

# FINAL PROJECT REPORT FOR PRACTICE SCHOOL-1

## Web Based Platform for Geo-Spatial Data Analysis using Datacubes

By:

SIVARAMAKRISHNAN KN	2017A7PS0153H
GARVIT SONI	2017B3A70458H
VAIBHAV CHAUDHARI	2017B5A70834G
CHETAN KHANNA	2017B4A70591G
SHREYASH CHAUDHARI	2017A7PS0941G
NRUPESH SURYA U	2017B2A70767G

*A report submitted in partial fulfilment of the requirements of the course  
**BITS F221 - PRACTICE SCHOOL-1***



**BIRLA INSTITUTE OF TECHNOLOGY AND  
SCIENCE, PILANI**

12 July, 2019

## Declaration

We, the students of BITS Pilani, Practice School - 1 (BITS F221), Summer Term, hereby declare that the project work entitled “Web Based Platform for Geo-Spatial Data Analysis using Datacubes” is our original work and has been carried out under the supervision of Shri Nilay Nishant , Scientist/Engineer-SD and Dr. Puyam S. Singh , Scientist/Engineer-SE, North Eastern Space Applications Center (NESAC) . This work has not been submitted to any other University, College or institution for evaluation or examination.

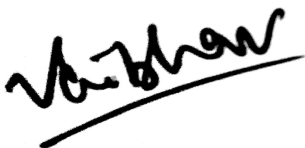
Date:12 July,2019

Place: Shillong

Student's Name and Signature:



SIVARAMAKRISHNAN K. N.



VAIBHAV CHAUDHARI



NRUPESH SURYA



CHETAN KHANNA



SHREYASH CHAUDHARI



GARVIT SONI

## CERTIFICATE

It is to certify that the work contained in this Report entitled “Web Based Platform for Geo-Spatial Data Analysis using Datacubes,” by the students of BITS Pilani, Practice School - 1 (BITS F221), Summer Term, has been carried out under my supervision and this work has not been submitted elsewhere for any type of evaluation.

Mr. Nilay Nishant

## ACKNOWLEDGEMENT

First of all, we are deeply grateful to the supreme almighty with whose blessings, we were able to accomplish this humble piece of work.

We would like to express our sincere thanks to our guide, Mr. P.S. Singh and Mr. Nilay Nishant for their continuous guidance, supervision and constant encouragement throughout the duration of this project. We would always remain thankful to our guide for giving me an opportunity to explore and learn in the field through this project.

We express our wholehearted thanks to Mr. PLN Raju, Director, North Eastern Space Application Centre, Umiam and Ms. Ritu Anilkumar for allowing us to undertake this project in this esteemed organization. We are greatly thankful to our colleagues and management of NESAC for their valuable support and advice during the course of this project.

We would like to express our sincere thanks to our PS coordinator Prof. D. Sriram for his cordial support, valuable information and guidance.

# LIST OF ILLUSTRATIONS

Fig 1.1	Workflow of ODC	9
Fig 2.1	ODC	9
Fig 3.1	Workflow of Web Applications	10
Fig 3.2	Homepage of Website	11
Fig 3.3	Sample Page from Website	11
Fig 4.1.1	NDVI Output	15
Fig 4.1.2	NDBI Output	17
Fig 4.1.3	NDWI Output	19
Fig 4.1	Band Map Algorithm Using NDVI+NDBI+NDWI	19
Fig 4.2.1	Cloud Free Mosaic	20
Fig 4.3.1	Example of Land Cover Classification	21
Fig 5.1	Example of Random Forest Classification	23
Fig 5.2	Example of Simplified Random Forest Classification	23
Fig 5.3	Gini Coefficient	24
Fig 6.1	GetGeoTransform Indexes	27
Fig 6.2	Notebook Workflow	28
Fig 6.3	Notebook Code Snippet 1	28
Fig 6.4	Notebook Code Snippet 2	29
Fig 6.5	Notebook Code Snippet 3	29
Fig 6.6	Image of Guwahati taken from Sentinel-2	30
Fig 6.7	Output Image of Guwahati after Classification	30

Fig 7.1	Tiling Services	31
Fig 8.1	Tools Used	32

# ABSTRACT

The Open Data Cube (ODC) initiative seeks to increase the value and impact of global Earth observation satellite data by providing an open and freely accessible data exploitation for further utilization in analysis and algorithms. NE-GeoCloud aims to facilitate convenient interaction between diverse satellite imagery and scientists without the additional complexity of local data hosting and processing. Through a user-friendly front-end and a Data Cube-based back-end, NE-GeoCloud reduces the complexity of geospatial algorithm execution on data to simple product/algorithm selection. The platform also enables scalability of future complex algorithms (machine/deep learning scripts) and larger datasets (tiling, mosaicing, ingestion configuration).

## **TABLE OF CONTENTS**

ACKNOWLEDGEMENT	4
LIST OF ILLUSTRATIONS	5
ABSTRACT	7
1. Introduction	9
2. Objective	10
3. Implementation	11
4. Applications	13
4.1. Band Math Algorithms	14
4.1.1. NDVI	14
4.1.2. NDBI	16
4.1.3. NDWI	18
4.2. Cloud Free Mosaic	20
4.3. Land Cover Classification	21
5. Random Forest Classification	22
6. Python Notebook Creation for classification	27
7. Tiling Services	31
8. Datasets and Tools used	32
9. Challenges	33
10. Discussion and Scope	34
References	35



# 1. Introduction

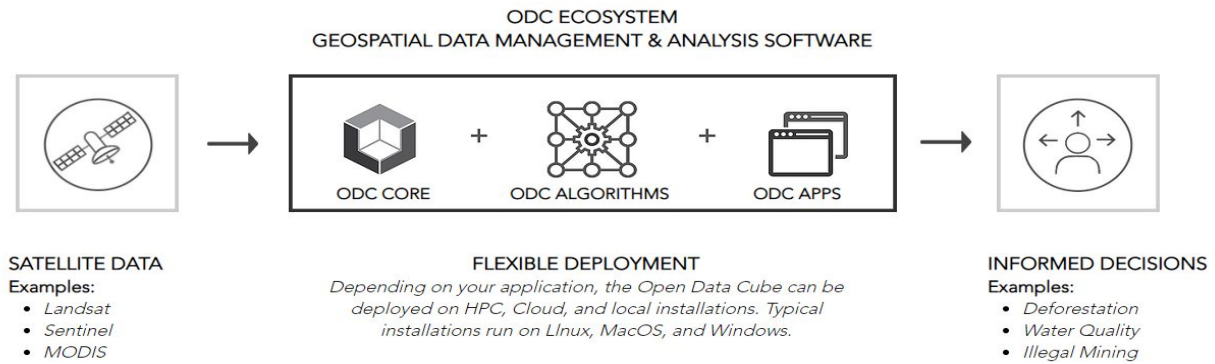
A Datacube is a multi-dimensional ("n-D") array of values. Typically, it is applied in contexts where these arrays are massively larger than the hosting computer's main memory; examples include time series of image data.

DataCubes are aiming to realize the full potential of earth observation(EO) data repositories by addressing Volume, Velocity, and Variety challenges, providing access to large spatio-temporal data in an analysis ready form.

The Open Data Cube (ODC) is an Open Source Geospatial Data Management and Analysis Software project that helps you harness the power of Satellite data. At its core, the ODC is a set of Python libraries and PostgreSQL database that helps you work with geospatial raster data.

The ODC seeks to increase the value and impact of global Earth observation satellite data by providing an open and freely accessible exploitation architecture.

The ODC project seeks to foster a community to develop, sustain, and grow the technology and the breadth and depth of its applications for societal benefit.



(Fig 1.1) Workflow of ODC

## 2. Objective

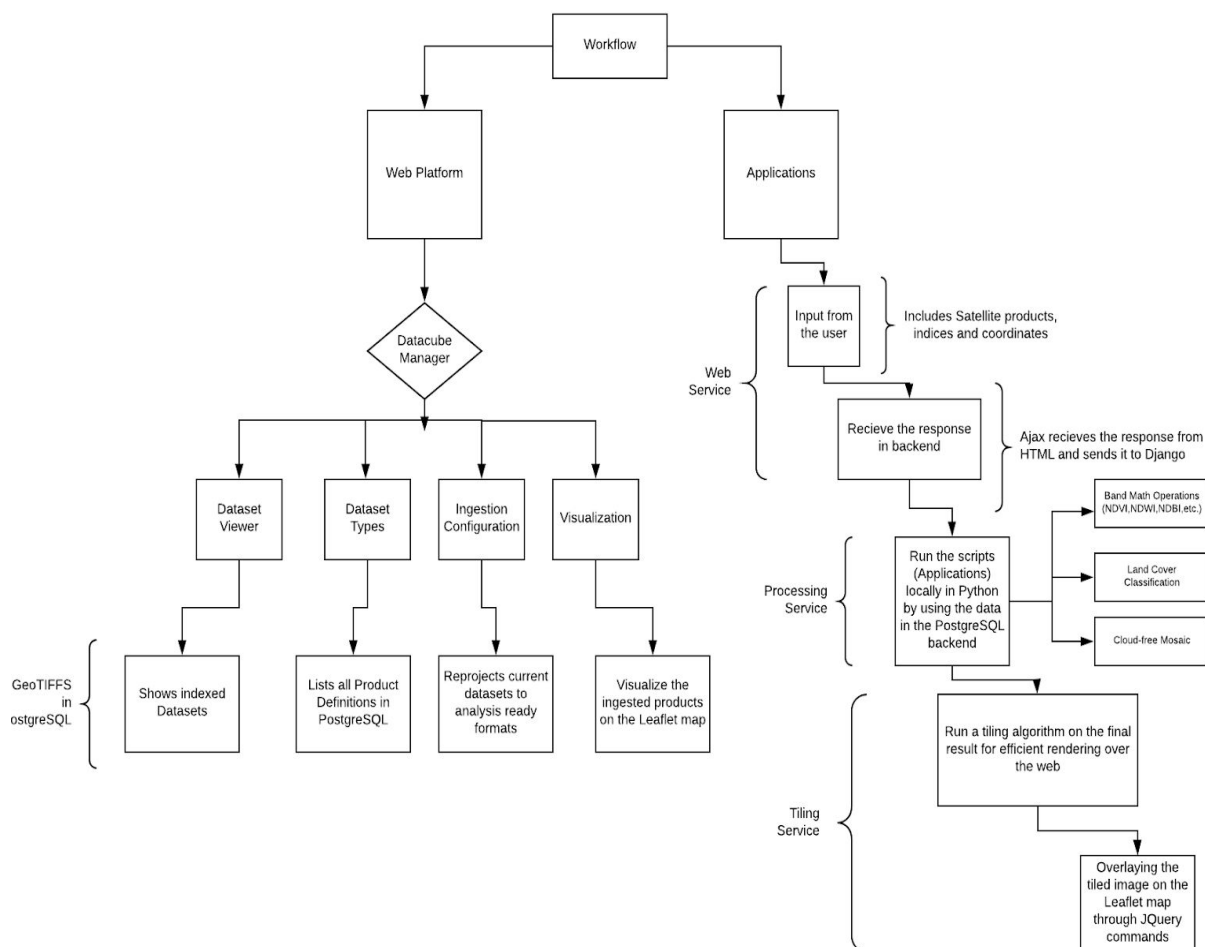
- Creating an online platform for the Geospatial applications on the EO Data obtained by satellites using Open Data Cube.
- Creation of the Deep Learning Algorithms (Notebooks) for manipulating the data over a particular region in the world.
- Developing an interactive website using which the user can customize the results of the data as per his need.
- Efficient time series analyses to support land change applications.



(Fig 2.1) ODC

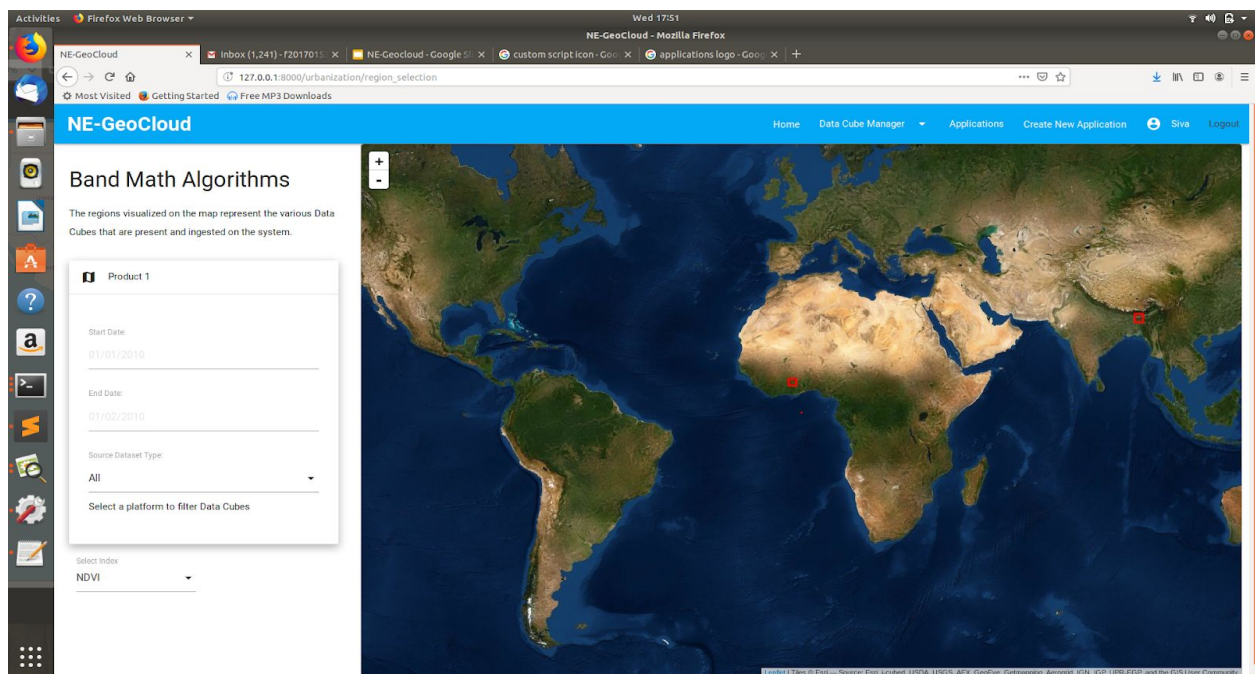
### 3. Implementation

- Datacube structure provides convenient consolidation of varied satellite data.
- Interaction between user and data facilitated via a front-end framework and a back-end PostgreSQL database and Django framework.
- A central web platform for running algorithms allows for instant results without local setup, and supports scalability and expansion of both data and applications.



(Fig 3.1) Workflow of Web Application

(Fig 3.2) Homepage of the website



(Fig 3.3) Sample page from website

## 4. Applications

We provide the following four applications out of the box in our cloud based application. These are :

1. Band Math Algorithms (NDVI, NDWI, NDBI)
2. Cloud Free Mosaic
3. Land Cover Classification

The first two are basic band math operation while the third is done using Random Forest classification. This was done to demonstrate that apart from basic band math operations our application can handle ML applications as well. It can also be easily extended to Deep Learning Applications. We describe these notebooks in detail below.

## 4.1 Band Math Algorithms

### 4.1.1 NDVI

- NDVI stands for Normalized Difference Vegetation Index
- It is a simple graphical indicator that assesses whether the given target contains vegetation
- $NDVI = (NIR - RED) / (NIR + RED)$

```
# # NDVI
# <br>
#
# > **NDVI(Normalized Difference Vegetation Index**
# > A derived index that correlates well with the existence of vegetation.
#
# <br>
#
# $$ NDVI = \frac{(NIR - RED)}{(NIR + RED)} $$
#
# <br>

# In[2]:
```

```
def NDVI(dataset):
    return (dataset.nir - dataset.red)/(dataset.nir + dataset.red)
```



*(Fig 4.1.1) NDVI Output*

## 4.1.2 NDBI.

- NDBI stands for Normalized Difference Built-up Index
- It is a simple graphical indicator that assesses whether the given target contains built-up
- $$\text{NDBI} = (\text{NIR} - \text{SWIR}) / (\text{NIR} + \text{SWIR})$$

```
# <br>
#
# # NDBI
# <br>
# > ** NDWI Normalized Difference Build-Up Index **
# > A derived index that correlates well with the existance of urbanization.
# <br>
#
# $$ NDBI = \frac{(\text{SWIR} - \text{NIR})}{(\text{SWIR} + \text{NIR})} $$
#
# <br>
```

```
# In[4]:
```

```
def NDBI(dataset):
    return (dataset.swir2 - dataset.nir)/(dataset.swir2 + dataset.nir)
```





*(Fig 4.1.2) NDBI Output*

### 4.1.3 NDWI

- NDWI stands for Normalized Difference Water Index
- It is a simple graphical indicator that assesses whether the given target contains water
- $$NDBI = (NIR - SWIR) / (NIR + SWIR)$$

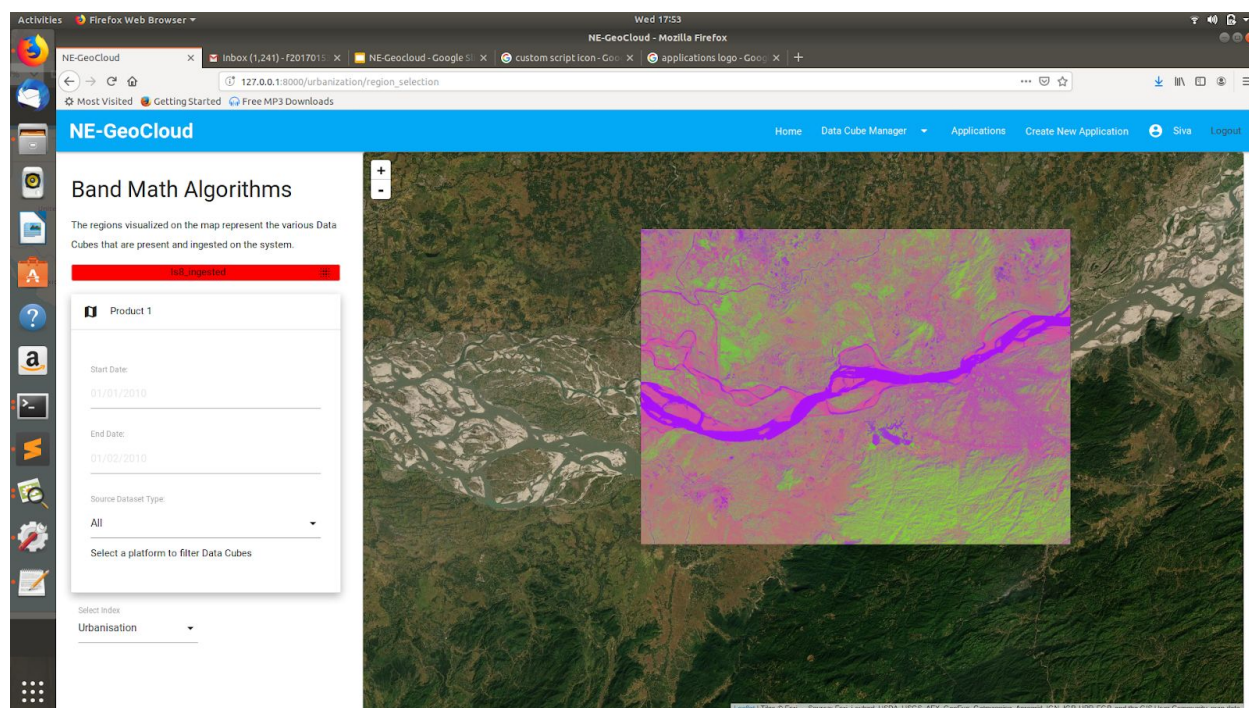
```
# <br>
# # NDWI
#
# <br>
# > ** NDWI Normalized Difference Water Index **
# > A derived index that correlates well with the existence of water.
# <br>
#
# $$ NDWI = \frac{GREEN - NIR}{GREEN + NIR} $$
#
# <br>

# In[3]:

def NDWI(dataset):
    return (dataset.green - dataset.nir)/(dataset.green + dataset.nir)
```



*(Fig 4.1.3) NDWI Output*

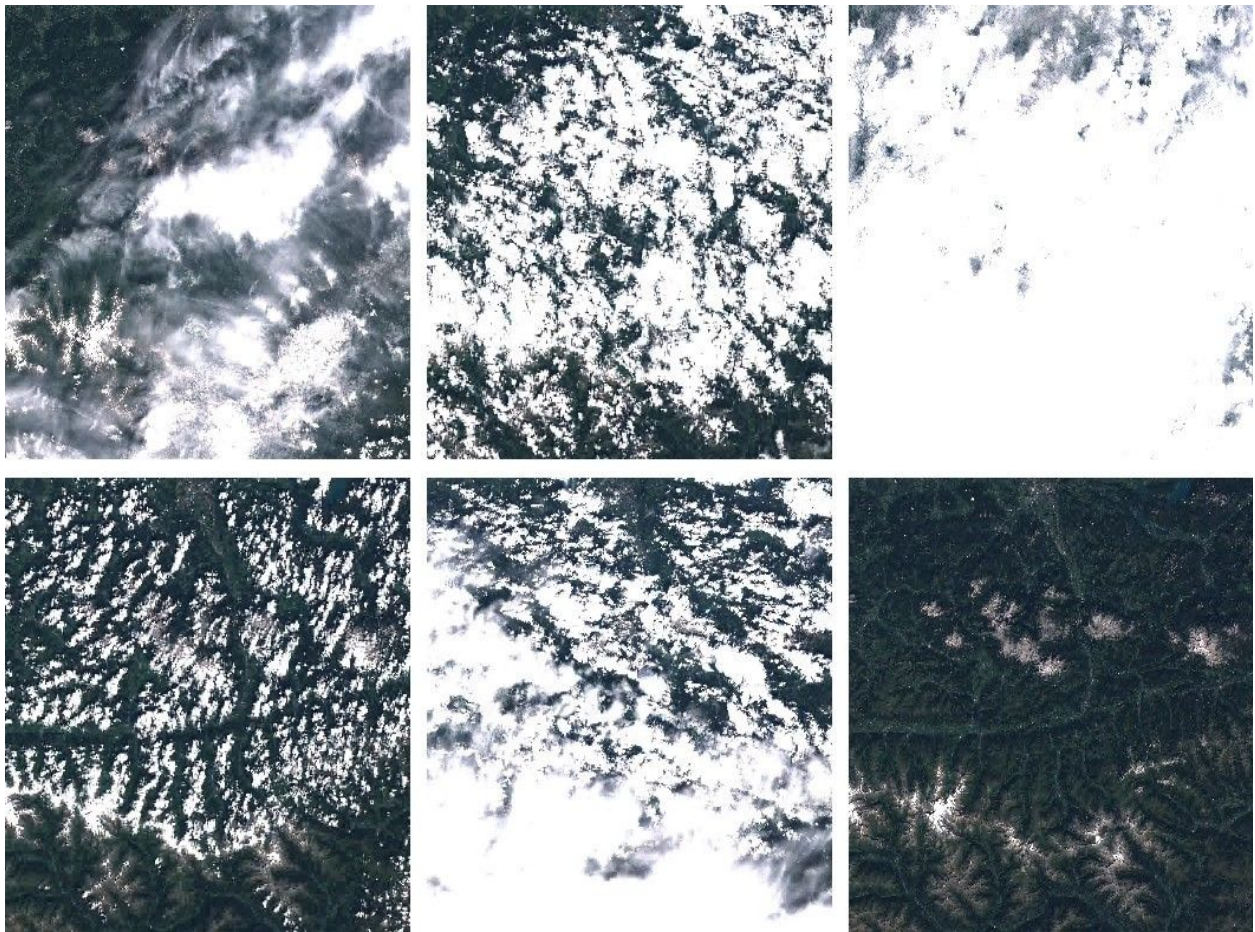


*(Fig 4.1) Band Math Algorithm using NDVI+NDBI+NDWI*



## 4.2. Cloud Free Mosaic

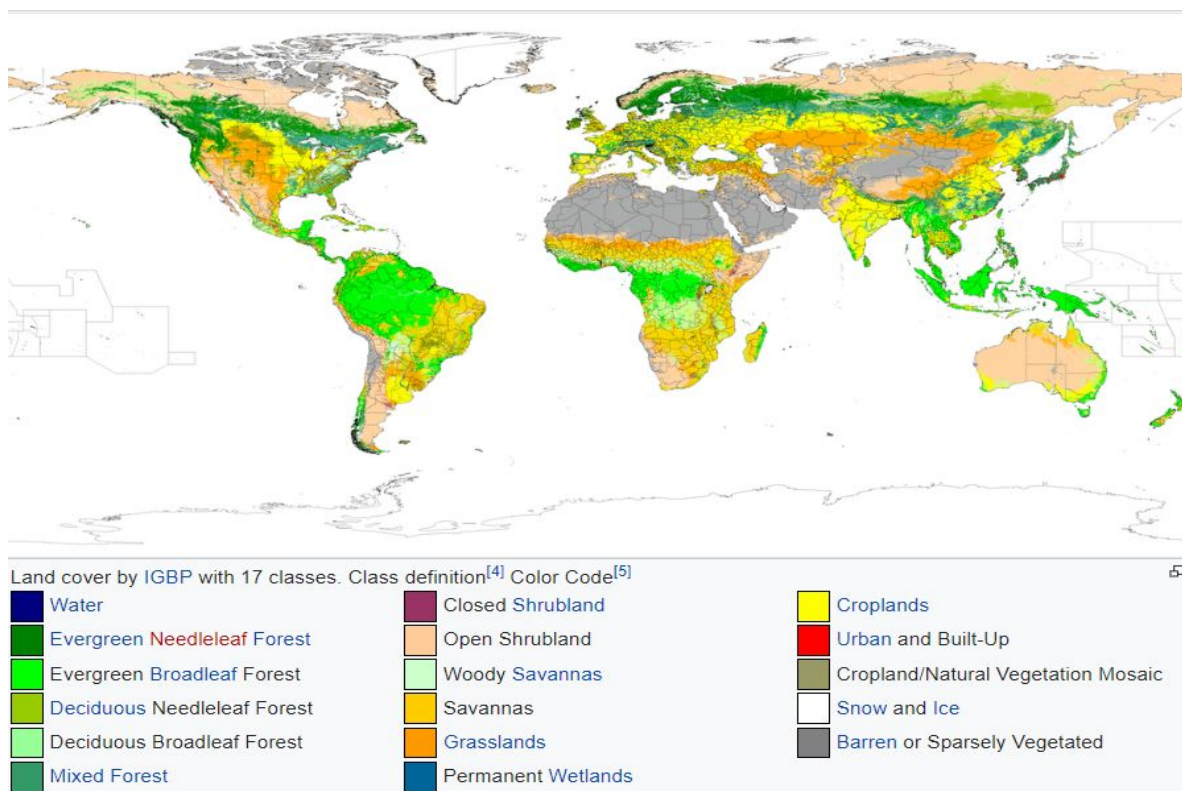
- Cloud Free Mosaic is another band math operation like NDVI
- It works on all pixels in the image and removes those where there is no cloud.



*(Fig 4.2.1) Cloud Free Mosaic*

## 4.3. Land Cover Classification

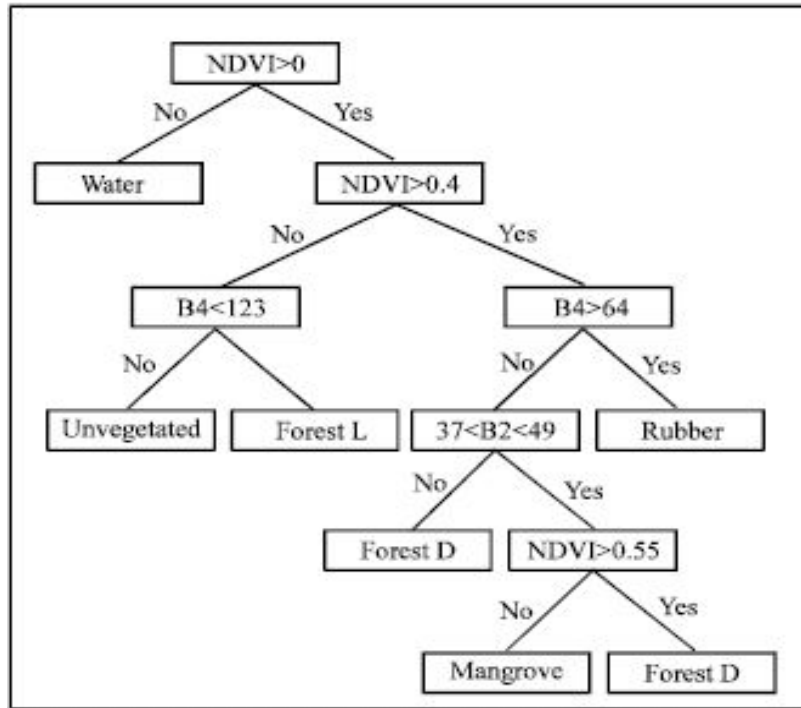
- Land cover is the physical material at the surface of the earth. Land covers include grass, asphalt, trees, bare ground, water, etc.
- There are two primary methods for capturing information on land cover: field survey and analysis of remotely sensed imagery. Land change models can be built from these types of data to assess future shifts in land cover.
- In this project we have tried to classify the land cover into various classes using the Random Forest Classification techniques.



(Fig 4.3.1) Example of Land Cover Classification

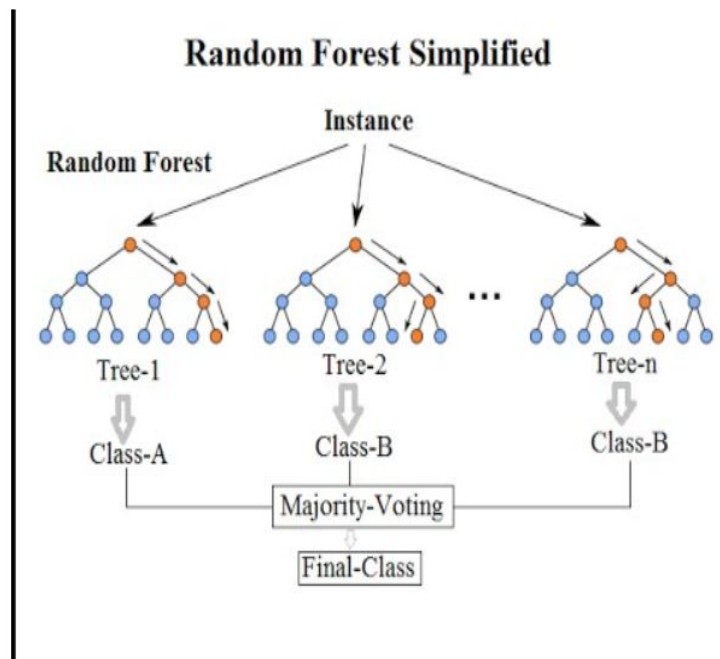
## 5. Random Forest Classification

A brief explanation of the RandomForest algorithm comes from the name. Rather than utilize the predictions of a single decision tree, the algorithm will take the ensemble result of a large number of decision trees (a forest of them). The "Random" part of the name comes from the term "bootstrap aggregating", or "bagging". What this means is that each tree within the forest only gets to train on some subset of the full training dataset (the subset is determined by sampling with replacement). The elements of the training data for each tree that are left unseen are held "out-of-bag" for estimation of accuracy. Randomness also helps decide which feature input variables are seen at each node in each decision tree. Once all individual trees are fit to a random subset of the training data, using a random set of feature variable at each node, the ensemble of them all is used to give the final prediction.



(Fig 5.1) Example of Random Forest Classification

Forest: Collection of trees



(Fig 5.2) Example of Simplified Random Forest Classification

We use the Gini Index as our cost function used to evaluate splits in the dataset. We minimize it. A split in the dataset involves one input attribute and one value for that attribute. It can be used to divide training patterns into two groups of rows. A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a Gini score of 0, whereas the worst case split that results in 50/50 classes in each group results in a Gini score of 1.0 (for a 2 class problem). We calculate it for every row and split the data accordingly in our binary tree. We repeat this process recursively.

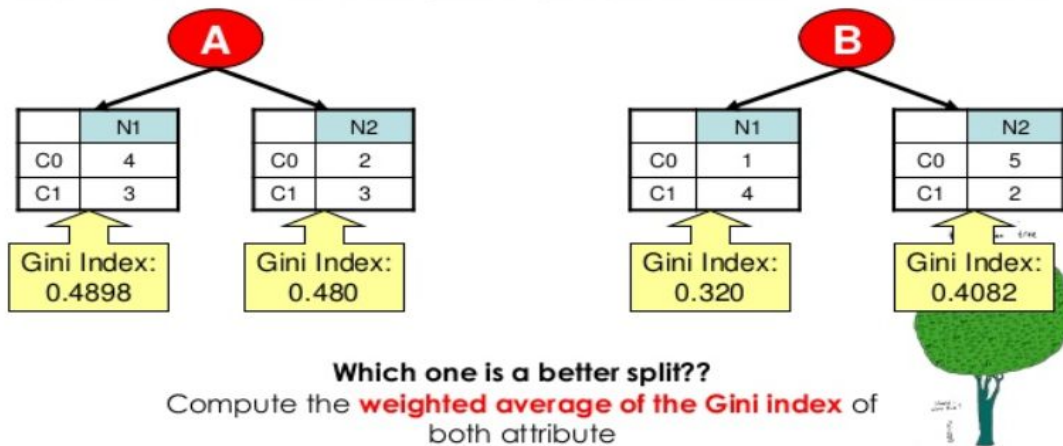
### Splitting Binary Attributes (using Gini)

Example :

	Parent
C0	6
C1	6
Gini = 0.5	

$$\text{Gini : } 1 - (6/12)^2 - (6/12)^2 = 0.5$$

Suppose there are two ways (A and B) to split the data into smaller subset.



(Fig 5.3) Gini Coefficient



**Advantages of Random Forest:**

- It gives you a measure of "variable important" which relates how useful your input features (e.g. spectral bands) were in the classification
- The "out-of-bag" samples in each tree can be used to validate each tree. Grouping these predicted accuracies across all trees can sometimes give you an unbiased estimate of the error rate similar to doing cross - validation.
- Can be used for regressions, unsupervised clustering, or supervised classification
- Available in many popular languages, including Python, R, and MATLAB
- Free and open source, and fast

In the classification mode, this means that if you were to have 5 classes being predicted using 500 trees, the output prediction would be the class that has the most number of the 500 trees predicting it. The proportion of the number of trees that voted for the winning class can be a diagnostic of the representativeness of your training data relative to the rest of the image. Taking the 500 trees example, if you have pixels which are voted to be in the "Forest" land cover class by 475 of 500 trees, you could say that this was a relatively certain prediction. On the other hand, if you have a pixel which gets 250 votes for "Forest" and 225 votes for "Shrub", you could interpret this as either an innately confusing pixel (maybe it is a mixed pixel, or it is a small statured forest) or as an indicator that you need more training data samples in these types of pixels.

## 6. Python Notebook Creation for Classification

### GDAL - Geospatial Data Abstraction Library

Library with useful functions to read raster data. Some useful functions are:

- Open
- GetRasterBand
- GetGeoTransform

```
geotransform[0] = top left x
geotransform[1] = w-e pixel resolution
geotransform[2] = 0
geotransform[3] = top left y
geotransform[4] = 0
geotransform[5] = n-s pixel resolution (negative value)
```

*(Fig 6.1) GetGeoTransform Indexes*

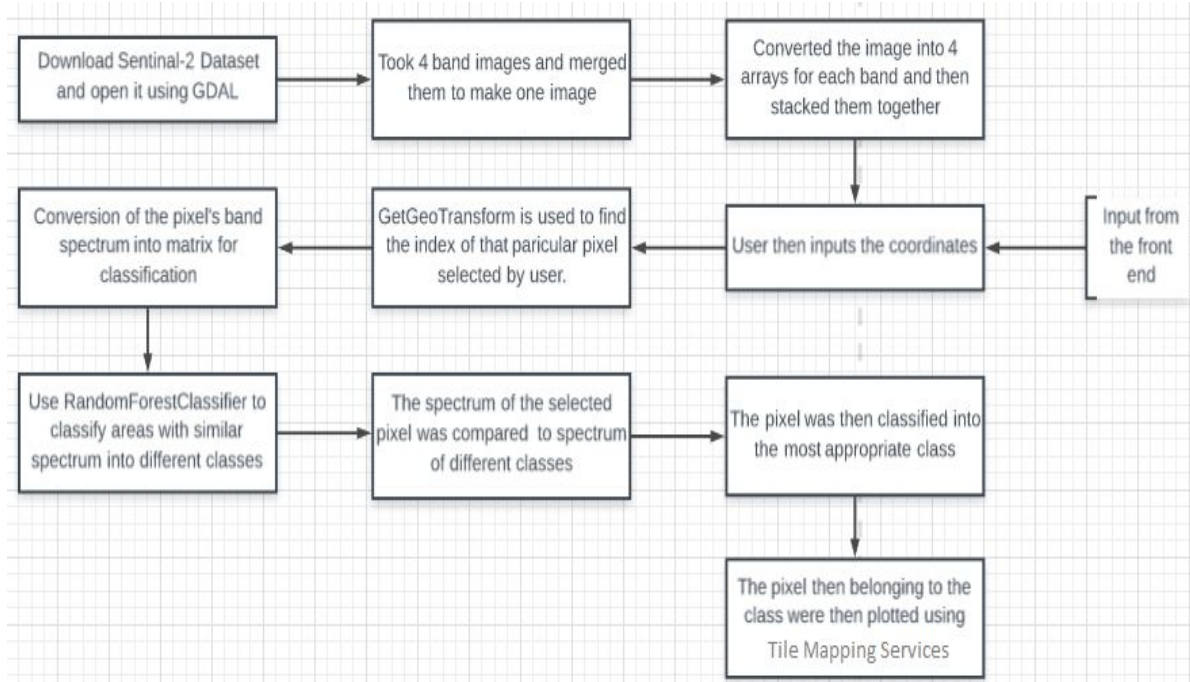
### What is Matplotlib?

- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats.

### What is Numpy?

NumPy is the fundamental package for scientific computing with Python:

- A powerful N-dimensional array object



(Fig 6.2) Notebook Workflow

```

1 import gdal
2 import numpy as np
3 from sklearn.ensemble import RandomForestClassifier
4 import matplotlib.pyplot as plt
5
6 img_ds = gdal.Open("/home/localuser/Downloads/S2B_MSIL1C_20190117T043119_N0207_R133_T46RCP_20190117T072349.SAFE/Geo/sentinel_stack_geo_subset.img")
7 band2= img_ds.GetRasterBand(1).ReadAsArray()
8 band3 = img_ds.GetRasterBand(2).ReadAsArray()
9 band4= img_ds.GetRasterBand(3).ReadAsArray()
10 band8 = img_ds.GetRasterBand(4).ReadAsArray()
11 x=img_ds.RasterXSize
12 y=img_ds.RasterYSize
13 gt=img_ds.GetGeoTransform()
14 minx=gt[0]
15 maxy=gt[3]
16 x_res=gt[1]
17 y_res=gt[5]
18 ans=np.stack((band2,band3,band4,band8),0)
19
20 latitude= float(input("Enter the Latitude: "))
21 longitude=float(input("Enter the Longitude: "))
22 horizontal=longitude-minx;
23 vertical=maxy-latitude;
24 horizontal/=x_res
25 vertical/=y_res
26 horizontal=int(horizontal)
27 vertical=int(vertical*-1)
28 roi = np.insert(ans[:, vertical, horizontal], 0, 1, axis=0)

```

(Fig 6.3) Notebook Code Snippet 1

```

29
30 latitude2= float(input("Enter the Latitude: "))
31 longitude2=float(input("Enter the Longitude: "))
32 horizontal2=longitude2-minx;
33 vertical2=maxy-latitude2;
34 horizontal2/=x_res
35 vertical2/=y_res
36 horizontal2=int(horizontal2)
37 vertical2=int(vertical2*-1)
38 roi2 = np.insert(ans[:, vertical2, horizontal2], 0, 2, axis=0)
39
40 latitude3= float(input("Enter the Latitude: "))
41 longitude3=float(input("Enter the Longitude: "))
42 horizontal3=longitude3-minx;
43 vertical3=maxy-latitude3;
44 horizontal3/=x_res
45 vertical3/=y_res
46 horizontal3=int(horizontal3)
47 vertical3=int(vertical3*-1)
48 roi3 = np.insert(ans[:, vertical3, horizontal3], 0, 3, axis=0)
49
50 latitude4= float(input("Enter the Latitude: "))
51 longitude4=float(input("Enter the Longitude: "))
52 horizontal4=longitude4-minx;
53 vertical4=maxy-latitude4;
54 horizontal4/=x_res
55 vertical4/=y_res
56 horizontal4=int(horizontal4)
57 vertical4=int(vertical4*-1)
58 roi4 = np.insert(ans[:, vertical4, horizontal4], 0, 4, axis=0)
59
60 new=np.stack((roi,roi2,roi3,roi4),0)
61 X=new[0:4,1:5]
62 y=[1,2,3,4]
63
64 rf = RandomForestClassifier(n_estimators=500, oob_score=True)
65 rf = rf.fit(X, y)

```

(Fig 6.4) Notebook Code Snippet 2

```

66
67 new_shape = (ans.shape[1] * ans.shape[2], ans.shape[0])
68 img_as_array = ans[:, :, :].reshape(new_shape)
69 class_prediction = rf.predict(img_as_array)
70 class_prediction = class_prediction.reshape(ans[0, :, :].shape)
71
72 def color_stretch(image, index, minmax=(0, 10000)):
73     colors = image[:, :, index].astype(np.float64)
74     max_val = minmax[1]
75     min_val = minmax[0]
76     colors[colors[:, :, :] > max_val] = max_val
77     colors[colors[:, :, :] < min_val] = min_val
78     for b in range(colors.shape[2]):
79         colors[:, :, b] = colors[:, :, b] * 1 / (max_val - min_val)
80     return colors
81
82 img543 = color_stretch(ans, [4, 3, 2], (0, 8000))
83 n = class_prediction.max()
84
85 # Next setup a colormap for our map
86 colors = dict((
87     (2, (0, 150, 0, 255)), # Forest
88     (1, (0, 0, 255, 255)), # Water
89     (3, (0, 255, 0, 255)), # Herbaceous
90     (4, (160, 82, 45, 255)) # Barren
91 ))
92
93 for k in colors:
94     v = colors[k]
95     v = [ v / 255.0 for _v in v]
96     colors[k] = _v
97
98 index_colors = [colors[key] if key in colors else
99                 (255, 255, 255, 0) for key in range(1, n + 1)]
100 cmap = plt.matplotlib.colors.ListedColormap(index_colors, 'Classification', n)

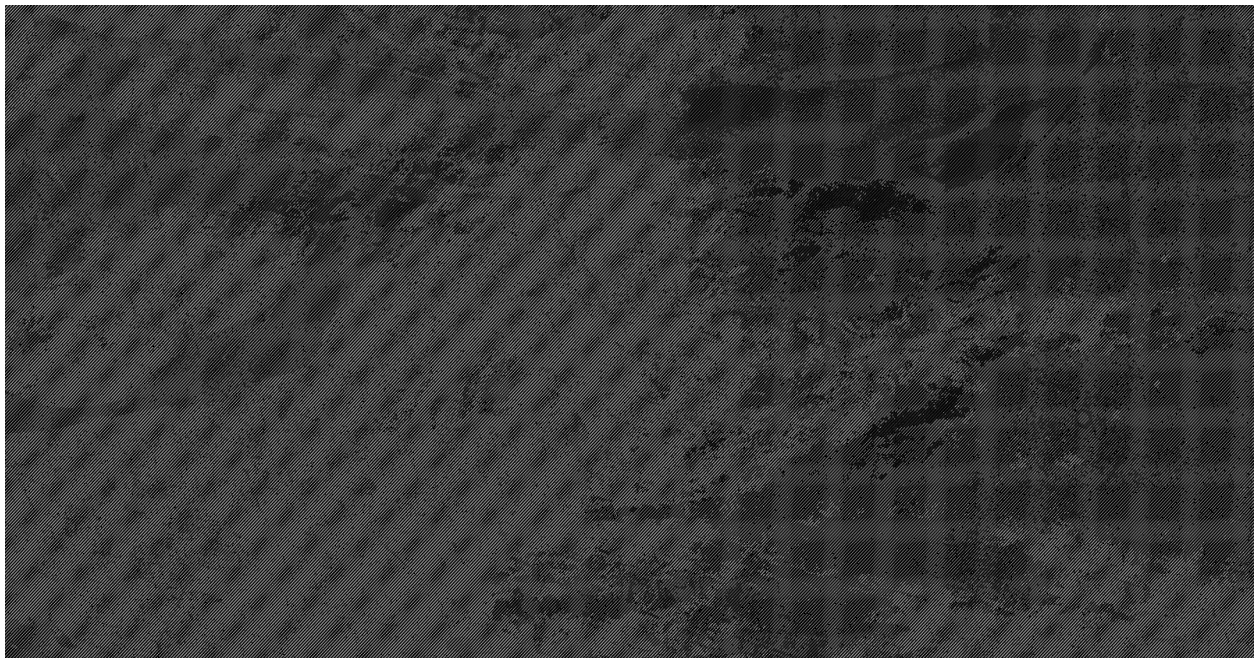
```

(Fig 6.5) Notebook Code Snippet 3





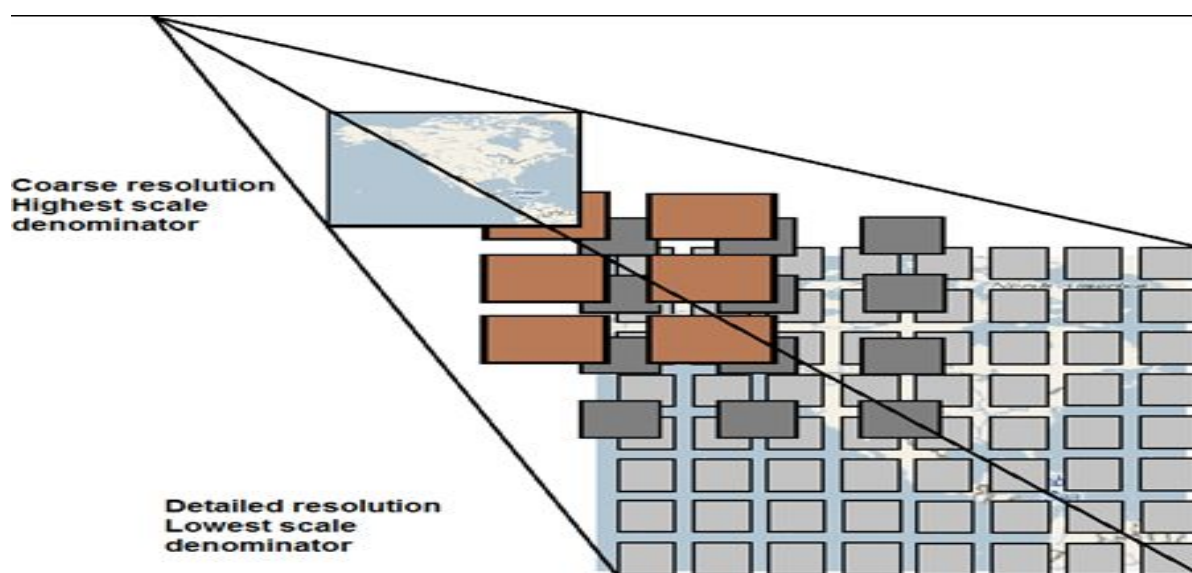
*(Fig 6.6) Image of Guwahati taken from Sentinel-2*



*(Fig 6.7) Output Image of Guwahati after Classification*

## 7. Tiling Services

**Tile Map Service** or TMS, is a specification for tiled web maps, developed by the Open Source Geospatial Foundation. The definition generally requires a URI structure which attempts to fulfill REST principles. The TMS protocol fills a gap between the very simple standard used by OpenStreetMap and the complexity of the Web Map Service standard, providing simple urls to tiles while also supporting alternate spatial We used tiling in order to render the resultant images of our applications on the leaflet maps seamlessly and on the fly.



(Fig 7.1) Tiling Services



## 8. Datasets and Tools Used

The Data Cube UI is a full stack Python web application used to perform analysis on raster datasets like Landsat 7 and GPM using the Data Cube. Our UI is a good tool for demonstrating the Data Cube capabilities and some possible applications and architectures. The UI's core technologies are:

- [HTML/CSS/Javascript](#): Core front-end web development
- [Django](#): Backend.
- [Data Cube](#): API for data access and analysis
- [PostgreSQL](#): Database backend for both the Data Cube
- [Bootstrap 3/Materialize](#): Simple, standard, and easy front end styling
- [RasterIO/GDAL](#): Geospatial imagery processing
- [Sklern, Matplotlib, Numpy](#): Machine Learning and Data Processing



(Fig 8.1) Tools Used



## 9. Challenges

- Setup and interoperability between a plethora of various languages, frameworks and libraries was challenging to carry out across different working systems.
- Obtaining and passing data between the front-end, back-end local databases and Python scripts required in-depth knowledge of data structure and format at each stage
- Limited processing power on personal machines as well as large volumes of satellite data required more efficient implementations of nearly each step of execution.
- Running nearly identical algorithms on data from different satellite sources required manipulation of ingestion configurations.

## 10. Discussion and Scope

- Further applications and utilities such as cloud coverage, mine detection, road maps, etc.
- Future expansion and implementation of more powerful GPUs and servers can allow for processing of large volumes of ingested satellite data off-system.
- Dedicated implementations of multiprocessing and scheduling will allow numerous users to run applications at once.
- Mosaicing and tiling of large geographical regions will allow rapid execution of applications on any user-defined regions.
- Implementation of deep learning algorithms like Convolutional Neural Networks for intelligent and more detailed predictions and classification, including object detection.
- Expansion of the Datacube backend through creation of more custom data ingestion scripts will allow wider compatibility of the database with various satellites, including Resoursat, MODIS, etc.
- Useful as a research tool to analyze sensitive satellite data that cannot be uploaded to similar online portals.

## References

- J. Ross, B. Killough, T. Dhu, M.Paget, *Open Data Cube and the Committee on Earth Observation Satellites DataCube Initiative, IAC, 2017.*
- Brian Killough, *Overview of the open data cube initiative, 2019.*  
[https://www.researchgate.net/publication/328995502\\_Overview\\_of\\_the\\_Open\\_Data\\_Cube\\_Initiative](https://www.researchgate.net/publication/328995502_Overview_of_the_Open_Data_Cube_Initiative)
- *Open Data Cube Website:*  
<https://www.opendatacube.org>
- *Open Data Cube GitHub Repository:*  
<https://github.com/opendatacube>
- *Open Data Cube web-based User Interface:*  
<http://ec2-52-201-154-0.compute-1.amazonaws.com>
- *GDAL Documentation:*  
<https://gdal.org/>