Method

We used Apache Zeppelin to run different methods in our project. Apache Zeppelin enables interactive data analytics on a web based notebook. Apache Zeppelin supports different languages like Python, Scala, R, SQL and more. The main advantage of using Apache Zeppelin is, it has built in Apache Spark integration. In many other notebook or tools we need to build separate module or import the libraries required. Some of the features Apache Zeppelin provides with spark integration are Automatic injection of data using SparkContext and SQLContext, load libraries or jar files from local file system or maven repositories, display the job progress and job cancelation. Normally Apache Zeppelin comes with whole package required for data analysis. For example, matplotlib library is mostly used for the data visualization and data analysis in python, so latest version of Apache Zeppelin comes with integrated Matplotlib library, we do not have to import it separately and also leverage the Spark performance speed by pyspark Interpreter which fast and reliable.

In this project we are going to use python in Zeppelin. In Apache Zeppelin we can write python scripts by using %python tag before writing python scripts. But we need to use spark integrated python too as for now we have only limited dataset and in future if we are going to build large scale project based on this, we might need to leverage spark parallel processing and high volume processing. For using python in spark we need to use %pyspark and this will change the interpreter into python on spark. Apache Zeppelin provides all these inbuilt so we do not have to reconfigure any setting to get this functionalities. More over Apache Zeppelin is open source which gives huge advantage to programmers and application builders and contribute in the development of tools which are helpful

First we need to import the data into the spark data file system. For this we can use the SQLContext function where we need to specify the directory of all the files. This will consider all the files in the directory and read them. Now that data is in the sqlcontext object we can use sql commands too to see the output and work on functions. This is very good approach because we need to configure so many things, change settings, paths, connections before so that we can import, validate, make changes and export where as in Apache Zeppelin we have all the functionalities, dataset and we can use which comfortable programming language we are comfortable with to build the project and run successfully.

# Flowcharts

A flowchart is a visual portrayal of the arrangement of steps and choices expected to play out a procedure. Each progression in the grouping is noted inside a chart shape. Steps are connected by interfacing lines and directional bolts. This permits anybody to see the flowchart and sensibly take after the procedure from start to finish.

The basic symbols we use in the flow charts are rounded ends, which represents start or end. Parallelogram used for input or output, rhombus is used for the decision tree like if conditions in the programming languages. The flow charts of our three methods are as shown below.



The charts shown above are the flow charts for read file, show heatmap of pollution around the city and verifying the time-series for the data. For instance, let’s take the first flow chart, where in the beginning we used rounded or oval shaped which represents the start of flow chart. Next step is input or output, we are reading all the files from the directory specified and load them into the spark data file system. We are introducing if statement in the next step it is a rhombus shaped. This basically used for the decision making. So in this if reading failed and if there are no files existed in the directory, it will as for the directory path again.

# Pseudo code

Pseudo code is used by programmers for better understanding and informal language used to develop algorithms.

**Pseudo code for reading files:**

Initialize list to empty list

Initialize path to all file’s directory path

While directory has file:

Read each csv file

Split each record with respect to comma delimiter

Read first record as header

Each record after splitting is appended to temporary list

All these temporary lists are appended to main list

End of while loop

Print main list which have each location’s readings

**Pseudo code to implement heat map:**

Import map libraries

For each in main list:

Read 449 records in list

Get latitudes and longitudes

Get values for those coordinates

Plot those values in the map

Increase the depth of the colour according to the value

End for loop

View whole map with locations and their respective heat map

**Pseudo code for running different regression models:**

First we need to understand the data and analysis which regressions models can be applicable. Before running all the models and end up only few models are applicable for this data set. And select which parameters we are going to use for the models. Scikit library is very powerful when it comes to the regression models.

Import scikit library

Initialize modeList to all the models to be run

Initialize runRec to all the records from main list

For loop to iterate through all the locations available:

Integrate all location’s records into one list

For loop to iterate all in modeList:

Run model on new list

Save results in temporary list

Compare the results of all the models from temporary list

Select the best model from the compared models

Use scikit library and implement the model we selected on the data

Print the results of the predicted values from this model

**Pseudo code for Time series analysis**:

Get all the location’s values

For each location:

Get the timestamp from the data

Get the value of pollution to the timestamp

Plot time series graph using this data

Observe any patterns of sudden fall or sudden rise

Predict the pollution values for the next timestamp

# Explain operations

There are four main operations considered in this project and one future implementation or operation for better understanding of the situation. The four main operations are show case location of sensors which collect the data of the pollution and records it. The different factors for the air pollution these sensors collect are ozone, particulate matter, carbon monoxide, sulphur dioxide, nitrogen dioxide, additional to these it also records latitudes, longitudes and time stamp. With this readings, these following operations are implemented:

**Show locations of sensors:**

In the data files we have, there are latitudes and longitudes of reading that took place. Each file is from one different location, so each file have data from one sensor. So using this locations we can show where the sensors are and how polluted those areas are. There are 449 locations in the directory so Brasov city has records of 449 pollution records.

**Heat map showcasing density of pollution:**

Now that we have the data of different locations of pollution records so we can see how the pollution is spread in the city. We have the locations of the sensors in our data, using those latitudes and longitudes and using maps library in python we can showcase the Brasov city. Those coordinate we locate them on the map, if it is highly dense colour it represents the highly polluted area.

**Time series for those locations or busy location**:

We can also plot the time series of a particular location throughout the day, how different factors of pollution distributed along the time. We can build these plots using matplotlib library. We can also compare different factors how they are related to each other and we can find correlation between each of them. The data we have is the records for every 5 minutes update, so we can have accurate time series plot. By observing the time series and density of pollution we can find the busy routes in the city. This could help in finding the highly polluted areas and people can avoid these routes as it will be harmful to health

**Regression models and predict:**

Predicting can be made using different regression models. Some of the regression models we can use are logistic regression model, multiple regression, box-cox, principle component regression and Ridge regression. Once we run all the models compare them with each other and observing which one is the best model and decide which one to use for further implementations.

**Cross validate Pollution rates and population:**

If we consider only the pollution rates of areas, it is not sufficient because highly populated area will obviously have high transportation rates. With increase in transportation there is high chances of air pollution around this area. We need to find the area wise population of Brasov city and as we already have the pollution rates of those areas, we can cross validate both of them and examine if there is any relation between pollution and population. Theoretically there should be relation. Vehicle usage is directly proportional to the air pollution. Not only vehicles emit harmful gases into the air, even some electrical tools which we use on daily bases will also emit harmful gases like Air Conditioners, refrigerator will also emit some pollution.

# Functions

Functions are basically used to eliminate the reuse of piece of code. In technical point of view functions are set of code which perform certain operations on the values provided from the parameters passed. This functions can be called as many times as required and perform operations as many times as required without writing the same code repeatedly over and over again. In simple words functions are explained with an example like, sum of two numbers calculation is repeating over and over again in code, this adding two numbers is written into one function. So whenever there is requirement of sum of two numbers, we can directly call this function by passing the two numbers which are needed to add. This will return the output, sum of those two numbers.

Functions are divided into two categories, user defined functions and inbuilt functions. Inbuilt functions are the functions which are written and available with the programming language, this does not need to write separately. For example “print” function in python. Print function is already written in programming language itself, so whenever we call “print” function with any value passing into it, it will print it into the console without any additional coding. Whereas user defined functions are written by programmers for their convenience and they define the rules of that function. For example sum of two numbers function. This function is written manually by asking the user to enter two numbers which are to be added. Inside the function these two numbers which are passed through function are added together by the addition operation which is mentioned inside the function. Then there is a return function which returns the final value after all the operations executed in that function.

For this project we are going to use three main user defined functions readFiles, locationValues, timeValues. ReadFiles function is built mainly to read csv file in the directory and return the list of all the data from csv files. This function takes the directory as input string, this input directory is accessed by the python and gets all the files from that directory. Now that we have all the files from directory (we have 449 csv files in this directory). Each file represents the location of sensor across the city and each file will have five air pollution factors, latitude and longitudes and timestamp for the readings. So in for loop we will be taking each file at a time and read it line by line. Each line represents one record in csv. So every line consists of 5 values of pollution factors, 2 location values and 1 timestamp. So each line is read and splits using delimiter “,”, now these 8 values are stored in the list. As we are in the loop of all the files, these lists generated by each file is added to another list and this big list consists of all the 449 files. Using all these lists, looping them we can work on our operations using pyspark.