A Crop Recommendation Tool for Organic Farmers

Jasmine Hsu and Dennis Shasha

Technical Report #975
Courant Institute of Mathematical Sciences
New York University
New York, New York 10003
`{jch550, shasha} at {nyu.edu, courant.nyu.edu}`

We describe the data sources and machine learning algorithms that go into the current version of www.whatcanifarm.com, a website to help prospective organic farmers determine what to grow given the climate characterized by their zip code.

Introduction

In developed countries, food is abundant, but is grown with too much pesticides and too much herbicide. Environmentally-minded citizens would like to grow foods with minimum chemical intervention. ”Green farming” eschews farming methods such as routine antibiotics, growth hormones, and laboratory genetic modification. As the US Department of Agriculture puts it:

> Organic agriculture is an ecological production management system that promotes and enhances biodiversity, biological cycles and soil biological activity. It is based on minimal use of off-farm inputs and on management practices that restore, maintain and enhance ecological harmony. (*8*)

As such, this project grew out of a need for developing an online resource for novice farmers that want to adhere to these practices.

Motivation

The motivation of this project is that our tool will be one of many resources online that new farmers can use to get a practical sense of what type of crops they can grow organically in their area. Below are some of the most useful resources we have found during our research:

1. Understanding the pure principles of organic agriculture, this resource provides ethical principles in health, fairness, ecology, and care. These principles are in encouragement of:

    (a) Maintaining health in soil, plants, and animals

    (b) Respect for all living things

    (c) Sustaining the natural biodiversity in the system

    (d) Management in a precautionary and responsible manner to protect the well-being of future generations in our environment

    Cultivating Change, the Organic Info Hub (*3*)

2. Novice farmers require much information in addition to what this application currently provides such as pest management, soils & compost, water management, and energy alternatives. For such information, the ATTRA is a program that is funded by the USDA Rural Business-Cooperative Service and provides information about the business aspect of sustainable economic communities. Thus, this program provides a database of many free publications that cover the topic of organic production, livestock, and etc. ATTRA, National Sustainable Agriculture Information Service (*2*)

3. Besides general information about organic farming principles and an overview of business practices, the following resource provides regional specific guides to How To Go Organic (*1*)

4. Based on certain crop predictions generated from our website, a farmer may be inclined to do further research, especially on these plants, crops, or animals. Below is a list of specific databases we recommend for such purposes:

   (a) Plants For a Future: Over 7,000 plants and their usages: edible, medicinal, and etc.

   (b) Web Soil Survey: provides an interactive way of exploring soil data from the National Cooperative Soil Survey

   (c) NewCROP: New Crop Resource Online Program - providing new and specialty crop profiles

   (d) Iowa State Entomology Index of Internet Resources: integrated pest management

   (e) There's an app for that! (Even in farming) An article providing links to some ways new technologies are used for analytics in all aspects of farming tasks. These apps are likely for profit, but still may be an incredible resource should the farmer feel the expense is worth the cost.

Data Sources

Here is a description of the data sources that currently underlie our website. We sought data having three qualities: 1. structure, 2. availability, and 3. coverage. When this project expands beyond just crop recommendations, these requirements will still hold.

Requirements

Structure means a data source that has a consistent format for its attributes (e.g., precipitation, soil temperature).

Availability means that the information was freely available for use. In practice, this meant finding sites having an easily accessible API for data queries.

Coverage means that the data should cover our universe, in this case United States zip codes.

Feature Data

We use the term feature data to mean the entire collection of environmental feature values for the classifier to train on. The National Oceanic and Atmospheric Administration (NOAA) provides users access to climate and historical weather data and information. (*9*) Through their API, a user can access data at particular locations or study climate change and variability through monthly climate reports.

Our data was aggregated from NOAA's collection of data sources by accessing their Climate Data Online (CDO) program through their developer API. Scripts were written in R to generate the feature data for US locations based on their county code (FIPS). Query responses were in JSON and parsed within the R script to be written to a local directory, organized by station ID and FIPS code.

These data points were organized into monthly statistical summaries consisting of means, extremes, various thresholds, and etc. Some high level features include:

1. Temperature (monthly means, extremes)
2. Precipitation (monthly totals, extremes, various thresholds)
3. Snowfall (max depth)
4. Degree days

5. Soil and soil temperature

Each feature had a computational name and was collected for each month of one year.

1. CLDD - cooling degree days
2. DP01 - number of days in month with greater than or equal to 0.1 inch of precipitation
3. DT32 - number of days in month with min. temperature less than or equal to 32.0 F
4. MNYzop - monthly mean min soil temperature (y is soil cover, z is soil depth)
   (a) Soil cover code (y) examples: 1-grass, 2-fallow, 3-bare ground...
   (b) Soil depth code (z) examples: 1-"2in.", 2-"4in.", 3-"8in."

For further details, please see: `https://www.ncdc.noaa.gov/cdo-web/webservices`

All these feature names are well documented on the NOAA API developers page. The station data that was aggregated best met the requirements for our data source and was chosen for our model training. As a note to potential future collaborators, the API does have a rate limit per day, and thus to plan accordingly. (We, in particular, scraped over 6000 weather station data across several days and weeks)

Organic Farms & Crops Data

There are many farm websites pertaining to particular cities and states. However, many local websites can not be used to verify certification of meeting the "organic" farms practices set forth by USDA. Also, many of these sites are very outdated and do not have consistent information across their pages. In following our practices of locating data sources that satisfy the criteria of consistent structure, availability, and coverage, we found that the USDA organizations (which provide the organic certification) had a database of farms that existed across the US.

Each certified organic operation provided the list of products that they produced. These are our target labels, enables us to use ”supervised learning”. Based on the products at each organic farm, we are able feed our model, for each crop, the corresponding feature vector (i.e. environmental data).

Implementation: Data

Preprocessing

The weather station data collected were organized by ID and FIPS code. Our preprocessing scripts were written in python and mainly a collection of merging, averaging, imputing, and cleaning scripts. We will try to explain these steps as at high a level as possible. The scripts themselves are further documented inline.

1. Concatenate all the weather stations together into one CSV.
    (a) This meant that each row represented the values for a particular station and each column represented the values for a specific computational feature for all stations.
2. Map the FIPS code to zip code.
    (a) The FIPS code used by NOAA has to be mapped to the proper zip code since the list of organic farms do not have an associated FIPS code as an identifying location parameter.
3. Map the environment values to each farm based on zip code.
    (a) Each data point represents a farm, and based on their zip code, we have the corresponding feature vector. This feature vector is consistent across all organic farms.
4. Map the environmental values to each crop based on each farm.

   (a) Each data point represents a crop, and based on the farm, we have the corresponding feature vector. If a crop is grown on multiple farms, it has multiple feature vectors. Duplicates are dropped.

5. Use natural language preprocessing techniques to clean crop labels.

   (a) ”Apple” should be the same as ”Apples”. Otherwise, the model will learn that ”apple” is a different class than ”apples”, which will skew our prediction.

6. Format the data for classifier input.

   (a) This is dependent on what library of machine learning tools that one chooses, which require certain formatting of the input for the model. Generally, an X input is required, followed by its y target label for supervised learning.

Imputer Methods

While this is a form of preprocessing, it was a large experimental exploration in preparing the most ideal values for model learning, and deserves its own subsection.

A major challenge in machine learning is dealing with missing values. Though our data source was consistent, there was no guarantee that all values for every month, for ever computation (feature) existed.

Station failure or downtime throughout the year is a possibility, and to address that, methods had to be developed to fill these missing values. Through discussions with machine learning researchers, general imputer methods were not very effective and should be used as a last resort. For example:

1. Global imputation transformation (*4*)

   (a) Replace missing values using the mean of all global values along an axis

(b) Replace missing values using the median of all global values along an axis

(c) Replace missing values using the most frequent of all global values along an axis

While simple, for our dataset, these techniques would heavily skew values for weather stations if their features were simply averages of all the other stations.

There are two custom imputer methods that can be easily implemented for much better results.

1. Time-locality

2. Space-locality

Time and space are aspects of our station data. They can be identified by time (month, year) and space (zip code).

A time-locality based imputer would choose to produce a regression model that fit the features values per month along the x-axis. Thus, months that were simply missing those values can be imputed by the regression line.

A space-locality based imputer would choose to replace a missing value for a zip code with the average of all of nearby zip codes in the area. This idea is similar to a global average, but restricted in space (only nearby zip codes).

Our model is currently trained on values that used a space-locality based imputer as a first pass, and then a global imputer as a second pass. Feature computations that still did not have any values for all stations were simply dropped.

We will briefly illustrate the implementation of our space-locality imputer below:

1. For each zip code, get all zip codes within the nearest x miles.

   (a) A script was developed to query the Geonames API for all nearby zip codes.

2. For each zip code, average the values for all its neighboring zip codes and replace the value (only if it is missing the value)
    (a) The features values were analyzed post-imputation to find the best radius. Our model is trained on values that were imputed with zip codes within 30 miles. Thus, if a zip code was missing several values, its crop recommendation will be very similar to the average recommendations of its area.

A time-locality implementation can be implemented as discussed above, and we further elaborate on this in a later section in which we discuss model and application improvements.

Implementation: Model

The model used in our application is a one vs. all classifier. To explain briefly: our problem is of multi-class classification. Each training point can belong to one of N different classes. This means that our model is a function that, given a new data point, can correctly predict the class(es) the data point belongs to.

This leads us into the basic concept of a one vs. all classification. Essentially, it is N different binary classifiers, where N represents the number of target values. For each target value, we have a classifier f_i that has either positive or negative examples for all points x_i. Positive examples are points that are in class i and negative examples vice versa. (5)

$$F(x) = \arg\max_i f_i(x) \tag{1}$$

The base classifier of a one vs. all model is interchangeable, thus our model was tested with base classifiers using linear SVC, decision trees, and random forests. The current model hosted is trained using a random forest base classifier.

The strategy for the one vs. all classifier in generating a ranking is through the ability for multi-label learning. Each classifier is used to predict multiple labels by fitting a 2D matrix in which cell $[i, j]$ is 1 if sample x_i has label j or 0 otherwise.

We will briefly illustrate the implementation of one vs. all classifier below:

1. Create a multi-label binarizer to transform all sets of labels for each sample into a indicator matrix.
    (a) An example is a dataset with three samples [(1,2),(3),(2,3)] which transforms into

    $$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

    (b) This can similarly be applied to label names, which underscores again the importance of the NLP step of cleaning target labels.
2. Train each classifier on all samples (train the one vs. all classifier).
    (a) This is a generalized step, but a lot of work and experimentation falls into this step. The dataset should be split into training and test sets and cross-validation can be used to tune models on various parameters of the base classifier.
3. Return a probability estimate of the sample for each class in the model.
    (a) Sort the list in a descending fashion essentially giving a ranking. The first class in the list is the class that is predicted to have the highest probability given the test sample.

In our application, the test sample X is the feature vector which contains environmental data. The y target labels used for supervised learning are the crops; these target labels, after preprocessing and cleaning, undergo a multi-label binarizer transformation and reduce to the matrix seen above. These inputs train the model and can be split into train/test sets for validation. A query from the user results in a new x_i sample with a feature vector unseen before by the model and thus given to the model for prediction.

Implementation: Web Application

A brief summary of the web application stack as follows:

The web application is built in python using the Flask web framework, deployed on Heroku. Flask is a lightweight micro-framework based on Jinja2 templating. (*6*). Heroku is a cloud application platform for developers to build and run applications entirely in the cloud. (*7*) The back-end is on Postgres database with static files hosted on Amazon S3. The front-end uses Bootstrap on top of JQuery and Javascript to enable a responsive web design.

Testing

User testing was done on the web application through an audience with a variety of backgrounds. Some basic UI responses to the questions asked have been incorporated into the design. Some example questions were:

1. Was it intuitive to use?
2. Did you understand the purpose of the application from the initial page?
3. Did you think there was too much information given at once?
4. Do you think the information was organized well? / displayed well to the user?
5. Did you think anything was out of place?
6. Did at any point you feel confused using the application?

7. What were your initial thoughts after submitting your query?

8. Do you have any questions?

Feedback on UI greatly improved many aspects of the web application. Such examples are:

1. ”Initial page loading and search query seemed a little longer than normal.”

2. ”Next step: add if it’s legal to grow those crops. And then, if an agriculture company sues you for somehow getting your crops contaminated with proprietary GMOs, you have precautionary measures.”

3. ”Maybe you could add links for each of the crop suggestions? To wikipedia or somewhere, ... I don’t know what ”alfalfa” is.”

4. ”Is the map just to make sure that the person searching knows that the website got the right area? Or would other organic farms show up on the map if there were any nearby?”

5. ”A historical temperature+rainfall graph would be fancy. Since you’re giving current weather conditions too.”

6. ”Does the picture next to Location change based on if you live in a city or not? ”

7. ”Are you online learning? Or is your model static?”

8. ”Where are you hosting this? What if your site gets popular and you need to scale to 100000 users?”

9. ”..possibly having the page after landing have a smaller search area, seems out of place when you need to scroll to see a majority of the information, maybe re-positioning the search tool to the bottom would work as well on the second page. ”

10. ”Felt that current conditions didn’t add anything to the site, possibly a monthly average would be better? ”

11. ”My initial reaction was how clean the layout was, no clutter and down to business.”

12. ”As I’ve mentioned, the site is pretty straightforward and intuitive. One thing that I didn’t like was having to read sentences when everything else seemed like lists in the weather and geography descriptive. Instead of using sentences maybe try and portray the weather and topographic info in ’x = y’ format?”

13. ”Like that it’s simple and intuitive. Huge points there. Extra points if you give it a bit more thought on the kinds of feelings you want to appeal to the user; you are promoting ”green living” but the black and white makes me wanting a bit more.”

14. ”If you were to make your landing page prettier (add farm themes and old MacDonald, etc.), I would also recommend adding a navigation bar.”

15. ”If I were a farmer, wouldn’t I want to know more specifics about the weather in my area? Your site provides the weather today, and I am assuming it bases crop choices from the highest and lowest data points in aggregated weather data. However, there is no information on whether the region has a consistent environment or changes every season.”

16. ”Maybe for version 2.0 you could consider breaking the information down by season.”

17. ”You probably need some padding for the <div class=”container”> because once I resize the screen the words stick to the grey box.”

18. ”The moment the landing page opens I’d like the cursor to be in the zipcode entry box.”

19. ”Future app could ask for browser location to autofill.”

20. ”Future app could preload a zipcode.”

21. ”May be good to limit zip inputs to numeric entries. And perhaps a char limit of 5 or 10.”

All beta testers are listed in a final acknowledgment in the citations section.

Future Work

We hope that this project will be able to continuing expanding. Some initial thoughts on general areas of improvements include:

1. Data Collection Stage
   (a) Explore and incorporate additional datasets at NOAA.
   (b) Incorporate economic indexing for crops and allow the the user to run a query with ”profitable crops” in mind.
2. Preprocessing Stage
   (a) NLP techniques to improve cleaning crop labels: i.e. ”brown rice” is: ”rice, brown”,”rice - brown”, ”rice (brown)”.
   (b) NLP techniques in creating a crop ontology will also simplify the data collection stage as crop profiles can be retrieved.
   (c) Time-based locality imputer method.
3. Model Stage
   (a) This stage is open-ended in any machine learning problem. Simply to continue experimenting with various techniques.
4. Web Application Stage

   (a) Improve framework to allow for scalability, i.e. larger model sizes, larger databases.

   (b) Increase application speed.

5. Beta Tester responses and suggestions

   (a) Time-series temperature or weather graph.

   (b) Additional crop information linked from rankings.

   (c) Online learning as base model.

Code

Our code is hosted on github.com at:

```
https://github.com/hellojas/whatcanifarm
```

The git repository can be cloned via the command:

```
git clone https://github.com/hellojas/whatcanifarm.git
```

References and Notes

1. Organic Trade Association. How to go organic, 2015.

2. NCAT: ATTRA. Attra — national sustainable agriculture information service, 2015.

3. IFOAM Organics International. Cultivating change, the organic info hub.

4. Scikit learn Developers. Scikit-learn. machine learning in python, 2010.

5. Ryan Rifkin. Multiclass classification, February.

6. Armin Ronacher. Flask. web development, one drop at a time., 2014.

7. Salesforce.com. Heroku: A platform as a service (paas) that enables developers to build and run applications entirely in the cloud., 2015.

8. Alternative Farming Systems Information Center United States Department of Agriculture, National Agriculture Library. Organic product/organic food : Information access tools, May 2015.

9. NOAA United States Department of Commerce. National centers for environmental information - national oceanic and atmospheric administration, 2015.

10. Final acknowledges include:

    1. Roy Lowrance, Ph.D., Managing Director, NYU Center for Data Science.

    2. All web application QA Engineers:

        (a) David Johnson, CEO, Narlycode, Inc.

        (b) Marie Han, QA Engineer, Booz Allen Hamilton, Inc.

        (c) Wonju Chang, Multi-disciplined Language Analyst, US Department of Defense.

    3. All web application Beta Users:

        (a) Adarsh Jois, MS Computer Science, New York University

        (b) Kshitiz Sethia, MS Computer Science, New York University

        (c) Mihir Jadhav, MS Computer Science, New York University

        (d) Taesung Lee, Alumni Law, Georgetown University

        (e) Jenn Sun, Senior Consultant, Acumen Solutions, Inc.

        (f) Tiffany Yim, Senior Consultant, Booz Allen Hamilton, Inc.

        (g) Ann Tran, Director of Operations, ProcureLinx, LLC.

        (h) James Silverman, Business Manager, Sound Hearing Centers, LLC.

(i) Marshall Masterson, Underwriting Analyst, Freddie Mac.

(j) Joseph Hsu, International Business, NVCC.