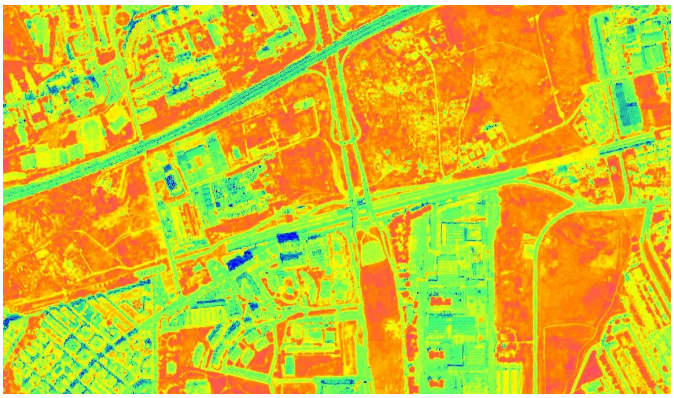
NATIONAL TECHNICAL UNIVERSITY OF ATHENS

MSC DATA SCIENCE AND MACHINE LEARNING

Geospatial Big Data Analytics
Project 1



Pavlos Kalfantis
MSc Student
pavloskalfantis@mail.ntua.gr
Student ID: 03400134

April 2022

Introduction

The first semester project of the graduate course 'Geospatial Big Data Analytics', offered by the Remote Sensing Laboratory at the National Technical University of Athens, consists of analyzing geospatial data and time series data, recovered from Data and Information Access Services. During the first part, different Data and Information Access Services were explored and compared, such as the services offered by the EU under the Copernicus program, as well as services offered by private companies like Google and Amazon. Next, different geospatial questions were analyzed and answered using Google Earth Engine, like calculating the Normalized difference vegetation index (NDVI) for different areas of Greece or analyzing NDVI time series and fitting harmonic models. Finally, a small web application is built on Google Earth Engine, allowing users to retrieve, show and export different images from the Landsat 8 satellite.

Question 1

There are currently many different services that provide free access to Earth Observation (EO) data and other geospatial data, as well as provide computation engines that can analyze them. Since satellites started gathering and saving this type of data around the world, the amount of information that can be retrieved has reached the petabyte scale. To facilitate and standardise access to this EO data, the European Commission has funded the deployment of five cloud-based platforms. These platforms are known as the DIAS, or Data and Information Access Services. The five DIAS online platforms allow users to discover, manipulate, process and download Copernicus data and information. All DIAS platforms provide open EO data from the Copernicus program through standard web services and protocols such as OpenSearch, WFS, WMS, WCS and CSW. Besides that, these systems allow users to contract on-demand services to store and process EO data sets as well as to directly access these data sets. The DIAS systems allow users to execute their applications in a cloud environment where data is stored, avoiding the movement of data through the network and so improving the processing performance. These platforms are CREODIAS, sobloo, WEkEO, ONDA and mundi. Moreover, private companies Google and Amazon provide similar services, Google Earth Engine and Amazon Web Services Earth respectively. As part of the first question of this project, four of the services mentioned above were studied and compared. Below, the key functionality of these services is briefly presented.

Google Earth Engine

The most advanced service and the one that is subsequently used to answer the questions of this project is Google Earth Engine (GEE). GEE is a cloud-based platform that enables large-scale scientific analysis and visualization of geospatial data sets. It provides a Javascript API and a Python API for data management and analysis. For the JavaScript version, a web Integrated Development Environment (IDE) is also provided,

where the user has easy access to available data, applications and real-time visualization of the processing results. The Python API is available through a module and has a structure similar to its JavaScript version. GEE uses four object types to represent data that can be manipulated by its API. The Image type represents raster data that can consist of one or more bands, which contain a name, data type, scale, and projection. A stack or a time series of Images is represented by the ImageCollection type. GEE represents vector data through the Feature type. This type is represented by a geometry (point, line, or polygon) and a list of attributes. The FeatureCollection type represents groups of relatedFeatures and provides functions to manipulate this data, such as sorting, filtering, and visualization.

CREODIAS

CREODIAS is an environment that brings processing to Earth Observation data. Its design allows Third Party Users to prototype and build their own value-added services and products. Its set of pertinent tools guarantees simplicity, scalability and repeatability of any services' value chain. Four applications are the cornerstones for users' activities within the CREODIAS environment:

- Portal is the main hub for account management and information handling, with FAQs, forum and news feed available.
- EO Browser is a satellite imagery browser with ability to visualize chosen Earth Observation images.
- EO Finder is an advanced search engine, designed for selecting products based on sets of specific variables - such as time, place, collection, processing level. It can be accessed with JSON-based API.
- Cloud Dashboard serves as a tool for overview and management of ones processing resources.

Data processing is a fundamental function offered by CREODIAS. Pursuing most efficient and favorable solutions, the platform offers a full set of virtual resources: Virtual Machines with a choice of different operating systems available, easily mounted storage volumes with object storage solutions, virtual network and appliances like firewalls and VPN concentrators. Finally, the Earth Observation data repository covers full data sets of the Sentinel satellite family - full sets of products from Sentinel-1, 2, 3 and 5p. Additionally, Landsat-5, 7 and 8 and Envisat products are available as well.

WEkEO

WEkEO is the EU's Copernicus DIAS reference service for environmental data, virtual environments for data processing and skilled user support. The WEkEO DIAS addresses a wide range of users like institutions, private companies, entrepreneurs and scientists and enables them to develop their products, applications and value-added

services, or to access a one-stop-shop for all Copernicus data and information products. It provides users with a single distributed tool for accessing, visualising and analysing all Copernicus data and services, including big-data analysis tools, to develop applications tailored to their specific needs. Another key functionality of WEkEO is their JupyterHub. The Jupyter Environment is embedded in WEkEO and it allows writing and iterating on Python code for data analysis. Finally, WEkEO provides the WEkEO drive tool for users to share their work performed, as well as virtual machine services, only in the advanced paid subscription mode.

Mundi

Mundi is a consortium of public and private companies, led by Atos, a global leader in hi-tech transactional services, unified communications, cloud, big data and cybersecurity. It gives unlimited, free and complete access to Copernicus data and information. Mundi provides also a scalable computing and storage environment for third parties, either individual or companies. Third parties are empowered to offer advanced value-adding services integrating Copernicus with their own data and tools to the benefit of their own users. Mundi provides a cloud-based one-stop shop for all Copernicus satellite data and imagery, as well as information from the Copernicus services, and also gives access to sophisticated processing tools and resources. Its data collections include all Sentinel and Landsat EO data, which can be accessed by its Geodata Access Graphical User Interface (GUI) or its API. Finally, Mundi provides a Download service, either through their Geodata GUI or with REST API.

Question 2

The second question of the project was about analyzing an area of interest on the map and producing a true color and a false color composite image, as well as the NDV index, for the date in 2019 with the least cloud coverage. Two more indices, the Normalized Difference Water Index (NDWI) and the Enhanced Vegetation Index (EVI) are also calculated. True color composite uses visible light bands red (B4), green (B3) and blue (B2) in the corresponding red, green and blue color channels, resulting in a natural colored result, that is a good representation of the Earth as humans would see it naturally. A false color image is used to reveal or enhance features otherwise invisible or poorly visible to a human eye. In other words, a false color composite is a multispectral image interpretation using the standard visual RGB band range (red, green, and blue). In our case bands B5 (Near Infrared - NIR), B4 and B3 are used on the red, green and blue band respectively.

Using the standard bands, some indices can be calculated that when rendered on an image can convey useful information. The Normalized Difference Vegetation Index is the normalized difference of bands B5 and B4, showing areas with high vegetation, and the Normalized Difference Water Index is the normalized difference between bands B3 and B5, showing areas that are covered in water. Finally, the Enhanced Vegetation

Figure 1: Region of Interest

Index is calculated using bands B5, B4 and B2 and is a different way to show areas with high vegetation. This and the subsequent questions are answered using Google Earth Engine's JavaScript code editor. The code used to answer these questions is presented throughout this report, but it can also be found **here**. The region of interest that was selected is a rectangle southeast of the city of Larissa, containing mostly croplands. The area of interest and the produced images are shown on figures 1,2,3 and 4

```
// Question 2

// Filtered dataset for 2019 with less than 20% cloud coverage
var data19 = L8
  .filterDate('2019-01-01', '2019-12-31')
  .filterBounds(ROI)
  .filter(ee.Filter.lte('CLOUD_COVER', 20));

// NDVI spectrum
var ndviSpectrum = {min: 0, max: 0.5, bands: 'NDVI',palette: ['red','yellow','
    green']};
// Makes colors "pop out" more; NDVI Default Spectrum Range is {min: -1, max: 1};

var trueColorSpectrum = {bands : ['B4','B3','B2'],min:0,max:20000,};
var falseColorSpectrum = {bands : ['B5','B4','B3'],min:0,max:30000};

// Compute NDVI
var addNDVI = function(image) {
  var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
};

```

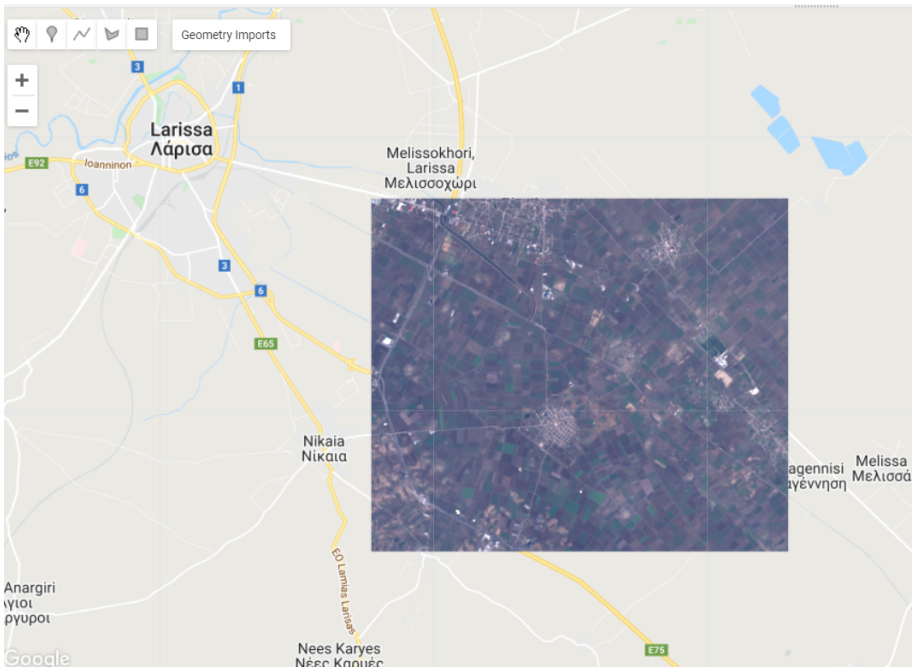


Figure 2: True Color Composite for least clouded day of 2019

```
// Compute NDWI
var addNDWI = function(image){
  var ndwi = image.normalizedDifference(['B3', 'B5']).rename('NDWI');
  return image.addBands(ndwi);
};

var addEVI = function(image){
  var evi = image.expression('2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE +
      1))',
                             {'NIR' : image.select('B5'),
                              'RED' : image.select('B4'),
                              'BLUE': image.select('B2')
                             }).rename('EVI');
  return image.addBands(evi);
};

//Apply NDVI, NDWI, EVI to collection
var data19 = data19.map(addNDVI);
var data19 = data19.map(addNDWI);
var data19 = data19.map(addEVI);
```

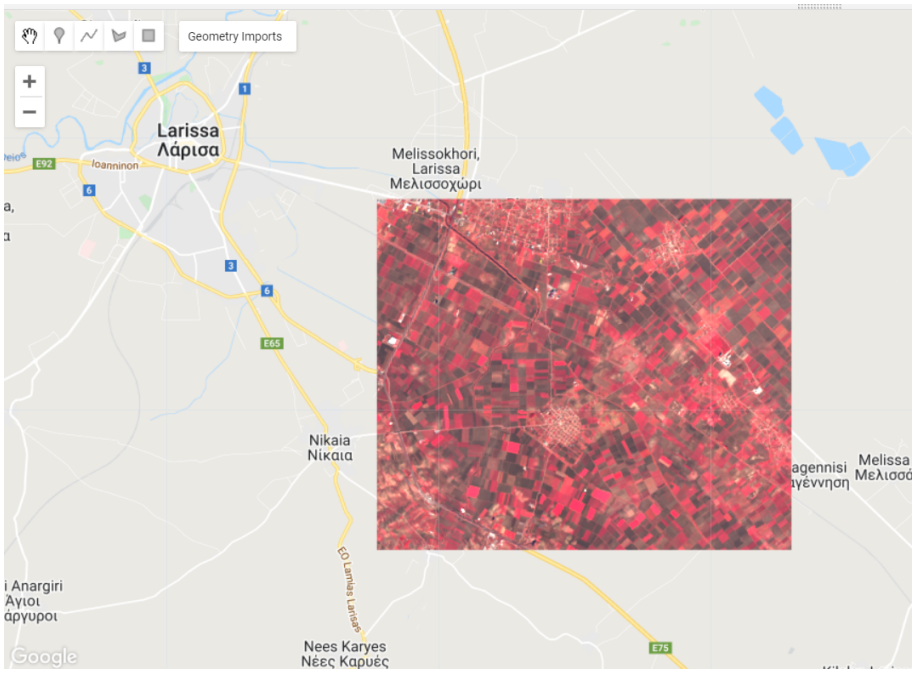


Figure 3: False Color Composite for least clouded day of 2019

Question 3

The third question of the project was about calculating and visualizing the day of the year, both for 2018 and 2019, that each pixel of the map had the highest NDVI value. So for each year, two images were created: A picture showing the maximum NDVI value of each pixel throughout the year and a map showing which day of the year (from 0 to 365 mapped from black to white, black meaning early in the year and white meaning late in the year) this value was achieved. We can see the differences between the different years, both at the NDVI values and the date of the year that these maximum values were observed on figures 5, 6, 7 and 8.

```
//Question 3

//DOY for max ndvi
var addDOY = function(image) {
  var img_date = ee.Date(image.date());
  var img_doy = ee.Number.parse(img_date.format('D'));
  return image.addBands(ee.Image(img_doy).rename('doy').toInt());
};

var data18 = L8
  .filterDate('2018-01-01', '2018-12-31')
  .filterBounds(ROI)
  .filter(ee.Filter.lte('CLOUD_COVER', 20));

var l8_2019_ndvi = data19.map(addDOY); //NDVI already added
var l8_2018_ndvi = data18.map(addNDVI).map(addDOY);
```

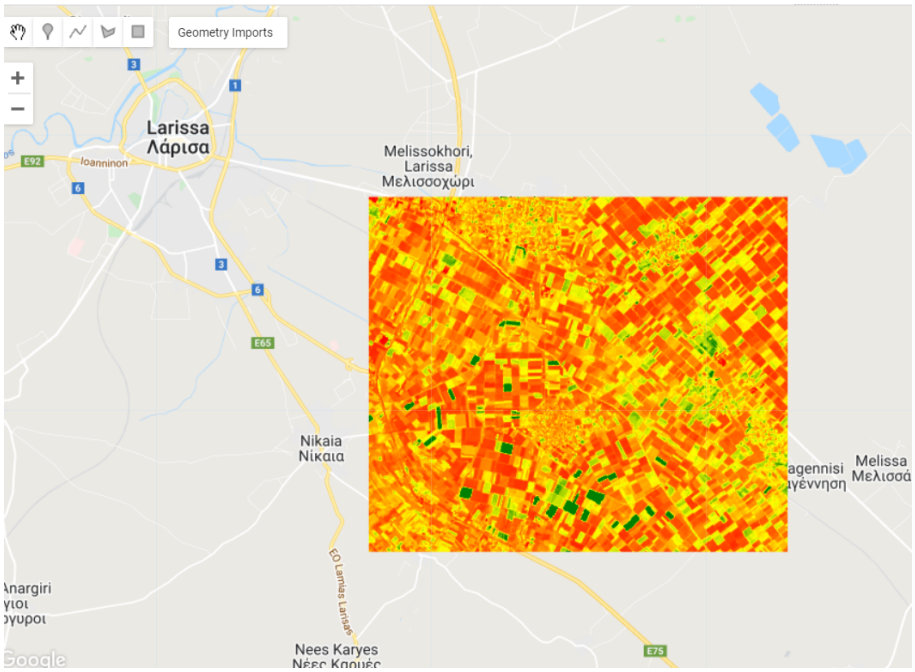


Figure 4: Normalized difference vegetation index for least clouded day of 2019

```
var greenest_2019 = l8_2019_ndvi.qualityMosaic('NDVI');
var greenest_2018 = l8_2018_ndvi.qualityMosaic('NDVI');


// Display the result.
Map.addLayer(greenest_2019, ndviSpectrum, 'Greenest pixel composite 2019');
Map.addLayer(greenest_2018, ndviSpectrum, 'Greenest pixel composite 2018');

// Visualize the 'date' image
Map.addLayer(
    greenest_2019.select('doy'),
    {'palette': ['black', 'white'], 'min': 1, 'max': 365},
    'Greenest doy 2019'
)

Map.addLayer(
    greenest_2018.select('doy'),
    {'palette': ['black', 'white'], 'min': 1, 'max': 365},
    'Greenest doy 2018'
)
```

Question 4

The fourth question of the project was about creating 4 different polygons within the region of interest. For these polygons, all Landsat 8 data was recovered in order to create time series of the NDVI value. The goal then was to add the best fitted harmonic

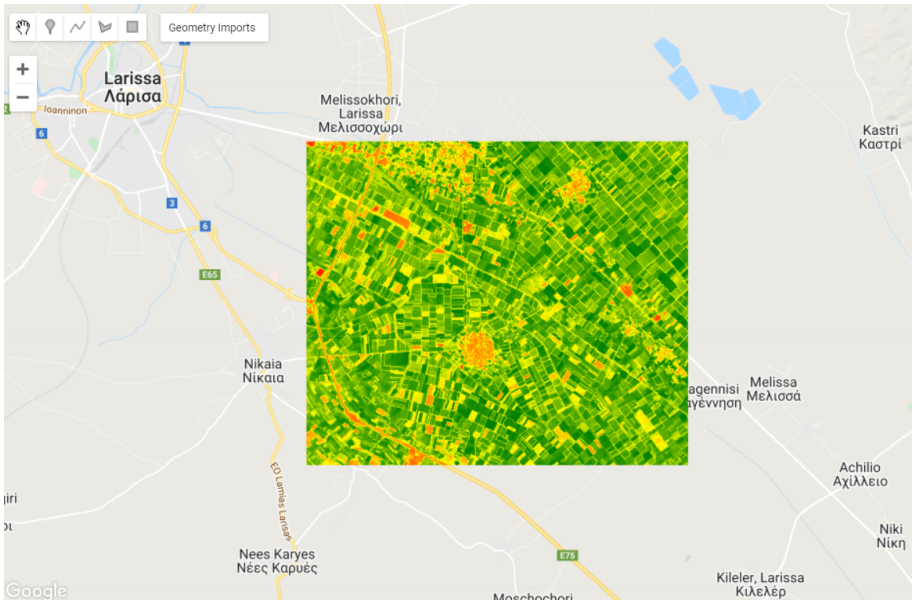


Figure 5: Greenest Pixel Composite for 2018

lines to these time series and visualize the seasonality of the NDVI value throughout the years. The harmonic model line equation (consisting of a constant, a linear and a harmonic term) is:

$$NDVI_t = \beta_0 + \beta_1 t + \beta_2 cos(2\pi\omega t) + \beta_3 sin(2\pi\omega t) \quad (1)$$

The code used in Javascript to fit the harmonic line is shown below. The four polygons that were analyzed are shown on figure 9 and the four created time series with the fitted lines are shown on figures 10, 11, 12, 13.

```
//Question 4
var data = L8
  .filterBounds(TS1)
  .filter(ee.Filter.lte('CLOUD_COVER', 20))

// This field contains UNIX time in milliseconds.
var timeField = 'system:time_start';

// Use this function to add variables for NDVI, time and a constant
// to Landsat 8 imagery.
var addVariables = function(image) {
  // Compute time in fractional years since the epoch.
  var date = ee.Date(image.get(timeField));
  var years = date.difference(ee.Date('1970-01-01'), 'year');
  // Return the image with the added bands.
  return image
    // Add an NDVI band.
    .addBands(image.normalizedDifference(['B5', 'B4']).rename('NDVI')).float()
    // Add a time band.
    .addBands(ee.Image(years).rename('t').float())
    // Add a constant band.
```

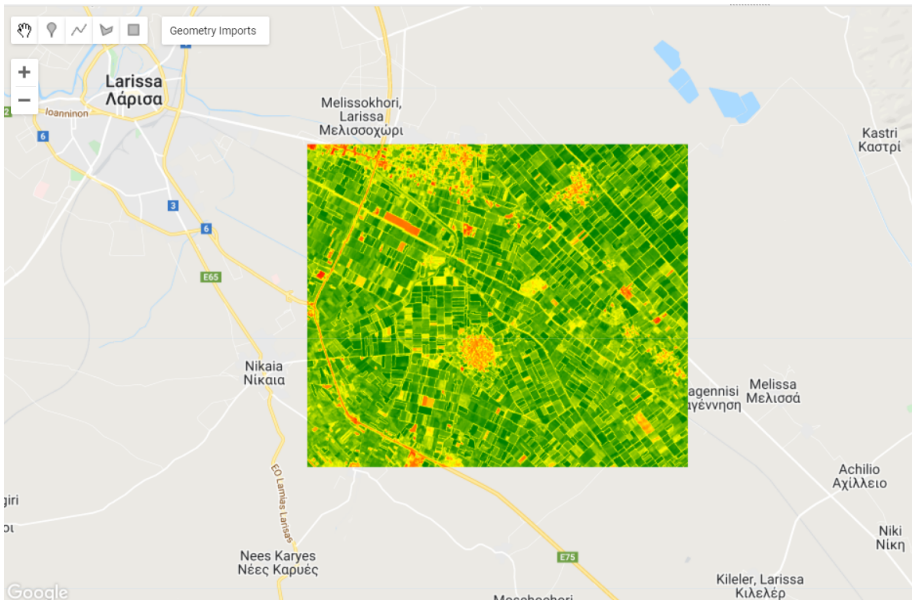


Figure 6: Greenest Pixel Composite for 2019

```
    .addBands(ee.Image.constant(1));
};

var data = data
  .map(addVariables);

// Harmonic trend ----------------------------
// Use these independent variables in the harmonic regression.
var harmonicIndependents = ee.List(['constant', 't', 'cos', 'sin']);

// Add harmonic terms as new image bands.
var harmonicLandsat = data.map(function(image) {
  var timeRadians = image.select('t').multiply(2 * Math.PI);
  return image
    .addBands(timeRadians.cos().rename('cos'))
    .addBands(timeRadians.sin().rename('sin'));
});

// Name of the dependent variable.
var dependent = ee.String('NDVI')

// The output of the regression reduction is a 4x1 array image.
var harmonicTrend = harmonicLandsat
  .select(harmonicIndependents.add(dependent))
  .reduce(ee.Reducer.linearRegression(harmonicIndependents.length(), 1));

// Turn the array image into a multi-band image of coefficients.
var harmonicTrendCoefficients = harmonicTrend.select('coefficients')
  .arrayProject([0])
  .arrayFlatten([harmonicIndependents]);

// Compute fitted values.
var fittedHarmonic = harmonicLandsat.map(function(image) {
```

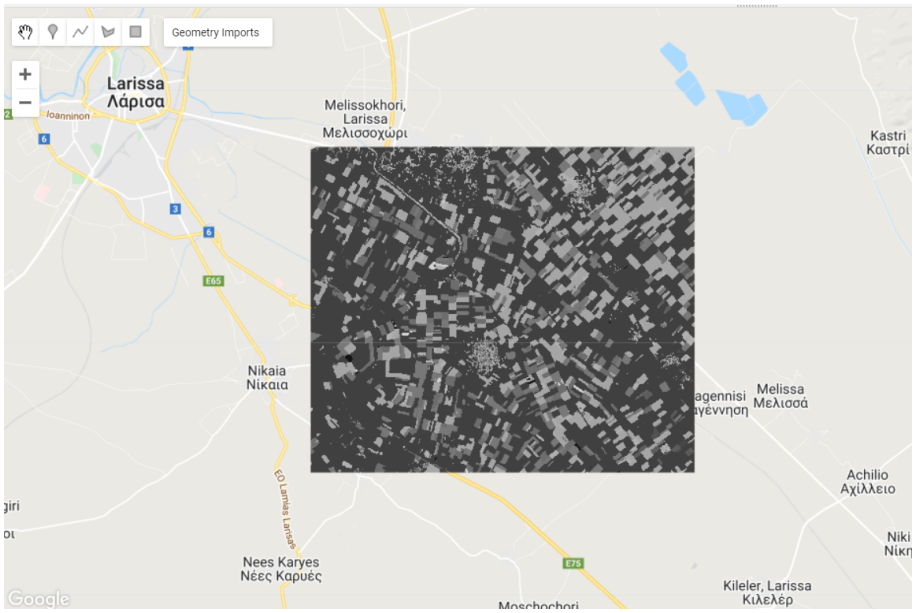


Figure 7: Day of Year of Maximum NDVI for 2018

```
  return image.addBands(
    image.select(harmonicIndependents)
      .multiply(harmonicTrendCoefficients)
      .reduce('sum')
      .rename('fitted'));
});

// Plot the fitted model and the original data at the ROI.
print(ui.Chart.image.series(
  fittedHarmonic.select(['fitted','NDVI']), ROI, ee.Reducer.mean(), 30)
    .setSeriesNames(['NDVI', 'fitted'])
    .setOptions({
      title: 'Harmonic model: original and fitted values',
      lineWidth: 1,
      pointSize: 3,
}));
```

Question 5

On question 5, a bigger region was selected, that contained different types of land usage. The area analyzed in shown on figure 14 By containing urban areas, crops, water and forests on a bigger picture, the goal is to build a classifier that can distinguish between these **4 classes** and produced an image showing the classification of each pixel, based on the values of several bands of this pixel. In order to build the classifier, a composite image of the classification region was created (mean of all available pictures). The characteristics of the image that are used to train the classifiers are all bands from B1 to B8, plus the previously calculated NDV index.

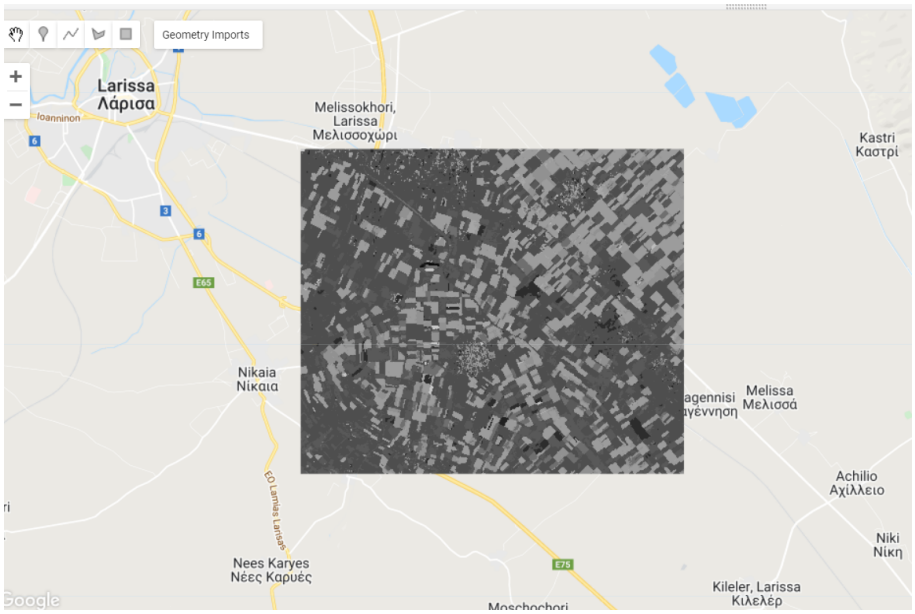


Figure 8: Day of Year of Maximum NDVI for 2019

Next step was to specify training points by assigning pins on the map to the four classes (around 100 points for each class, out of which 20 are assigned to a test set and 80 to a training set, shown on 15). Two different classifiers are used, a Decision Tree Classifier and a Support Vector Machine Classifier. By training the two models, two classified images are produced, shown on figures 16 and 17 as well as the classification report showing how many data points of the test set were correctly assigned, on figures 18 and 19. Both classifiers have high accuracy (0.91 for the decision tree and 0.96 for the support vector machine). We can visually observe that the decision tree classifier has a worse performance identifying between cropland and urban areas, assigning the urban class (red) to more areas that are in fact lands.

```
1  //Question 5
2
3  // CART Classification
4
5  var classification_collection = L8
6    .filterBounds(classification_region)
7    .filter(ee.Filter.lte('CLOUD_COVER', 20))
8
9  //Apply NDVI to collection
10 var classification_collection = classification_collection.map(addNDVI);
11
12 var classification_image = classification_collection.mean().clip(
       classification_region)
13 Map.addLayer(classification_image,trueColorSpectrum,'trueColor-
       classification_image');
14 Map.addLayer(classification_image,falseColorSpectrum,'falseColor-
       classification_image');
15 Map.addLayer(classification_image,ndviSpectrum,'ndvi-classification_image');
```

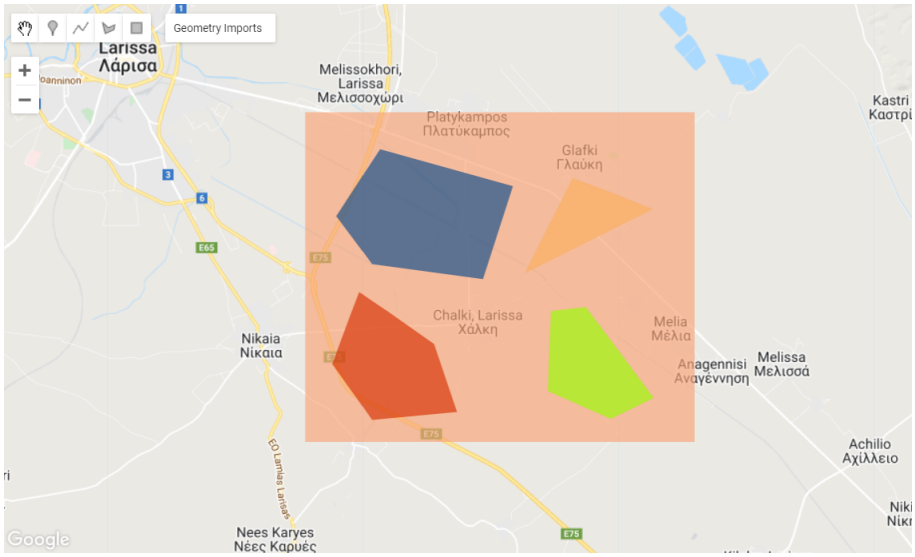


Figure 9: Polygons for NDVI time series

```
16
17
18 var label = 'Class';
19 var bands = ['B1','B2','B3','B4','B5','B6','B7','B8','NDVI'];
20
21
22 var input = classification_image.select(bands);
23
24 var training = Urban.merge(Water).merge(Vegetation).merge(Forest);
25
26 var trainImage = input.sampleRegions({
27   collection: training,
28   properties: [label],
29   scale:30
30 });
31
32 var trainingData = trainImage.randomColumn();
33 var trainSet = trainingData.filter(ee.Filter.lessThan('random',0.8));
34 var testSet = trainingData.filter(ee.Filter.greaterThanOrEquals('random',0.8));
35
36 //CART and SVM classifiers
37
38 var classifier = ee.Classifier.smileCart().train(trainSet,label,bands);
39 var classifier_svm = ee.Classifier.libsvm().train(trainSet,label,bands);
40
41 var classified = input.classify(classifier);
42 var classified_svm = input.classify(classifier_svm);
43
44 var landcoverPalette = [
45   '#5DADE2', //water
46   '#C0392B', //urban
47   '#ABEBC6', //vegetation
48   '#117A65', //forest
49 ];
50
```

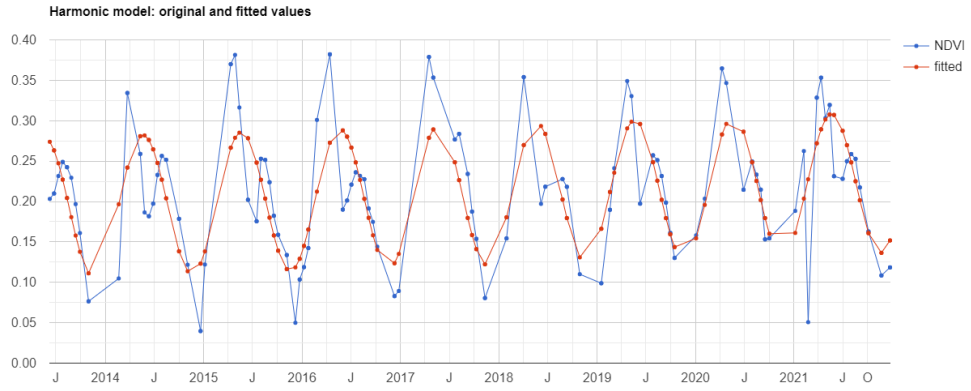


Figure 10: Time Series of Polygon 1

```
Map.addLayer(classified, {palette:landcoverPalette, min:0, max:3},'
    classification_cart');
Map.addLayer(classified_svm, {palette:landcoverPalette, min:0, max:3},'
    classification_svm');

var confusionMatrix = ee.ConfusionMatrix(testSet.classify(classifier)
  .errorMatrix({
    actual: 'Class',
    predicted: 'classification'
  }));
var confusionMatrix_svm = ee.ConfusionMatrix(testSet.classify(classifier_svm)
  .errorMatrix({
    actual: 'Class',
    predicted: 'classification'
  }));

print(confusionMatrix)
print(confusionMatrix.accuracy())

print(confusionMatrix_svm)
print(confusionMatrix_svm.accuracy())
```

Question 6

The final task of the project was building a custom web application on Google Earth Engine, that can provide some functionality to the user and can answer some geospatial questions. The app built allows the user to select an available image from the Landsat 8 collection within the dates that they specify and render one of four visualizations. They can produce a true color composite or a false color composite, similar to question 2 of this report, or visualize the NDV or NDW index. The user scrolls to the location of the map they want to create the image, they can choose the visualization and the date

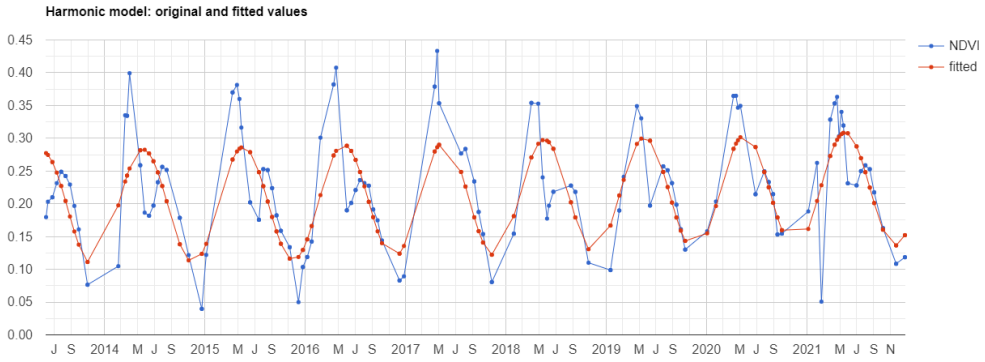


Figure 11: Time Series of Polygon 2

range they want to produce and the picks one of the available images for the specific location on the map. Finally, they have the option to export and save the created image to their Google Drive. This is an example of an app created on Google Earth Maps with user functionality and a limited user interface, as they have the option to select specific dates, areas and images they want to create out of the Landsat 8 collection. The code used for the app can be found **here**, while the app can be found on this link: **https://pavkalfantis.users.earthengine.app/view/geo-data-project**.

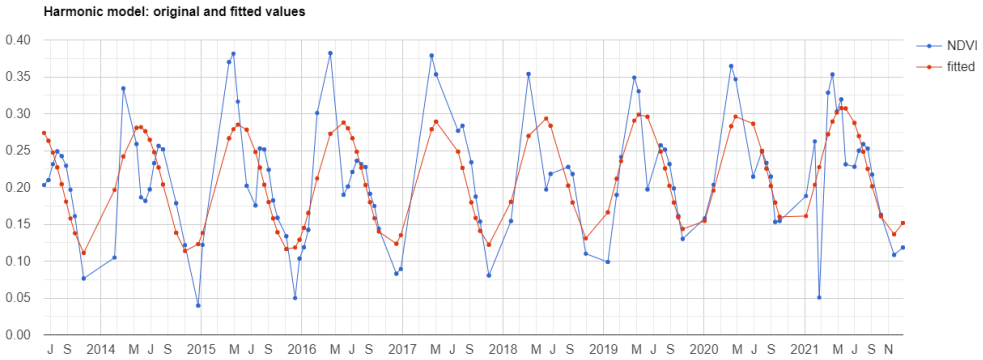


Figure 12: Time Series of Polygon 3

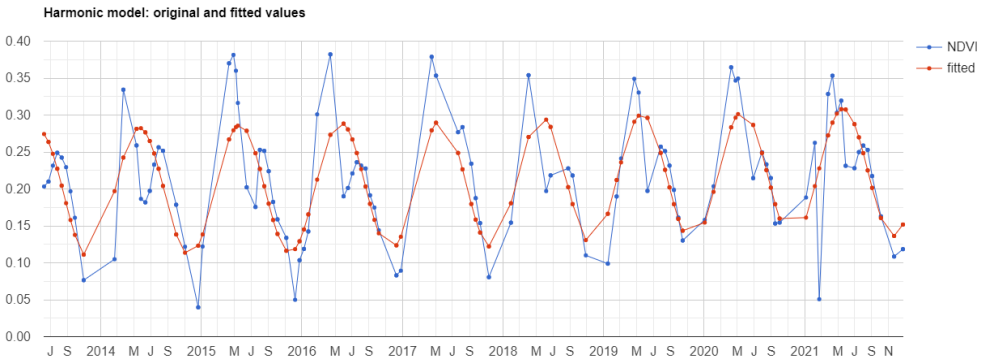


Figure 13: Time Series of Polygon 4

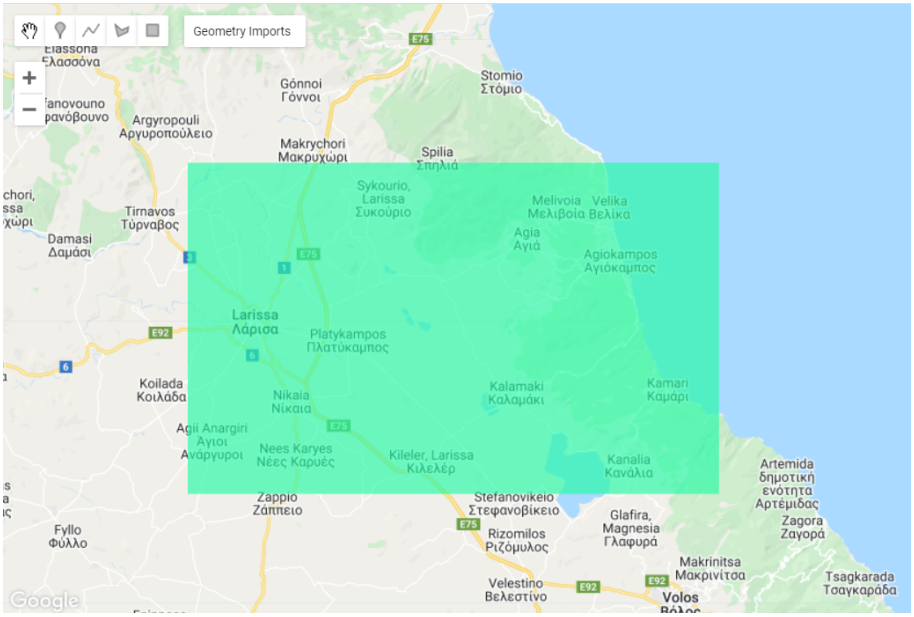


Figure 14: Region Analyzed for Classification

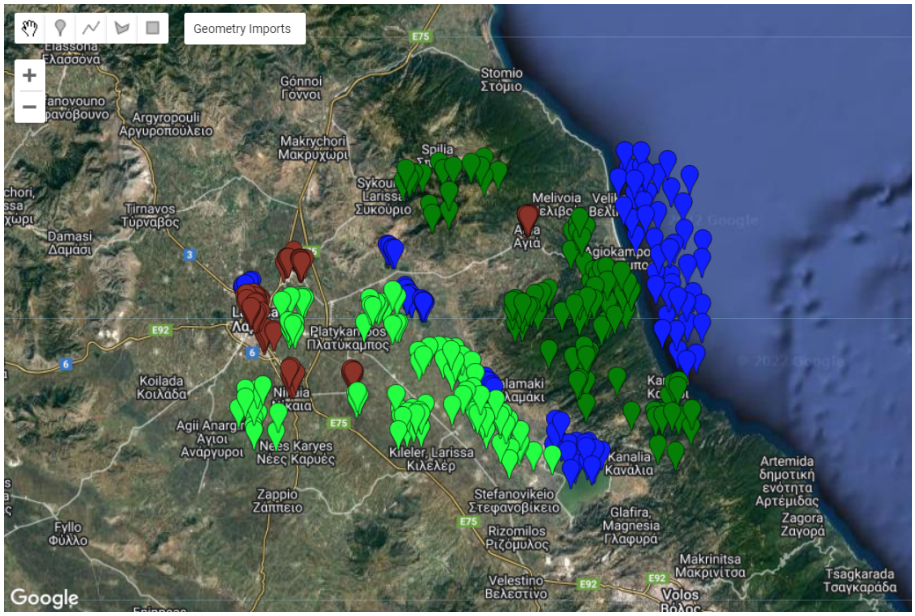


Figure 15: Training Points and Classes for Classification

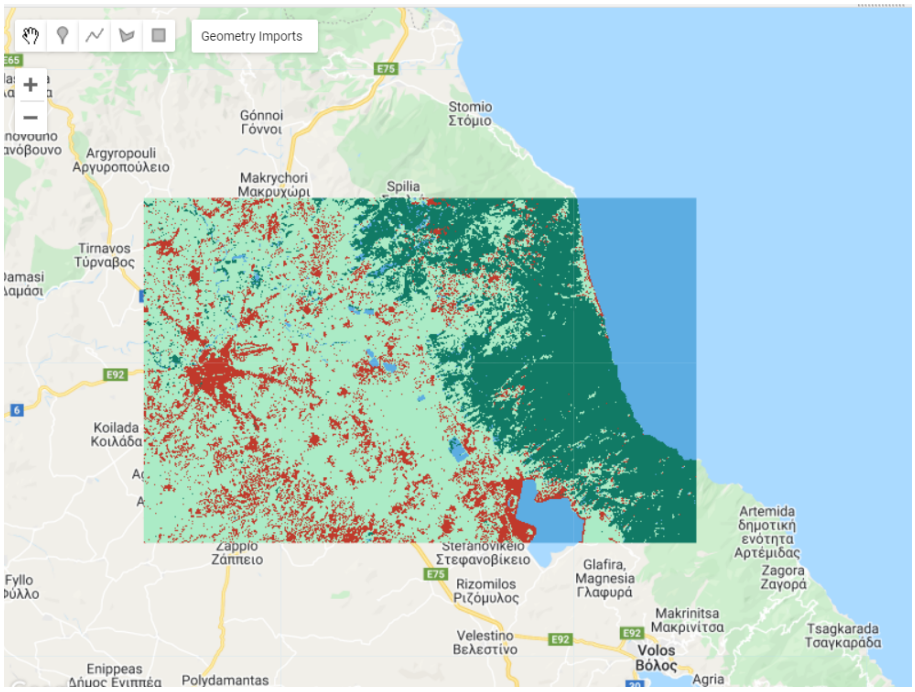


Figure 16: Classified Image using CART

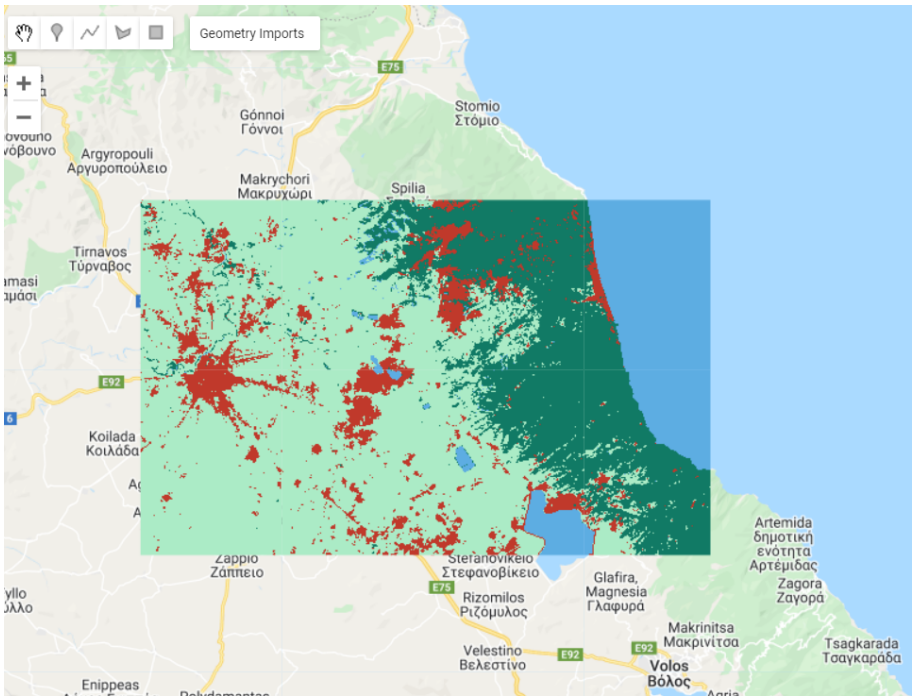


Figure 17: Classified Image using SVM

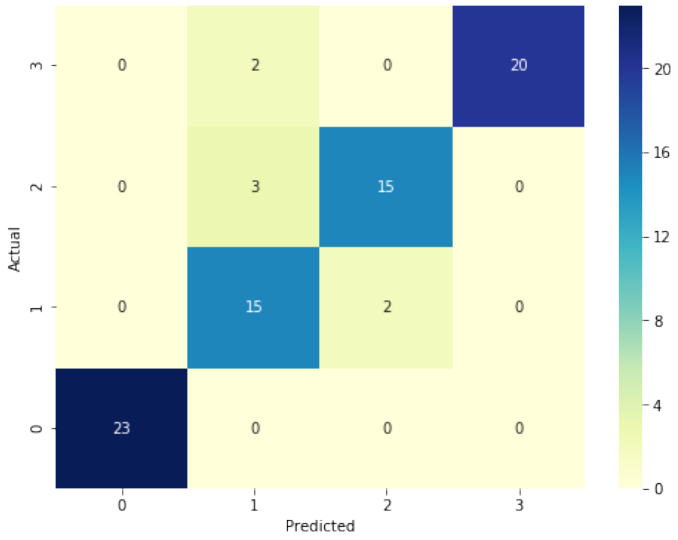


Figure 18: Classification Report of Test Set using CART

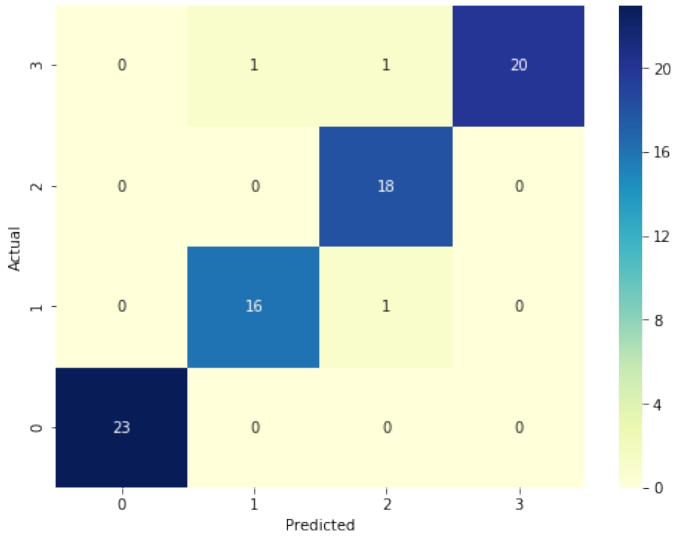


Figure 19: Classification Report of Test Set using SVM