School of Engineering and Computer Science

Final Thesis Report

"Observing Mega Cities to Grow from Space and Predicting Its Growth Using Machine Learning Techniques"

Submitted by:

Mitesh Chakma - 13101082

Sadia Zannat - 12201045

Thesis Supervisor:

Dr. Md. Khalilur Rhaman

TABLE OF CONTENTS

Declaration

This is to certify that the following research work titled "Observing Mega Cities to Grow from Space and Predicting Its Growth Using Machine Learning Techniques" is submitted by Mitesh Chakma and Sadia Zannat to the Department of Computer Science and Engineering, BRAC University. We hereby declare that; this thesis is based on results found from our own work. Materials of work found by other researcher are mentioned by reference. We carried our research under the supervision of Dr. Md. Khalilur Rhaman.

Supervisor's Signature

_____

Dr. Khalilur Rhaman

Author's Signature

_____ _____

Mitesh Chakma Sadia Zannat

Acknowledgement

The work presented in this thesis would not have been possible without the uttermost support of the family members and the respected supervisor Dr. Md. Khalilur Rahman. It would have been difficult to bring this work towards a completion without his guidance, enormous encouragement, and continuous support.

Lastly, we convey gratitude from the depth of our hearts towards the faculty members of department of Computer Science and Engineering for providing us with all the knowledge throughout our undergraduate life so that we could pursue this research.

ACRONYMS AND ABBREVIATION

NDVI = Normalized Index Vegetation Index

NDBI= Normalized Index Built-up Index

NDWI= Normalized Index Water Index

LULC= Land Cover and Land Cover Changes

ETM+ = Enhanced Thematic Mapper Plus

OLI = Operational Land Imager

GIS = Geographical Information System

USGS= U.S. Geological Survey

ANN = Artificial Neural Network

RNN= Recurrent Neural Network

ERTS = Earth Resources Technology Satellite

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Overview

This section discusses our research work briefly and gives a complete overview of the work.

1.1 Abstract

In recent years, developing countries are undergoing a massive change in growth in the urbanization. Urbanization is yielding many positive effects for these developing countries in infrastructure, economy, social status and so on. However, this extensive urbanization mostly resulting some adverse effects. This study aims to evaluate and observe the various changes like vegetation, built-up and water in the urban area of the greater Dhaka area of Bangladesh which has one of the highest growth rate among developing countries using Landsat 7 ETM+ and Landsat 8 OLI images between 2012 to 2018 using different GIS and remote sensing techniques. The analysis shows the superabundant growth of the buildup areas as well as degradation in the vegetated and water areas of greater Dhaka, Bangladesh. Based on such data, different machine learning techniques have been applied to show the performance in terms of accuracy and forecast a predicting rate for the future growth of Dhaka city. The experimental results provides an idea of the subsequent changes in terms

of water body, vegetation and built-up areas for Dhaka city, with an aim to provide these useful information for the policy makers and urban planners.

1.2 Introduction

Rapid changes in LULC (land use and land cover) in urban areas becoming a major problem for the mega cities like Dhaka, Bangladesh. The conversion for this rapid change are currently occurring at an unprecedented rate. Only monitoring and proper management plans can solve the emerging problems timely and effectively. LULC change due to human activities is currently proceeding more quickly in developing countries than the developed world and proceeding more quickly in developing countries than the developed world, and it has been projected that by the year 2020, most of the world's mega cities will be from developing countries [1]. By using satellite images and their data's we can measure the subsequent changes and take the necessary procedure for the upcoming changes in urban expansion.

Bangladesh has witnessed a rapid change in the urban growth in recent times. Dhaka, the capital of Bangladesh is the prime center for this change. However, with astonishing growth in population of 3 million in 1980's to over 16 million, Dhaka has become one the most densely populated cities in the world where its expansion is disordered and irregular. This growth

however has placed without proper planning which is resulting in a city with inadequate urban structure, contraction of river banks, vegetated area and so on [2]. Remote sensing and GIS (Geographic Information Systems) are powerful yet most cost-effective tools for accessing the spatial and the temporal dynamics of LULC [3]. Multi-temporal data are provided through Remote sensing which is useful for the processes and the patterns of LULC change whereas GIS is useful for mapping and analyzing those patterns [4]. Most developed countries have both recent and extensive LULC information, but such scenario is prevalent for developing countries, particularly Bangladesh. For instance. City does not have official land uses patterns or proper statistics, nor the master plans contain maps or quantitative information's on the existing pattern of land uses map in the city[4].Therefore , this study is an attempt to identify the spatial and temporal dynamics of LULC changes of greater Dhaka to identify a scientific approach to LULC change detection and for the policy makers to access the LULC change pattern in this area of Bangladesh.

The objective for this approach is to collect, record, analyze and to explore multi-temporal data of greater Dhaka using Landsat ETM+ and OLI images ranging from 2012 to 2018 and to predict the future phases of changes in the urban expansion using machine learning techniques.

1.3 Study Area

As shown in figure 1, Dhaka, the capital of Bangladesh, has been chosen as the study area for this research. As of World Bank data, this city has over 16 million people while the city itself has a population estimated at about 8.5 million with population growth of 4.2% annually and expected to be the third largest city in the world by 2020 [4] and the rapid urban expansion experienced by the city in recent decades is one of the highest in the world[6]. This city is in the center of Bangladesh between 23° 55`N and 90° 23`E and 23°39`N and 90° 26`E respectively. The study area surrounded by four river systems: The Bur-iganga, Turag, Tongi and the Balu, which flow to the south, west, north, and east, respectively. It is one of the most densely populated areas and one of the fastest megacities around the world carrying a density of 23,000 people living per square kilometer within a total area of approximate 300 square kilometers.

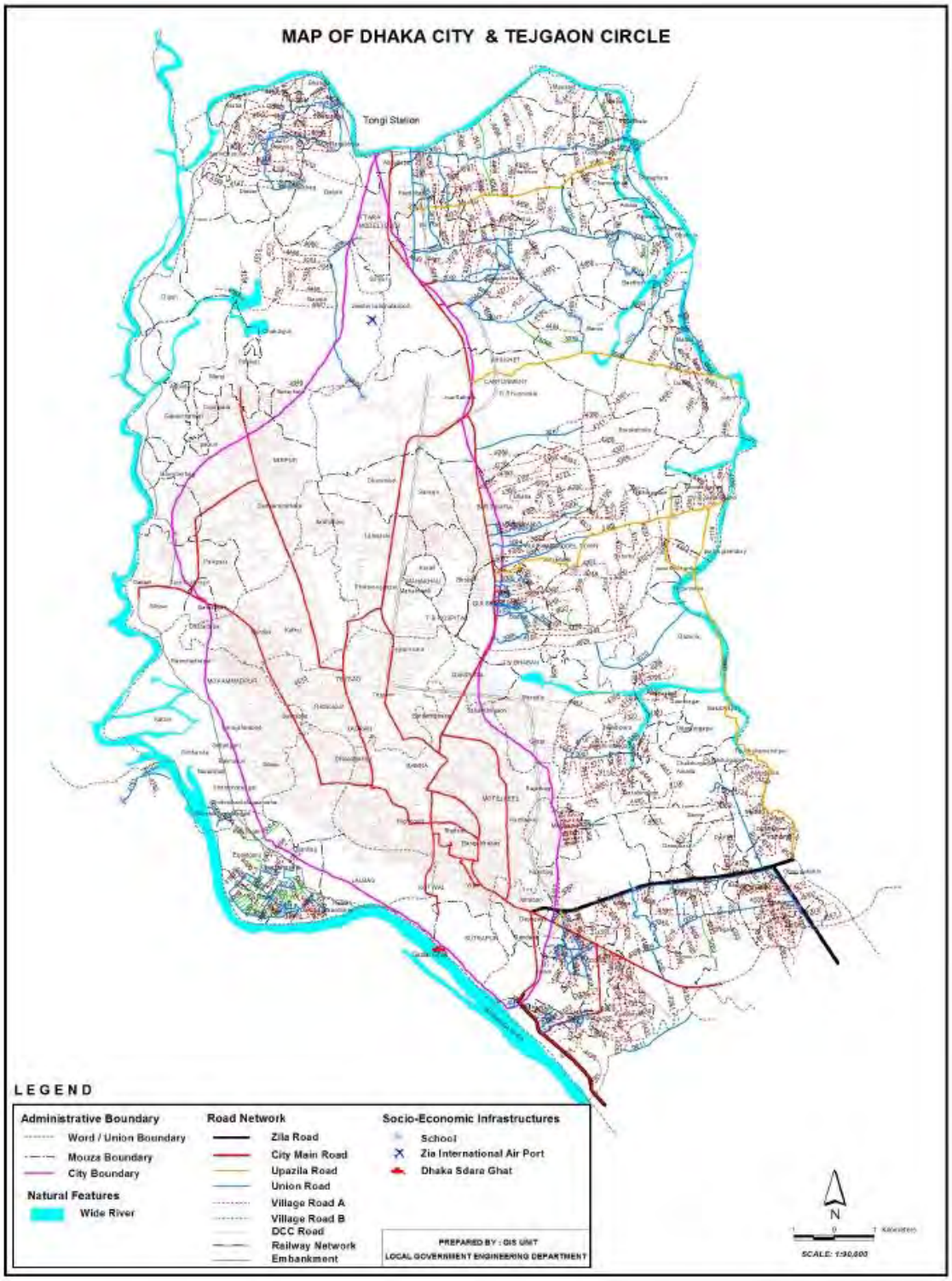


Figure 1. Map of Dhaka City

1.4 Thesis Objective

The objective of this thesis to create a database of subsequent LULC changes based on four different classifications namely, Built-up area, vegetation, water index and bare soil and to predict the future growth patterns based on such classifications using RNN techniques.

1.5 Thesis Contribution Summary

The summary of the main contributions are as follows-

There are only few research has been done on observing the growth of Dhaka, Bangladesh using GIS and remote sensing techniques and on those few researches, the predicting growth has been done by raw data and some unknown calculations. Our study is different form those as we are creating our data sets using recent data and those data are collected and analyzed by us. Moreover, for evaluation process we are generating two Landsat data ranging from 2012-2018 and the growth pattern has been evaluated by different machine learning techniques and predicted growth has been done by Neural network technique.

Chapter 2

DATA ACQUISITION

This chapter focuses on from where we collected our images, types of images to extract data and reason behind choosing them.

2.1 United State Geological Survey (USGS)

The USGS or United States Geological Survey is a scientific agency of the United States Government. This organization study the natural resources and the natural hazards that threaten it. The current moto of USGS is "Science for a changing world".

2.2 Landsat Program

Landsat program represents the worlds longest continuously acquired collection of space based moderate-resolution land remote sensing data. Four decades of imaginary provides a unique resource for those who work in agriculture, geology, forestry, regional planning, education, mapping, and global change research. Landsat images are also invaluable for emergency response and disaster relief. As a joint initiative between the U.S. Geological Survey (USGS) and NASA, the Landsat Project, and the

data it collects support government, commercial, industrial, civilian, military, and educational communities throughout the United States and worldwide.

At over 40 years, the Landsat series of satellites provides the longest temporal record of moderate resolution multispectral data of the Earth's surface on a global basis. The Landsat record has remained remarkably unbroken, proving a unique resource to assist a broad range of specialists in managing the world's food, water, forests, and other natural resources for a growing world population. It is a record unmatched in quality, detail, coverage, and value.

Figure 2. Logo of USGS and NASA

2.2.a Landsat Missions

In the mid-1960s, stimulated by U.S. successes in planetary exploration using unmanned remote sensing satellites and a September 21, 1966 announcement by the Department of the Interior Secretary Stewart Udall, the Department of the Interior, NASA, and the Department of Agriculture embarked on an ambitious effort to develop and launch the first civilian Earth observation satellite. Their goal was achieved on July 23, 1972, with the launch of the Earth Resources Technology Satellite (ERTS-1), which

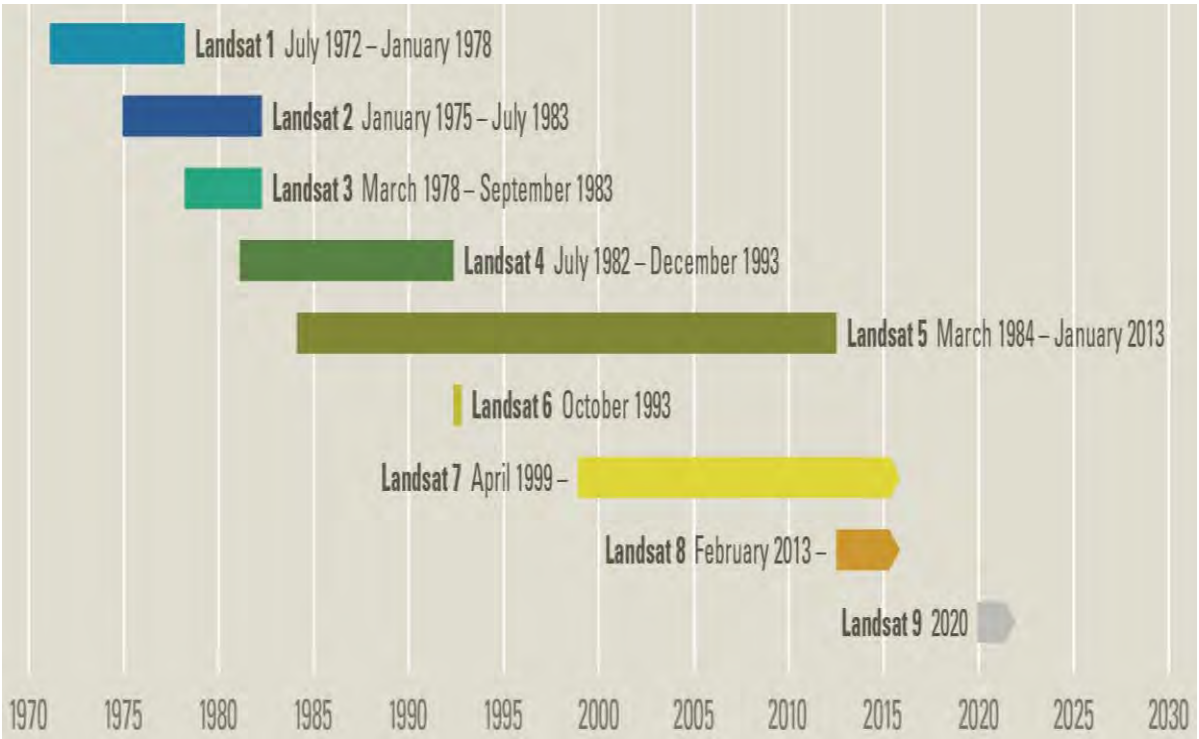


Figure 3. Landsat Missions

was later renamed Landsat 1. The launches of Landsat 2, 3, 4 followed in year 1975, 1978, and 1982.

2.2.b Image Collection

All the images collected and shown in this research has been collected from Landsat 7 and Landsat 8 and those images are openly distributed in USGS website for research purpose. The number images collected from USGS are as follows in Table 1–

Satellite Data	Spatial Resolution (meters)	Date of Acquired data	No. Images
Landsat 7 ETM+	30	16 Dec-2013; 22 Dec-2016; 27 Dec-2013	26
Landsat 8 OLI	15	24 Dec-2013; 30 Dec 2015; 03 Dec-2017	33

Table 1. Collection of Satellite Images

2.3 Reasons for Choosing

The reasons for choosing Landsat program –

1. Landsat program contains images from four decades.

2. The most resourceful platform for geo-scientists.

3. Openly distributed which is free to use for research purpose.

The USGS delivers high quality systematic, geometric, radiometric, and terrain corrected data to users worldwide, and since December 2008, without any cost to users. Furthermore, The Landsat 7 and Landsat 8 satellites both orbit the Earth at an altitude of 705 kilometers(438 miles) in a 185-kilometer (115-mile) swath, moving from north to south. Each satellite makes complete orbit every 99 minutes, completes about 14 full orbits each day, and crosses every point on Earth once every 16 days. Although each satellite has a 16-day full-Earth-coverage cycle, their orbits are offset to allow 8-day repeat coverage of any Landsat scene area on the globe. Between the two satellites, more than 1,000 scenes are added to the USGS archive each day. Landsat 4 and 5 followed the same orbit as Landsat 7 and 8, whereas Landsat 1, 2, and 3 orbited at an altitude of 920 kilometers (572 miles), circling the earth every 103 minutes, yielding repeat coverage every 18 days [7].

Chapter 3

Software's

This section discusses the software's we have used for our research.

For image stacking and segmentation and processing-

-ERDAS Imagine 2015

-ArcMap 10.1

--MATLAB

For machine learning -

-Anaconda

-Jupyter Notebook

ERDAS IMAGINE supplies tools for remote sensing, photogrammetry, and GIS needs. We have used ERDAS (Figure 4) for processing our image data.

Figure 4. ERDAS Imagine

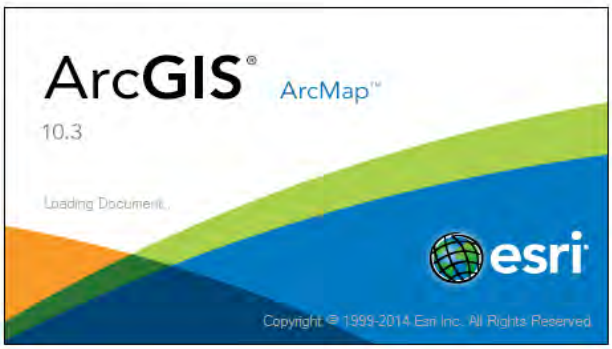


Figure 5. ArcGIS

ArcGIS (Figure 5) is a geographic information system (GIS) for working with maps and geographic information. We have used this for analyzing mapped information, using maps and geographic information in a range of applications.

Figure 6. MATLAB

MathWorks (Figure 6) is a mathematical computing software. We used it for data analysis and simulations. We used ANACONDA (Figure 7) for machine learning calculations, as it is an open source platform of the Python and R programming languages for large-scale data processing.

Figure 7. ANACONDA

Chapter 4

Data Analysis

This chapter will describe the work methodology of our research briefly and gives complete overview of the work.

4.1 Working Process

Our working procedure flowchart (Figure 8) is below-

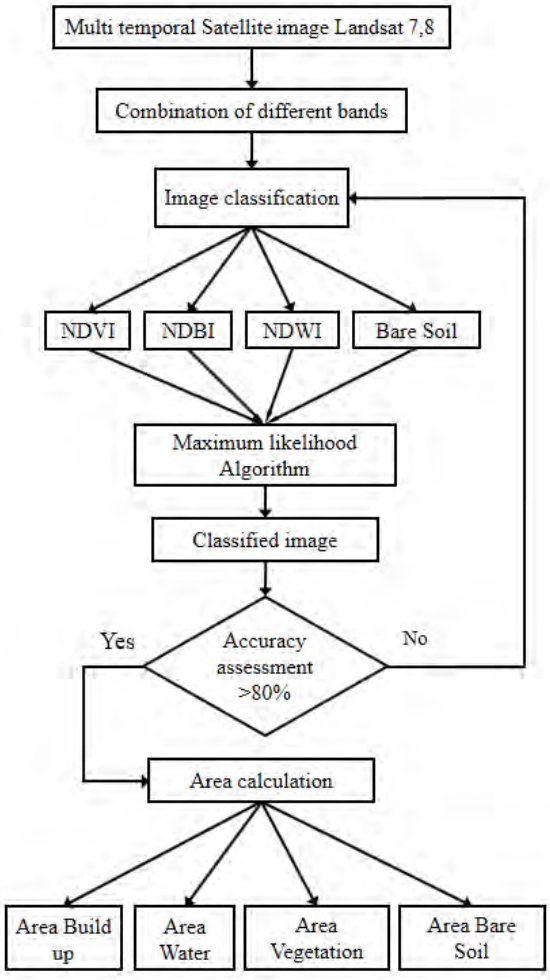


Figure 8. Working process flowchart

4.1.a Stacking Images

We have used Layer stacking process to combine multiple image bands into a single multispectral image file. The images we got from USGS is not a single image file rather it is a combination of separate spectral bands, which have different spatial resolution. We have used Landsat 7 and Landsat 8 images. Landsat 7 images consist of seven spectral bands (Table 2) with a resolution of 30 meters for bands 1-7. The resolution of band 8 is 15 meters, which is

	Landsat 7	Wavelength (micrometers)	Resolution (meters)
Enhanced Thematic Mapper Plus (ETM+)	Band 1	0.45-0.52	30
	Band 2	0.52-0.60	30
	Band 3	0.63-0.69	30
	Band 4	0.77-0.90	30
	Band 5	1.55-1.75	30
	Band 6	10.40-12.50	60 * (30)
	Band 7	2.09-2.35	30
	Band 8	.52-.90	15

Table 2. Landsat 7 band specification

panchromatic band. Band 6, a thermal infrared channel has 60-meter spatial resolution. Landsat 8 (Table 3) has nine spectral bands with a spatial resolution of 30 meters for bands 1 to 7 and 9. The panchromatic

(band 8) has 15-meter spatial resolution. Band 10 and 11 (Thermal bands) has 100-meter resolution. The software we have for stacking images is ERDAS Imagine 2015. It took nearly 3-7 minutes.

	Bands	Wavelength (micrometers)	Resolution (meters)
Landsat 8 Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) Launched February 11, 2013	Band 1 - Coastal aerosol	0.43 - 0.45	30
	Band 2 - Blue	0.45 - 0.51	30
	Band 3 - Green	0.53 - 0.59	30
	Band 4 - Red	0.64 - 0.67	30
	Band 5 - Near Infrared (NIR)	0.85 - 0.88	30
	Band 6 - SWIR 1	1.57 - 1.65	30
	Band 7 - SWIR 2	2.11 - 2.29	30
	Band 8 - Panchromatic	0.50 - 0.68	15
	Band 9 - Cirrus	1.36 - 1.38	30
	Band 10 - Thermal Infrared (TIRS) 1	10.60 - 11.19	100
	Band 11 - Thermal Infrared (TIRS) 2	11.50 - 12.51	100

Table 3. Landsat 8 band specification

After that, we applied subset or clipping. This to separate the area of interest from the original raster dataset. We have used the reference of Dhaka city map to generate a study map which is based on the actual Dhaka map and we made a shape file out of it using ArcMap 10.1 software. Below is the figure of that shape file we have produced.

Below in Figure 9, shown a diagram of stacking procedure.

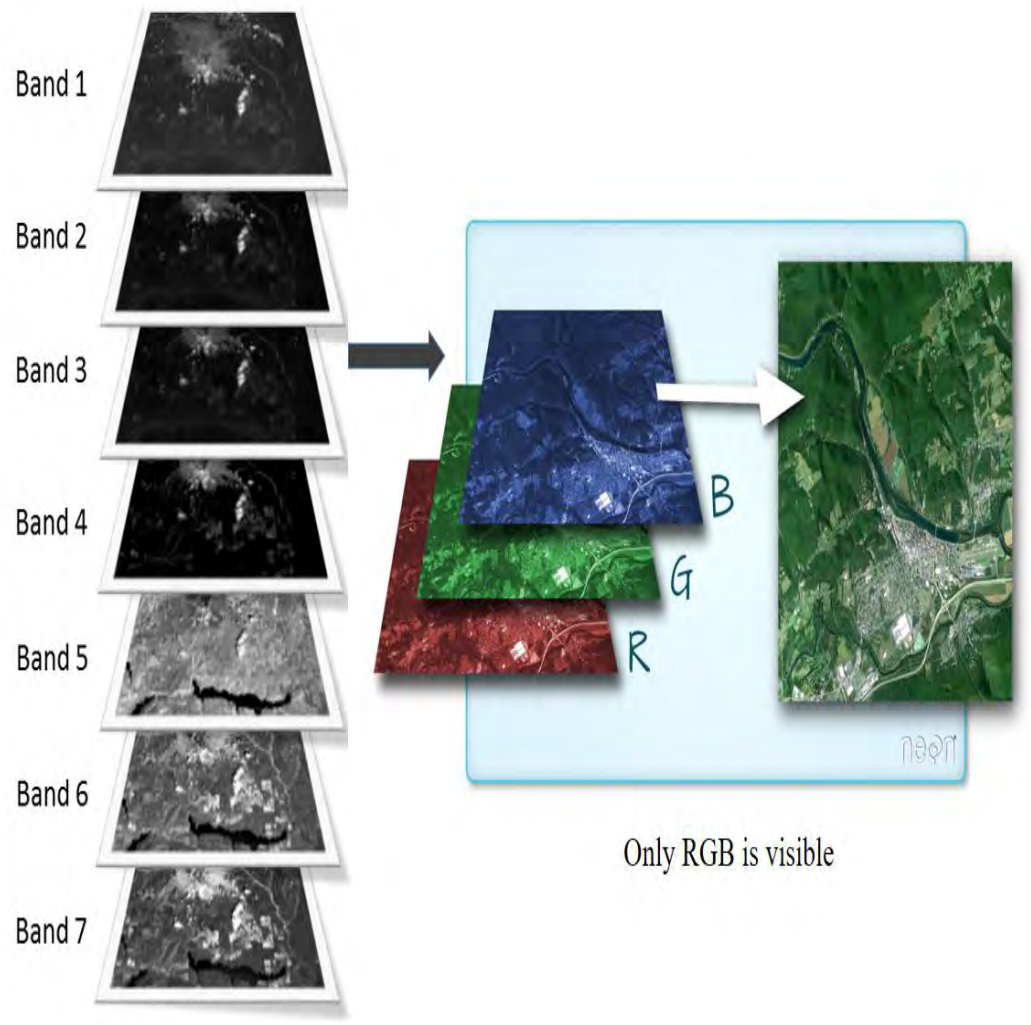


Figure 9. Stacking procedure

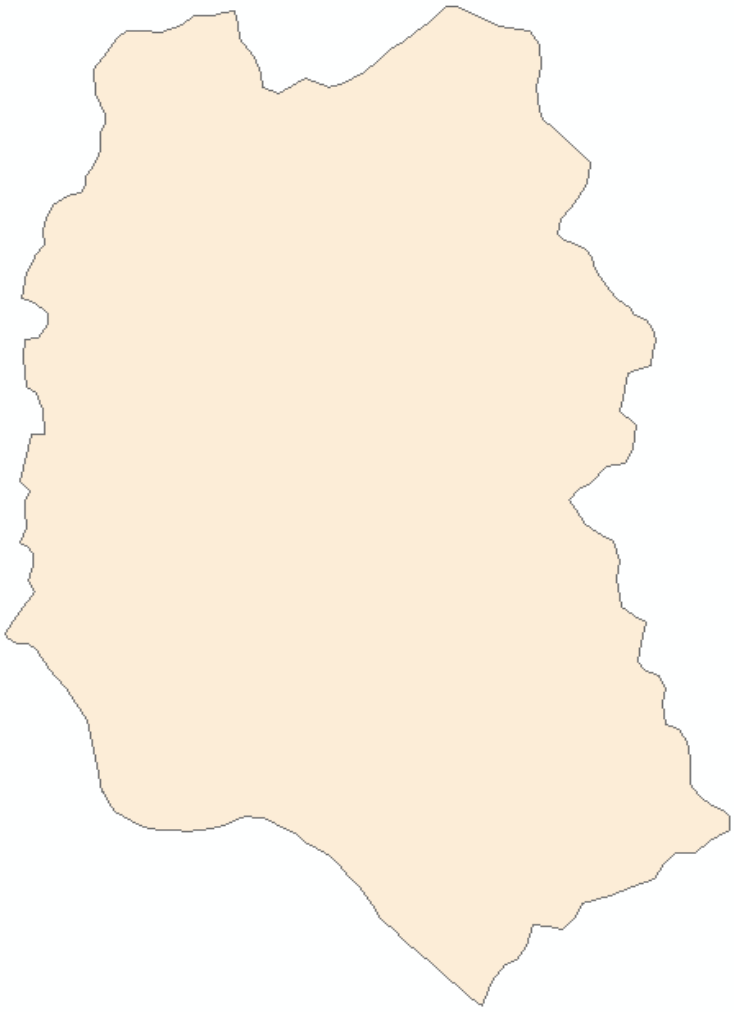


Figure 10. Image of the shape file of Dhaka city

We than manually inserted the projection system for the shape file which is WGS_1984_UTM_Zone_46N for Dhaka. The subset image is shown in Figure 10.

We have created this subset image by geotagging process using shape file of Dhaka city. It is a technique of adding geographical identification metadata to various media such as geotagged photograph or video. Geotagging is to find out the exact location on a map. We have used a geotagged digital image that include precise latitude and longitude coordinates to locate Dhaka city form Landsat 7 and Landsat 8 images. In Figure 11 and 12 below, our geotagging process is shown with subset image.

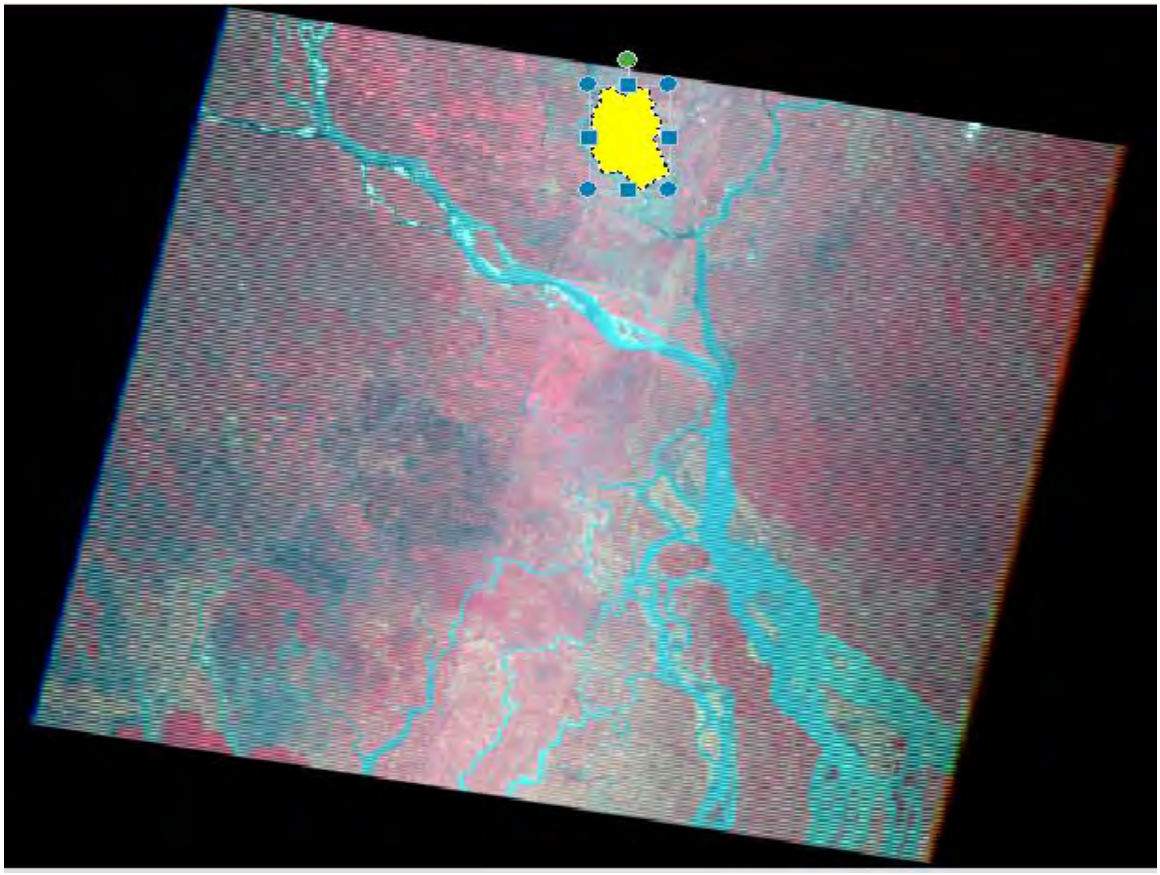


Figure 11. Study area geotagging

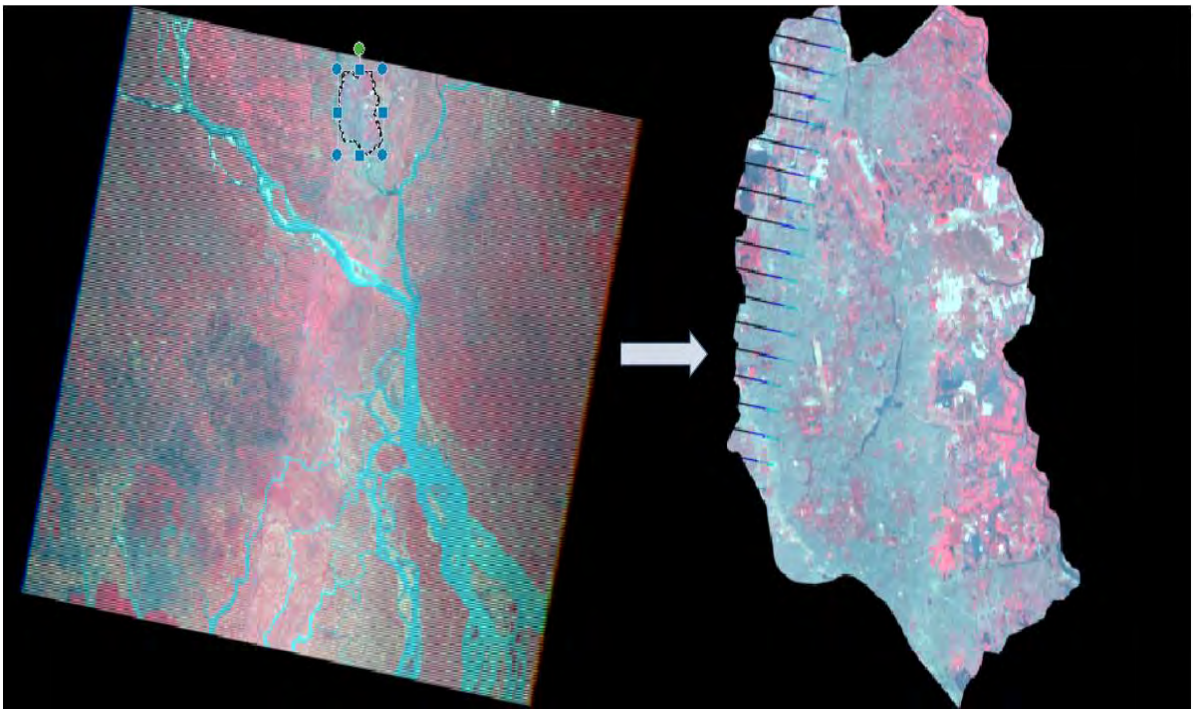


Figure 12. Study area geotagging with subset image

We have faced some problem during our research, SLC error is one of them. After performing stacking images of Landsat 7 we faced that some dead pixels were present in those images. The reason is for Scan Line Corrector (SLC) failure of Landsat 7 satellite. The Scan Line Corrector (SLC) in the Enhanced Thematic Mapper plus (ETM+) instrument failed on 31 May 2003. The SLC was designed with a pair of small mirror that rotate about an axis in tandem with the motion of the main ETM+ scan mirror. The purpose was to compensate for the forward motion of the spacecraft so that the output scans are parallel to each other, but it failed to

perform properly. Without the effect of the SLC the others instrument scans the earth in a “zig-zag” way because

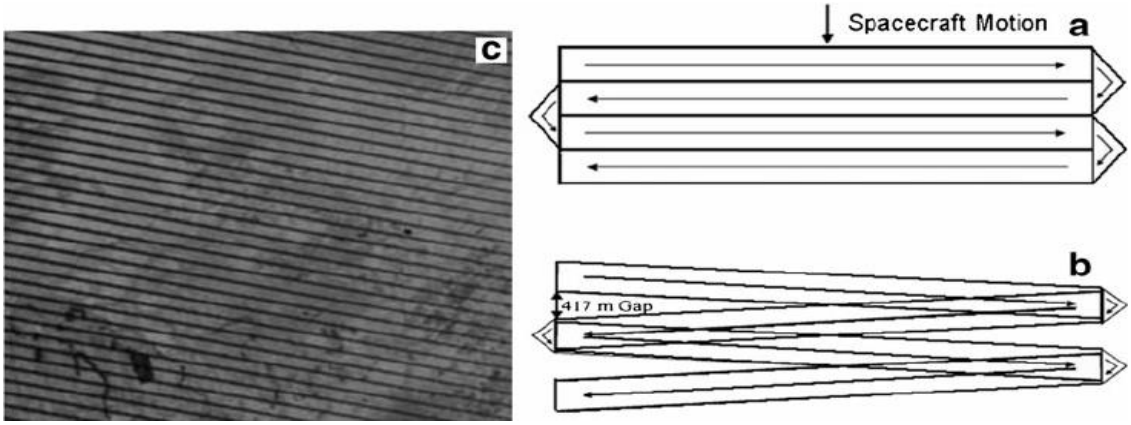


Figure 13. Landsat 7 SLC error

some areas were imaged twice while others are not imaged at all shows in Figure 13. That is how some pixels or areas of Landsat 7 images remains as dead pixels. Approximately 22% of the data is missing without a functional SLC in a Landsat 7 scene [10]. There are several processes to overcome this SLC error of them Neighborhood Similar Pixel Interpolator (NSPI) is most commonly and easy to use. [11] .Even though we have Landsat 8 recent images which are SLC error free, still Landsat 7 images is important for remote sensing analysis[12]. In Figure 14, our study area with SLC error is shown.

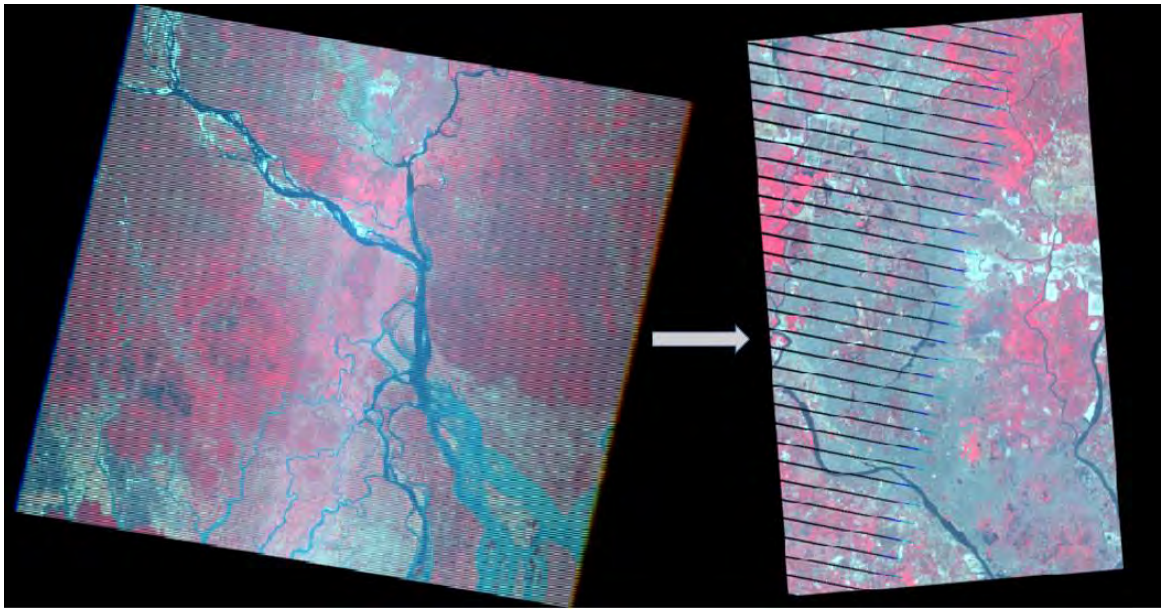


Figure 14. Our study area with SLC error

As most of the remote sensing software have a separate module for it. ERDAS Imagine 2015 is the software we have used. This process mainly modifies neighboring pixels in a single Landsat 7 SLC- off scene and it creates an aesthetic final image. We keep doing this process nearly 4-7 times until the image is dead pixel free. In Figure 15 focal analysis steps are shown.

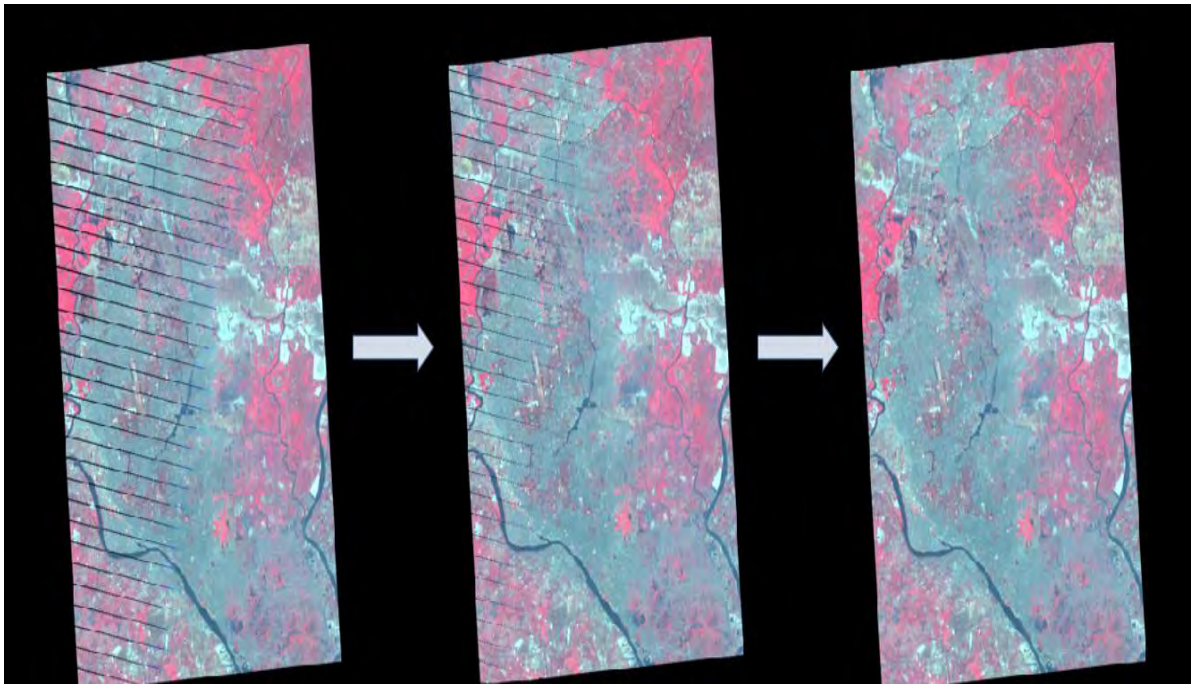


Figure 15. SLC error correction with focal analysis

This method was designed using ERDAS Imagine 2015, along with for final filled-image verification. The approximate time to complete a scene with all the steps is 20 minutes.

4.1.b Image Classification

One of the most important functions of remote sensing data is the production of LULC (Land Use and Land Cover Changes) and thus can be managed through a process called image classification. In other words, Image classification is the process of assigning land cover classes to pixels.

For example, in our study 4 different land cover classify image into Built-up area, Vegetation, Water index and Bare soil.

There are various classification approaches that have been developed and widely used to produce land cover maps[8].However, there are two broad types of classification procedure and each finds application in the processing of remote sensing images: one is referred to as supervised classification and the other one is unsupervised classification . For our approach, we followed supervised classification approaches.

4.1.c NDVI (Normalized Difference Vegetation Index)

Normalized Difference Vegetation Index (NDVI) quantifies vegetation by measuring the difference between near-infrared (which vegetation strongly reflects) and red light (which vegetation absorbs). The formula of generating NDVI maps is as follows-

-NDVI = (NIR - RED) / (NIR + RED)

4.1.d NDBI (Normalized Difference Built-up Index):

NDBI or Normalized Difference Built-up Index is based on the unique spectral response of built-up lands that have higher reflectance in MIR wavelength range than in NIR wavelength range. This index highlights urban areas where there is typically a higher reflectance in the shortwave-infrared(SWIR) region, compared to the near- infrared (NIR) region. The formula of generating NDBI maps is as follows-

NDBI = (MIR – NIR) / (MIR + NIR)

4.1.e NDWI (Normalized Difference WATER Index):

The Normalized Difference Water Index (NDWI) (Gao, 1996) is a satellite-derived index from the Near-Infrared (NIR) and Short-Wave Infrared (SWIR) channels. It is an index to extract water bodies from satellite imagery. The formula of generating NDWI maps is as follows-

NDWI = (GREEN - NIR) / (GREEN + NIR)

4.1.f Supervised Classification:

Upon generating each maps , supervised classification is than applied which follows “Maximum Likelihood Algorithm” to relate the study classification area pixels.

The algorithm used by the Maximum Likelihood Classification is based on two principles:

- The cells in each class sample in the multidimensional space are normally distributed.
- Bayes' theorem of decision making.
- The algorithm for Maximum likelihood classifier is as follows-

  t -1 Gi(X)= -1/2(X-Ui) Ci (X-Ui) - (d/2) log(2TT) - (1/2)log(|Ci|) + log(Pi)

The algorithm is explained in Table 4.

Part	**Description**
Gi(X)	Result for class *i* on pixel *X*
d	Number of channels in the classification
X=(x1,...,xd)	(*d* by 1) pixel vector of gray-levels
Ui	(*d* by 1) mean vector for class *i*
Ci	(*d* by *d*) covariance matrix for class *i*
TT	pi = 3.1415...

|Ci|	Determinant of the covariance matrix
Pi=Bi/SUM(Bi)	Priori probability for class *i*
Bi	BIAS for class *i*
SUM(Bi)	Sum of Biases for all classes used
t	As a superscript, denotes transpose
-1	As a superscript, denotes inverse
Ti	Threshold value THRS for class *I*
d, Ui, Bi, Ti, Ci and |Ci|	Obtained from the signature segment

Table 4. Maximum likelihood classifier

Below in Figure 16 is the figure after applying the above process-

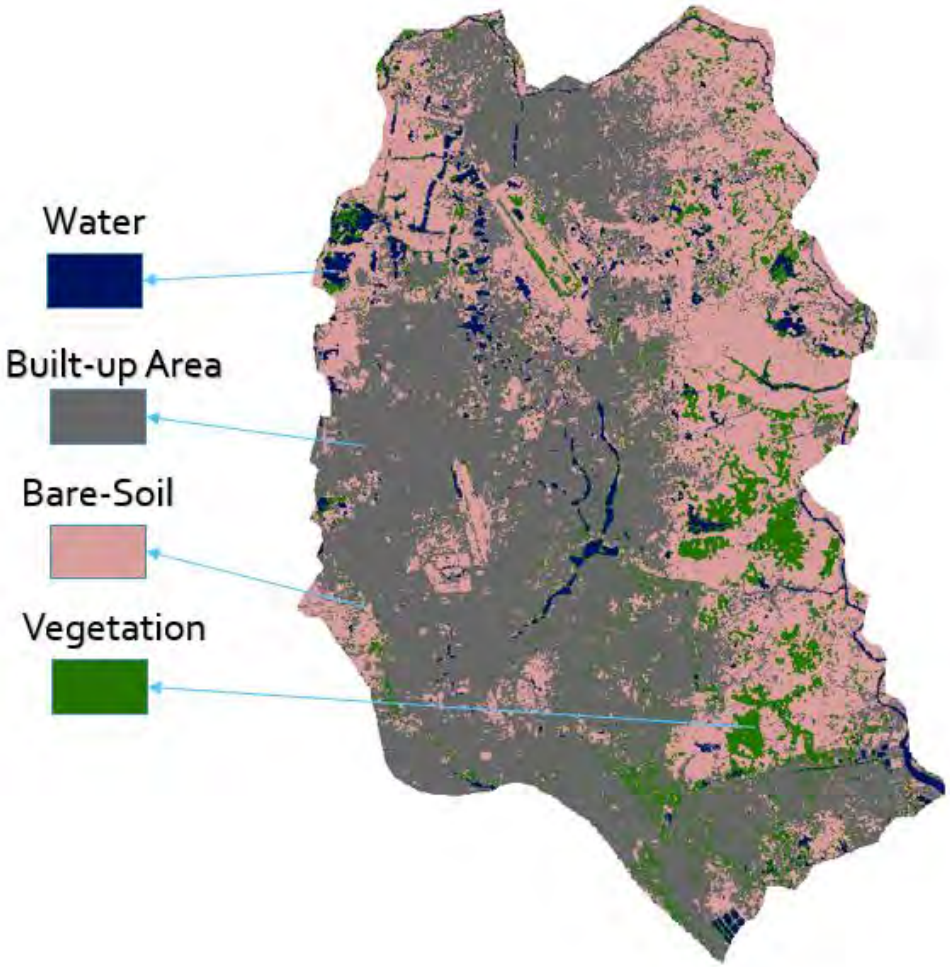


Figure 16. Classified Image

4.1.g Accuracy Assessment

A post classification method was used to improve the accuracy of the classification for its simplicity and effectiveness [13]. Accuracy assessment compares the "Ground truth" data with the reference of the classified map. It is done to assess how well the classification has worked. This study followed "simple random sampling". The accuracy assessment was done

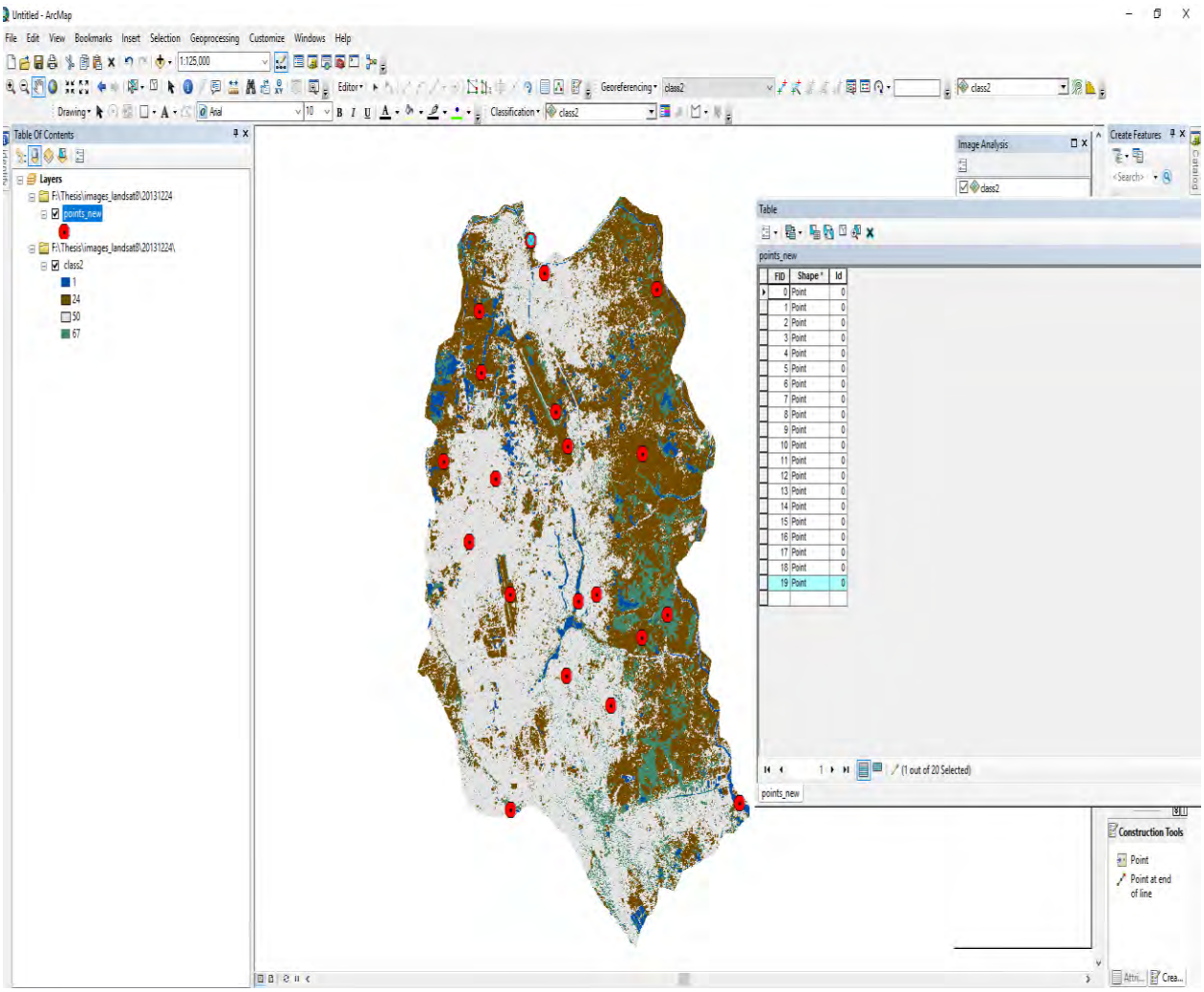


Figure 17. Random sampling points

with the reference of Google Earth data as "ground truth data" and

compares the classified image for assessment. Figure 17 is presenting some random points.

Comparing with Google Earth data (Figure 18).

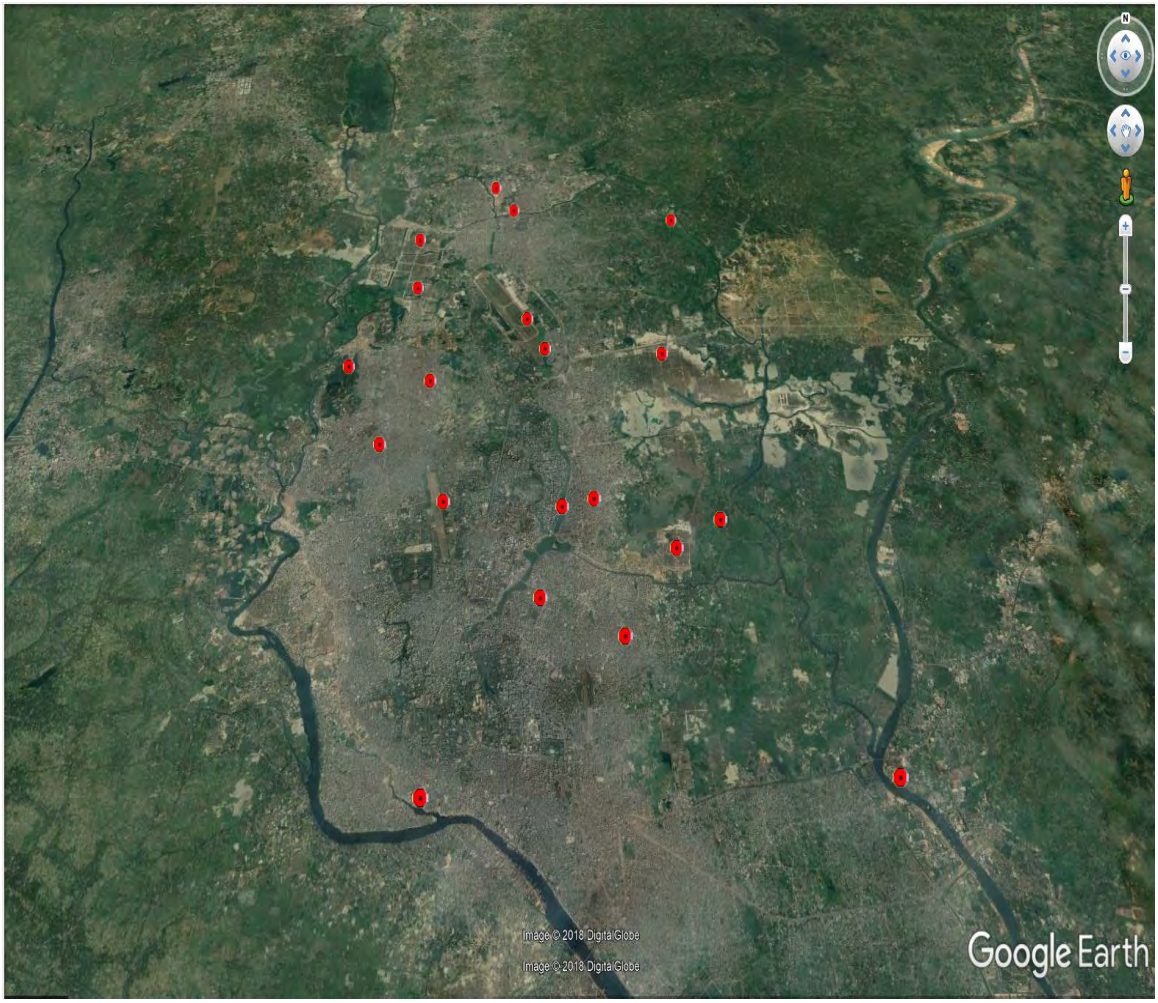


Figure 18. Random sampling points for Google Earth

Accuracy result ranges from 82.7-83.6% for the images ranging from the year 2012-2018 which is within the result recommended by Anderson [14].

4.1.h Area Calculations:

Based on classified maps the area has been calculated. Area calculation was done in python script and the aprtial script has been shown in figure 14 Exapmles of area calculation is shown in the table 5 below-

Features	Year 2013(Sq. km)	Year 2015(Sq. km)	Year 2017(Sq. km)
Water	20.18	19.09	14.31
Built-up	122.43	157.13	163.23
Bare soil	87.31	66.25	73.45
Vegetation	69.23	56.78	48.12
Total Area (Sq.km)	299.15	299.25	299.11

Table 5. Classification report for the year 2013, 2015, 2017

Area Calculation algorithm in python is as follows in Figure 19-

```
# Calculate AREA values

# Import system modules
import arcpy

# Local variables...
workspace = "C:/data"
input = "tracts.shp"
calculate_output = "tracts_with_area_field.shp"

try:
    # Set the current workspace (to avoid having to specify the full path to the feature classes each time)
    arcpy.env.workspace = workspace

    # Process: Calculate Areas...
    arcpy.CalculateAreas_stats(input, calculate_output)

except:
    # If an error occurred when running the tool, print out the error message.
    print arcpy.GetMessages()
```

Figure 19. Study Area after area calculations

Example image after area calculation in Figure 20-

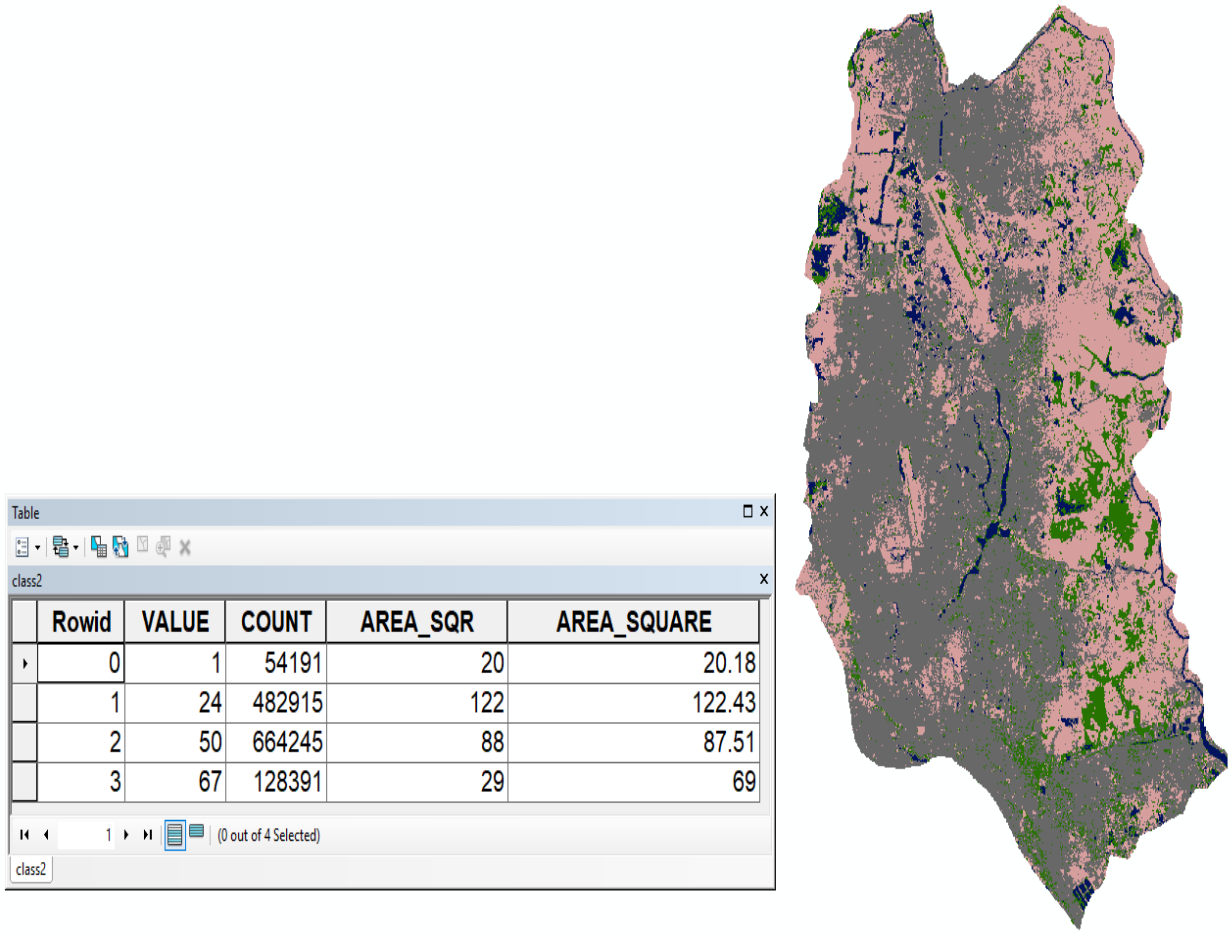


Figure 20. Study Area after area calculations

Chapter 5

Machine Learning

5.1 What is Machine Learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

5.2 Reasons for Using Machine Learning in GIS

Urban growth is one of the most important topics in urban studies. A city is considered as a complex system. It consists of numerous interactive sub-systems and is affected by diverse factors including governmental land

policies, population growth, transportation infrastructure, and market behavior. To understand the driving forces of the urban form and structure change, the satellite-based estimates are considered as the appropriate methods to monitor these dynamically change in a long term. Furthermore, modeling and simulation are believed to be powerful tools to explore the mechanisms of urban evolution and provide planning support in growth management.

5.3 Machine learning to analyze data

A human being has the ability to make accurate generalizations from a few scattered facts provided by a teacher or the environment using inductive inferences. This approach is time consuming and sometimes for a big dataset it is almost humanly impossible to generate a result. Our approach to utilize machine learning to simulate and predict the mechanisms of urban expanding and evolution based on the dataset we made from the above satellite images that was mentioned before in this study.

5.4 Approach for machine learning

Our approach for machine learning can be described by the folowing figure -

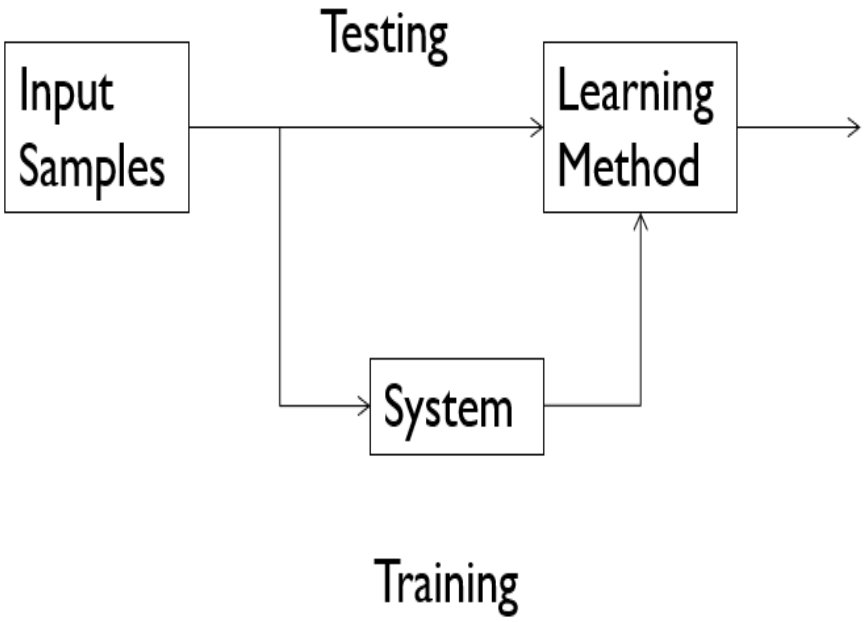


Figure 21. Machine learning workflow

5.5 Libraries used

Numpy -NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to integrate with a wide variety of databases seamlessly and speedily.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Pandas -pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Matplotlib-Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts,

the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

5.5.a Training & Testing Datasets

Among the different types of ML tasks, a crucial distinction is drawn between supervised and unsupervised learning:

- **Supervised machine learning:** The program is "trained" on a pre-defined set of "training examples", which then facilitate its ability to reach an accurate conclusion when given new data.
- **Unsupervised machine learning:** The program is given a bunch of data and must find patterns and relationships therein.

As the dataset we are providing are statistical data. We compared different Supervised learning algorithms analyze the success rate of those algorithms based on our data. Supervised machine learning solves this problem by getting the computer to do the work for us. By identifying patterns in the data, the machine can form heuristics. The primary difference between this and human learning is that machine learning runs on computer hardware and is best understood through the lens of computer science and statistics, whereas human pattern-matching happens in a

biological brain (while accomplishing the same goals). This study analyzed different supervised machine learning approaches namely-

1.Linear Regression

2.Decision Tress Regressor

3.Random Forest Regressor

5.5.b Evaluating the learned model

We have divided the data we have in two sets,

-Training

-Test

We learn on our training data and after evaluating the performance on test data, the data we have is known as validation data. Here, the test data is unseen for the model, as it has not seen this during training phase. The performance of a model is depending on how as model has done generalized well, comparing the other machine learning approaches.

Below is the python code for training, testing, and evaluating the datasets. The comparison result is given after the code-

In [61]:

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
```

In [44]:

```
#read csv from source
df= pd.read_csv("habijabi.csv")
```

In [45]:

```
#read csv
df.head()
```

Out[45]:

	Year	Water Body	BuildUp	Vegetation
0	2012	19.00	59.63	21.37
1	2013	17.50	63.50	19.00
2	2014	16.00	69.00	15.00
3	2015	14.97	71.03	14.00
4	2016	13.11	74.69	12.20

In [52]:

```
#datafrmae - x & y split

x= df[["Year"]]
y= df[["Water Body","BuildUp","Vegetation"]]
```

In [53]:

```
x_test=np.array([[2003]])
clf=DecisionTreeRegressor()
clf.fit(x,y)
y_predic=np.array([14,56.7,29.3])

#y_pred=clf.score(x_test,y_predic)

y_pred=clf.predict(x_test)
```

```
print(y_pred)
```

```
[[19.   59.63 21.37]]
```

In [54]:

```
x_test=np.array([[2017],[2031],[2005]])
clf=DecisionTreeRegressor()
clf.fit(x,y)
y_predic=np.array([[14,56.7,29.3]])
#y_pred=clf.score(x_test,y_predic)

y_pred=clf.predict(x_test)


print(y_pred)
```

```
[[12.91 77.09 10.  ]
 [12.37 77.88  9.75]
 [19.   59.63 21.37]]
```

In [55]:

```
total = 0

for i in range (100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.3)
    clf=DecisionTreeRegressor()
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    total=total+clf.score(x_test,y_test)
    #r2score rsquareError/accuracy
    print (clf.score(x_test,y_test))
avg =total/100
print("average " ,avg)
```

```
-0.31102029843396173
0.7515988668831801
0.5081351136263144
0.5081351136263144
-2.788519129647941
0.7178637403814819
-14.749925911291117
0.2449681525725761
0.7244329418477663
0.8367002350482009
0.7244329418477662
0.20273697414563308
0.48796301830043765
0.8150796207393576
```

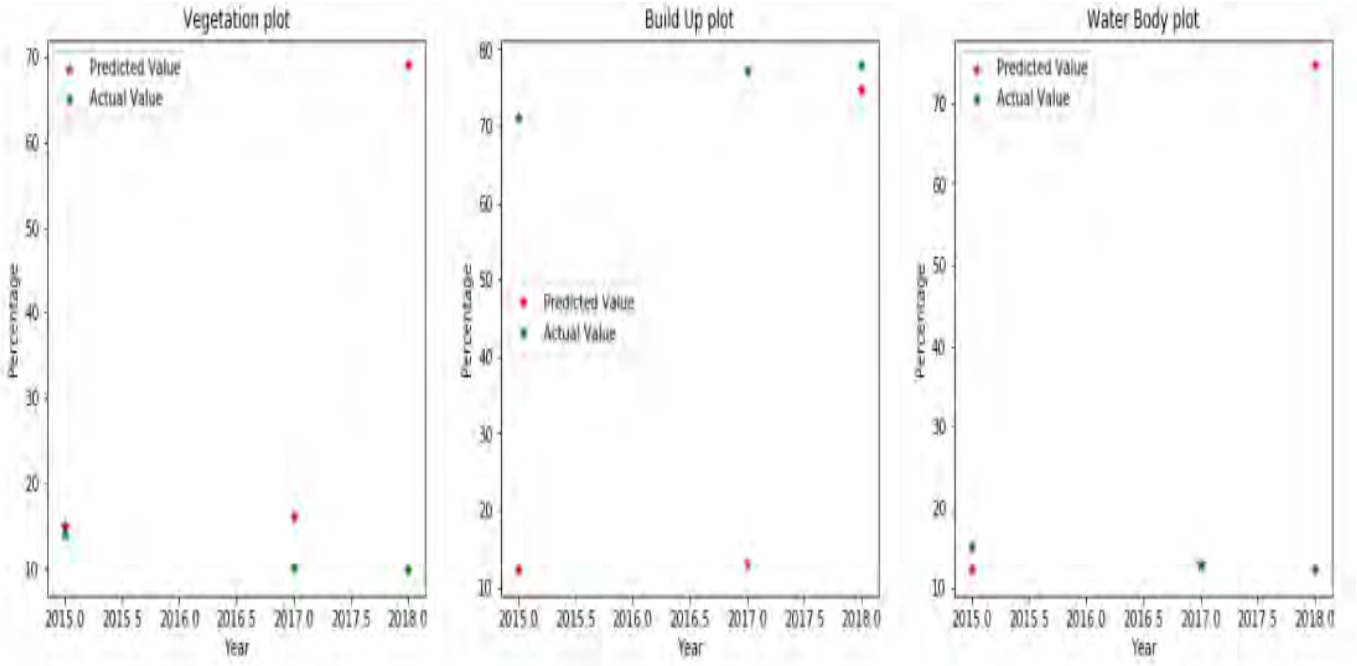
-14.749925911291117
0.7515988668831801
0.7300860190557984
0.22396723273272304
0.5081351136263144
-0.031874041679625736
0.7898316771388645
0.34326626375400465
-0.0567009213216244
-0.9062716641733845
0.7300860190557984
-4.901853425470174
0.8367002350482008
-0.0567009213216244
-2.788519129647941
0.8367002350482008
0.34326626375400465
0.20273697414563302
0.48390729645515385
0.8367002350482008
0.749577946552695
0.8150796207393576
0.6019178041864652
0.7515988668831801
-0.12889951873618763
-0.031874041679625736
-2.788519129647941
0.7178637403814819
-0.1288995187361878
-4.901853425470174
-0.056700921321624395
0.6528986012436588
-0.31102029843396195
0.8469578413201838
0.8118925683151011
0.6528986012436588
-3.2373244997067823
-4.901853425470174
-0.10506967100093254
0.8407722506445173
0.34326626375400465
0.34326626375400465
0.7515988668831801
0.34326626375400465
0.48390729645515385
0.34326626375400465
0.22396723273272304
0.7266027711736913
0.7515988668831801
-0.031874041679625736

```
0.7515988668831801
0.20273697414563302
-0.31102029843396173
-0.031874041679625736
-0.4570186557741195
0.34326626375400465
-0.031874041679625736
-0.12889951873618763
0.2449681525725761
0.48390729645515385
0.2449681525725761
0.48796301830043765
-0.031874041679625736
0.8367002350482008
0.7515988668831801
0.5081351136263144
-4.901853425470174
-0.9062716641733845
-0.10506967100093254
-0.031874041679625736
-0.031874041679625736
0.8852344223547272
0.7898316771388645
0.48796301830043765
0.8852344223547273
0.8367002350482009
0.2449681525725761
0.8367002350482008
-0.8498700007769774
0.8407722506445173
0.7898316771388645
-0.05670092132162455
-14.749925911291117
0.6528986012436587
0.8407722506445173
0.48796301830043765
average  -0.41265260139088
```

In [14]:

```
#Visualization with pyplot for DecisionTreeRegressor

veg_pred=y_pred[2]
bu_pred=y_pred[1]
wb_pred=y_pred[0]

veg_test=y_test['Vegetation']
bu_test=y_test['BuildUp']
```

```
wb_test=y_test['Water Body']

plt.figure(figsize = (20,5))
plt.subplot(131)

plt.plot(x_test,veg_pred,'r*',label="Predicted Value")
plt.plot(x_test,veg_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Vegetation plot")
plt.legend()

plt.subplot(132)
plt.plot(x_test,bu_pred,'r*',label="Predicted Value")
plt.plot(x_test,bu_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Build Up plot")
plt.legend()

plt.subplot(133)
plt.plot(x_test,wb_pred,'r*',label="Predicted Value")
plt.plot(x_test,wb_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Water Body plot")
plt.legend()
plt.show()
```



In [34]:

```
total = 0

for i in range (100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.3)
    clf=linear_model.LinearRegression()
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    total=total+clf.score(x_test,y_test)
    #r2score rsquareError/accuracy
    print (clf.score(x_test,y_test))
avg =total/100
print("average " ,avg)
```

```
0.8408105639503667
0.7949994472009472
0.9736125226097867
0.6037251820363785
0.8409592439402287
0.7579989711240961
0.7715118739571096
0.9736125226097867
0.6452548367333754
0.6796432265518542
-0.12294189695677793
0.5593421538061256
0.8369863716111445
0.9513314761267522
```

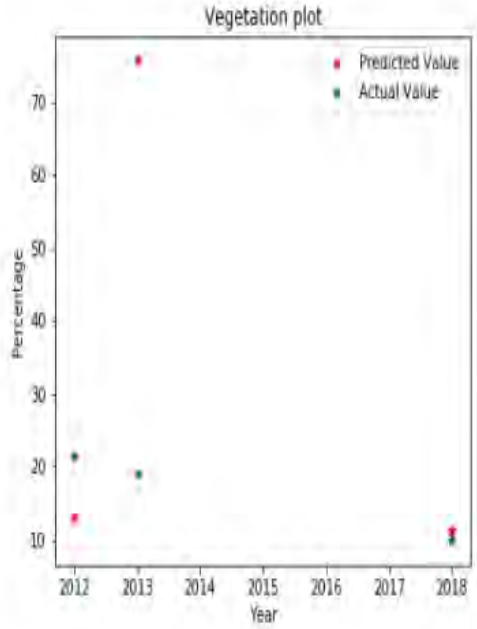
0.7579989711240961
0.9695724521387723
0.968134524662281
0.9515681276896513
0.8964719811404714
0.840810563950352
0.9720512747599543
0.8025538609551408
-0.1229418969563878
0.8964719811404714
0.8964719811404714
-0.27491226122252055
0.9736125226097867
-0.12294189695677793
0.7680638202720937
0.9736125226097867
0.9513314761267522
0.6796432265519341
-0.6259409866057273
0.8369863716111421
-3.4432197123079398
-497.3090233202741
0.6037251820363785
0.9695724521387794
0.9093613918305246
0.6796432265518542
0.7075092353696507
0.7824656266018407
0.7824656266018407
0.6452548367334031
-3.4432197123079398
0.8328075360333788
0.7715118739571096
0.8369863716111445
0.008705456656861954
-3.44321971230794
0.6534007401481976
0.00870545656861954
-0.6259409866057273
0.7680638202720937
0.9515681276896513
0.951331476126761
0.8369863716111445
0.7824656266018407
0.7824656266018407
0.951331476126761
0.840810563950352
0.9240145159369791
-0.6259409866057273
0.707509235369668

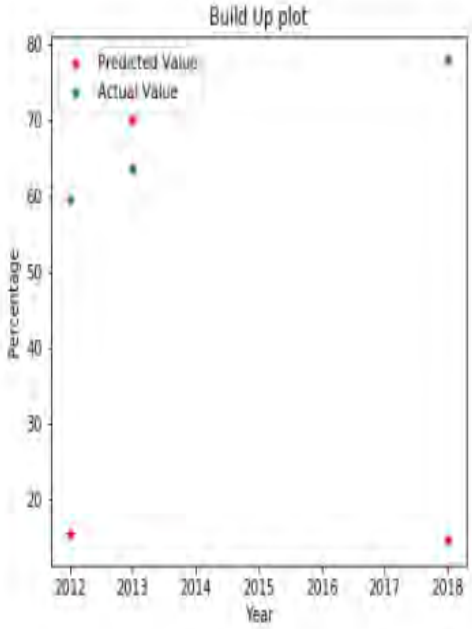
```
-0.6259409866052548
0.5593421538062944
0.008705456656861954
0.7075092353696507
0.8409592439401664
0.008705456656861954
0.8409592439402814
0.6037251820363785
0.951331476126761
0.945887081556817
0.7579989711240961
0.679643226551894
-497.3090233202741
0.5593421538061256
-3.4432197123084585
0.6037251820363785
0.679643226551894
-0.1229418969563878
0.9093613918305246
-497.3090233199164
0.8964719811404714
0.3472432297705354
-3.443219712309884
0.008705456656861954
-0.12294189695638913
0.7715118739570671
0.8964719811404714
-0.27491226122252055
0.9695724521387723
0.8272191075227906
0.9515681276896513
-0.27491226122268486
0.7075092353696413
0.9240145159370042
0.8964719811404714
0.794999447200877
average  -14.523839747612687
```

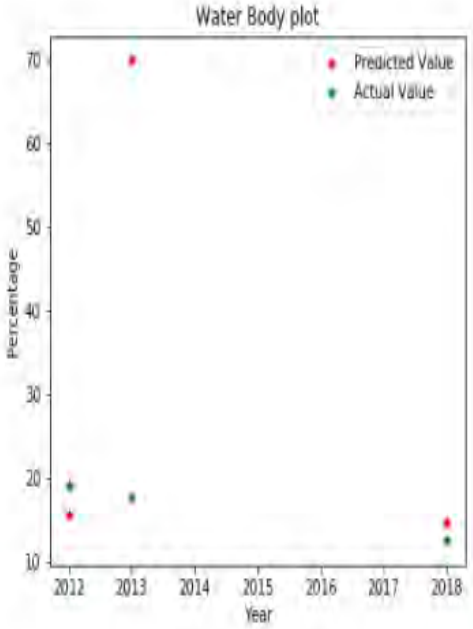
In [22]:

```
#Visualization with pyplot for LinearRegressor

veg_pred=y_pred[2]
bu_pred=y_pred[1]
wb_pred=y_pred[0]

veg_test=y_test['Vegetation']
bu_test=y_test['BuildUp']
```

```
wb_test=y_test['Water Body']

plt.figure(figsize = (20,5))
plt.subplot(131)

plt.plot(x_test,veg_pred,'r*',label="Predicted Value")
plt.plot(x_test,veg_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Vegetation plot")
plt.legend()

plt.subplot(132)
plt.plot(x_test,bu_pred,'r*',label="Predicted Value")
plt.plot(x_test,bu_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Build Up plot")
plt.legend()

plt.subplot(133)
plt.plot(x_test,wb_pred,'r*',label="Predicted Value")
plt.plot(x_test,wb_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Water Body plot")
plt.legend()
plt.show()
```







In [57]:

```
total = 0

for i in range (100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.3)
    clf= RandomForestRegressor()
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    total=total+clf.score(x_test,y_test)
    #r2score rsquareError/accuracy
    print (clf.score(x_test,y_test))
avg =total/100
print("average " ,avg)
```

```
-0.07122903637038404
0.09186782914195336
-2.619660819932337
0.6224279090768016
0.648609294705594
0.8220143667006191
-2.8819358873131753
0.6496523163153243
-0.15330287231866813
-1.074504670532718
0.3825442827024701
0.9631366995334198
0.6769041346965621
0.723475889530403
0.8992447818760121
0.6415271763201431
```

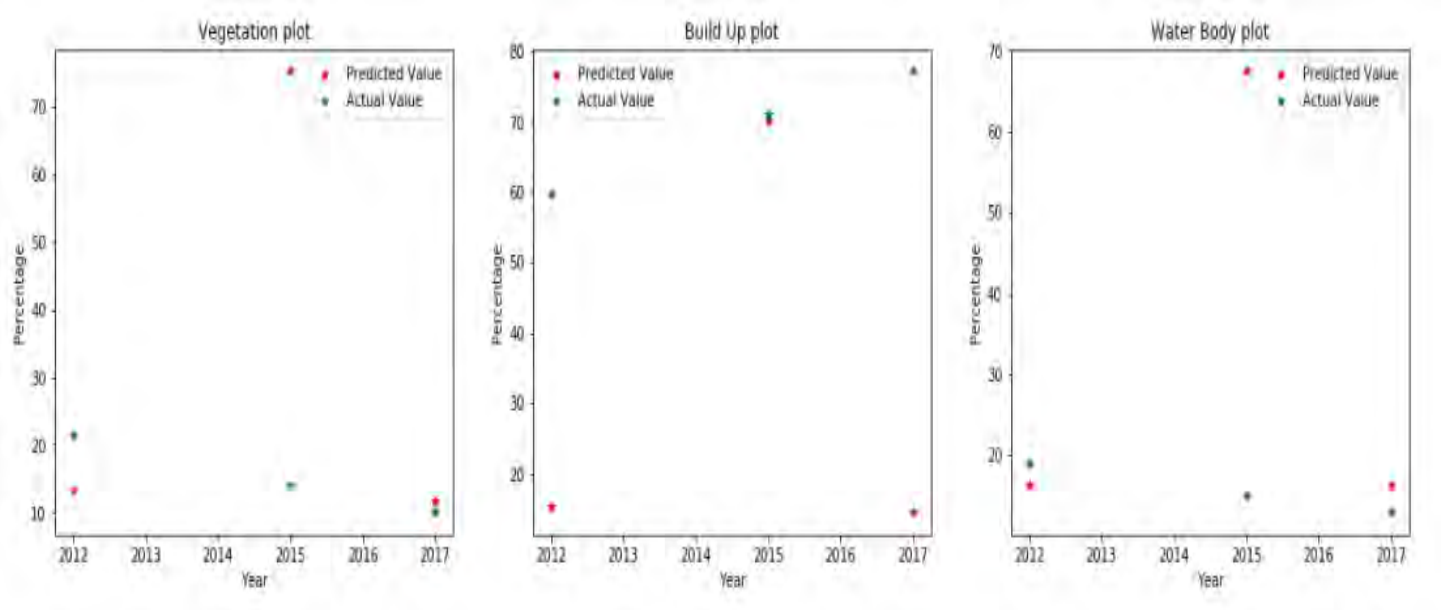
0.5005360935840275
0.8262380941563598
0.7761577739740831
0.5783419180865481
0.022610826643385182
-16.96230959201819
0.5770871260253015
0.7090902319523485
0.9045592370161681
0.8691728874996347
0.12544873405611415
-0.21376023379616815
0.25607258816838796
0.8749304205784937
0.4968861878220202
0.34330726525212363
-0.28773089248105194
0.6043315655170564
0.9514915197280901
-3.2373244997067734
0.21930661822507388
-22.897336757878136
0.8880495655956773
0.9499326159026622
0.7998869115725916
-1.0528433106375403
0.9654963291790414
0.09737418236922611
0.8268648565816413
0.8707819095334564
0.8415743977339532
-0.13703308305689194
-0.28270178979252614
0.768684470598068
0.5778635226819612
0.8844883402269196
0.9645503777763756
-5.837019628435728
0.9009704435961591
-0.9464428978551936
0.6033143744984234
-5.361853921279277
0.272907815510584
0.9193797922135896
0.8595671110219958
0.1574607645186928
-3.357849112453026
-0.5437094326538763
-0.35175493487065296
-16.96230959201819

-0.48527207921333876
-0.9551793502063423
0.4712808891832037
0.9601643307379196
0.7592918429091952
0.7105050679129828
0.6495040577961821
0.7692871694182866
0.7678285885413245
0.3940781185602966
0.9342866845582942
-0.023766182069760614
0.6294279276351575
-1.4542654835206965
0.7475808239536341
0.11390985425025853
0.0672227844541225
0.6919788996363723
0.10795959768273614
-0.17448829134783767
0.7586111660323022
0.9600292712430172
0.5265073481958598
0.6298566159732635
0.22747376016339832
0.7285045071947147
-26.770948476595112
0.4511535979572803
0.9670748861153761
-0.356519404189277
0.2919841173424045
0.6445701058109133
0.7343513365818929
0.6415659988891051
average -0.6921094133381783

In [11]:

```
#Visualization with pyplot for LinearRegressor

veg_pred=y_pred[2]
bu_pred=y_pred[1]
wb_pred=y_pred[0]

veg_test=y_test['Vegetation']
bu_test=y_test['BuildUp']
wb_test=y_test['Water Body']
```

```
plt.figure(figsize = (20,5))
plt.subplot(131)

plt.plot(x_test,veg_pred,'r*',label="Predicted Value")
plt.plot(x_test,veg_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Vegetation plot")
plt.legend()

plt.subplot(132)
plt.plot(x_test,bu_pred,'r*',label="Predicted Value")
plt.plot(x_test,bu_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Build Up plot")
plt.legend()

plt.subplot(133)
plt.plot(x_test,wb_pred,'r*',label="Predicted Value")
plt.plot(x_test,wb_test,'g*',label="Actual Value")
plt.ylabel("Percentage")
plt.xlabel("Year")
plt.title("Water Body plot")
plt.legend()
plt.show()
```



In [58]:

```
print ("xtest \n" , x_test)
print ("y_test \n", y_test)
print ("prediction \n" , y_pred)
print("average " ,avg)
```

```
xtest
    Year
4  2016
5  2017
0  2012
y_test
    Water Body  BuildUp  Vegetation
4       13.11    74.69       12.20
5       12.91    77.09       10.00
0       19.00    59.63       21.37
prediction
 [[15.279 70.421 14.3  ]
 [13.253 75.622 11.125]
 [16.75  66.25  17.   ]]
average  -0.6921094133381783
```

In [59]:

```
print (x_test)
print (y_test)
print (y_pred)
```

```
   Year
4  2016
5  2017
0  2012
   Water Body  BuildUp  Vegetation
4       13.11    74.69       12.20
5       12.91    77.09       10.00
0       19.00    59.63       21.37
[[15.279 70.421 14.3  ]
 [13.253 75.622 11.125]
 [16.75  66.25  17.   ]]
```

In [60]:

```
print (x_test)
print (y_test)
print (y_pred)
```

```
   Year
4  2016
```

```
5  2017
0  2012
   Water Body  BuildUp  Vegetation
4       13.11    74.69       12.20
5       12.91    77.09       10.00
0       19.00    59.63       21.37
[[15.279 70.421 14.3  ]
 [13.253 75.622 11.125]
 [16.75  66.25  17.   ]]
```

In [11]:

```
x_train
```

Out[11]:

	Year
10	2013
13	2016
1	2004
12	2015
11	2014
8	2011
0	2003
5	2008
4	2007

In [12]:

```
y_train
```

Out[12]:

	Water Body	BuildUp	Vegetation
10	4	86.00	10.00
13	1	94.60	4.40
1	19	52.50	28.50
12	2	90.60	7.40

11	***3***	***87.25***	***9.75***
8	***9***	***77.00***	***14.00***
0	***23***	***54.00***	***23.00***
5	***18***	***59.90***	***22.10***
4	***11***	***69.00***	***20.00***

5.5.c Evaluating the learned model and their comparisons

Machine learning techniques	Accuracy result
Decision Tree Regressor	-0.15
Linear Regressor	0.73
Random Forest Regressor	-0.22

Table 6. Model comparison

The above table shows the Linear Regressor does well than the other algorithms for our proposed model.

5.5.d Artificial Neural Network

Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behavior of biological systems composed of "neurons". ANN's are computational models inspired by an animal's central nervous systems. It is capable of machine learning as well

as pattern recognition. These presented as systems of interconnected "neurons" which can compute values from inputs.

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

A neural network may contain the following 3 layers:

- **Input layer –** The activity of the input units represents the raw information that can feed into the network.
- **Hidden layer –** To determine the activity of each hidden unit. The activities of the input units and the weights on the connections between the input and the hidden units. There may be one or more hidden layers.
- **Output layer –** The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

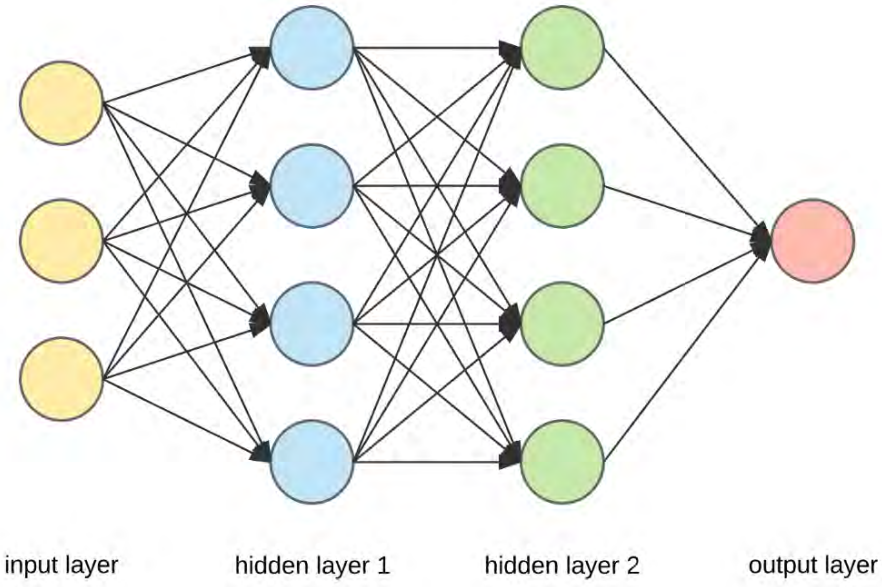


Figure 22. Artificial Neural Network Model

5.5.e Recurrent Neural Network

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking

back only a few steps (more on this later). Here is what a typical RNN looks like:

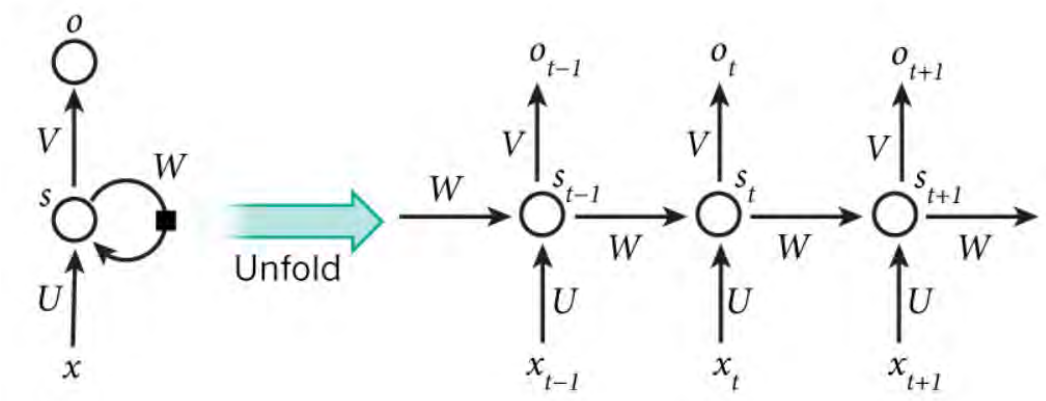
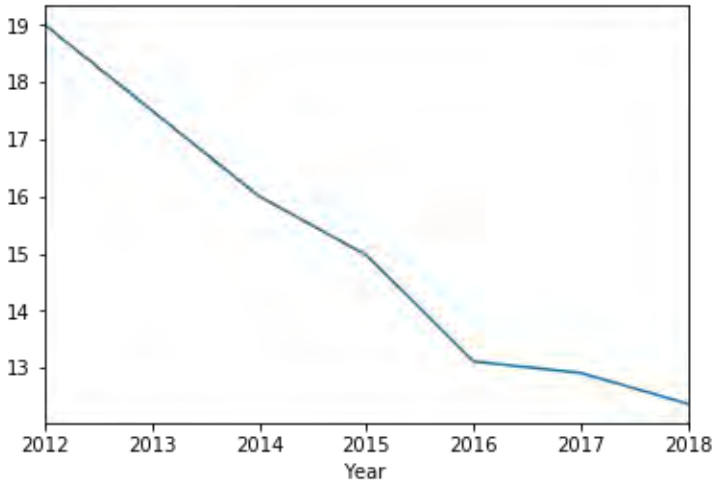


Figure 23. A RNN and the unfolding in time of the computation involved in (t+1)

5.5.f Reasons for using Recurrent Neural Network

As our model is a Univariate time series and we are going to predict one step ahead we have generated the following code for our train test and predicting data-

```
from pandas import read_csv
from pandas import datetime
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
def parser(x):
return datetime.strptime(x, '%Y')
series = read_csv('Data/water.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)
 series.plot()
 pyplot.show()
```



In [64]:

```
values = DataFrame(series.values) dataframe = concat([values.shift(1), values],
axis=1) dataframe.columns = ['t-1', 't+1']
print(dataframe.head(5))
```

```
      t-1    t+1
0     NaN  19.00
1   19.00  17.50
2   17.50  16.00
3   16.00  14.97
4   14.97  13.11
```

In [65]:

```
# split into train and test sets
X = dataframe.values
train_size = int(len(X) * 0.66)
train, test = X[1:train_size], X[train_size:]
train_X, train_y = train[:,0], train[:,1]
test_X, test_y = test[:,0], test[:,1]
```

In [66]:

```
# persistence model
 def model_persistence(x):
 return x
```

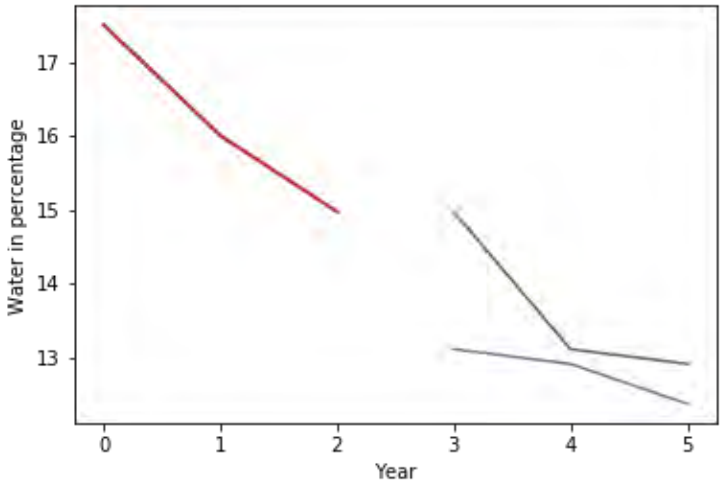
In [67]:

```
# walk-forward validation
predictions = list()
for x in test_X:
    yhat = model_persistence(x)
    predictions.append(yhat)
test_score = mean_squared_error(test_y, predictions)
```

```
print('Test MSE: %.3f' % test_score)
```

Test MSE: 1.264
In [72]:

```
# plot predictions and expected results
pyplot.plot(train_y)
 pyplot.xlabel("Year")
pyplot.ylabel("Water in percentage")
pyplot.plot([None for i in train_y] + [x for x in test_y])
pyplot.plot([None for i in train_y] + [x for x in predictions])
pyplot.show()
```




```
#creating TensorFlow graph that will do the computation

tf.reset_default_graph() #if theres any prev graph obj running , this will reset the graphs


num_periods=20   #no of periods per vector we are using to predict one period ahead

inputs = 1       #no of vectors submitted

hidden = 100     #no of neurons we will recursively work through , can be changed to improve accuracy

output = 1       #no of output vectors


X=tf.placeholder(tf.float32, [None, num_periods, inputs])  #create variable objects

y=tf.placeholder(tf.float32, [None, num_periods, inputs])


basic_cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden, activation = tf.nn.relu) #creating our RNN obj
```

```
rnn_output, states = tf.nn.dynamic_rnn(basic_cell, X , dtype=tf.float32) #chosing dynamic over static


learning_rate = 0.001 #creating small learning curve so we dont pass the minimum


stacked_rnn_output = tf.reshape (rnn_output, [-1, hidden])            #changing the form into a tensor

stacked_outputs = tf.layers.dense(stacked_rnn_output, output)          #specify the type of layer

outputs = tf.reshape(stacked_outputs, [-1, num_periods, output])       #shaping the results


loss= tf.reduce_sum(tf.square(outputs - y))    #defing the cost functions which evalutes the quality of our model

optimizer =tf.train.AdamOptimizer(learning_rate=learning_rate)         #gradient decent method

training_op = optimizer.minimize(loss)         #train the result of the application of the cost_function

init = tf.global_variables_initializer()     #initialization of all vatiables


#implementing the above model into our training data

epochs= 1000 #no. of iterations or training cycles, incluides both the FeedForward and Backpropagation

with tf.Session() as sess:
   init.run()
   for ep in range (epochs):
      sess.run(training_op, feed_dict={X: x_batches , y: y_batches})
      if ep % 100 ==0:
         mse = loss.eval(feed_dict={X: x_batches , y: y_batches})
         print (ep, "\tMSE" , mse)


   y_pred = sess.run(outputs, feed_dict={X: X_test})
```

This is a very useful result as it suggests the result reported above was probably a statistical fluke. The experiment suggests that the model is probably about as good as the persistence model on average (1.26), if not slightly worse.

Chapter 6

Previous work Vs our work

There have been some work previous work using remote sensing and GIS to detect and monitor LULC change in Dhaka city of Bangladesh but none of them used machine learning techniques to forecast results. Also previous works were not based on recent data. We have used the most recent spatial image dataset from 2012 to 2018 of Dhaka city for predicting future change.

Chapter 7

Future Plan

One of our paper titling “Determining land use and land cover changes and predicting the growth of Dhaka, Bangladesh using remote sensing and GIS

techniques" is under review in ICESSAT2018. We are planning to improve our dataset to generate results that are more accurate. Results can be more precised. We also are planning to publish a journal after refining our work.

Chapter 8

Conclusion

Satellite remote sensing and GIS technology has proven its usability in determining LULC analysis. That is why this kind of study is a good way organize as well as cost and time effective for urban planners and decision makers. This study shows the changes with machine learning techniques in terms of built-up, water, bare soil and vegetation indexes of Dhaka city has experienced a massive change risen high as it was from 2012 and predicted its growth much higher than the year 2018. Furthermore, this study analysis the given data and applies different machine learning techniques and forecast the future growth patterns of Dhaka city using neural network techniques. As frequent remote sensing data's and their analysis are not quite available for Bangladesh, the result of this study will contribute for better planning in urban expansion, also predicting the possible changes of Dhaka city.

References

[1] The World Bank. (2007). Dhaka: Improving living conditions for the urban poor. Sustainable Development Unit, South Asia Region, Report No. 35824-BD.

[2] Islam, N. (2005). Dhaka now: Contemporary development. Dhaka: The Bangladesh Geographical Society.

[3] Hathout, S. 2002 The use of GIS for monitoring and predicting urban growth in East and West St Paul, Winnipeg, Manitoba, Canada. Journal of Environmental Management, 66, 229–238.

[4] Zhang, B. P., Yao, Y. H., Cheng, W. M., Zhou, C. H., Lu, Z., & Chen, X. D. (2002). Human -induced changes to biodiversity and alpine pastureland in the Bayanbulak Region of the East Tianshan Mountains. Mountain Research and Development, 22, 1–7.

[5] The World Bank. (2007). Dhaka: Improving living conditions for the urban poor. Sustainable Development Unit, South Asia Region, Report No. 35824-BD.

[6] Siddiqui, K., Ahmed, J., Awal, A., & Ahmed, M. (2000). Overcoming the governance crisis in Dhaka City. Dhaka: University Press Limited.

[7] https://pubs.usgs.gov/fs/2015/3081/fs20153081.pdf

[8] Aplin, P. & Atkinson, P.M. 2004, "Predicting missing field boundaries to increase per-field classification accuracy", Photogrammetric Engineering and Remote Sensing, vol. 70, no. 1, pp. 141-149.

[9] Gao, B.-C. 1996. NDWI - A normalized difference water index for remote sensing of vegetation liquid water from space. Remote Sensing of Environment 58: 257-266.

[10]https://landsat.usgs.gov/gap-filling-landsat-7-slc-single-scenes-using-erdas-imagine-TM

[11] Jin Chen, Xiaolin Zhu, James E. Vogelmann,Feng Gao, Suming Jin A simple and effective method for filling gaps in Landsat ETM+ SLC-off images. Remote Sensing of the Environment.

[12] https://yceo.yale.edu/how-fill-gaps-landsat-etm-images

[13] Harris, P. M., & Ventura, S. J. (1995). The integration of geographic data with remotely sensed imagery to improve classification in an urban area. Photo-grammetric Engineering & Remote Sensing, 61(8), 993–998.

[14] Anderson, R., Hardy, E. E., Roach, J. T., & Witmer, R. E. (1976). A land use and land cover classification system for use with remote sensor data. USGS Professional Paper 964. Washington, DC.

[15]https://landsat.usgs.gov/gap-filling-landsat-7-slc-single-scenes-using-erdas-imagine-TM