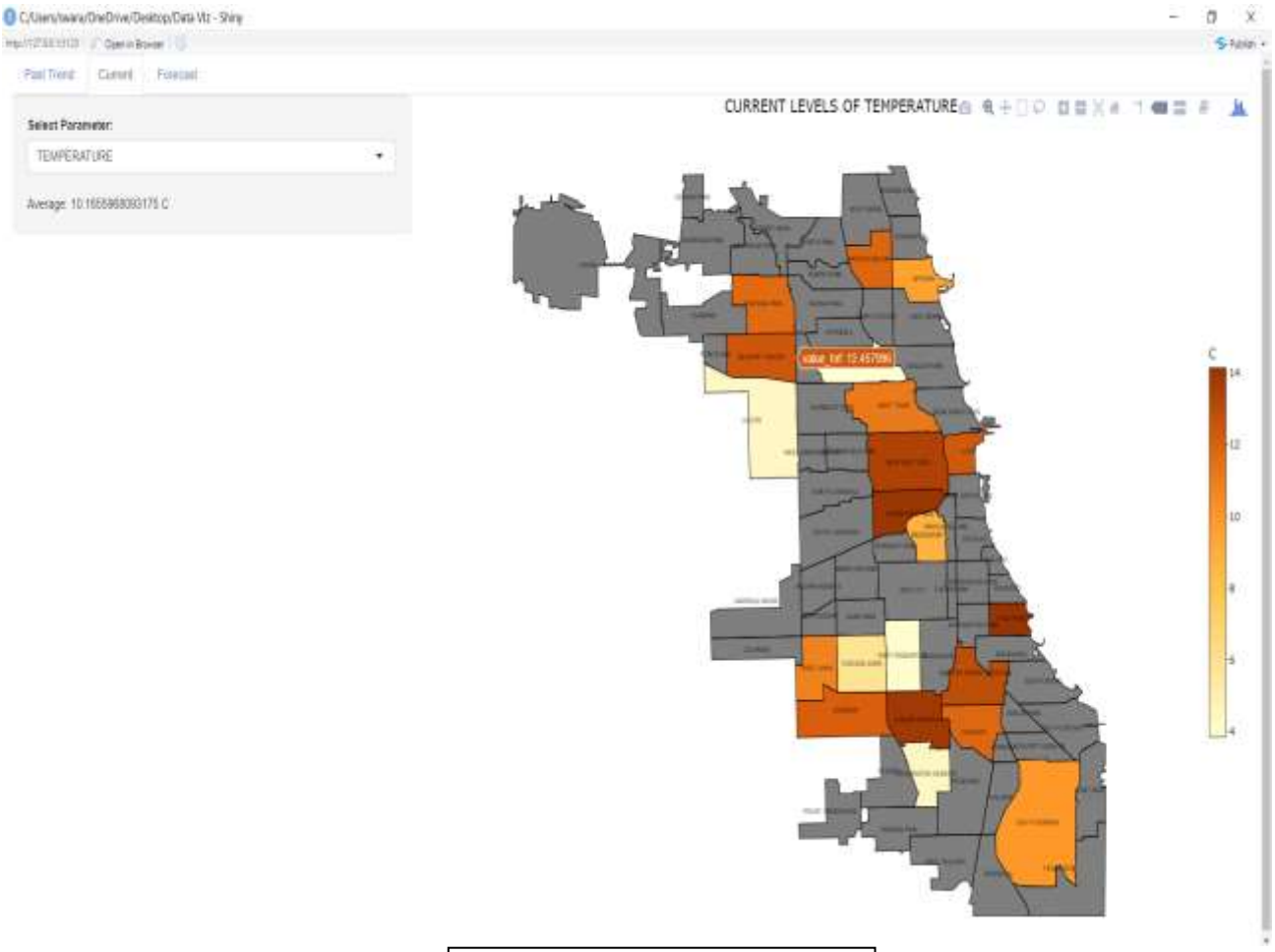


Fig6. Current Tab – View1

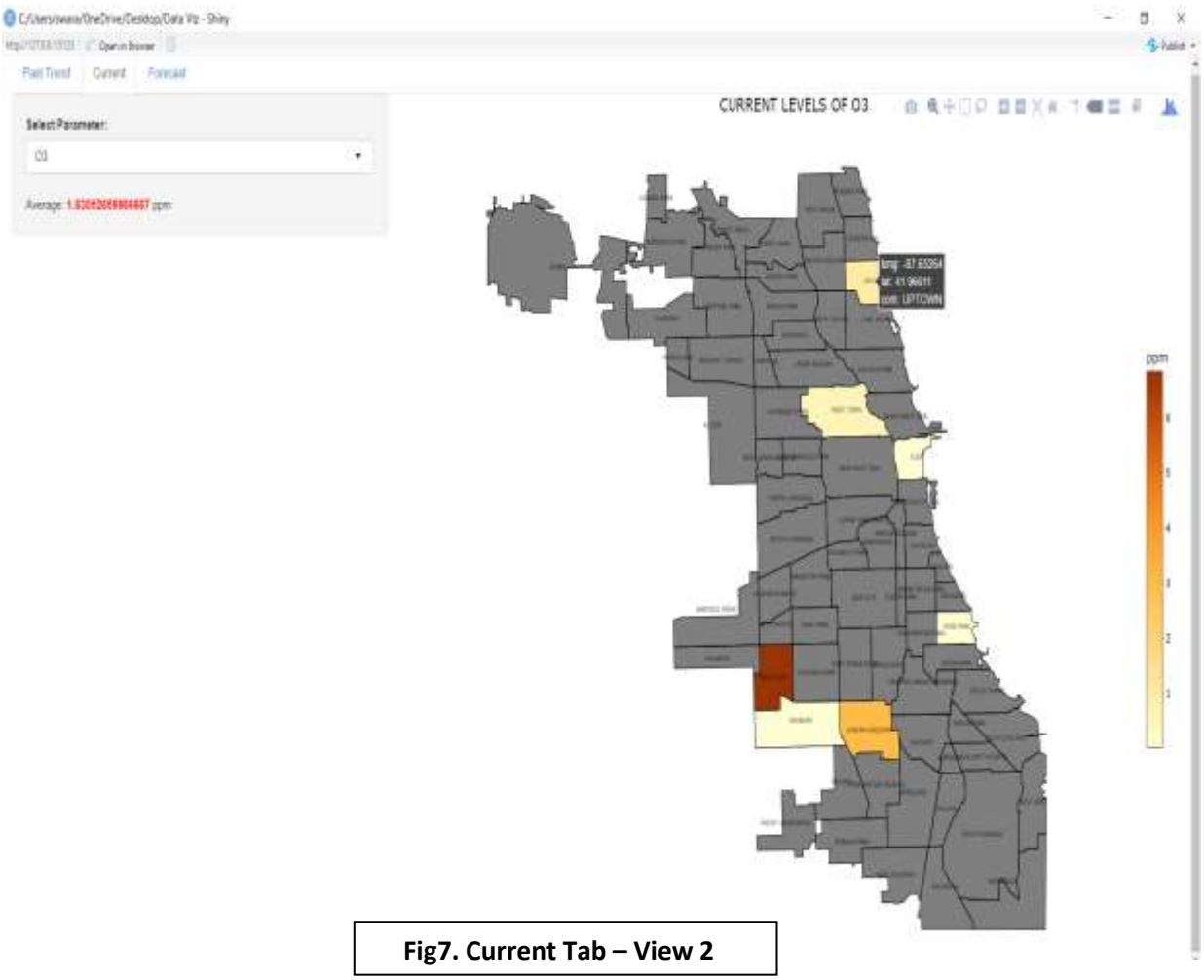


Fig7. Current Tab – View 2

The final tab is the forecast tab. This shows the forecast generated by the model. Although we are currently able to forecast only for SO_2, the sidebar filter is present to give an idea of how it would ideally work. To indicate the permissible limit the dashed red line is also plotted on the graph. The prediction is done for the next 24 hours as there wasn't enough data to test the model over a longer period. The tab is shown in figure 8 given below. Note that each tab has a panel in the top right corner as seen in figure 8 which allows the user basic functionalities such as selection, zooming, panning and downloading the plot.

Fig8. Forecast Tab

5. Conclusion

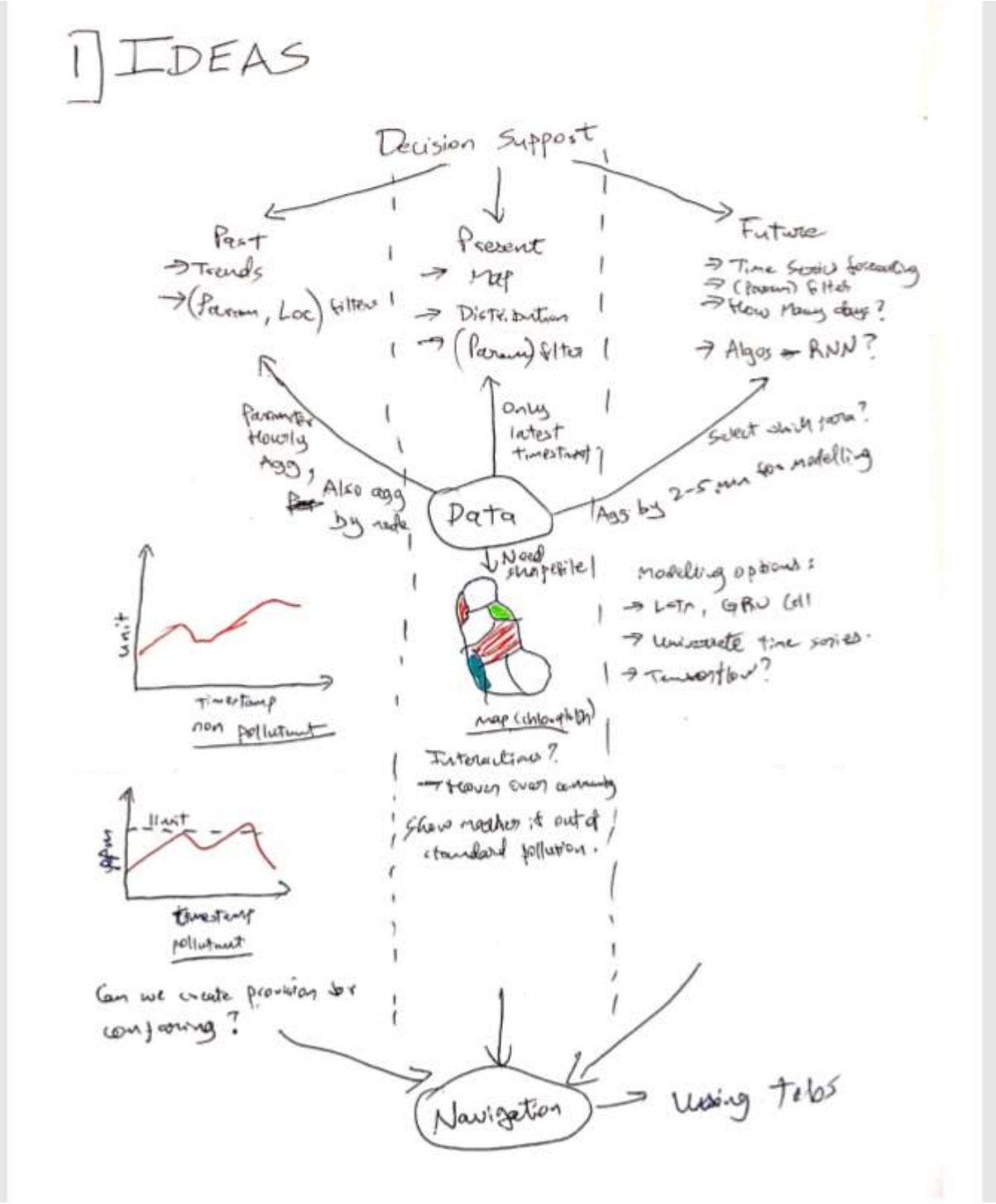
This project has laid down implementational foundations towards working with Array of Things data which can be applied to any IOT data. The project has covered IOT sensor data wrangling, predictive data modelling and interactive data visualization. This project helped me venture into the realm of front-end development, UI/UX principles and interactive visualizations. This is just the start towards extracting value out of the AOT project. Big data processing capabilities would enable large scale machine learning deployment on the AOT infrastructure which can be used to tackle several use cases such as flu prediction or flood prediction. I would have liked to worked more in terms of predictive modelling and would have pursued climate forecasting using multivariate time series comprising of temperature, humidity, pressure to start with. To summarise, the AOT is a very good open data initiative and will certainly allow data scientists to impact various cities. IOT engines coupled with machine learning models and big data processing capabilities may very well be the wave of the future.

6. References:

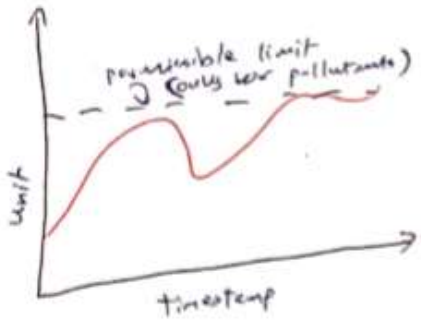
- https://arrayofthings.github.io/
- https://tensorflow.rstudio.com/tensorflow/
- https://www.epa.gov/criteria-air-pollutants/naaqs-table
- https://github.com/waggle-sensor/sensors/raw/master/sensors/datasheets/mma8452q.pdf

7. Appendix

5 design sheets:



2 INITIAL DESIGN 1



Location filter
OVERALL
LOOT
WESTLAWN

Parameter filter
Temp
Pressure
SO2
CO

Title : AOT dashboard
Author : Swaraj Purohit
Date : 2/6/2019
Sheet : 1 - FDS 2
Task : Plot past trends

Operations

→ User can filter based on location, parameter.

→ Provision to compare parameters across locations?

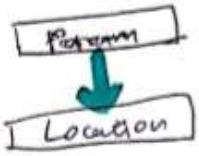
→ Easy Navigation to other panel.

Layout

→ User should be able to see trend :. line graph

→ Limit line to alert user of high pollution.

→ ~~Parameter filter~~ Location filter will update based on param.



Decision

→ Give ability to select multiple locations?

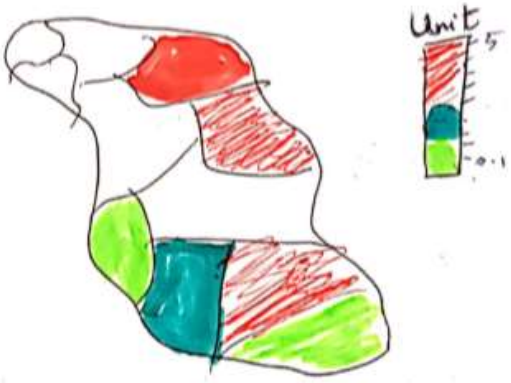
→ Use community area instead of node to make user friendly.

→ Colour : Perhaps use red to show limit & blue for line.

focus

3 INITIAL DESIGN 2

Param filter



B·4

current average

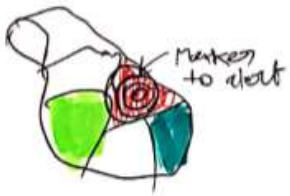
Layout

Title : AQT Dashboard
Author : Swaraj Purohit
Date : 2/6/2019
Sheet : 2 of FDS 3
Task : Show Current Data

Operations

→ Filter Parameter.
→ Hover on community to see its value ?.

→ Try to denote some marker on community if pollution > standards.



→ Show Avg in red if exceeds limit.

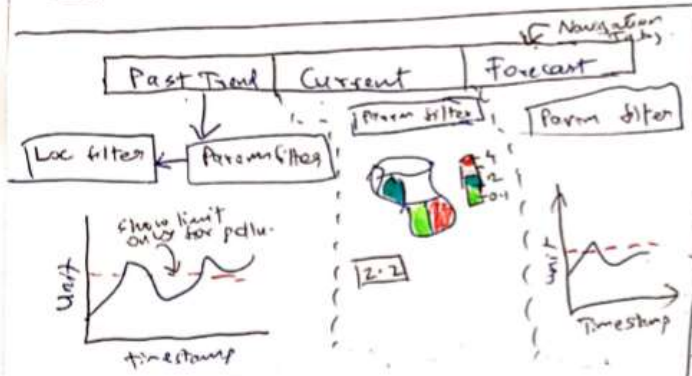
3·4

Focus

Discussion

→ Colour Scheme : Since various params should we use windows ?
→ Should community names be on map ?
→ Hover should show name, value for that community.

4 INITIAL DESIGN 3



Title : AQT Dashboard
Author : Suvaraj Purohit
Date : 2/6/2019
Sheet : 3 - FDS 4
Task : show overall Dashboard

Operations

→ Change panel easily using tabs
→ filter each panel suitably.
→ Compare different locs for same param (past trend).
→ Hover map to see value and name for the community.

Layout

focus

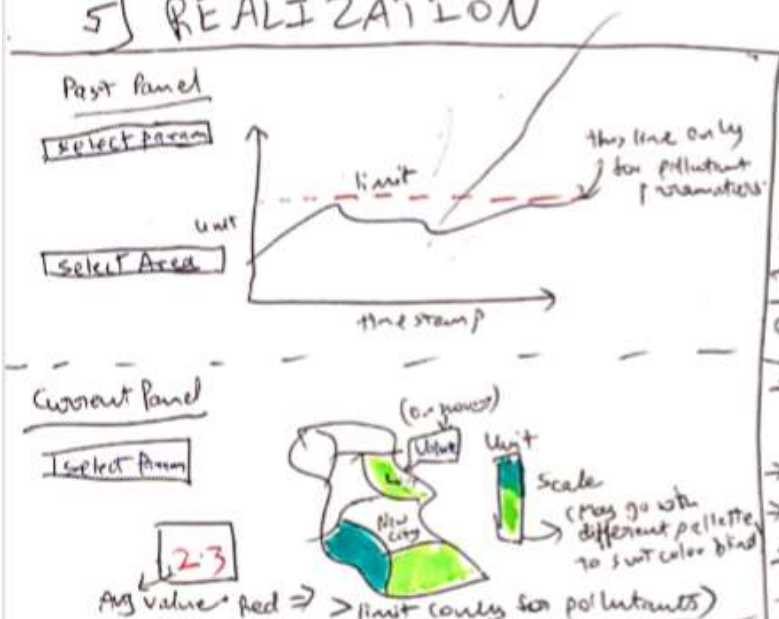
→ Using permissible limit line wherever required.
→ Show current avg. in red if pollutant exceeds standards else green.

10.2 / 4.2

Discussion

→ Each Tab needs its own sidebar filter.
→ Loc filter depends on param filter for past trend.
→ Use uni coloured for map?
→ For forecast load model (trained) and predict.

5] REALIZATION



LAYOUT

→ Denote pollution → limit line in line graph.
→ Colour coded average in current panel.

FOCUS

Title : AQI Dashboard
Author : Swaraj Purohit
Date : 3/6/2019
Sheet : 4-FDS 5
Task : Finalize project & detail.

OPERATIONS

→ Allow user to search using filters.
→ Easy panel navigation.
→ Hover to see exact details.
→ forecasting
→ Zoom, pan, download, edit.

DETAIL

→ Estimated work hours : 40hrs.
→ Data from AQI website
→ Uni-coloured scale to suit colour blind as well (Y10, B7)
→ Use red to denote pollute alerts
→ Software required : Rstudio, Anaconda (64 bit).
→ Important libraries : plotly, ggplot, Tensorflow, dplyr, maptools.
→ Algorithms : LSTM / GRU cell based RNN.