Machine Learning Engineer Nanodegree

Capstone Project

Larion Babych
24 June, 2017

I. Definition

Project Overview

There are lots of issues with fast and accurate monitoring in the agriculture industry. Satellite images can help solve this problem. I am working in the company that helps to monitor agri fields. But such images can have obstacles, clouds, and snow, they could be very annoying. We can't change the weather but we could improve the effectiveness of users if we label such bad images.

In this project, I created standalone Python app that could classify images of fields. I used Decision Tree Classifier which was trained on data which you can find by this URL.

Problem Statement

I want to create image processing application that can determine is image cloudy, covered by snow or just clear; these tasks involved the following:

1. Prepare data;
2. Create model which could represent such terms as "clear", "cloudy" and "snow covered" field image;
3. Train classifier which could predict the class of image;
4. Test prediction of such classifier;

Metrics

F1-measure is the common scoring function for data classification problems. It considers both the precision and the recall of the test to compute the score.

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

Where

$$precision = true\ positives / (true\ positives + false\ positives)$$

and

$$recall = true\ positives / (true\ positives + false\ negatives)$$

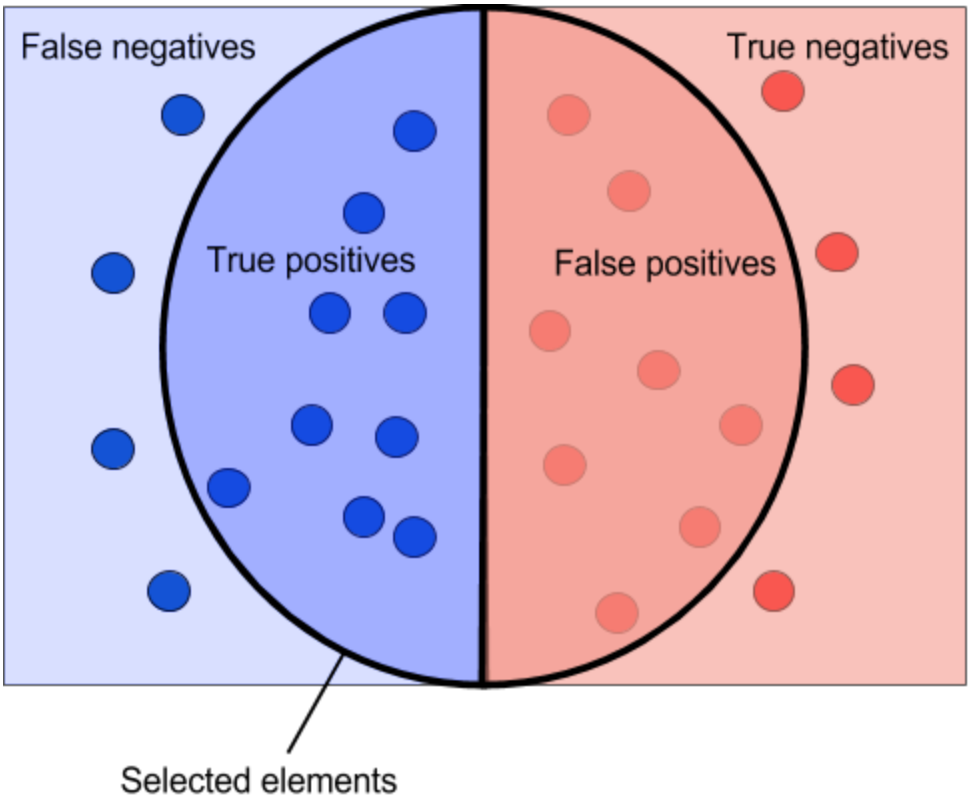


Image above represents abstract classifier. In this sample we have 13 blue circles and 12 red circles, 9 true positives, 4 false negatives, 8 false positives and 4 true negatives, recall = 9 / (9 + 4) = 0.69 , precision = 9 / (9 + 8) = 0.52, and finally F1 = 2 * 0.69 * 0.52 / (0.69 + 0.52) = 0.59.

The classificator is trained with average F1-measure, which uses all labels for measurement. I will mark this measure as F1(0, 1, 2, 4, 5).

But the most important for users of such system would be the fact that bad images, which are labeled by “5”, would not contain good ones, and good images {0, 1, 2, 4} don’t contain bad ones. I mapped {0, 1, 2, 4} as “good” and 5 has “bad” and added F1(good, bad) measure to the benchmark.

Another important feature of my classifier is the ability to recognize snow in the fields and don’t mark such images as bad ones. I added F1(good, snow, bad) measure to the benchmark also.

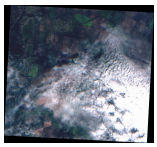
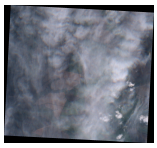
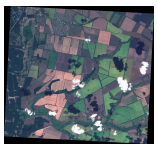
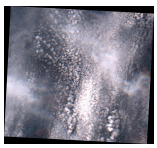
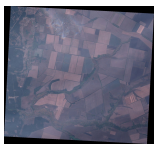
II. Analysis

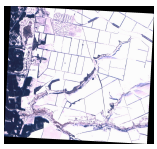
Data Exploration

I used images from our system as a dataset. They are not well annotated and don’t have standard sizes. I divided images of farms into subgroups with the same sizes because this step will affect data preprocessing and model building step, which would be described later. I added desc.txt file in each subgroup of training dataset with such labels:

- 0 - clear or almost clear image.
- 1 - the image has a lot clouds.
- 2 - image with snow coverage.
- 4 - the image has snow and cloud coverage.
- 5 - very bad image, without any piece of fields. Such images should be shrunk on a server, because they don’t have any useful information for agronomists.

The final dataset sample is presented in the table below. I will discuss parameters later in the Data preprocessing.

name	1	2	3	4	5	6	7	cor	date	label	image
g1/20150806.tif	0.014	0.140	0.547	0.248	0.038	0.002	0.007	0.44	0.66	0	
g1/20150813.tif	0.023	0.364	0.1883	0.121	0.106	0.069	0.125	0.08	0.66	1	
g1/20150816.tif	0.013	0.058	0.3880	0.303	0.209	0.027	0.021	0.0089	0.66	1	
g1/20150823.tif	0.014	0.199	0.4375	0.277	0.046	0.005	0.019	0.49	0.66	0	
g1/20150826.tif	0.014	0.184	0.4642	0.292	0.042	0.001	0.0	0.53	0.66	0	
g1/20150902.tif	0.015	0.038	0.6512	0.266	0.027	0.0	0.0	0.31	0.75	0	
g1/20150905.tif	0.013	0.061	0.2972	0.266	0.209	0.110	0.041	0.0	0.75	5	
g1/20151005.tif	0.014	0.014	0.5204	0.442	0.007	0.0	0.0	0.06	0.83	0	

g1/20160103.tif	0.01	0.093	,0.0410	0.046	0.049	0.06184	0.6963	0.2448	0.083	2	

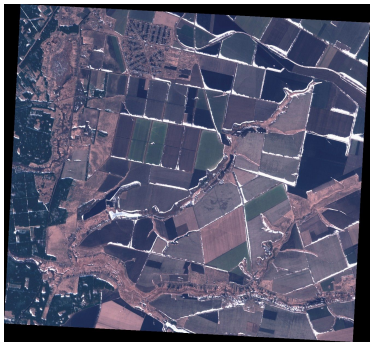
Dataset has such stats:

- Total number of images: 427
- Number of features: 10
- Number of 0 labels: 142
- Number of 1 labels: 63
- Number of 2 labels: 34
- Number of 4 labels: 14
- Number of 5 labels: 174
- Total number of good images: 253

Each training directory has images of similar size. Each image has RGB colorspace.

- g1 - 1337 × 1235
- g2 - 1707 × 1564
- g3 - 1222 × 1453
- g4 - 965 × 1130
- g5 - 1505 × 1512
- g6 - 1933 × 1944

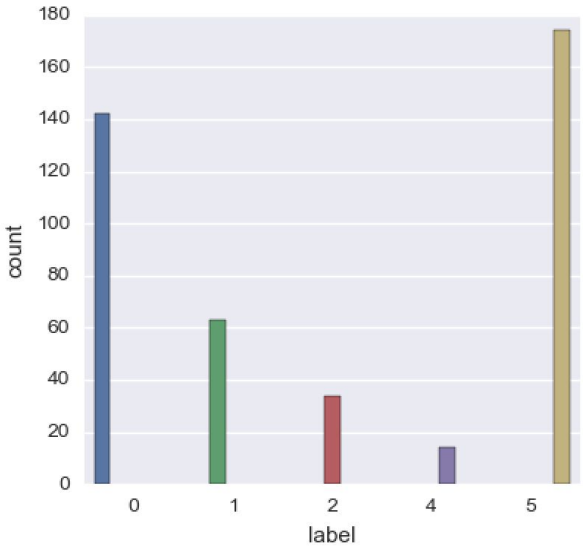
When I started the project my approach was different. I marked absolutely clean images with 0, with a small cloud coverage as 1, and with a heavy coverage as 2. But there were abnormal images.

First	Second	Ethalon
		

The first one has a little snow coverage, and a second one has a large amount of bright yellow color. Both of these samples could cause misclassification. Bright yellow color after gray scale operation could be almost white. Whence I decided to reduce number of green labels, and added additional date parameter into the model, these actions reduced amount of misclassifications.

Exploratory Visualization

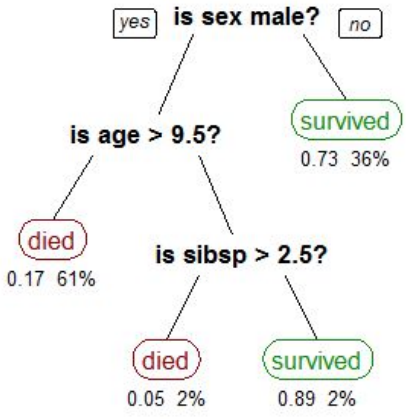
I did data exploration in exploration notebook, which you can find in the repository. I've used g1, g2, g3, g4, g5, g6 data frames. The process of building datasets will be discussed in Data pre processing project later.



We can see a lack of samples with 2(snow) and 4(snow and cloud) images. This may cause overfitting problem during the training process. Due to lack of time I did not find and label enough images from each class, so there probably will be errors of classification of images with second and fourth labels.

Algorithms and Techniques

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown in the diagram right.



Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable.

Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. [1]

Positives:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation.
- Able to handle multi-output problems.

Negatives:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree [2]

The choose of this algorithm was justified by the best F1 score among others default. You could find the comparison in Exploration notebook. Results are listed below.

Name	F1 if training set	F1 of testing set
Decision Tree Classifier	1.0000	0.9170
Random Forest Classifier	0.9866	0.8798
SVC	0.8097	0.8357
Extra Trees Classifier	1.0000	0.8855
AdaBoost Classifier	0.7467	0.7755

[1] "Decision tree learning - Wikipedia." https://en.wikipedia.org/wiki/Decision_tree_learning. Accessed 13 Aug. 2017.

[2] "Decision Trees - Scikit-learn." http://scikit-learn.org/stable/modules/tree.html. Accessed 13 Aug. 2017.

Benchmark

As was mentioned in Metrics session I used three F1 scores for the classifier.

F1(0, 1, 2, 4, 5) will show F1 score of classifier for (good, good & cloudy, snow, snow & cloudy, bad) images. My goal is to achieve 0.8.

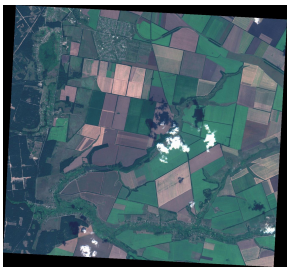
F1(good, bad) will show F1 for ({good, good & cloudy, snow, snow & cloudy}, bad) images. It is very important to train the classifier that would not miss with good and bad classes of images, our clients could understand imperfect labels of good ones, but if we missed and mark a good image as bad one it would be very bad situation. F1(good, bad) must be above 0.9.

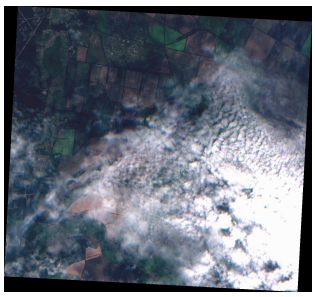
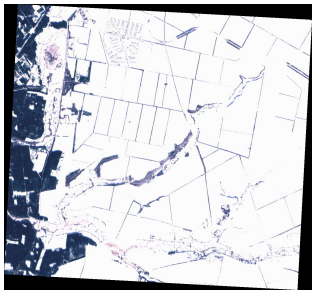
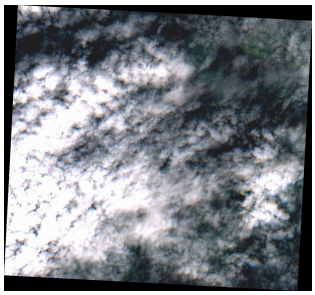
I spent much time for building model that depict snow on the image, whence I would be great if classifier recognizes such images and would have a good metric. F1(good, snow, bad).

III. Methodology

Data Preprocessing

The main question of this project is “How to build a model that could represent such features as snow coverage, cloud coverage and clearance of fields?”. When I started to work on this project the first idea was just to calculate histograms of the colour distribution of each image (table below).

	Normalized histogram of gray scaled image color distribution							
Image	1	2	3	4	5	6	7	Label
	0.0149	0.1405	0.5477	0.2482	0.0385	0.0027	0.0071	0

	0.0238	0.3649	0.1883	0.1215	0.1063	0.0694	0.1254	1
	0.0097	0.0793	0.02812	0.0255	0.0244	0.0263	0.8064	2
	0.0701	0.1723	0.2017	0.14870	0.1184	0.0976	0.1909	5

But as you can see snow covered and clouded images have similar distributions of color and it would be impossible to train a classifier with such model. I added another parameter - cor. This parameter represents a correlation of edges between image and mask which was created for a particular group of images.

Each folder contains images of one particular farm with same sizes. One farm can have several group folders. For example folders, g1 and g2 have data from a single farm but contains images with different size.

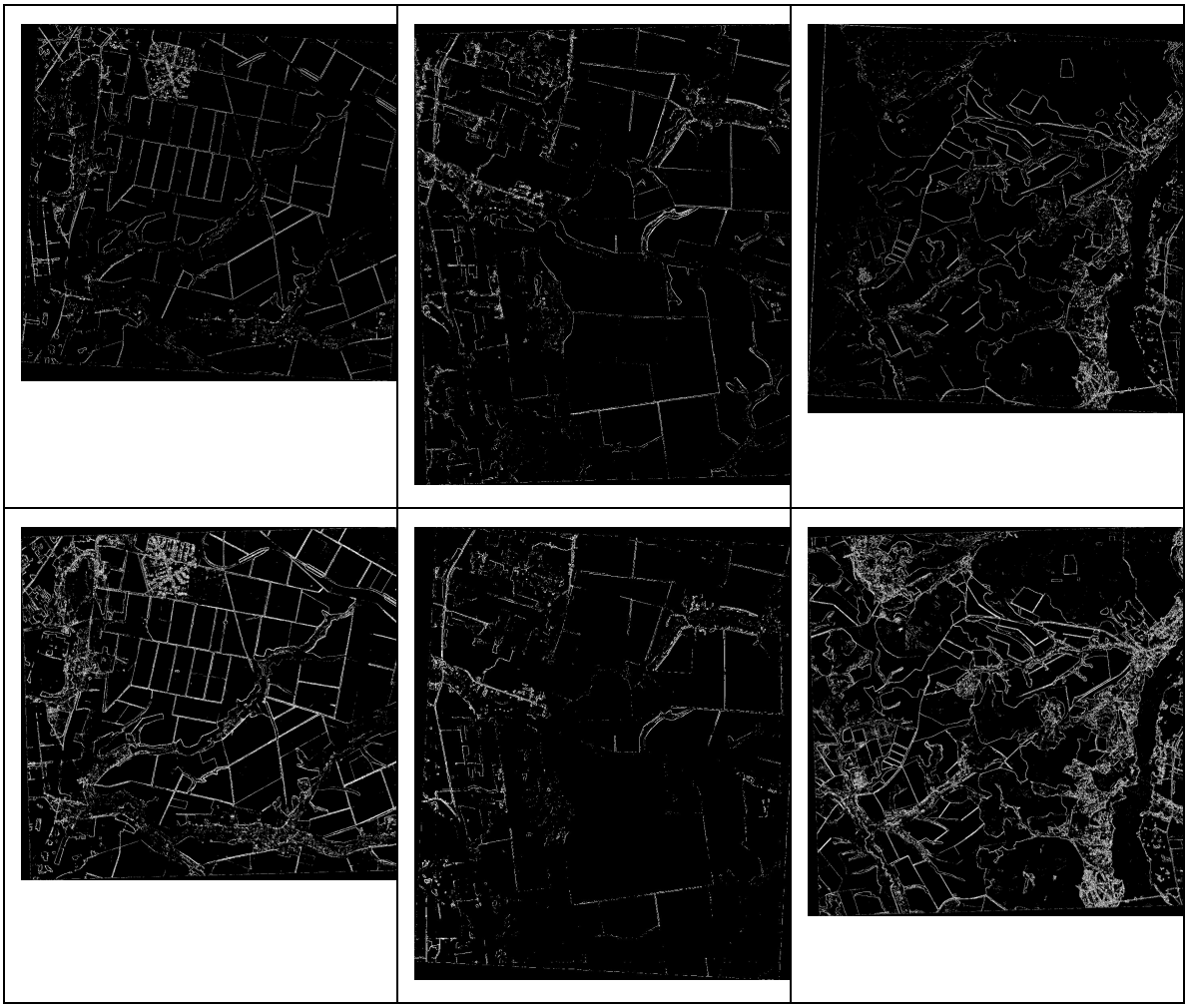
After farm images splitted to same size folders I calculated cor parameter. Follow next steps:

- Find edges of each image in folder with help of Canny Edge Algorithm[3].
- Sum and normalize edges.

[3] "Canny edge detector - Wikipedia." https://en.wikipedia.org/wiki/Canny_edge_detector. Accessed 13 Aug. 2017.

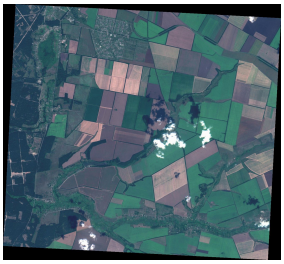
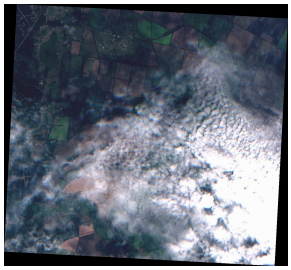
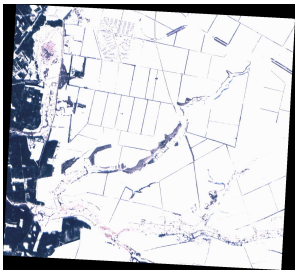
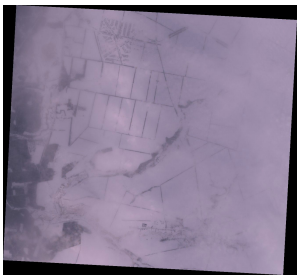
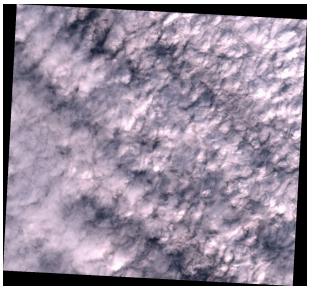
- If value from sum-edge matrix is between 0.1 and 0.3 mark it as 255 if not - 0.
- After 1 - 3 steps in each folder you can find mask.tif file, you can find alg implementation in model/mask.py file.

  For each folder I got such masks.



These masks depict edges of fields and if we calculate correlation (model/cov.py) with a single image we could get a parameter that represents image “clearance”.

Image	Cor	Label

	0.445766903067	0
	0.0863889079906	1
	0.22695653604	2
	0.0216221685329	4
	0.0101419492232	5

I also added date parameter due to a fact that already yielded fields will have the same color palette as snow fields if you gray scale them. It's obvious that snow

in summer would be very rare weather condition, whence such parameter could help to train a better classifier.

Implementation

The Implementation can be split into stages:

1. Preprocess images and build a representative model. (This part is most creative one)
    - Split images of with the same sizes and farm into directories.
    - Calculate histograms of gray scaled images.
    - Calculate Canny edge mask of groups.
    - Calculate correlation of images in the same group with created mask.
    - Combine histogram, correlation, and date the into single dataset, which represents each image via the model.
2. Split datasets into training and testing sets
3. Train Decision Tree classifier on training set
4. Check F1(0, 1, 2, 4, 5), F1(good, bad), F1(good, snow, bad) scoring results of predicted labels.
5. Predict labels for test folder(t1, t2, t3, ….)

Created application was designed for simple illustration purpose, it demonstrates main stages of learning and can use trained classifier to predict labels of groups which were not included in the train set.

There are simple button list:

- <Add model> will add pre built model list for further training.
- <Learn> will launch train and test methods, and print results to the log.
- <Predict> can be used for testing classifier on datasets, which were not included in the data set. Additional images could be found here. Results will be printed in the choosable log frame.
- Choose labeled image and click <Image> button for manual validation.

Refinement

Due to a fact that case is multi labeled it is impossible to use meta-parameter optimization such as Grid Search. So I tuned parameters of DTC by myself. I have added table which shows results.

	Default	Custom
F1(0, 1, 2, 4, 5)	0.9256	0.9391
F1(good, bad)	0.9623	0.9815
F1(good, snow, bad)	0.9521	0.9843

IV. Results

Model Evaluation and Validation

I used final meta-parameters because they performed the best among tried ones.

The final DTC has such parameters:

- min_samples_split=2. The minimum number of samples required to split an internal node.
- min_samples_leaf=1. The minimum number of samples required to be at a leaf node.
- max_depth=16. The maximum depth of the tree.
- max_leaf_nodes=10. Grow a tree with `max_leaf_nodes` in best-first fashion.

These parameters get robust results, especially in prediction bad and snow images, F1(good, bad) = 0.98, F1(good, bad, snow)=0.98. The final model aligns good with my expectations, it has high accuracy and predicted results could be trusted.

Justification

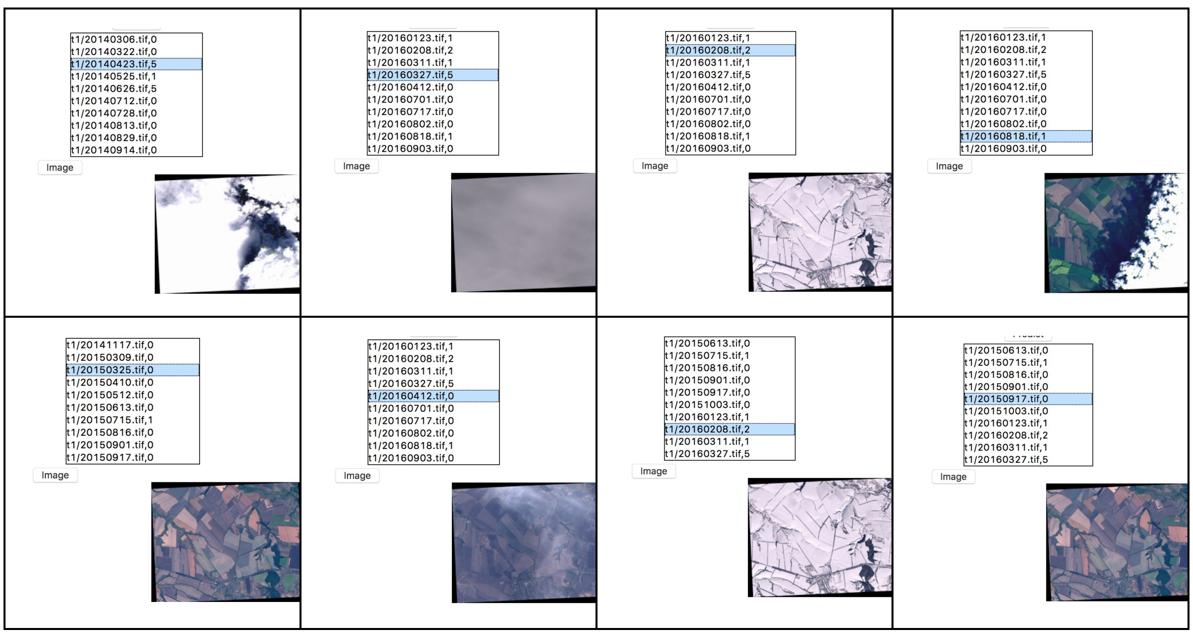
DTC and model work great. We can use it for image labeling and improve the efficiency of our clients. As was planned it could recognize bad images and snow very well even better than was predicted in the benchmark.

I have added additional images to t1, t2, t3 directories, which you could find by this URL. These images were not a part of training data set and were prepared for demonstration only. You could find samples below in Visualization section.

V. Conclusion

Free-Form Visualization

You can use t1, t2, t3 folders for testing. I added some examples.



Reflection

The process used for this project can be summarized using the following steps.

1. The problem is discussed.
2. Datasets of images were loaded and preprocessed.

3. Model or feature building process is established.
4. A benchmark was created for the classifier.
5. DTC is trained and tested and scored.
6. Simple standalone app is developed for presentation purpose.
7. Data for testing is prepared and labeled.

The most difficult part was creating a feature that would show fields contours. I watched Computer Vision course on Udacity for this purpose. After that, I began working on the preprocessing and feature building parts of my project.

Improvement

Created application is a good starting point for our global system improvement. It has a good F1 score of good vs bad classification and it can recognize snow images.

Due to lack of balanced data DTC misses with 0 and 1 labels.

During implementation, I recognized that It would be useful if our system shows a more vast set of marks for clouds, not just 0 and 1 but from 0 to 5 for good images, 6, 7 for snow covered and 10 for really bad images.

All these causes further improvement of the dataset with a more balanced distribution of images.

During data preprocessing cor parameter is being calculated, but if a group folder has only bad images this parameter will be high and classifier recognize such images as good ones or with snow. I must add another classifier which recognizes only good images and then run already trained one.