Regen Organics Analytics App (v1.0.0)
User Manual

Table of Contents

1 Introduction

The Regen Organics Analytics App, hereby referred to as 'the app', is a Python web application developed by a team of volunteers at DataKind in partnership with Regen Organics. The primary goal of this application is to provide Regen Organics staff and ground team, the 'end users', with an analytics dashboard that can be used to make more informed and data-driven business decisions regarding agricultural practices and trends in Kenya by referring to pre-existing data generated and analyzed during the research of the work as well as being able to generate and incorporate their own data. Being an open-source product, the entire source code is available to the public from the project repository: https://github.com/RM503/Regen_analytics_app. We hope that the open-source nature of the app can be leveraged to make further improvements and make it a more robust agricultural analytics dashboard in the future.

This manual aims to provide a very detailed description of how to run the app on a local environment, its various features and functionalities, as well as notes on data interpretation. The workings of the app are based on the extensive initial research phase of the project that included satellite data generation, polygon segmentation from satellite rasters and model training on vegetation indicators (please click on the GitHub link appearing in the footnote to be directed to the project repository).[1]

2 Running the app

i As Docker containers

One way the app can be run is through Docker containers. Hence, users who wish to operate the app will required a running version of Docker installed on their PCs. For ease of use, we recommend installing Docker Desktop application, which has a simple graphical user interface (GUI), enabling users to monitor images and containers. This can be downloaded from https://docs.docker.com/desktop/ and is available for all major operating systems. After downloading Docker, the app's image needs to be pulled from the repository, which can be done using the following command -

```
docker pull rmahbub503/regen_organics_analytics_app:v1
```

The app can be built in several ways. The Docker Desktop app provides a Terminal, which can be opened through its GUI. Alternatively, the Terminal for MacOS/Linux or the Command Prompt (or Power Shell) for Windows users can also be used for the build. Building the app requires executing the following command -

```
docker build -t <image_name:image_tag> .
```

1 Link to research project repository: https://github.com/RM503/DataKind_Geospatial

where the image tag can be any chosen name (preferably version names). The `.` at the end of the end of the command will build the app directly in the user's current directory. This can be changed at the user's discretion. After the image is build, it can be run as *containers* through -

```
docker run --env-file .env.docker -p 8000:8000 <image_name:image_tag>
```

The `.env.docker` file contains API keys, passwords and other authentication items required for smooth operation of the app. A list of required services can be found in Sec. (3iii). Once the app is run, it can be accessed through `localhost:8000`.

ii Through AWS

This section will be completed after AWS deployment.

3 Important notes

i A note on region names

One aspect of the app that might be confusing to end users is in regards to the names of the regions where the distributors are located. The locations of the various distributors in the initial data were inconsistent in their positions in the administrative level hierarchy in that some had either subcounty or county level information, while for others these informations were absent with only a street address provided. Hence, for the sake of consistency, we decided to subsume them to the county level. Region names appear as the county name with numbers succeeding an underscore. This is done to keep distributors in the same county separate (there are cases where this happens and may happen in the future). For example, a distributor in Trans-Nzoia county is described as Trans_Nzoia_1 and so on if there are more distributors in this location. In Table 1 we provide a table that describes the correspondence between distributor and county, with an example where two distributors are located in the same county (hence the use of '_1' and '_2' in the region labels):

Table 1: Distributors by region.

Distributor	County
Jawapa Stores	Kajiado_1
Esther Naserian Karanja	Kajiado_2
Solai Agrovet	Laikipia_1
Paves Vet Ago	Trans_Nzoia_1

As of writing, `v1.0.0` of the app contains data on these four regions in the form of geographical polygons, soil data, planting cycles, moisture levels and more. These polygons were segmented from satellite raster images surrounding these distributor regions during the research phase and the farm polygons amongst them were identified by training a machine learning (ML) model of vegetation indicators using training data from the Trans-Nzoia region. A lack of consistency in phenological data between regions meant that the ML model trained on Trans-Nzoia could not be properly applied to others, with inconsistencies in the Sentinel satellite's data acquisitions in the various regions further impacting data quality. Generalization may not be possible and training data from different regions will be required for more accurate farm classification. Future versions may will contain custom region creation capabilities, allowing for a more expansive user experience.

ii Application architecture and design

The app was designed with simplicity in mind. It uses Flask, a Python-based web microframework, as a backend on which multiple analytics dashboards, built using Plotly Dash, are served.[2] This was done regardless of other, more robust, web frameworks like Django. Data related to the broader project and the app are stored in a PostgreSQL database hosted on Supabase. The database is capable of accepting newly queried farm polygons (an update from `v1.0.0-alpha`), perform calculations on the backend, update tables using triggers and `INSERT` the new data into appropriate tables. Although both anon and authenticated users can `SELECT` data (view the data), Row-Level-Security (RLS) policies on the different Supabase tables restricts `INSERT` functionality to only authenticated users. Hence, any user with access to the app can view the data and download geometries and statistics related to newly queried polygons without being able to push them into the database. However, `UPDATE` and `DELETE` functionalities are limited to administrators, who are members/users with direct access to the Supabase project dashboard.

The frontend landing page is created using Python's Jinja2 library with minimal use of HTML and Javascript coding, which includes a login form for authenticated users. As of writing, the landing page links to four dashboards, which are -

1. **Initial Market Data:** This contains results from analyses carried out on market data initially provided by Regen Organics. This is a static dashboard with no download/insert capabilities. However, the dashboard can be updated with newly obtained market data upon future releases with more analyses.

2. **Polygon Generator:** This contains an interactive map which can be used to not only view existing farm polygons but also query new polygons. Furthermore, these newly queried polygons can be inserted into the database. Currently, the app does not contain functionalities to add new regions.[3] This feature will be incorporated in a newer release. Moreover, the current version also lacks location searches through longitude/latitude (users can only pan and zoom and user region bounding boxes through the preloaded polygons), a feature which will also be added upon a newer release.

3. **Farmland Characteristics:** This is the dashboard with arguably the most computationally intensive backend. Polygons queried from the *Polygon Generator* dashboard can be uploaded to generate important summary statistics related to soil quality, planting cycles and moisture

2 The app was initially prototyped with FastAPI. However, passing login information to the dashboard proved to be challenging since Plotly Dash runs on Flask. This arises primary due to the fact that FastAPI uses ASGI while Dash, which is built on top of Flask, uses WSGI.

3 In actuality, polygons outside the provided regions can be drawn. However, such polygons will not contain any appropriate 'region' tag that may be required for further analysis.

content of the farms. The newly generated data can then be inserted into their appropriate tables in Supabase, trigger functions for updates on additional tables.

4. **Farmland Statistics:** This dashboard contains an aggregate of all the data contained in the Supabase database. These include a farm polygon map showing distributions of vegetation indices and soil quality markets, distribution of planting cycles and more. A future version of the app will also contain results on drought risk assessment for the various regions, which is extremely important for scientific and market research.

Fig. 1 provides an architectural flowchart of the app, showing how the different components interact with each other. Details of its workings are found in Sec. (4).

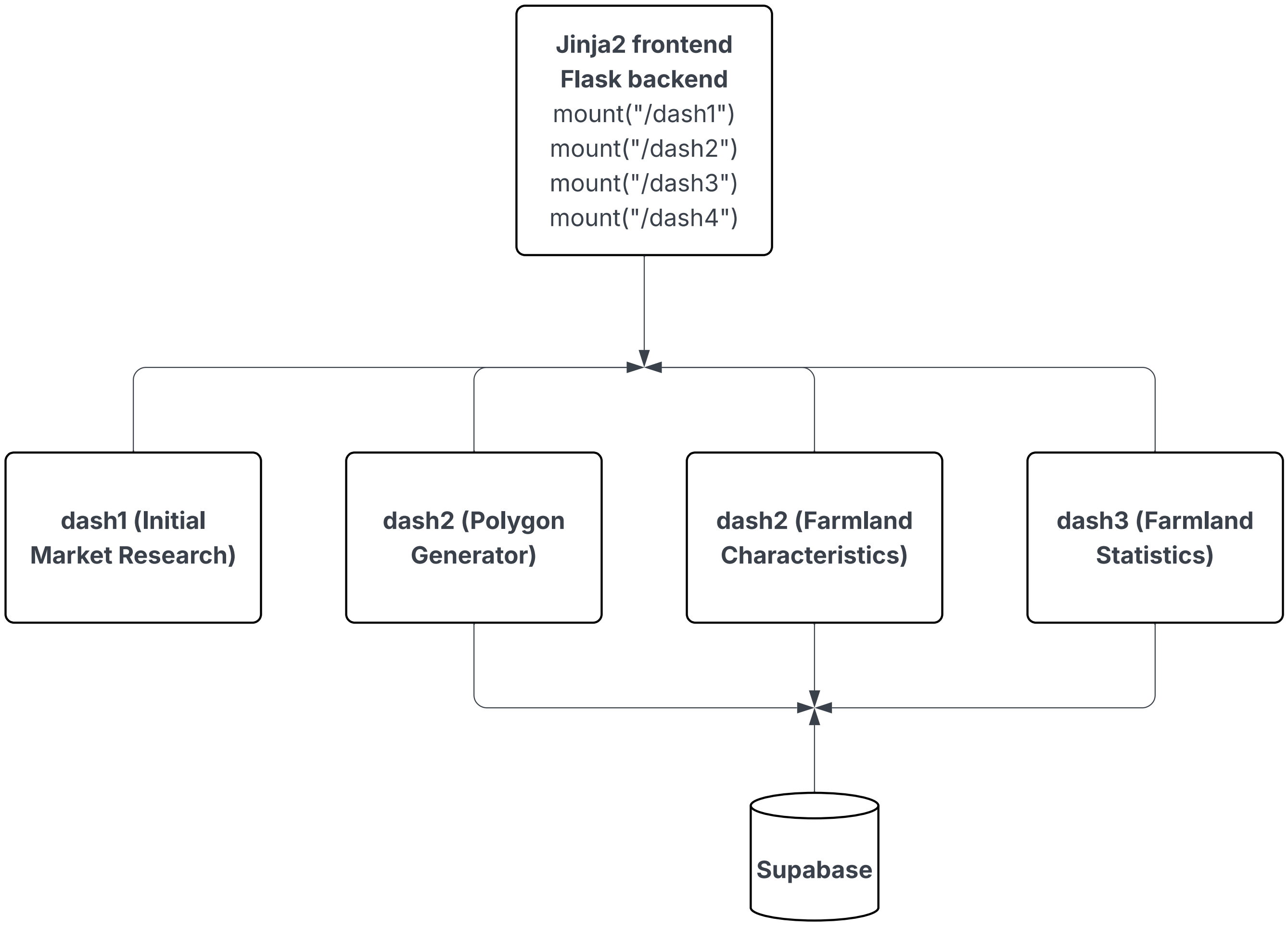

Jinja2 frontend
Flask backend
mount("/dash1")
mount("/dash2")
mount("/dash3")
mount("/dash4")
dash1 (Initial Market Research)
dash2 (Polygon Generator)
dash2 (Farmland Characteristics)
dash3 (Farmland Statistics)
Supabase

Figure 1: The architectural flowchart of the app showing the different components and their interactions.

iii Required services and APIs

Needless to say, the app requires the use of multiple services and APIs for its backend for performing data requests for computations, most of which are used in the *Farmland Characteristics* dashboard of the app. Here we provide a brief description of the requirements and where to obtain them.

1. **Google Earth Engine (GEE):** This is used to obtain the various indices used to assess vegetation health and moisture content for further calculations. A GEE account can be created for free and registered as ‘noncommercial’ for free-tier usage indefinitely. More details on this can be found here: https://earthengine.google.com/noncommercial/. Although the free-tier version will lack many advanced features, it is sufficient for a small scale app. Once an account has been created, a GEE project can be created with a given `GEE_PROJECT` name. Furthermore, through the Google Cloud Platform (GCP), an Earth Engine Service Account, with its own `EE_SERVICE_ACC_EMAIL`, must be created to generate a `credentials.json` in order for GEE to run on a Docker container environment (the `GEE_PROJECT` is enough for running it standalone on a personal device). Details on creating a service account obtaining the credentials file can be found here: https://developers.google.com/earth-engine/guides/service_account.

2. **iSDA soil API:** The iSDA API is used for making soil data requests to the iSDA servers. Developer access can be obtained easily from https://www.isda-africa.com/isdasoil/developer/, with which the `ISDA_USERNAME` and `ISDA_PASSWORD` can be set.

3. **Supabase:** Supabase is a Postgres development platform where the project database is hosted. Even though it is a paid service, Supabase offers a free-tier with storage and compute sufficient enough for a small scale project. However, with an increase in the project’s scale, a professional version, or higher, is highly recommended. From Supabase, the following will be required -

    - `SUPABASE_URL`: This is the base URL for the Supabase project of the form `https://<project-ref>.supabase.co`. This is how the end users interact with the database.

    - `SUPABASE_KEY`: This is a public/anon key for client-side operations where we do not require full admin access.

    - `SUPABASE_SERVICE_ROLE_KEY`: This is a special service role key with administrative access which can be used to create new authenticated users.

    - `DB_URL`: This is a direct connection to the PostgreSQL database. Even though the `SUPABASE_URL` gives access to the database, it is limited to a certain number of rows for `SELECT` operations. This greatly limits visualization aspects of the dashboard. A direct database connection bypasses this limitation.

    - `SUPBASE_USER_EMAIL` and `SUPABASE_USER_PASSWORD`: These are the Supabase user account and password information for the project.

iv Important terminologies

Significant portions in both the research phase and the app use remote sensing phenological indices for assessing vegetation health and moisture content. These calculations are performed by measuring how much light is reflected off of the surfaces across the different bands of the electromagnetic (EM) spectrum. Here we provide somewhat of a detailed description of two such vegetation indices, without turning the manual pedantic, namely the 'Normalized Difference Vegetation Index' and 'Normalized Difference Moisture Index'.

1. **Normalized Difference Vegetation Index (NDVI):** The NDVI is an index that is used measure the *greenness* of vegetation. It is calculated via

   $$NDVI = \frac{NIR - RED}{NIR + RED}$$

   where NIR and RED represent the near infrared and red bands of the EM spectrum. This formula is motivated by the fact that green, healthy vegetation reflects a lot of the near infrared radiation incident on it while absorbing a lot of the incident red light. In the ideal case where all the incident red light is absorbed results in an $NDVI = +1$. On the other hand, vegetation that has been stressed less near infrared and more red, lowering the NDVI. Apart from vegetation health, the NDVI can also be used to track crop growth, allowing one to determine growing seasons, planting cycles and other patterns. The following is a breakdown of how to interpret different ranges of NDVI values:

   1. $[+0.5, +1.0]$: This range can indicate the presence of denser vegetation (forests and canopy cover) or mid-to-full season crops with ample foliage.
   2. $[+0.2, +0.5]$: This range usually indicates areas with sparser vegetation (grasslands and similar regions) or early-to-mid season crops with developing foliage.
   3. $[0.0, +0.2]$: This range typically represents exposed soil or regions with very little vegetation or planting stages of crops.
   4. $[-1.0, 0.0]$: This range of values typically indicate a distinct lack of vegetation, usually characteristic of barren rock, urban areas, cloud/snow/water cover. Negative values also frequently appear in satellite data for pixels which contain cloud contamination.

   As stated previously, peaks on the NDVI time-series curves typically represent peak growing season for crops and the presence of multiple growing seasons during a calendar year indicate multiple crop cycles. In many regions in Kenya, especially those with climates similar to Trans-Nzoia, there are typically two planting seasons annually coinciding with long short rainy seasons during the middle and end of the year. A sample NDVI time-series illustrating this is shown in Fig. (2). In the computational backend of the app, a peak-finding algorithm is used to detect the calculate the number of planting cycles per year. We note that, despite smoothing, the time-series data remain imperfect and there is generally some uncertainty regarding the number of planting cycles calculated from the data. Regardless of this limitation, one can infer important patterns related to planting cycles that may be characteristic of certain regions or even groups of adjacent farms.

2. **Normalized Difference Moisture Index (NDMI):** The NDMI is used to measure the amount of water retained in vegetation. It is calculated via

$$NDMI = \frac{NIR - SWIR}{NIR + SWIR}$$

where SWIR refers to short-wave infrared. The reason behind using SWIR arises from the fact that vegetation with good water content strongly absorbs waves in the SWIR band. Hence, higher NDMIs indicate better water retention. Unlike NDVI, this particular index is not as widely used in agronomy. However, it can be a simple way to assess potential irrigation and moisture levels of farms. There is generally no acceptable interpretation for ranges of NDMI values unlike NDVI. Moreover, there are regional variations to how large the NDMI peaks are. However, for this work we provide ranges of values for baseline interpretations

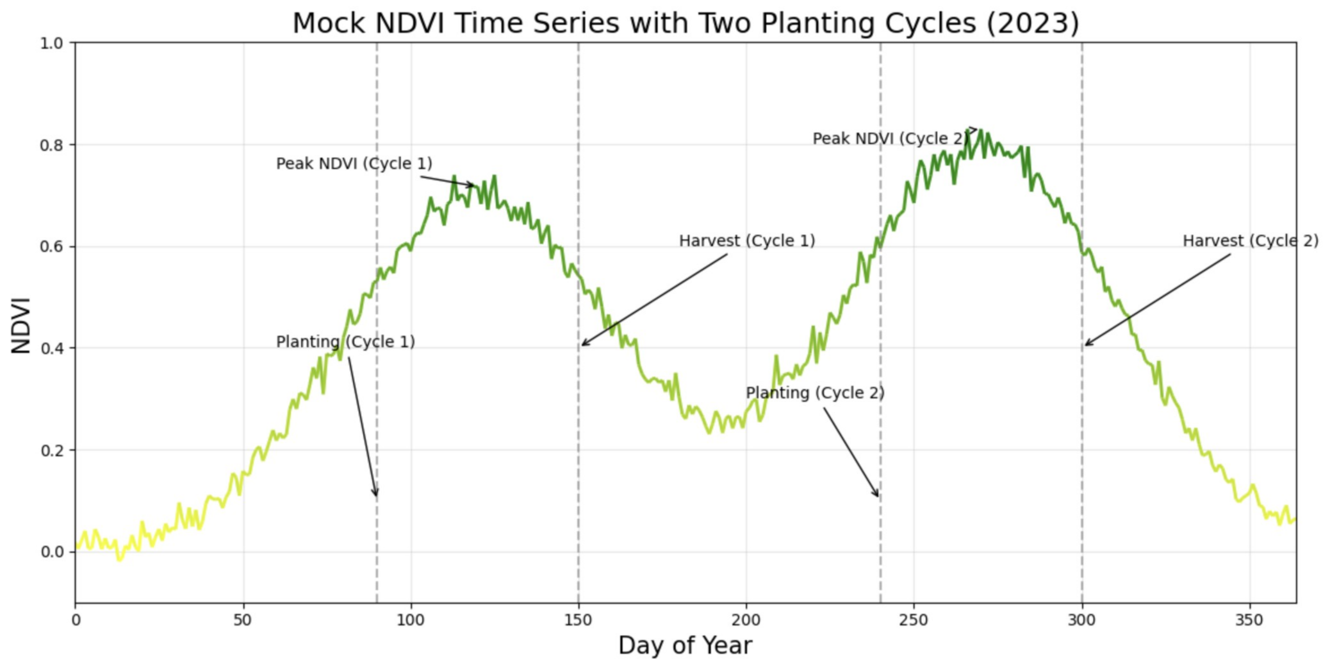


Figure 2: The figure shows a mock NDVI time-series curve with two planting cycles during a given year. The curve tends to more greener colors as peak growing season begins.

1. $NDMI \geqslant 0.38$: high moisture (possible indications of irrigation)
2. $0.25 \leqslant NDMI < 0.38$: medium moisture
3. $0.20 \leqslant NDMI < 0.25$: approaching low moisture
4. $NDMI < 0.20$: low moisture

v List of tables in Supabase

Finally, we discuss the various data tables that are stored in Supabase. It is important to understand the data flow and triggers that comprise of this app to ensure whether or not all the tables are being populated correctly and in the correct order upon `INSERT` actions. We note that even though we are generating NDVI and NDMI time-series data, and the research phase having heavily relied on them, Supabase currently does not store them for pre-existing or newly queried polygons. This merely comes down to the fact the during the app's development, the free-tier version of Supabase was used which offers very limited storage space. In any event, such time-series data can be generated using GEE for the required polygons.

The following lists the tables stored in Supabase with their corresponding columns-

- `farmpolygons`: This stores all the farm polygons that are present in the project, both pre-existing and newly queried ones. In essence, the *Polygon Generator* dashboard makes use of this table for visualization and data inserts. The columns of this table are `uuid`, `region`, `area` (of the polygon, in acres), `geometry` (polygon vertices in longitude and latitude) and `created_at` timestamp.
- `highndmidays`: This table keeps a tally of all polygons that spend any number of days in the high NDMI region. The columns contained here are `uuid`, `year`, `region`, `high_ndmi_days` and `created_at`.
- `peakvidistribution`: This table stores the peak values of NDVI and NDMI for all the polygons in the database. The columns are `uuid`, `year`, `region`, `ndvi_max`, `ndmi_max` and `created_at`.
- `ndvipeaksperfarm`: This table stores all the peaks occurring in the NDVI time-series across the years for all the polygons. The columns are `uuid`, `ndvi_peak_date`, `region`, `ndvi_peak_value`, `peak_position` and `created_at`. `ndvi_peak_date` refers to the date at which the peak occurs while `peak_position` refers to the ordering of the peak (1st, 2nd etc.).
- `soildata`: This table contains soil quantities for polygons obtained through iSDA, with the largest number of columns amongst all tables in the database. They are `uuid`, `region`, `bulk_density` (g/cm^3), `calcium_extractable` (ppm), `carbon_organic` (g/kg), `carbon_total` (g/kg), `clay_content` (g/kg), `iron_extractable` (ppm), `magnesium_extractable` (ppm), `nitrogen_total` (g/kg), `ph`, `phosphorous_extractable` (ppm), `potassium_extractable` (ppm), `sand_content` (g/kg), `silt_content` (g/kg), `stone_content` (g/kg), `sulphur_extractable` (ppm), `texture_class`, `zinc_extractable` (ppm), `geometry` and `created_at`. For more details on iSDA soil quantities, please visit https://www.isda-africa.com/isdasoil/.

The preceding tables are ones that directly updated through data `INSERT`s from the dashboard.[4] The following tables are those that are triggered by modifications to the preceding ones.

4 Technically, the `region` and `geometry` columns of `soildata` are also triggered by `farmpolygons`. However, all the other columns are populated through API calls to the iSDA server.

- `moisturecontent`: This table stores a count of farms in the different moisture classifications (high, medium, approaching low and low) with `region`, `moisture_content`, `counts` and `updated_at` as columns. This is one of the several pieces of data that is available in the *Farmland Characteristics* dashboard. Data updates in this table are triggered when updates in the `peakvidistribution` table occur.
- `ndvipeaksmonthly`: This tables stores the monthly distribution of the number of NDVI peaks occurring in each region, containing the columns `ndvi_peak_year`, `region`, `ndvi_peak_month`, `ndvi_peaks_per_month` and `updated_at`. The `ndvi_peaks_per_month` refers to the number of NDVI peaks occurring per month in the given region. Data updates in this table are triggered when updates in the `ndvipeaksperfarm` table are detected.
- `ndvipeaksannual`: Similar to `ndvipeaksmonthly`, this table stores the annual distribution of the number of NDVI peaks occurring in the different regions across the years, essentially counting the numbers of NDVI peak occurrences. This information can be used to infer the distribution of the number of planting cycles in the different regions. Data updates in this table are also triggered when updates in the `ndvipeaksperfarm` table are detected.

vi Notes on soil quantities

The soildata table comprises of a broad range of indicators related to soil quality and health and will prove essential in terms of market research and potential customer outreach. These data are available at two different levels: 0 − 20 cm and 20 − 50 cm. In this project, we retrieve the former. The various quantities can be broadly categorized into those that are measured in ‘grams per kilogram of soil’ (g/kg) and the ‘extractables’ that are measurement in ‘parts per million’ (ppm). We will not attempt to provide a detailed account of how to interpret the various soil quantities and the amounts required for soil to be considered healthy will vary from region to region, for which an expert should be consulted. We note (at least for the app’s administrators) that representations of values for `texture_class` will depend on whether it is showing up in the data table in the *Farmland Characteristics* dashboard or in the `soildata` table in Supabase. Table 2 provides the correspondence between USDA soil texture classification and integer encodings-

Table 2: This table contains USDA soil texture encodings and interpretations.

USDA soil texture classification	Integer encoding	Characteristics
Sand	1	Very gritty, drains quickly, holds less nutrient
Loamy sand	2	Slightly more silt/clay than sand
Sandy loam	3	Better water/nutrient retention than sand
Loam	4	Balanced mix; good fertility and drainage
Silt loam	5	Smooth and floury; holds water well
Silt	6	Almost all silt; compacts easily

Sandy clay loam	7	More clay, still gritty
Clay loam	8	Roughly equal sand, silt and clay
Silty clay loam	9	Silt and clay dominant
Sandy clay	10	Clay + sand, little silt
Silty clay	11	Clay + silt, little sand
Clay	12	Mostly clay; sticky, slow infiltration

One final note aimed at developers is that data generation should be treated distinctly depending on the needs of the project. In the app where polygon counts are limited, the data can be easily requested through API calls. However, for a large collection and regions and polygons, as was the case for the polygons generated during the research phase, bulk download is highly recommended. This is due to rate limits put in place for data requests and, even with asynchronous functions, such a method of data retrieval becomes extremely inefficient. The iSDA data are hosted in AWS and can be retrieve from https://registry.opendata.aws/isdasoil/. It contains a Jupyter Notebook tutorial on how to extract the soil quantities based on regions of interest.

4 Application functionalities

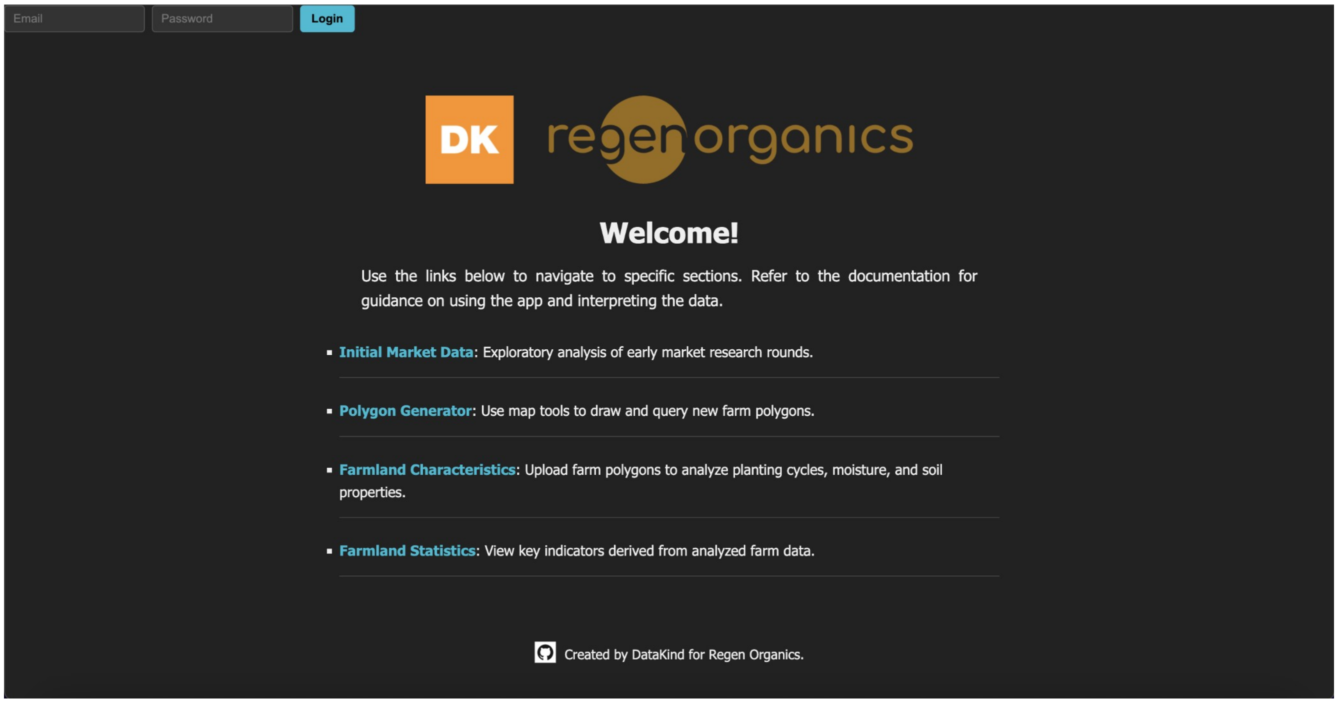
With the introductory notes out of hand, we can now discuss in detail the app's functionalities and recommended workflow. We highly recommend that the application workflow is followed in order to ensure that all the required data tables in Supabase are populated by the appropriate data. Google Drive links to tutorials related to the various dashboards will be provided in the corresponding subsections.

i Landing page

Launching the app directs users to the landing page shown in Fig. (3). The landing page consists of a login form at the top left and a list of the different dashboards currently available (see Sec. (3ii)). Authenticated users, i.e. those who have been provided with login credentials, will be able to log into the app. We note that both anonymous[5] and authenticated users have `SELECT` privileges – they will be able to view data. However, only authenticated users will be able to `INSERT` newly queried data. This is done with the consideration that, while all Regen end users should have the capability to view existing data, only a select group of approved personnel will have modification privileges.

Users with login credentials have their email id and passwords stored in the Supabase project's *Authentication Table.* Only project administrators can add/remove authenticated users. Creating these credential will fall under the purview of Regen Organics.

5 An *anonymous* user is someone who does not possess login credentials.



Welcome!

Use the links below to navigate to specific sections. Refer to the documentation for guidance on using the app and interpreting the data.

- **Initial Market Data**: Exploratory analysis of early market research rounds.
- **Polygon Generator**: Use map tools to draw and query new farm polygons.
- **Farmland Characteristics**: Upload farm polygons to analyze planting cycles, moisture, and soil properties.
- **Farmland Statistics**: View key indicators derived from analyzed farm data.

Created by DataKind for Regen Organics.

Figure 3: This is an image of the app's landing page with login form and the different dashboards.

ii Initial Market Data

This dashboard contains interactive graphs on exploratory data analysis carried out on initial market data provided by Regen Organics. As such, this particular dashboard has no data generation features. The dashboard contains the following plots-

1. New Customer Count Over Time By Region
2. Total Sales Volume Over Time By Region
3. Purchase/Sales Per Region
4. Number of Farms Per Distributor
5. Win Rates (by several categories)

iii Polygon Generator

As described in Sec. (3ii), this dashboard is meant for exploration of the distributor regions in order to both view pre-existing polygons and generate new polygons of interest. Ideally, these polygons are farms and that should be the case for newly queried ones that are being identified via visual inspection. However, users will note that various pre-existing polygons are not representative of farms. This has two primary reasons. Features contained in Sentinel-2 rasters of buffer regions around the distributors were first segmented using a geospatial version of Meta's *Segment Anything Model* (SAMGEO), producing numerous polygons. As it stands, SAMGEO has not been trained (or fine-tuned) on geospatial data and, as such, cannot distinguish between polygons representing different geographical/geological features. As a result, our research phase included an ML model training and evaluation phase where over 3000 polygons in the Trans-Nzoia region, containing various class labels (farms, fields, trees and man-made), were manually identified and trained on NDVI time-series which are characteristic of the various classes. Subsequently, the trained model was applied on the rest of the data for inference, arriving at a set of polygons which have been predicted to be farms.

However, vegetation phenological data is very inconsistent across regions and a model trained on one region cannot be easily generalized to others. We observed that while Trans-Nzoia, and counties surrounding it, have very distinct signatures of crop growing seasons in their NDVI time-series data, other places like Mashuru simply lack such a signature. Since the ML model relies on such patterns and signatures, it does not perform well in these regions. Further compounding the problem is the inconsistent fly-by rates of the Sentinel-2 satellite. While data is available at a 5-day frequency for Trans-Nzoia, in others it is much less frequent. These problems dramatically reduced the number of regions over which the model could be applied and without obtaining training data on the other regions and identifying a more unified approach for farmland detection, manual segmentation appears to be the most logical way ahead for many distributor locations.

For a video tutorial on the *Polygon Generator* dashboard, please use the link: Polygon Generator video tutorial.

Recommended workflow: The recommended workflow can be seen in the video. The region tab is used to navigate to a one of the specified regions which have been identified for the project. With v1.0.0, each region consists of a set of polygons identified as farms from the earlier research phase. Once selected, the map will zoom into the centroid of that region from where other farms can be identified and selected using from polygon draw tool on the map's panel. Each time a polygon is selected, the data table on the right hand side is updated with the polygon's `uuid`, `region`, `area` and `geometry`. The app restricts the number of polygons that can be drawn at any given time to 5. The reason behind this will be made clear in Sec. (iv). Furthermore, the area of the polygon is also restricted to 3,000 acres. This is done to prevent very large polygons to be drawn. After the required polygons are drawn on the map, they should be downloaded by clicking on the `Download polygons` button, which will automatically download a `polygon.csv` file (or a numbered variant if a file of this name already exists). Unless the user is authenticated, the `INSERT` button will remain greyed out. If authenticated users identify these polygons as farms of interest, they should `INSERT` the polygons into the `farmpolygons` table by clicking the button. We note that if a polygon is drawn without a chosen region, it will either default to the `Default` location or the one that was chosen previously.

We are working on an update which will include the following features -

- A longitude/latitude box which the users can use to zoom into locations of interest without actively having to find them from eye.
- The list of regions will be updated to include other distributor region bounding boxes. These new regions will likely not contain any pre-existing polygons. However, these bounding boxes will give users greater freedom to start collecting data on newer locations that might be of more immediate interest.

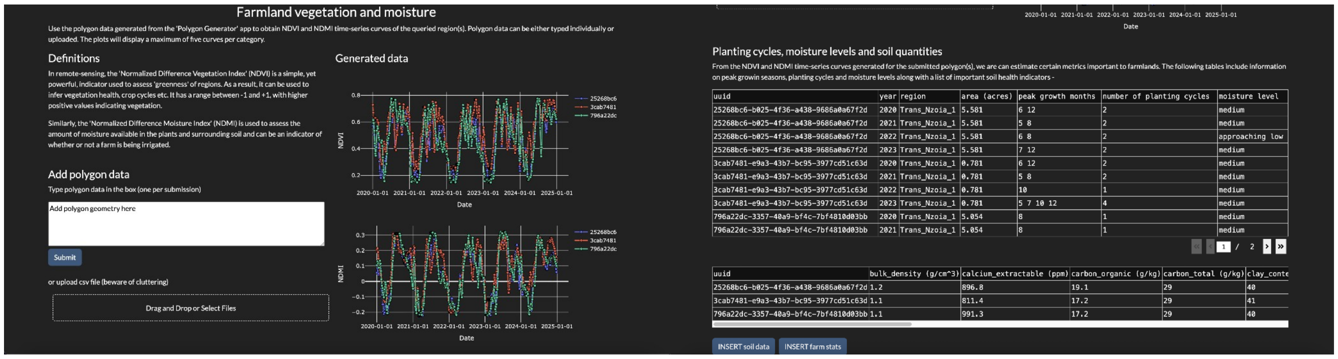
iv Farmland Characteristics

This dashboard is aimed at generating various data related to the polygons generated in the *Polygon Generator* dashboard. Data provided to this dashboard are processed using the GEE backend to generate the NDVI and NDMI time-series data, which are subsequently used for farm statistics calculations. Screenshots of the dashboard are shown in Fig. (4). The left panel shows the interface where polygon data are provided. This can be done in two ways-

1. **Data box:** Single polygon geometries can be pasted into the box and submitted. Data will be generated for the single polygon. The data box option should only be used to check farm characteristics and is not recommended for the workflow. This is because the data box only accepts a valid geometry and is agnostic of region information. Data on the polygon that are subsequently generated will contain 'None' for region.

2. **Drag and Drop/Upload:** Polygon data downloaded from the *Polygon Generator* dashboard can be dragged and dropped/uploaded to the dashboard. This then iterates through all the geometries in the file to generated the same set of data. Since the GEE backend can be slow to process the requests and generate the VI data, we decided to limit the number of polygons that can be downloaded to 5. To process a large number of polygons, GEE uses asynchronous batch processing and upload to Google Drive, which we felt introduces additional design complications. Furthermore, a large number of polygons will also obscure details from the plots.

Once the data are processed, the dashboard also displays two data tables as seen on the right panel. The first table contains `uuid`, `year, region`, `area (acres)`, `peak grow months`, `number of planting cycles` and `moisture level` as columns. These are all quantities calculated from the geometry data and NDVI and NDMI curves. `peak grow months` refers to the months where NDVI peaks are detected for each year of data. The second table contains soil quantities that have been discussed in detail.



Farmland vegetation and moisture

Use the polygon data generated from the 'Polygon Generator' app to obtain NDVI and NDMI time-series curves of the queried region(s). Polygon data can be either typed individually or uploaded. The plots will display a maximum of five curves per category.

Definitions

In remote-sensing, the 'Normalized Difference Vegetation Index' (NDVI) is a simple, yet powerful, indicator used to assess 'greenness' of regions. As a result, it can be used to infer vegetation health, crop cycles etc. It has a range between -1 and +1, with higher positive values indicating vegetation.

Similarly, the 'Normalized Difference Moisture Index' (NDMI) is used to assess the amount of moisture available in the plants and surrounding soil and can be an indicator of whether or not a farm is being irrigated.

Add polygon data

Type polygon data in the box (one per submission)

Add polygon geometry here

or upload csv file (beware of cluttering)

Planting cycles, moisture levels and soil quantities

From the NDVI and NDMI time-series curves generated for the submitted polygon(s), we are can estimate certain metrics important to farmlands. The following tables include information on peak growin seasons, planting cycles and moisture levels along with a list of important soil health indicators -

uuid	year	region	area (acres)	peak growth months	number of planting cycles	moisture level
25268bc6-b025-4f36-a438-9686a0a67f2d	2020	Trans_Nzoia_1	5.581	6 12	2	medium
25268bc6-b025-4f36-a438-9686a0a67f2d	2021	Trans_Nzoia_1	5.581	5 8	2	medium
25268bc6-b025-4f36-a438-9686a0a67f2d	2022	Trans_Nzoia_1	5.581	6 8	2	approaching low
25268bc6-b025-4f36-a438-9686a0a67f2d	2023	Trans_Nzoia_1	5.581	7 12	2	medium
3cab7481-e9a3-43b7-bc95-3977cd51c63d	2020	Trans_Nzoia_1	0.781	6 12	2	medium
3cab7481-e9a3-43b7-bc95-3977cd51c63d	2021	Trans_Nzoia_1	0.781	5 8	2	medium
3cab7481-e9a3-43b7-bc95-3977cd51c63d	2022	Trans_Nzoia_1	0.781	10	1	medium
3cab7481-e9a3-43b7-bc95-3977cd51c63d	2023	Trans_Nzoia_1	0.781	5 7 10 12	4	medium
796a22dc-3357-40a9-bf4c-7bf4810d03bb	2020	Trans_Nzoia_1	5.054	8	1	medium
796a22dc-3357-40a9-bf4c-7bf4810d03bb	2021	Trans_Nzoia_1	5.054	8	1	medium

1 / 2

uuid	bulk_density (g/cm^3)	calcium_extractable (ppm)	carbon_organic (g/kg)	carbon_total (g/kg)	clay_conte
25268bc6-b025-4f36-a438-9686a0a67f2d	1.2	896.8	19.1	29	40
3cab7481-e9a3-43b7-bc95-3977cd51c63d	1.1	811.4	17.2	29	41
796a22dc-3357-40a9-bf4c-7bf4810d03bb	1.1	991.3	17.2	29	40

Figure 4: This is a screenshot of the 'Farmland Characteristics' dashboard. (Left) Here, the generated polygon data are uploaded with the VI time-series graphs plotted on the right. (Right) The statistics and other data generated are displayed in data tables.

Once data are generated, authenticated users and push the data through two buttons- `INSERT soildata` and `INSERT farmstats`. The former pushes new data to the `soildata` table while the latter pushes data to multiple tables while trigger updates in others simultaneously. A video tutorial can be found in the following link: Farmland Characteristics video tutorial.

Recommend workflow: The workflow is quite simple. The `polygon.csv` file downloaded from the *Polygon Generator* dashboard is uploaded to the dashboard. After processing has been completed and the data generated, the data should be pushed to the database by clicking the `INSERT soildata` and `INSERT farmstats` buttons.

v Farmland Statistics

The final dashboard in the app is the *Farmland Statistics.* This dashboard aims to provide all the necessary statistics related to the pre-existing and newly generated farm polygons in the project database. A screenshot of the dashboard is shown in Fig. (5). The dashboard contains the following components-

- **Region selector:** This dropdown menu on the upper right is used to select the region of interest.
- **Variable selector:** This dropdown menu on the upper left is used to select variables that eventually appear on the polygon heatmap. The variables that are available are `ndvi_max`, `ndmi_max` and all the columns in `soildata`.
- **Polygon heatmap:** This is an interactive map that displays all the polygons currently available for the chosen region. The heatmap then provides a visualization of how the different variables of interest are distributed across the farms. The heatmap can then be used to identify farms with low soil quality and moisture.
- **Distribution of high moisture-level occurrences:** The distribution of farms spending time in high moisture levels is shown in this histogram. High moisture level days are determined by counting the number of days a polygon displays NDMI greater than 0.38.
- **Distribution of peak growing seasons:** This is a bar chart showing the distribution of peak growing seasons as function of months of the year. This essentially provides us with a window into farming practices in different regions. Since the data is further stacked by year, this plot can also reveal if planting practices vary over the years.
- **Distribution of annual planting cycles:** Similar to the previous one, this bar chart also reveals patterns in planting cycles. However, this plot shows how the number of planting cycles are distributed over the years, per region. The data is mostly consistent with two planting cycles per year. However, since determination of planting cycles can be difficult, due to noisy data, there are also cases which contain three or even four planting cycles.
- **Moisture-level breakdown:** This is a pie chart that provides a breakdown of the moisture content of the farms in the region.

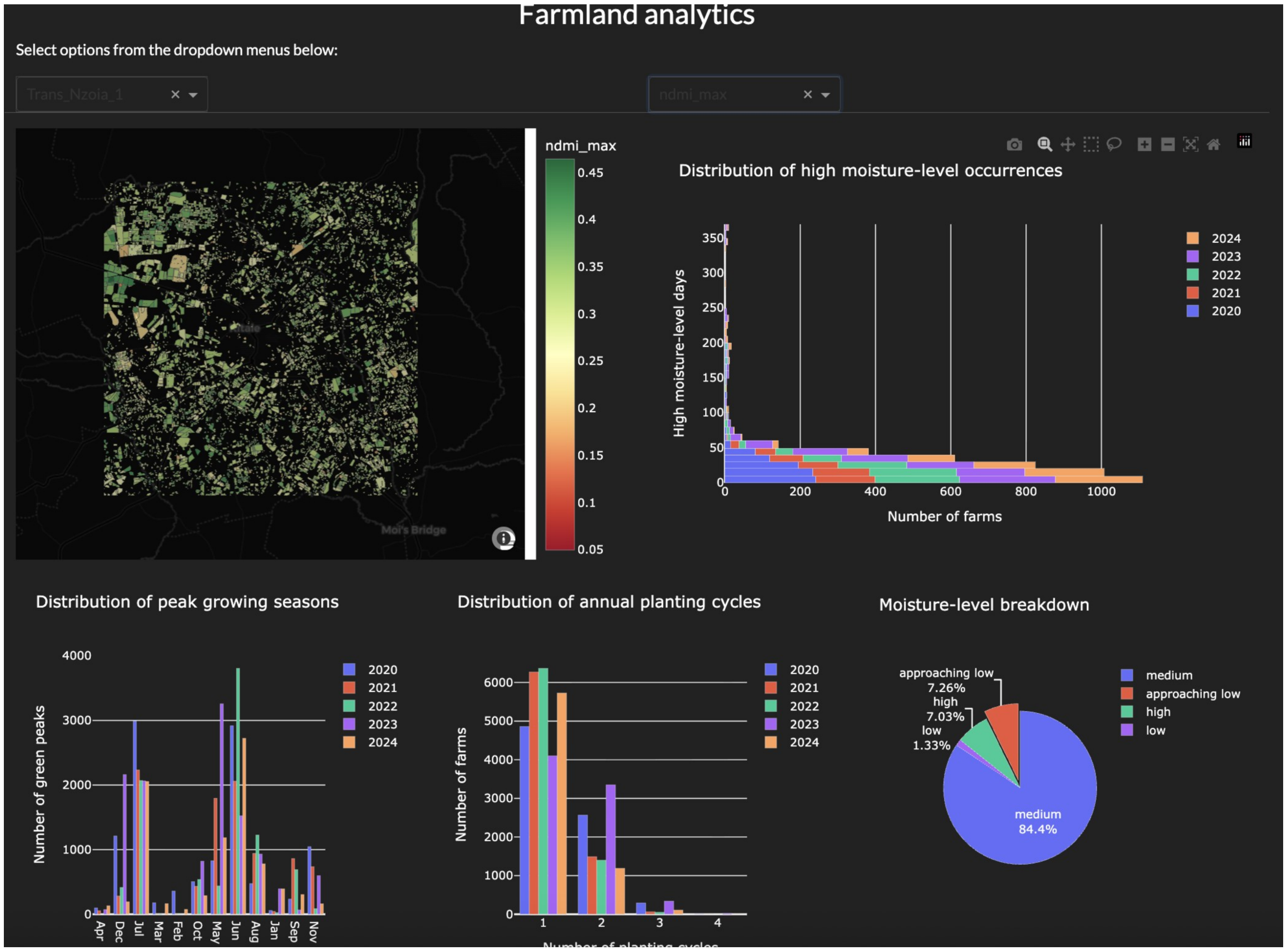


Figure 5: This image provides an overview of the 'Farmland Statistics' dashboard.

5 What to expect from future releases

Since `v1.0.0` is the first official release of the app, it lacks certain features that would make for a better user experience. Hence, we plan on including small, but significant, additions to that end. Even though most of the improvements slated for future release will eventually arise from feedback from the end-users, here are a few improvements that we foresee to be made available in the form of a minor update-

- **Latitude/Longitude search box:** Currently, the *Polygon Generator* dashboard does not possess a coordinate search box. This means that searching for locations becomes difficult. This means that users familiar with the geography of Kenya are able to find their desired location on the map. With the addition of a search box, finding locations will be much easier.
- **Additional distributor locations:** While we understand that being able to define one's own buffer is important, this feature is not part of the app's immediate future. However, we plan on expanding the number of available distributor regions to contain a list of priority regions

specified by Regen Organics. The newly added regions will only contain a bounding box, meaning that they will be devoid of pre-existing polygons (due to reasons previously discussed). With these new bounding boxes, authenticated users will be able to start adding new farm polygons and associated statistics to the project database.