DSC 540

Data Preparation

# Final Project Dataset

# Submitted By: Abhigyan Misra

I will be using the following datasets for my project.

1. **CSV/Excel/PDF or another flat file source.**

The data I downloaded is the District/Constituency wise Voter Information for the 2019 General Elections in India. The csv file being used was downloaded from the Election Commission of India website (eci.gov.in).

The csv/xls file we will be using for this step is: “13. PC Wise Voters Turn Out.xls”

1. **Website you want to pull data from**

The second data source that I will look at is the District/ Constituency wise candidate results from Wikipedia. The table that I will be looking at is *Table: - Results by constituency*. Link for the webpage is as below: -

<https://en.wikipedia.org/wiki/Results_of_the_2019_Indian_general_election>

1. **API you will pull data from**

The third data source, that I will use is openweathermap api or the iqair api for getting the historical weather and pollution data. I would pull the data for all 545 districts/constituency from the api.

The details of the same can be found at below urls:

<https://www.iqair.com/dashboard/api> - For now I am inclined more towards this as this has data available by city name, state name and Country. The city name is same as Parliamentary Constituency name or District Name.

<https://openweathermap.org/api> - Will use this, If I am able to figure out how to get the data based on city name. The documentation shows that I need to provide Latitude and Longitude information, then I might need another api to get the Lat and Long information for the constituency from there.

**Note:** Following files are located in below github location

<https://github.com/abhigyanmisra/dsc540/tree/master/finalproject>

**Relationship between the data sources**

The common relationship between them is the district or constituency name and I want to look at the pollution levels in each of the winning candidates’ constituencies. This may help in understanding what has been done by each of these candidates towards pollution control.

Below is the list of attributes and description from csv. The csv seems slightly improperly indented and split into columns.

**State Name:** This is the name of the state to which a constituency belongs to.

**PC. No:** This is the Parliamentary Constituency Number. This is unique per state.

PC**. Name:** This is the name of the Parliamentary Constituency.

**Electors:** This is the total number of Electoral Votes for the parliamentary constituency.

**Male:** This is the total number of votes cast using the EVM (Electoral Voting Machine) by males

**Female:** This is the total number of votes cast using the EVM (Electoral Voting Machine) by females

**Third Gender:** This is the total number of votes cast using the EVM (Electoral Voting Machine) by transgender.

**Total:** This is the total number of votes cast using the EVM (Electoral Voting Machine).

**NRI:** This is the votes by Non-Resident Indians cast through EVM’s. This is an additional information and sum is not separately counted towards total EVM votes.

**Postal Votes:** This is the number of votes by postal mail.

**Total Voters:** This is the sum of Total EVM Votes and Postal Votes.

**Voter Turnout (%):** this is the %age of total votes cast

**Voter Turnout (% - Male):**  This is the %age of Male Voters who voted.

**Voter Turnout (% - Female):** This is the %age of Female Voters who voted.

**Voter Turnout (% - Third Gender):** This is the %age of Transgender Voters who voted.

Then from the website, we have the following details

**State:** This is the name of the State to which Parliamentary Constituency belongs to.

**Parliamentary Constituency No:** This is the Parliamentary Constituency Number. This is unique per state.

**Parliamentary Constituency Name:** This is the name of the Parliamentary Constituency.

**Parliamentary Constituency Type:** This is the Type of Seat reserved based on the Reservation Rules

**Winner Candidate:** This is the name of the Winning Candidate.

**Winner Party:** This is the party name to which winning candidate belongs.

**Winner Votes:** This is the total votes received by the winning candidate.

**Runner Up Candidate:** This is the name of the Runner up Candidate.

**Runner Up Party:** This is the party name to which runner up candidate belongs.

**Runner Up Votes:** This is the total votes received by the runner up candidate.

**Margin:** This is difference of votes between the winning and runner up candidate.

To get the data from API, I will be hitting the following URL:

http://api.airvisual.com/v2/city?city=Los Angeles&state=California&country=USA&key={{YOUR\_API\_KEY}}

The detailed response would look like as follows:

{

"status": "success",

"data": {

"name": "Eilat Harbor",

"city": "Eilat",

"state": "South District",

"country": "Israel",

"location": {

"type": "Point",

"coordinates": [

34.939443,

29.531814

]

},

"forecasts": [ //object containing forecast information

{

"ts": "2017-02-01T03:00:00.000Z", //timestamp

"aqius": 21, //AQI value based on US EPA standard

"aqicn": 7, //AQI value based on China MEP standard

"tp": 8, //temperature in Celsius

"tp\_min": 6, //minimum temperature in Celsius

"pr": 976, //atmospheric pressure in hPa

"hu": 100, //humidity %

"ws": 3, //wind speed (m/s)

"wd": 313, //wind direction, as an angle of 360° (N=0, E=90, S=180, W=270)

"ic": "10n" //weather icon code, see below for icon index

},

… // contains more forecast data for upcoming 76 hours

]

"current": {

"weather": {

"ts": "2017-02-01T01:00:00.000Z",

"tp": 12,

"pr": 1020,

"hu": 62,

"ws": 2,

"wd": 320,

"ic": "01n"

},

"pollution": {

"ts": "2017-02-01T01:15:00.000Z",

"aqius": 18,

"mainus": "p1", //main pollutant for US AQI

"aqicn": 20,

"maincn": "p1", //main pollutant for Chinese AQI

"p1": { //pollutant details, concentration and appropriate AQIs

"conc": 20,

"aqius": 18,

"aqicn": 20

}

}

},

"history": { //object containing weather and pollution history information

"weather": [

{

"ts": "2017-02-01T01:00:00.000Z",

"tp": 12,

"pr": 1020,

"hu": 62,

"ws": 2,

"wd": 320,

"ic": "01n"

},

… // contains more weather historical data for past 48 hours

]

"pollution": [

{

"ts": "2017-02-01T01:15:00.000Z",

"aqius": 18,

"mainus": "p1",

"aqicn": 20,

"maincn": "p1",

"p1": {

"conc": 20,

"aqius": 18,

"aqicn": 20

}

},

… // contains more pollution historical data for past 48 hours

]

},

"units": { //object containing units information

"p2": "ugm3", //pm2.5

"p1": "ugm3", //pm10

"o3": "ppb", //Ozone O3

"n2": "ppb", //Nitrogen dioxide NO2

"s2": "ppb", //Sulfur dioxide SO2

"co": "ppm" //Carbon monoxide CO

}

}

}

Below are a few examples of return codes that we may get. This list is not exhaustive.

* success: returned when JSON file was generated successfully.
* call\_limit\_reached: returned when minute/monthly limit is reached.
* api\_key\_expired: returned when API key is expired.
* incorrect\_api\_key: returned when using wrong API key.
* ip\_location\_failed: returned when service is unable to locate IP address of request.
* no\_nearest\_station: returned when there is no nearest station within specified radius.
* feature\_not\_available: returned when call requests a feature that is not available in chosen subscription plan.
* too\_many\_requests: returned when more than 10 calls per second are made.

The details about each of the response attributes can be found in detailed response above. The one’s which I will specifically look at are pollution params units are

"p2": "ugm3", //pm2.5

"p1": "ugm3", //pm10

"o3": "ppb", //Ozone O3

"n2": "ppb", //Nitrogen dioxide NO2

"s2": "ppb", //Sulfur dioxide SO2

"co": "ppm" //Carbon monoxide CO

And

"pollution": [

{

"ts": "2017-02-01T01:15:00.000Z",

"aqius": 18,

"mainus": "p1",

"p1": {

"conc": 20,

"aqius": 18,

"aqicn": 20

}

Overall, I have everything needed to accomplish my 5 milestones which are

1. Identification of 3 different data sources
2. Cleaning / Formatting File Data Source
3. Cleaning / Formatting Website Data
4. Connecting to an API/Pulling in the data and cleaning/formatting.
5. Merging the Data and Storing in a Database/Visualizing Data