Centre for Cloud Computing and Big Data Assignment

Team:

Dhruva Kashyap	PES1201801457
Rakshith C	PES1201801903
K Vikas Gowda	PES1201801957
S Nikhil Ram:	PES1201801972

TASK 3: Bengaluru’s Greenest part

1. Obtaining Satellite Images

53 Satellite Images were obtained from taking screenshots from google earth, the project can be viewed here.

Karnataka Legislative Assembly Constituency lines and prominent localities were considered. Areas were given around 1-3 images per locality.

The list of areas is as follows

- Malleswaram
- Yelahanka
- K R Puram
- Byatarayanapura
- Yeshwanthpur
- R R Nagar
- Dasarahalli
- Mahalakshmi layout
- Hebbal
- Sarvagnanagar
- C V Raman Nagar
- Shivajinagar
- Shantinagar
- Gandhinagar
- Rajajinagar
- Vijayanagar
- Chamarajpet
- Chickpet
- Basavangudi
- Padmanabhanagar
- BTM Layout
- Jayanagar
- Koramangala
- Whitefield
- Indiranagar
- Ulsoor
- HSR Layout
- Marathahalli
- Mathikere
- Ballandur
- Kalyannagar
- Banashankari

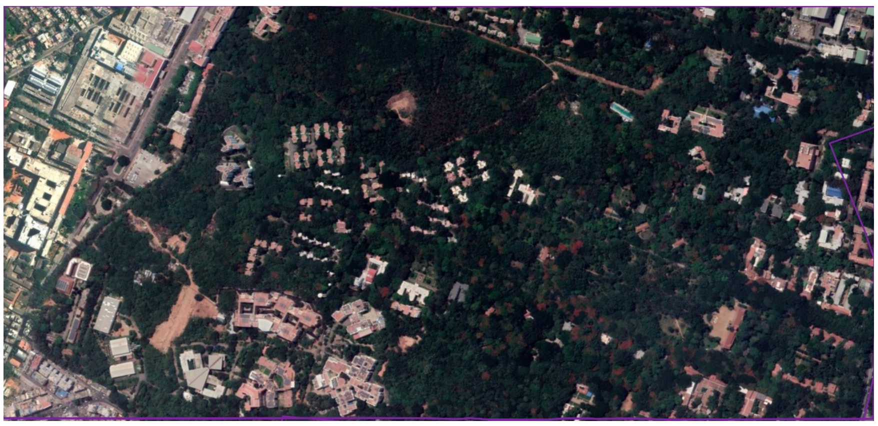
- Jalahalli
- JP Nagar
- Basaveshwarnagar
- RT Nagar
- RM Nagar
- Electronic City
- Sadashivnagar

2. Image Processing using OpenCV

Our code base was written in python. We converted the images we have obtained to HSV format and separated each image into chunks. A chunk being 1x1px or 5x5px or 10x10px. Each chunk was evaluated for the percentage of green and if it crossed a threshold percentage (50% or 60%) then entire chunk was considered to be a green area.

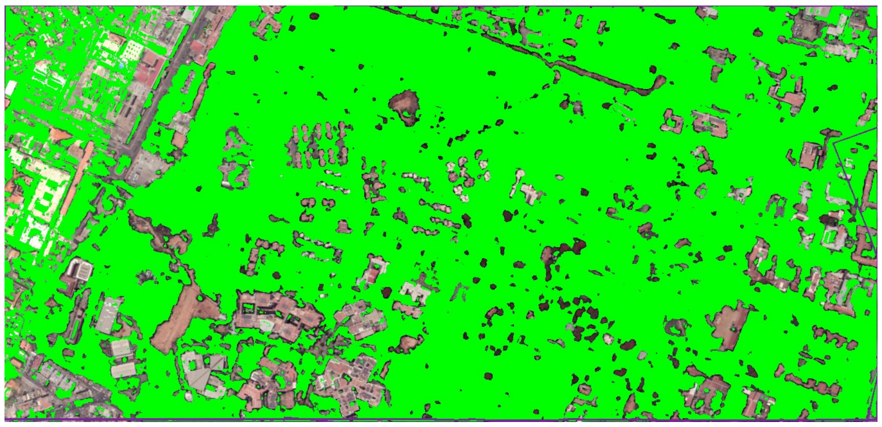
Some examples:

original image:



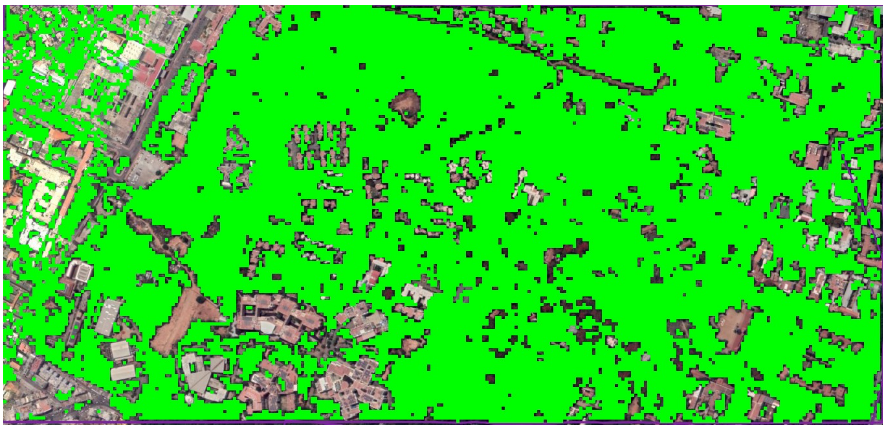
Areas recognized:

1x1px(100% threshold)



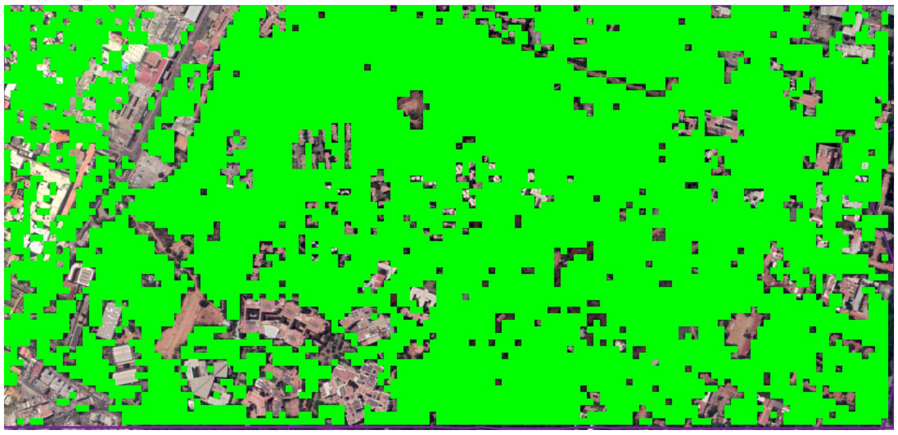
80% green

5x5px(70% threshold)



74% green

10x10px(50% threshold)



81% green

In case of multiple images for an area, the total green percentage was calculated as

$$p = \frac{\sum_{i=0}^{n} A^i * p^i}{\sum_{i=0}^{n} A^i}$$

We had added an additional 20% to the green area because we do not have a sufficient number of data points(images) to accurately determine the percentage of green in an area. This number was decided after observing results manually.

Instead of preprocessing the images to find the green percentages of images beforehand, we decided that this process could be done in the map phase of the hadoop job. This would theoretically speed up the process by reducing the amount of time spent on preprocessing the images. Instead, we would be calculating the green percentage during run time. If the job was distributed over multiple machines, this would make the process faster.

3. Hadoop Map Reduce

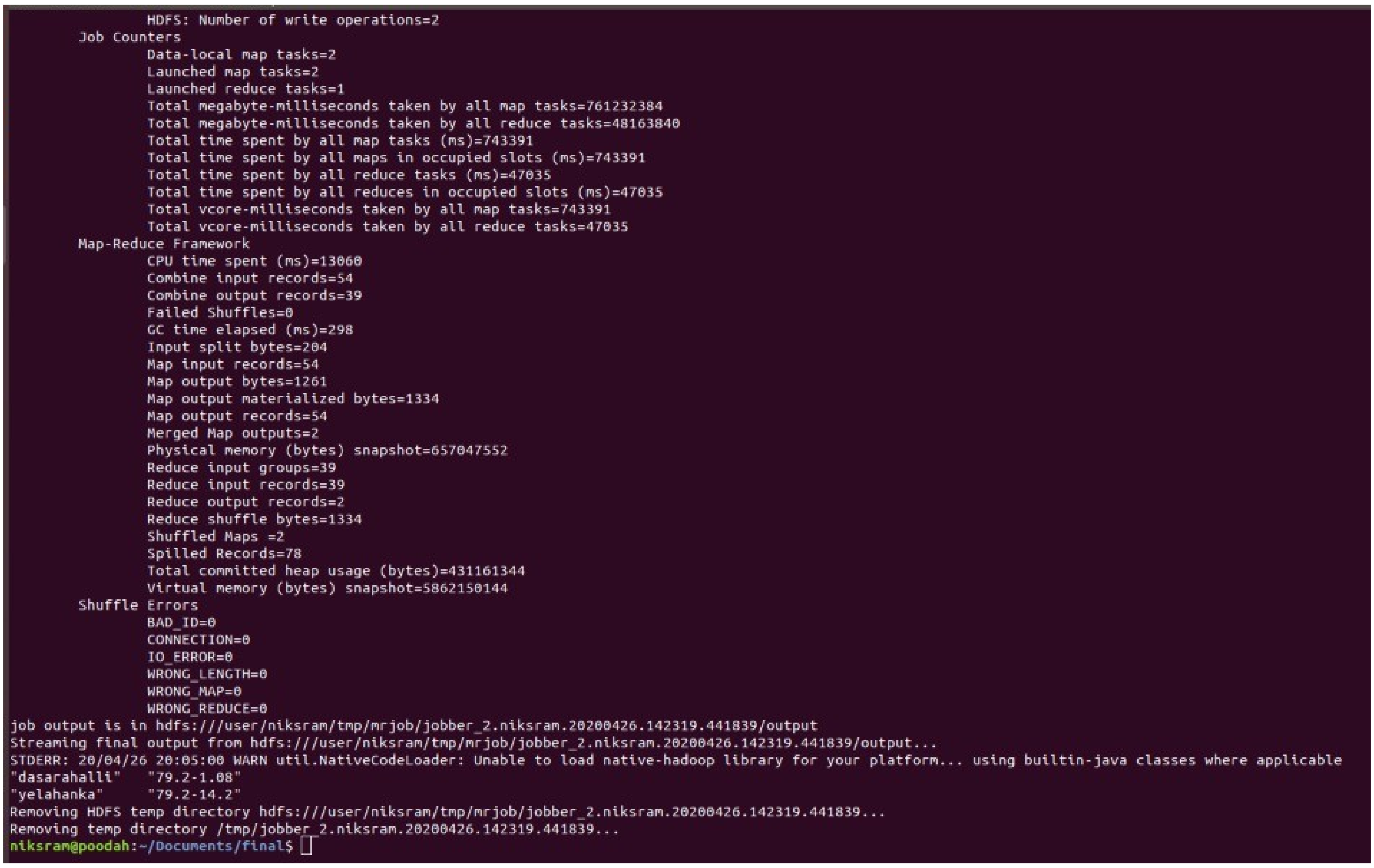
Our code base was written in Python 3. We have used the mrjob package in python 3 to run hadoop jobs. We set up a Pseudo-Distributed Operation on a virtual machine running Ubuntu 18.04. We have used Apache Hadoop 3.2.1.

After configuring hadoop to run in *pseudo-distributed operation* and configured hadoop to run map reduce jobs, we run the progaram as follows.

- Input file is generated from a json file that contains details of all images,i.e image name,locality name and area in sq kilometers.
- Start all dfs and yarn nodes.
- Move all images, numpy package files, opencv package files and the input file.
- The mapper reads from the input file and processes each image and calculates the percentage of green in each image.
- The combiner groups all images belonging to the same area.
- The reducer collects all images belonging to some locality and calculates the total green percentage and yeilds it if it is greater than 75%

run.sh contains furthur details.

Output:



```
                HDFS: Number of write operations=2
        Job Counters
                Data-local map tasks=2
                Launched map tasks=2
                Launched reduce tasks=1
                Total megabyte-milliseconds taken by all map tasks=761232384
                Total megabyte-milliseconds taken by all reduce tasks=48163840
                Total time spent by all map tasks (ms)=743391
                Total time spent by all maps in occupied slots (ms)=743391
                Total time spent by all reduce tasks (ms)=47035
                Total time spent by all reduces in occupied slots (ms)=47035
                Total vcore-milliseconds taken by all map tasks=743391
                Total vcore-milliseconds taken by all reduce tasks=47035
        Map-Reduce Framework
                CPU time spent (ms)=13060
                Combine input records=54
                Combine output records=39
                Failed Shuffles=0
                GC time elapsed (ms)=298
                Input split bytes=204
                Map input records=54
                Map output bytes=1261
                Map output materialized bytes=1334
                Map output records=54
                Merged Map outputs=2
                Physical memory (bytes) snapshot=657047552
                Reduce input groups=39
                Reduce input records=39
                Reduce output records=2
                Reduce shuffle bytes=1334
                Shuffled Maps =2
                Spilled Records=78
                Total committed heap usage (bytes)=431161344
                Virtual memory (bytes) snapshot=5862150144
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
job output is in hdfs:///user/niksram/tmp/mrjob/jobber_2.niksram.20200426.142319.441839/output
Streaming final output from hdfs:///user/niksram/tmp/mrjob/jobber_2.niksram.20200426.142319.441839/output...
STDERR: 20/04/26 20:05:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
"dasarahalli"   "79.2-1.08"
"yelahanka"     "79.2-14.2"
Removing HDFS temp directory hdfs:///user/niksram/tmp/mrjob/jobber_2.niksram.20200426.142319.441839...
Removing temp directory /tmp/jobber_2.niksram.20200426.142319.441839...
niksram@poodah:~/Documents/final$
```

Dasarahalli and Yelahanka seem to be very green!