Leaf Classification Playground Competition: Winning Kernels

The Leaf Classification playground competition ran on Kaggle from August 2016 to February 2017. Over 1,500 Kagglers competed to accurately identify 99 different species of plants based on a dataset of leaf images and pre-extracted features. Because our playground competitions are designed using publicly available datasets, the real winners in this competition were the authors of impressive kernels.

In these sets of mini-interviews, you'll learn:

- why you shouldn't always jump straight to XGBoost;
- what makes feature engineering in Kernels more exciting than any MOOC; and
- how to interpret visualizations of PCA and k-means, two of the most common unsupervised learning algorithms

Read on or click the links below to jump to a section.

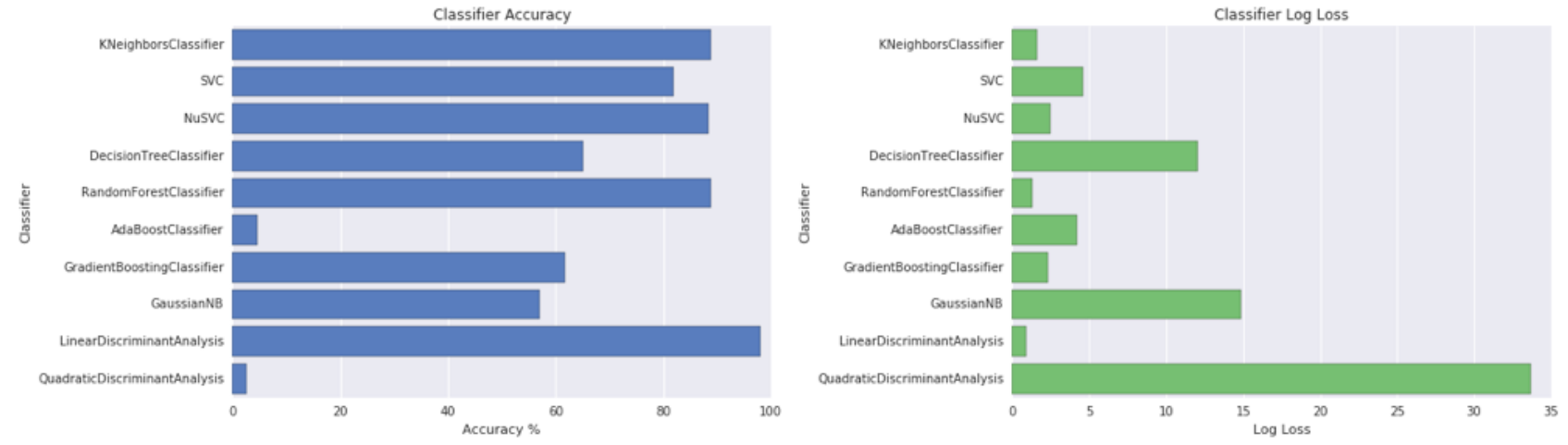
Data Exploration

10 Classifier Showdown

Created by: Jeff Delaney
Language: Python

What motivated you to create it?

Because the Leaf Classification dataset is small, I wanted a script that could run a bunch of different classifiers in a single pass. My goal was to provide a basic starting point for the competition and showcase the many classification algorithms in Scikit-Learn. It’s tempting to go straight to XGBoost for a competition like this, so it was important to point out that simple linear algorithms can be powerful as well.



Comparing classifier accuracy and logloss in the kernel [10 Classifier Showdown in Scikit-Learn].

What did you learn from your analysis?

I learned that Linear Discriminant Analysis was an effective algorithm for this problem, which is not something I would normally expect or even be looking for. However, it makes sense given that the leaves all follow distinct geometric patterns.

[3 Basic Classifiers and Feature Correlation]

Created by: [Federico C]
Language: Python

What motivated you to create it?

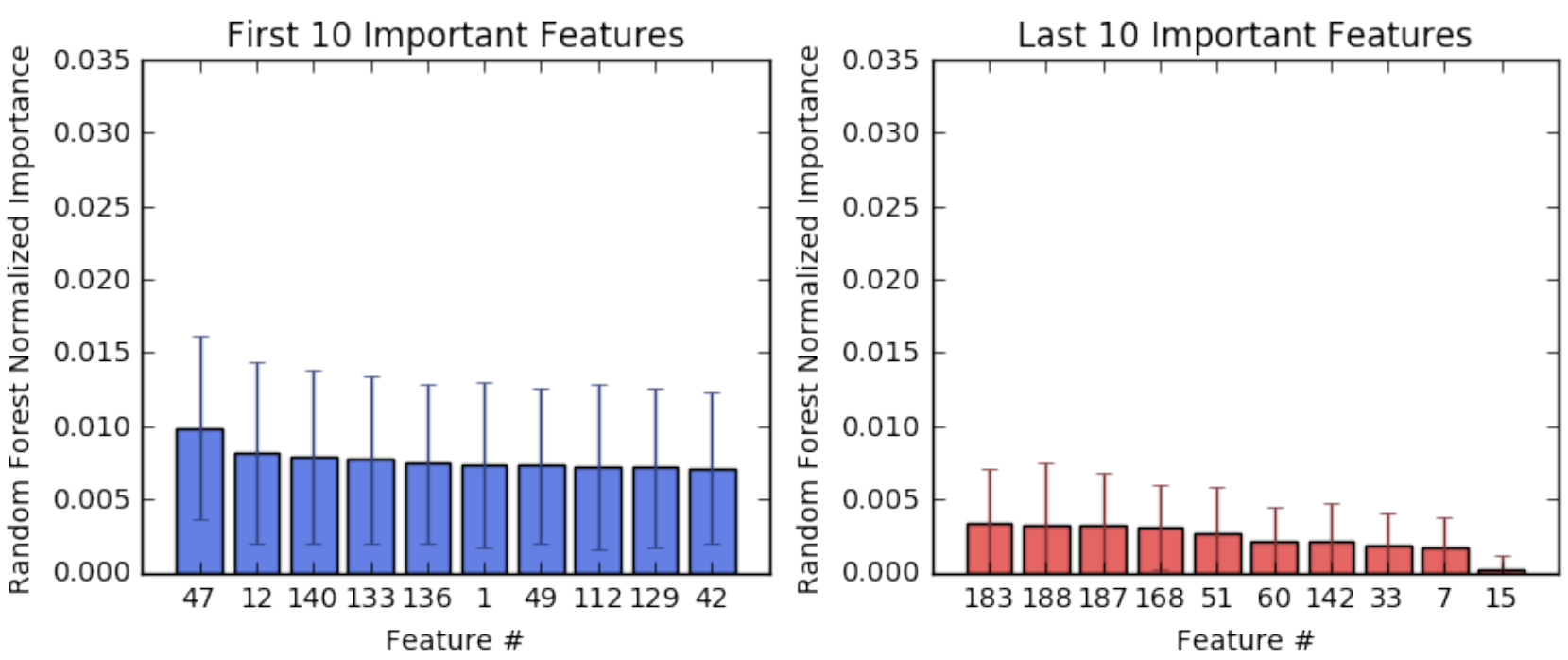
The Leaf Classification competition was my first attempt at a Kaggle challenge. As a beginner in the field, I started looking around for tips and suggestions among the kernels from the Kaggle community and decided to put everything I learned together for future reference.

As a first step, I thus chose to write my own notebook, where I tested a number of methods on the given dataset and identified their strengths and weaknesses. This helped me understand the logic behind each step, get experience on how different classifiers work and clarified why each algorithm may be more or less suitable for different kind of datasets.

Hopefully, the notebook also helped other participants who, like me, were looking for a good starting point.

What did you learn from your analysis?

In this analysis, I focused on the three main classifiers: Naive Bayes, Random Forest and Logistic Regression. Each one of them has specific pros and cons depending on the structure of the dataset at hand and should be carefully considered.



How to set up basic classifiers in the kernel [3 Basic Classifiers and Feature Correlation].

A necessary first step is to dive into the data, explore and prepare them, removing superfluous or redundant information and reconstructing (where possible) missing data. The Leaf Classification is a good dataset from this point of view, it offers a wide range of features, some of which are correlated and can be reshaped in a number of ways. This stage can strongly affect the final score of your algorithm, as exemplified clearly in the case of Naive Bayes, where making sure that the assumption of uncorrelated features is respected makes all the difference in the final outcome. With Random Forest and Logistic Regression, features reduction on this dataset has a negligible impact.

Can you tell us about your approach in the competition?

I approached the competition mainly with the intention to learn and understand the fundamental concepts of classification algorithms of machine learning. Thus, instead of choosing a single algorithm and optimise it, I decided to systematically apply different classifiers to the dataset prior and after treating it with PCA for features reduction.

Feature Engineering

Keras ConvNet w/ Visualization

Created by: Abhijeet Mulgund
Language: Python

What motivated you to create it?

Compared to a lot of competitors, I'm fairly new to the Kaggle. I only joined last

summer after an older friend of mine pointed me towards Kaggle while we talked about machine learning and courses. After dabbling in a few featured competitions, I moved to the Leaf Classification competition with 2 main goals: to score a <0.01 logloss, and to become involved in the Kernels and Discussions. Once I had achieved the former, I realized I had a solution based off a fairly simple idea, merging image data with Kaggle's pre-extracted features, which would be perfect for people who might be stuck or interested in new ideas, making this solution the perfect way for me to get involved with Kaggle Kernels.

What did you learn from your analysis?

From my analysis I came up with one conclusion and one (untested) hypothesis. After examining the data through several of the data exploration kernels and posts, I noticed that data augmentation would help my neural net learn a lot of the invariances in the data like rotational, scale, and orientation invariance (I didn't need to worry about shifting because I centered the images in my pre-processing phase). So I added data augmentation through keras to try to compensate for the size of the dataset. I found that some data augmentation did give a noticeable boost to my combined image and pre-extracted features model, but if I made the data augmentation too aggressive, it would do more harm than good.

Visualizing different convolutional layers in the kernel Keras Convnet with Visualizations.

When I first tried out my idea of merging the image and pre-extracted data, I was actually very surprised it even worked. I was fairly new to convolutional neural nets and was not quite sure why simply feeding the network the pre-extracted features produced such beautiful looking filters while trying to train purely on images produced garbage filters. Even with data augmentation, I could not get the pure image CNN to learn anything. After publishing my kernel and learning more about CNNs, I came up with a hypothesis. I think passing the pre-extracted features helps stabilize the

learning a little bit. With randomly initialized weights trained from solely high-dimensional images, the CNN will have trouble learning anything. But the inclusion of the pre-extracted features helps stabilize the learning and helps with convergence during gradient descent. If this is in fact true, one consequence I hope to explore is whether initializing the convolutional weights of a pure image model with the convolutional layer weights of the combined model will give the pure image a nice set of filters that might allow it to learn during training.

Can you tell us about your approach in the competition?

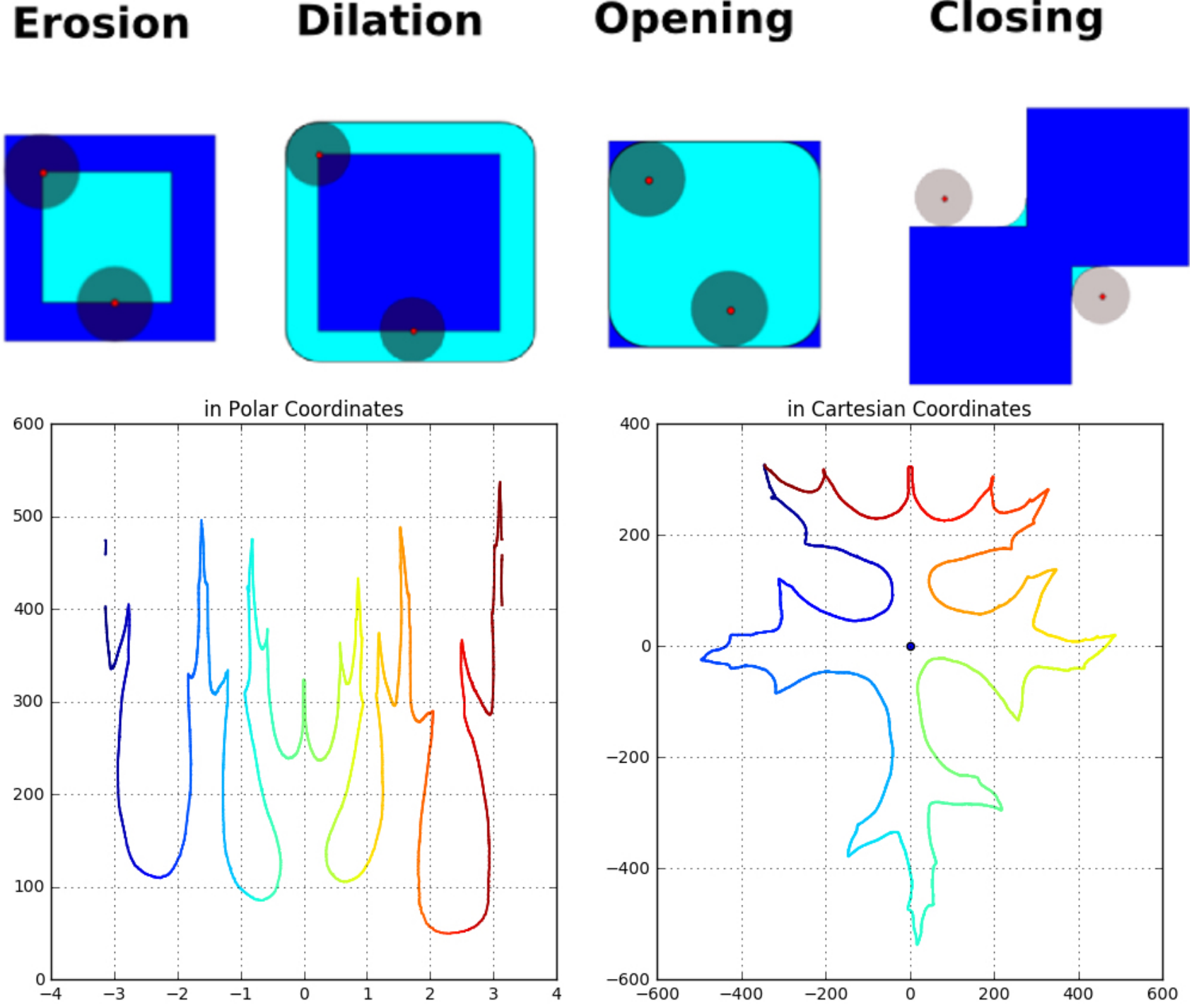
I started this competition by simply feeding the pre-extracted features into a multi-layer perceptron with one hidden layer and got surprisingly good results, but I still had all this image data that I wasn't using. My immediate thought then was to simply combine a convolutional neural network on the images with the pre-extracted features MLP and train the entire model end to end. To make up for the small size of the dataset I also threw data augmentation into the mix. Luckily, Keras already had built in image data augmentation for me to take advantage of.

Feature Extraction from Images

Created by: Lorinc
Language: Python

What motivated you to create it?

Even the best linear algebra MOOC puts me in sleep in seconds. Only working on something gives me enough reason to understand and comprehend the underlying principles. Therefore I often take on challenges I'm clearly not entitled to, and through countless hours of frustration (that I secretly enjoy a lot) I learn the subject. This time, I wanted to do the full lifecycle of a machine learning project, without relying too much on external libraries. And while I got way further than this notebook, I could not even finish the feature extraction, because I landed a dream job.



A step-by-step guide for extracting features from shapes by turning them into time-series in the kernel Feature Extraction from Images.

What did you learn from your analysis?

I never went deeper into any subject in my life, and I still know literally nothing. This business-maths-programming domain is not only huge, it is also infinitely deep. You can pick any niche topic in it and spend a lifetime mastering it. As I was sacrificing months of sleep reading up stuff at Wikipedia, I have learnt that there is an abyss of knowledge below the surface I considered as my universe until now. I wish someone opened this door for me when I was 13.

People underestimate the value of storytelling in data science. Magnitudes more valuable notebooks go down the drain unnoticed, because they are just a block of code. No story, no visuals. At the end, data scientists are just merchants, trying to sell their truths to their clients. Truth does not sell itself, take pride being a good merchant. For the greater good, of course.

https://github.com/lorinc/kaggle-notebooks/blob/master/extracted_leaf_shape.png

Insightful Visualizations

Visualization PCA and Visualizing k-means

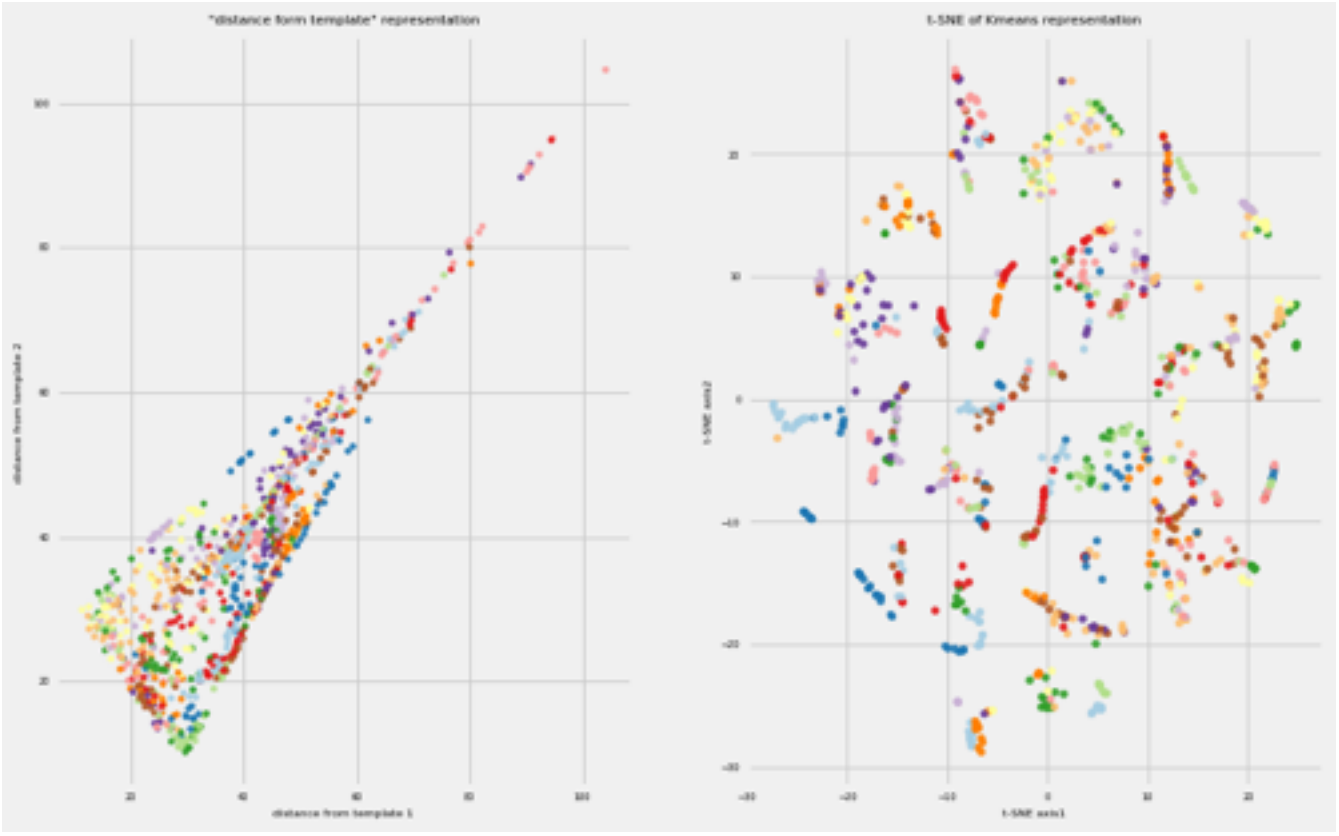
Created by: Selfish Gene
Language: Python

What motivated you to create it?

Until recently I was a heavy Matlab user, and have a lot of code accumulated in Matlab over the past several years.

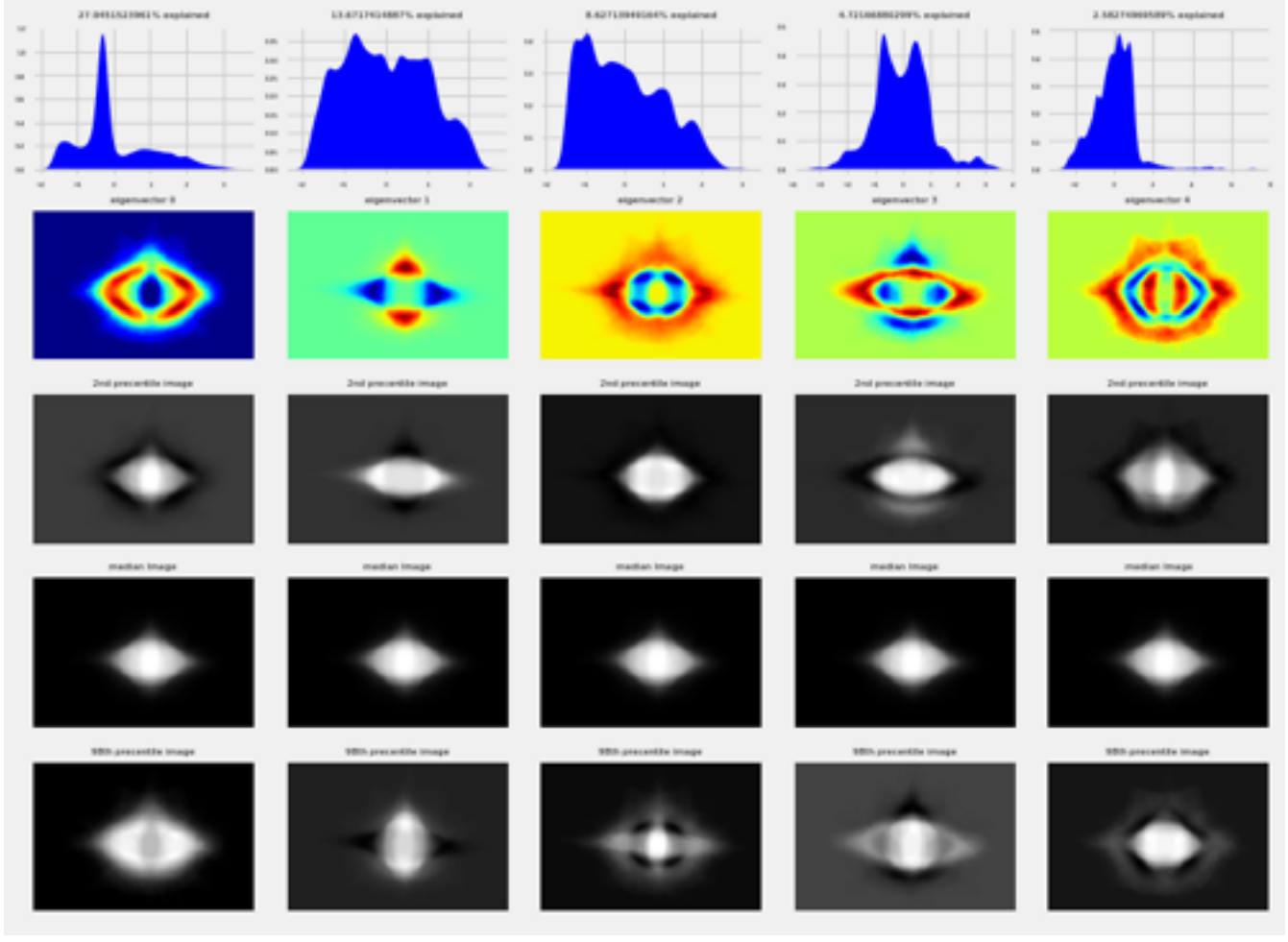
It’s funny but my main motivation for creating these two scripts was to force myself to translate the two extremely useful (for me personally) classes of GaussianModel and KmeansModel from Matlab to python.

Since Matlab is becoming less and less useful and python is already a much better tool in almost every aspect, I’m trying to completely back away from Matlab, and work exclusively with python.



Visualizing "distance from cluster centers" feature space in the kernel [Visualizing k-Means].

Another motivation was to illustrate how the results of the two simplest unsupervised algorithms, PCA and k-means, can be interpreted and visualized. I think it gives a lot of intuition about machine learning algorithms in general as it helps us understand how to think about objects (in this case, images) as points in a high dimensional space and understand what inner products in these high dimensional spaces actually mean.



How the leaf images vary around the mean image from the kernel Visualizing PCA.

What did you learn from your analysis?

I think applying basic visualization methods on data always helps understand how the data behaves and what are the main sources of variability in it. For example, I feel now that I understand a little bit more about what leaf shapes look like in real life.

Inspired to get started in computer vision? I recommend checking out our Digit Recognition getting started competition. You'll learn fundamentals of using machine learning to work with image data to classify handwritten digits in the famed MNIST dataset. We've curated a set of tutorials, but if you're brand new to Kernels, you can learn more here.