**Leaf recognition**

Name: Weiyi Lan, Yiqun Xu, Yang Zong

Section (6)

**Abstract:**

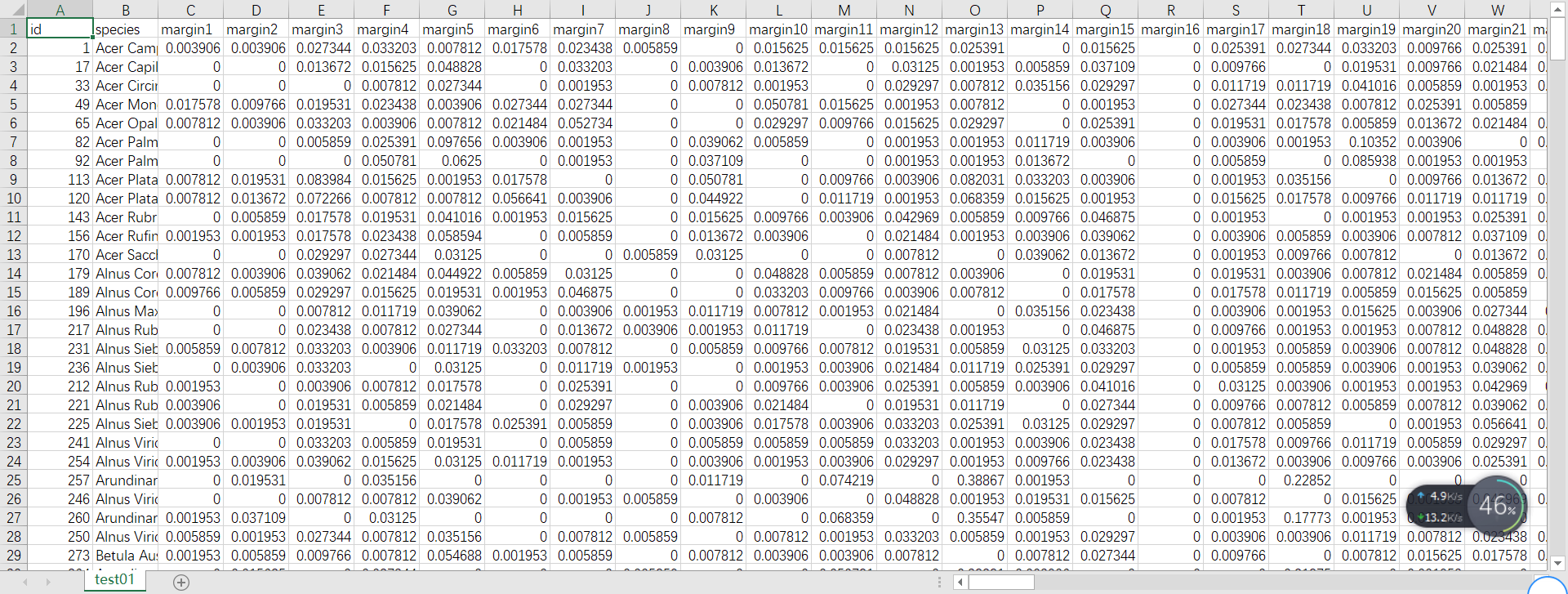
There are estimated to be nearly half a million species of plant all over the world. Here comes a problem that how can we classify these leaves, in other words, how can we get to know the name of leaves quickly and easily? Maybe we can use the knowledge in machine learning to do that. Machine learning has been a trending research and experimentation topic recently. But It is still a challenge when it comes to computer version. So we use neural network, which is a system of computer software that is patterned after the working of neurons in the human being. Deep learning refers to a subdivision of machine learning.one of the most popular deep learning technique is a convolutional neural network(CNN).it is commonly used for solving problems related to computer version. Our project is using CNN to recognize which kinds of leaves they are. Automating plant recognition might have many applications, including: Species population tracking and preservation, Plant-based medicinal research, Crop and food supply management and so on. If people can easily recognize different kinds of leaves, all the people can get full use of leaves very well.

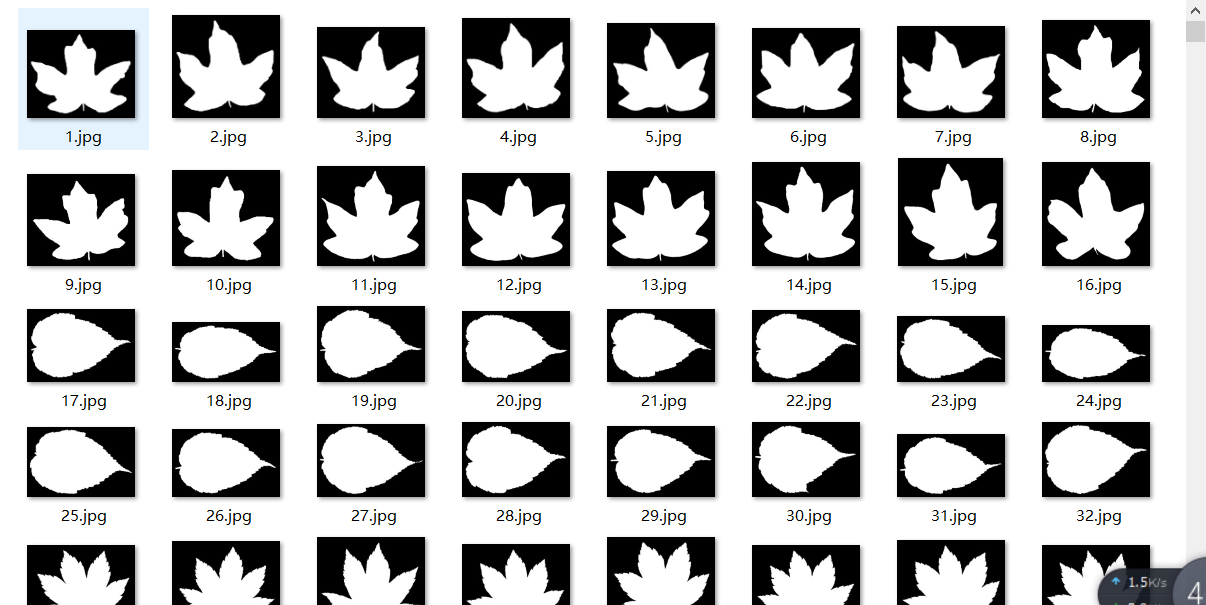
**Background:**

Modern description methods are used for plant classification through leaf recognition. These methods usually include color transformation, feature detection and description, dimension reduction, and classification. However, these methods use an original image as the input image from which to extract the features to be recognized. In this condition, computational complexity will increase. To reduce computational time, in the proposed method the Region of Interest (ROI) is extracted before extracting features from the image. Quality of image also plays an important role in increasing leaf classification rate. A good quality image gives better classification rate than noisy images. Using CNN, we can deal with the black-and-white graphs and get the vectors from different graphs and match the data in CSV file. Also we can get the similarity and difference among 100 hundreds kinds of leaves so that we can know the leaves better.

**Dataset:**

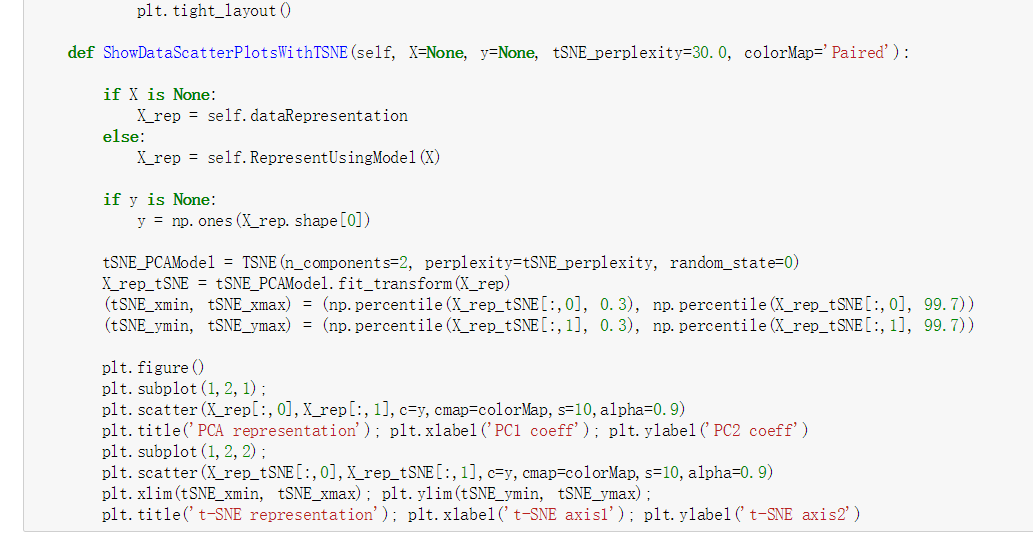
The dataset we use comes from UCI machine learning repository. And there are one-hundred plant species leaves as data. And Sixteen samples of leaf each of one-hundred plant species. For each sample, a shape descriptor, fine scale margin and texture histogram are given. And the number of each attribute is 64.

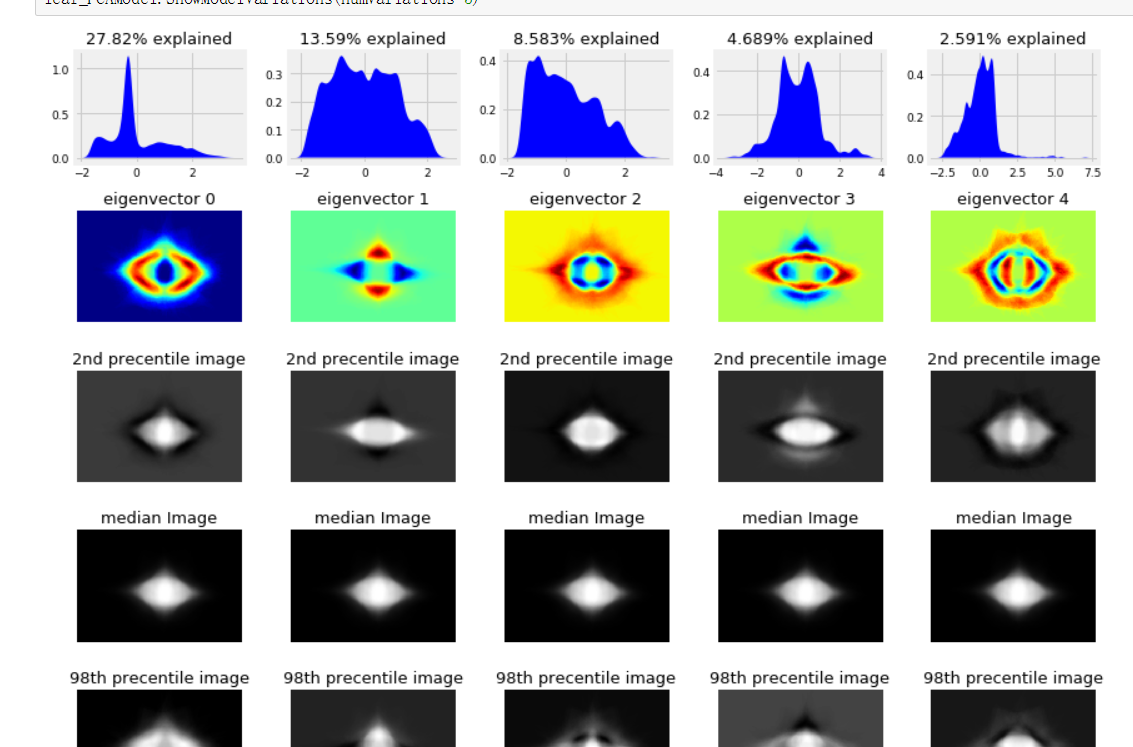




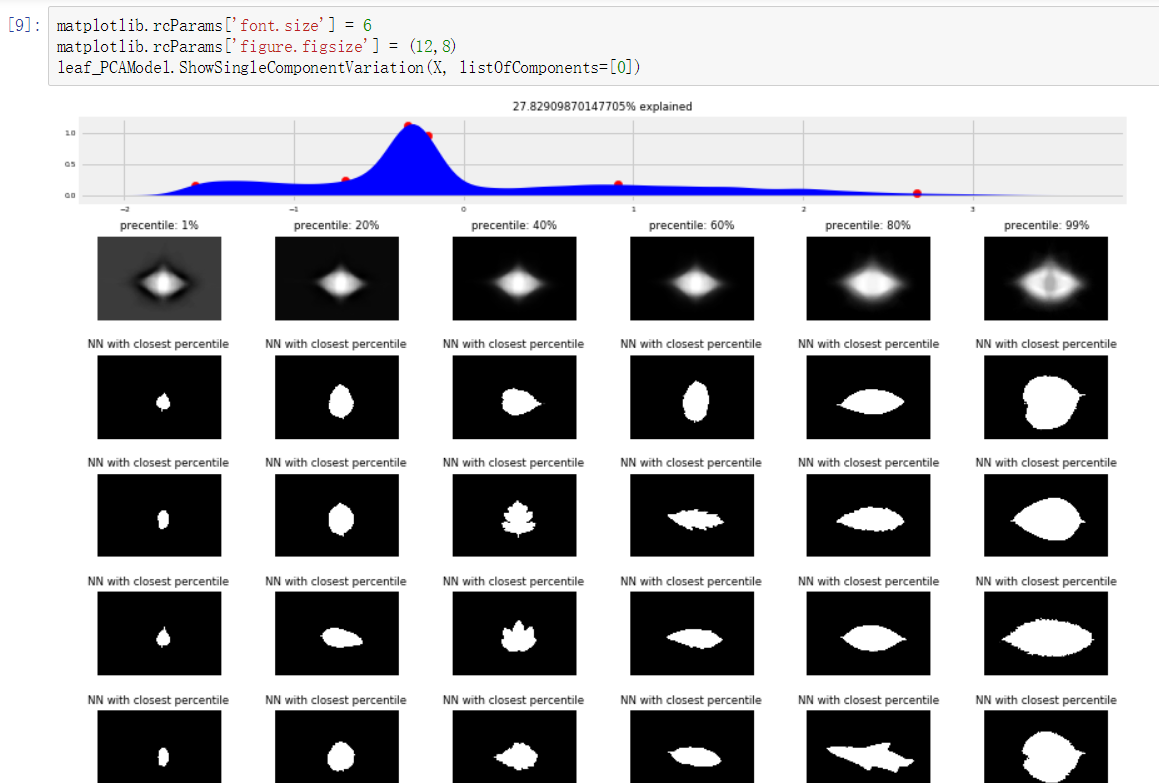
**Code with document:**

**Part 1: Use PCA to process data**

**mode a GaussianModel to reduce the dimensionality, and the eigenvalue** **left are most different ones.**

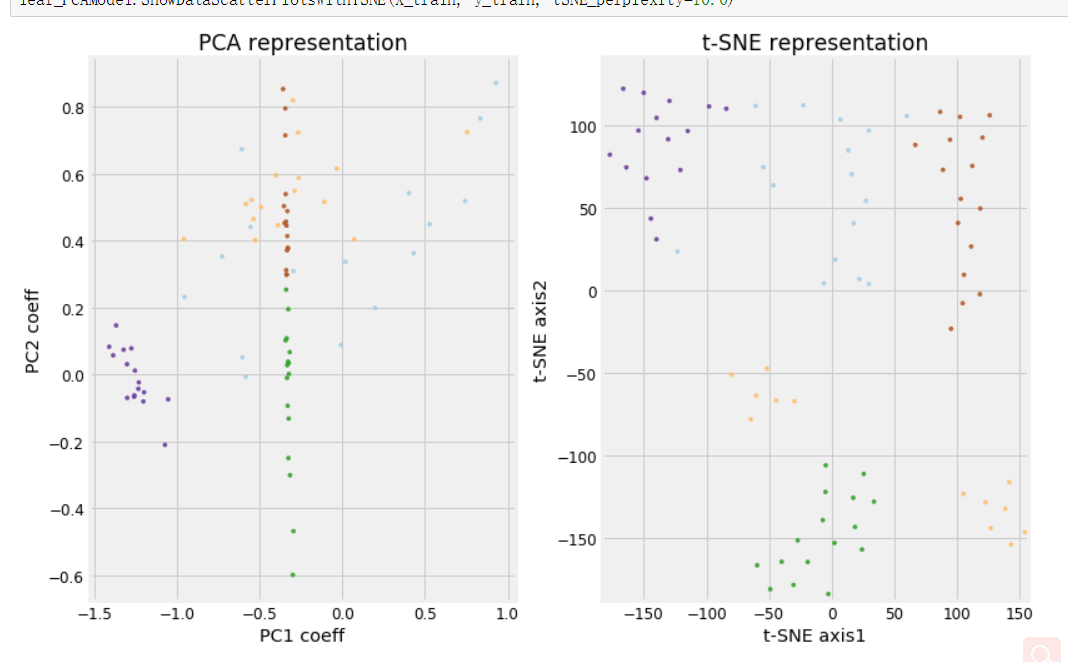


**Show model variations around mean image**

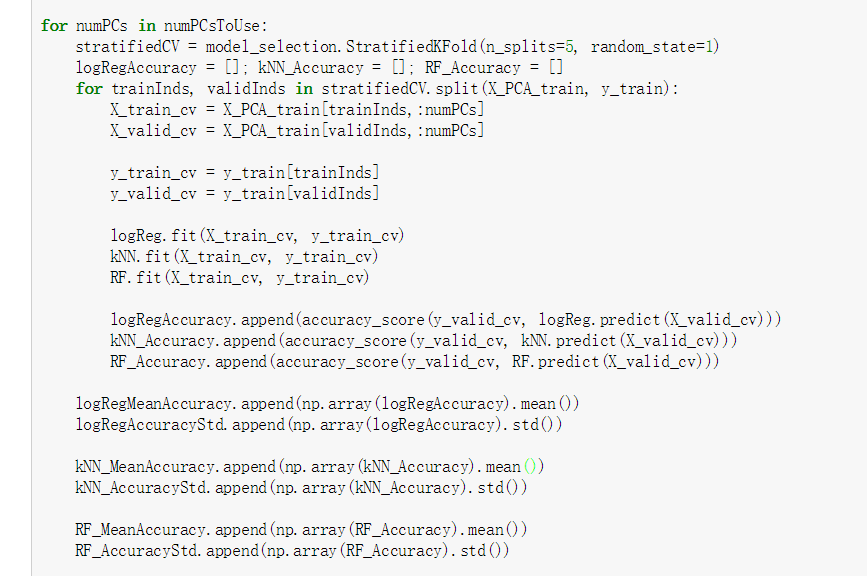
**Analyze Eigenvector 1 as an example: from left to right, the scale of the leaves becomes larger and larger.**

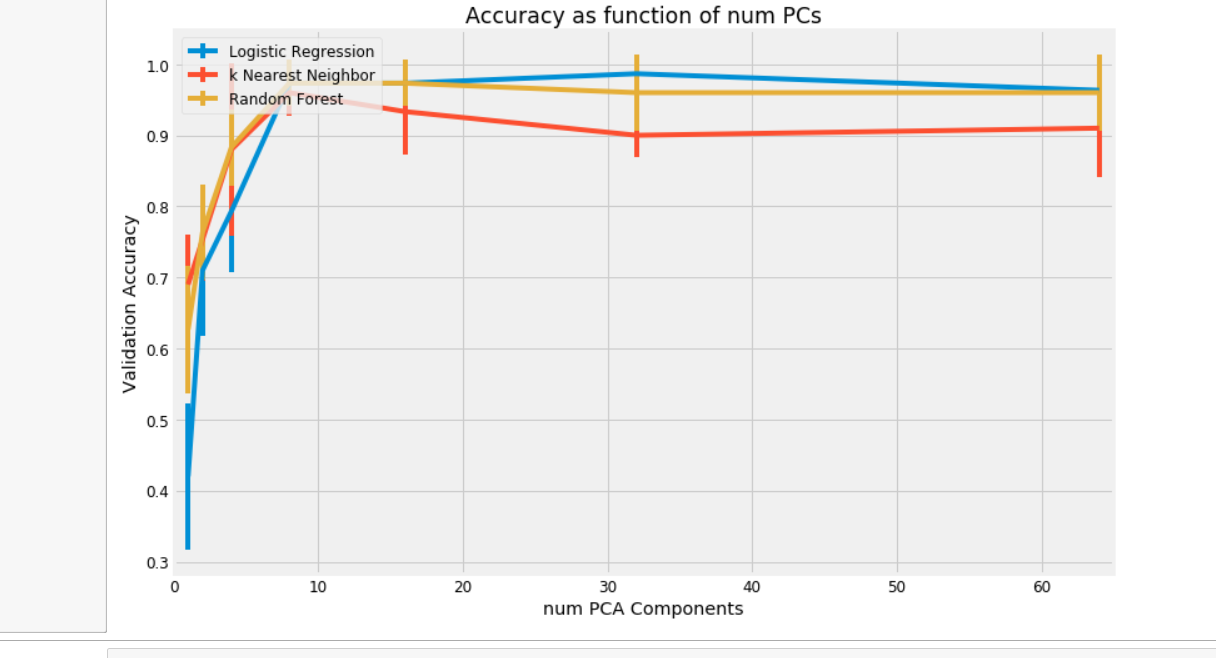
## 

**Show Scatter plot of Leaf images as points in high-dimensional space. PCA use only two parameters. And t-SNE means that showing the high-dimensional pictures into 2d picture.**



**Train the PCA using KNN:**



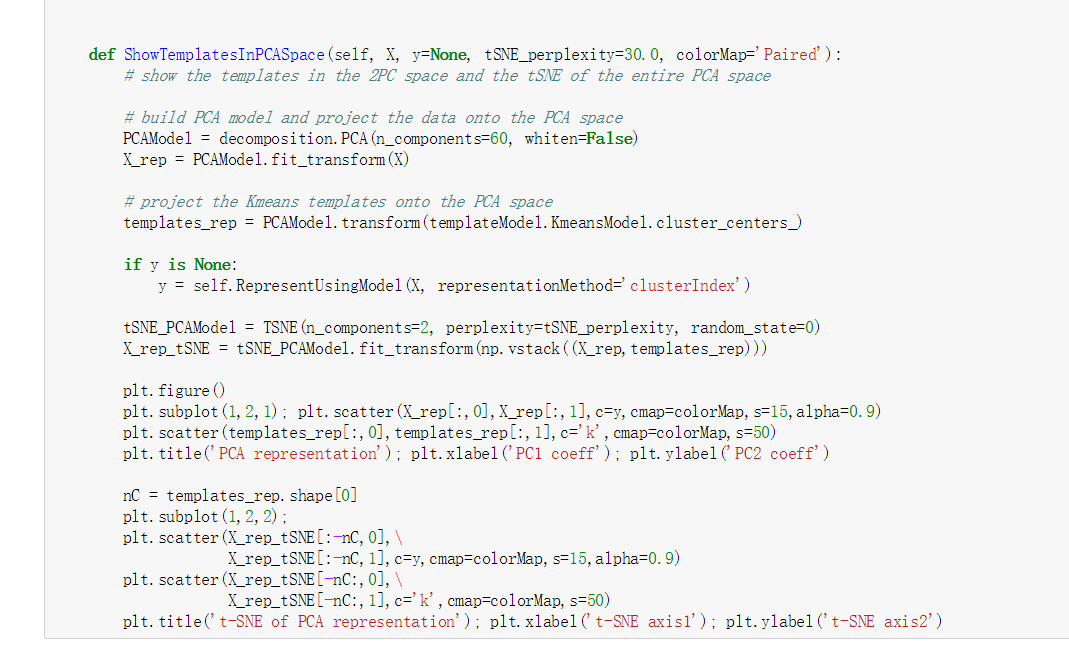
**show the Result:** 

**Show Model Accuracy as function of num PCA components**

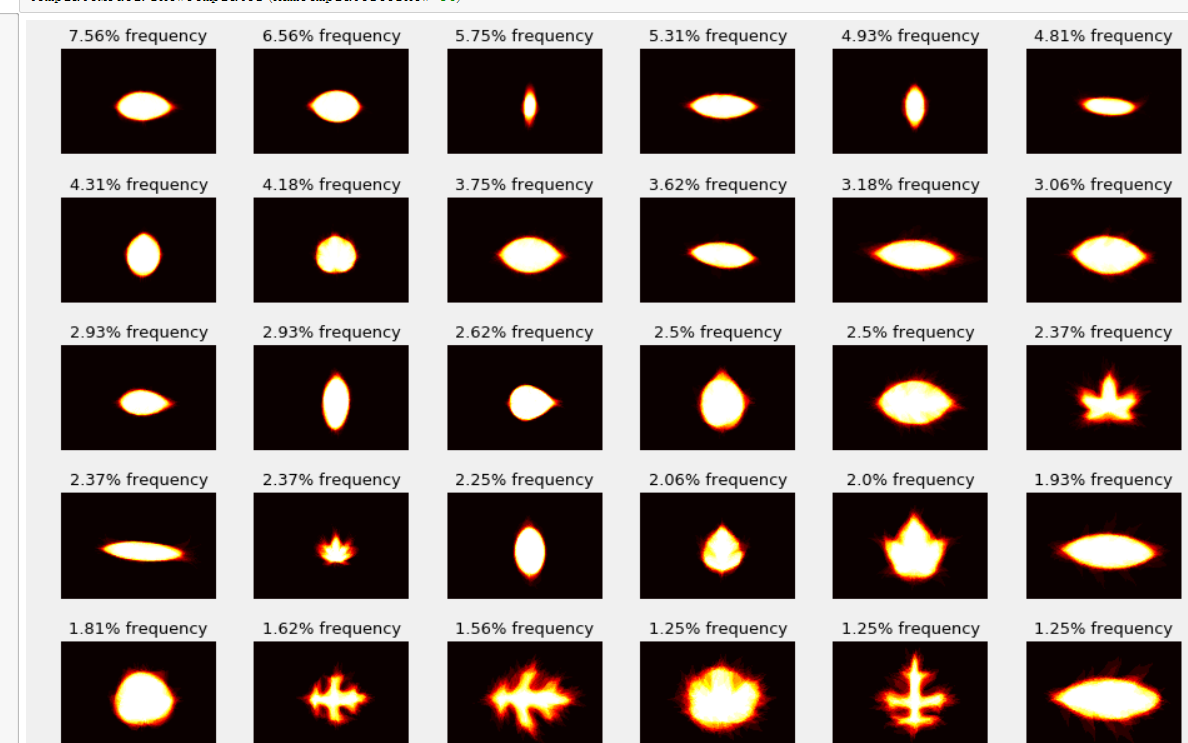
**Summary:**

**We can find that we can use PCA and T-SNE to reduce the dimension. We can get eigenvalues from the process and use clustering the data. As a conclusion, we can know that there are some rules in the dataset and we can use it to do the machine learning.**

**Part2 use k-mean to process data**

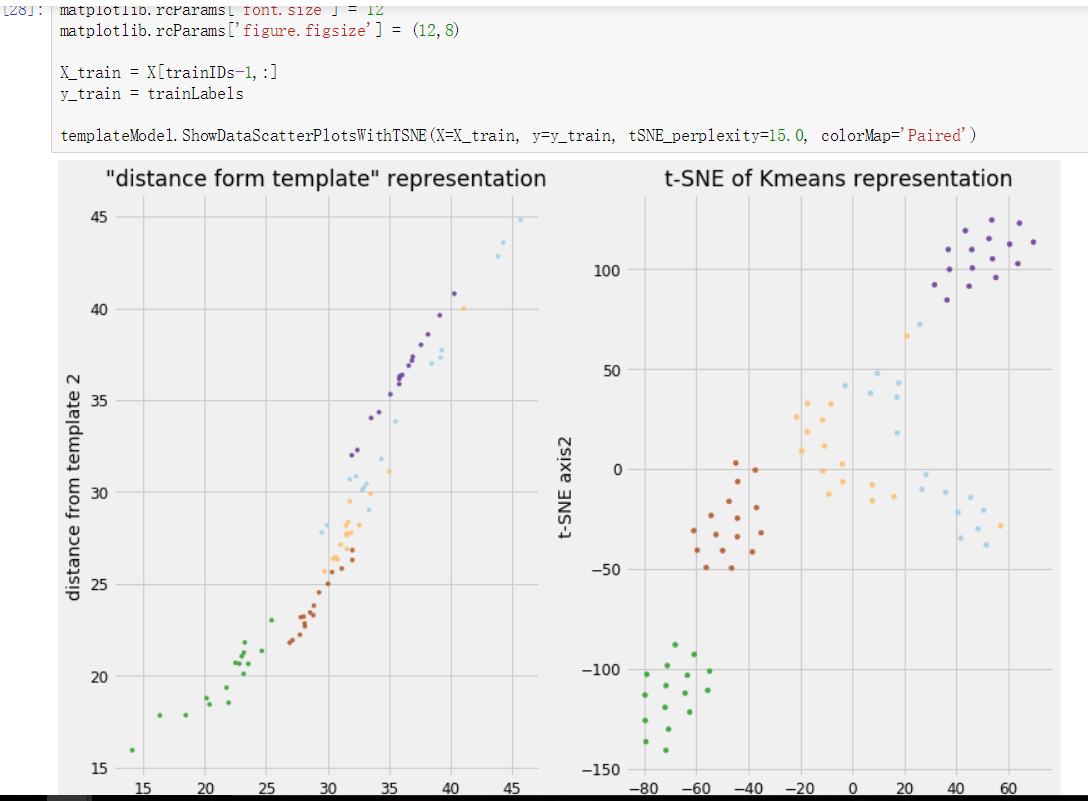


**Mode a k-mean model**



**Apply k-means and cluster all the data**

**Try k=36;**

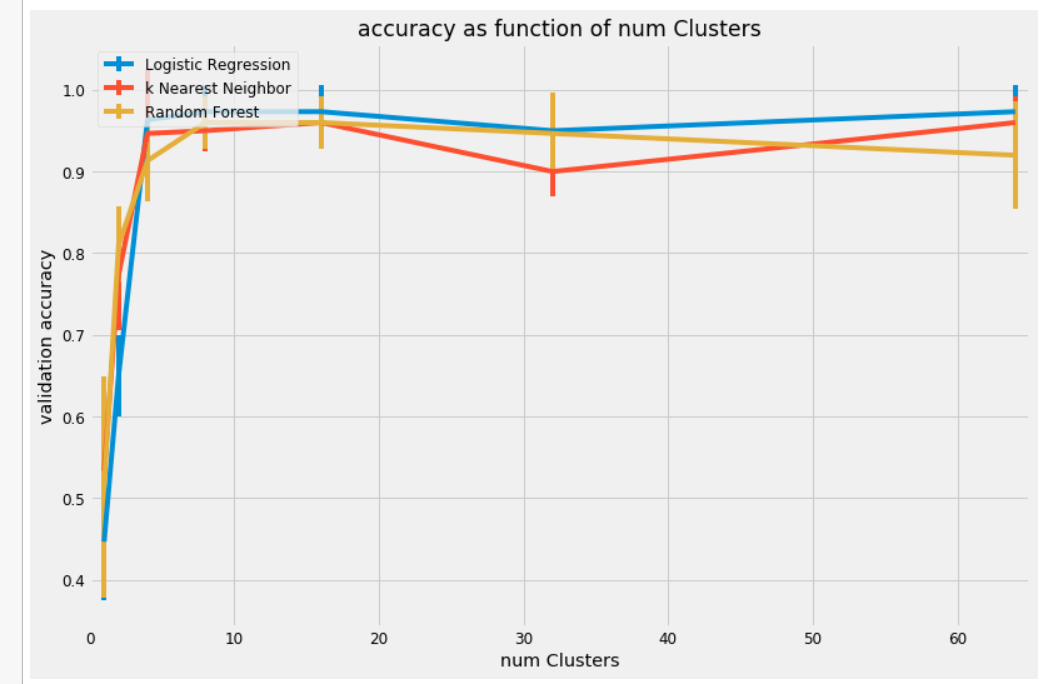
**Visualize "distance from cluster centers" feature space**

**Set 2 points and use a triangle method to decide the third points and doing the cluster work. T-SNE is also used for drawing a 2d picture with high-dimension parameters.**



**Train the mode and get the accuracy:**

**Result:**

**Summery:**

we build the k-means mode to show that when there is a center on the dataset. When the distance from the template is further, the difference between the eigenvalues are larger. For example, when there are two points respectively stay in the left and right side with the same distance, the eigenvalues are quietly different. And then we set 2 points which are next to each other, and then we can get the cluster and recognize the species.

## Summery about the first two parts:

1. PCA and K-Means image features are similarly useful in terms of classification.
2. The order between Logistic Regression and Random Forest has switched here compared to PCA case.

Even though these finding cannot be generalized because they heavily depend of this particular data distribution, we can speculate that there might be something complementary that Random Forest adds to the PCA feature representation, and that k-means features add to the classification abilities of the Logistic Regression classifier.

**Part4：simple train using CNN**

Train and test type: csv.file

Train set:1240

Single convolutional layer

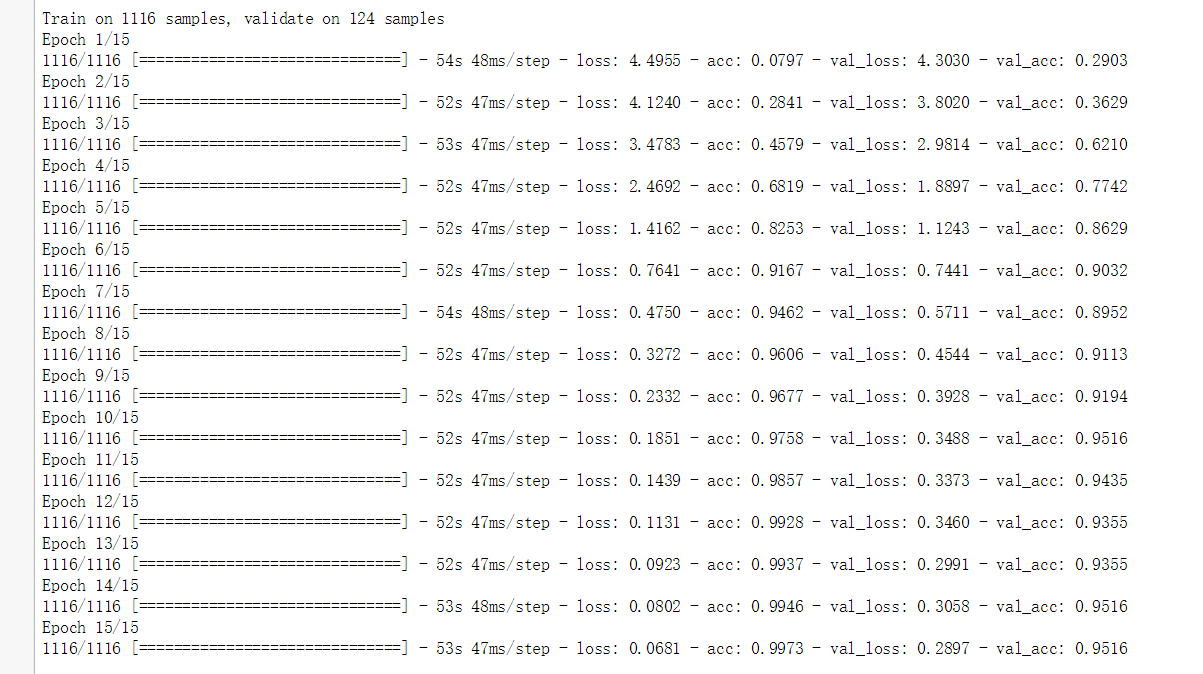
Activation method: relu;

Train method: softmax;

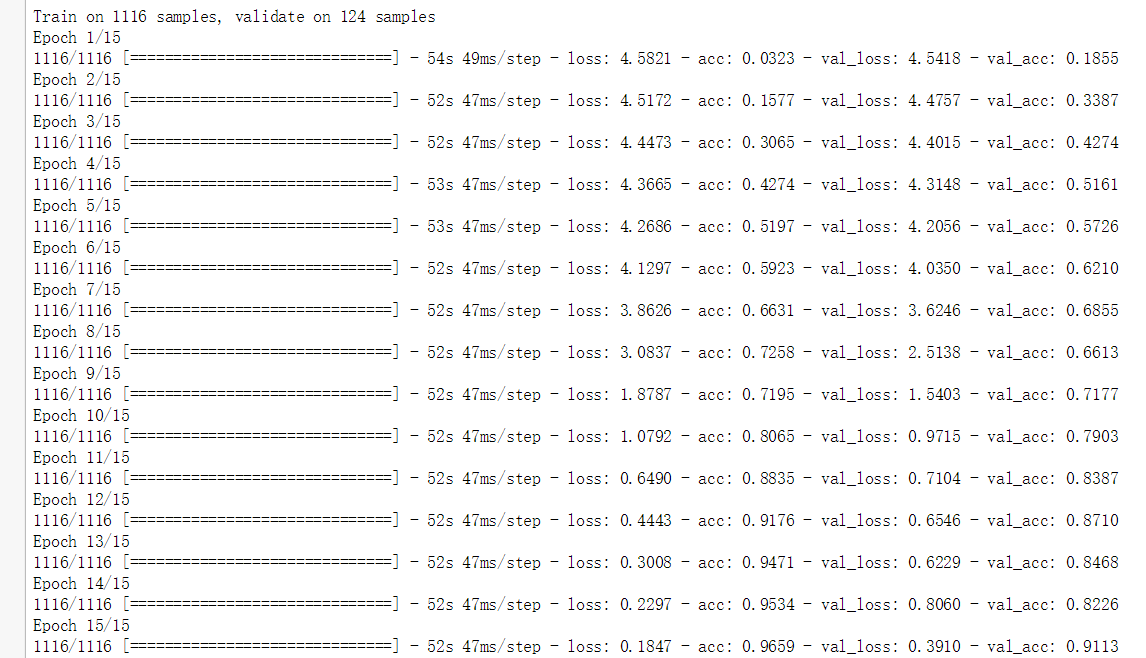
Loss: categorical\_crossentropty;

Opimizer: sgd;

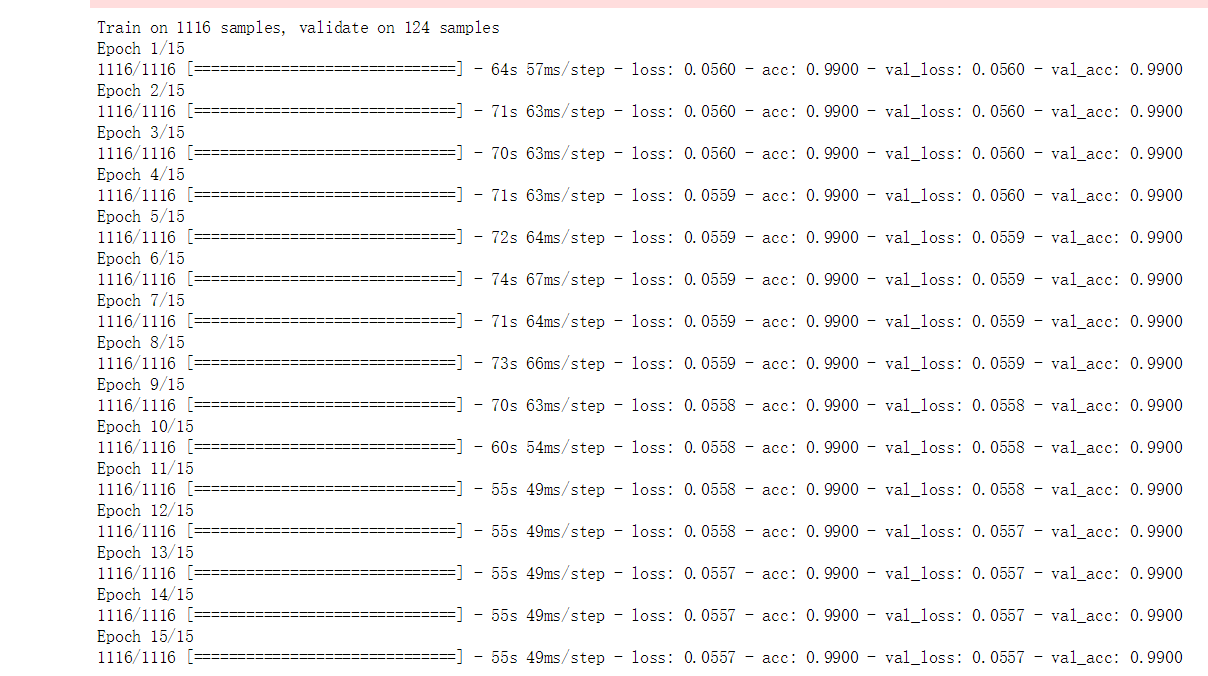




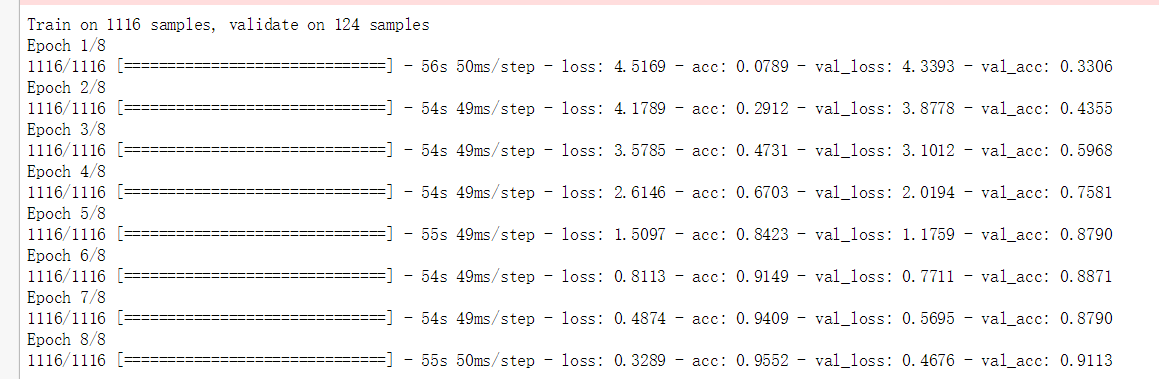
The original one:



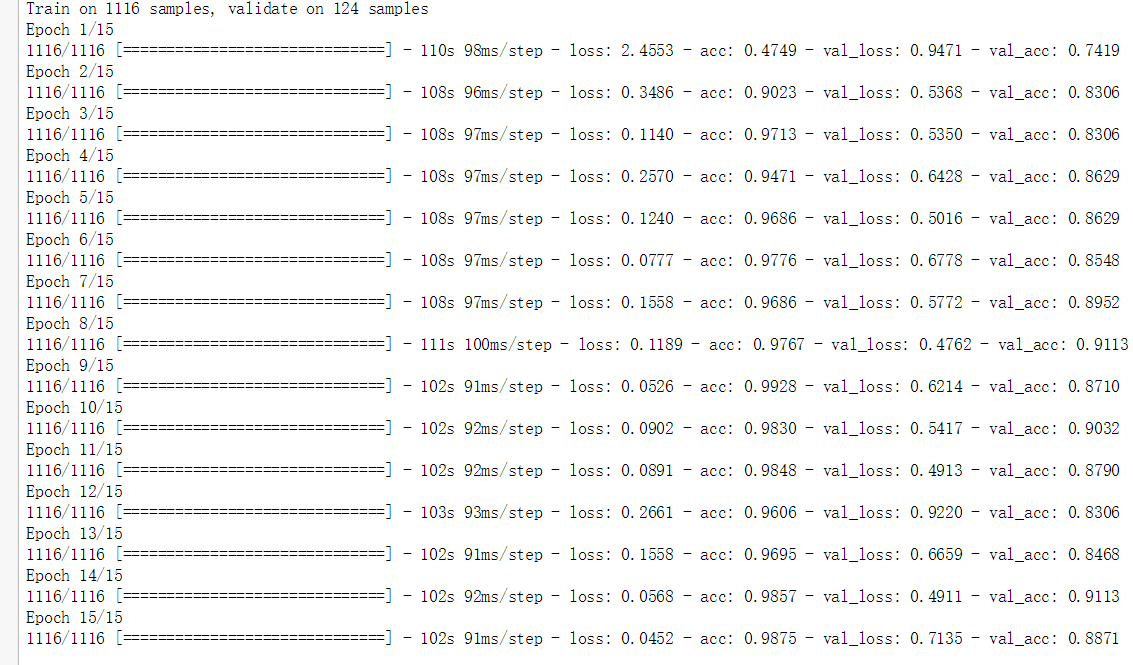
Change softmax to sigmoid:



Change categorical\_crossentropy to binary\_crossentropy:



Change the epoch from 15 to 8:



Change the optimizer from rgd to adam:

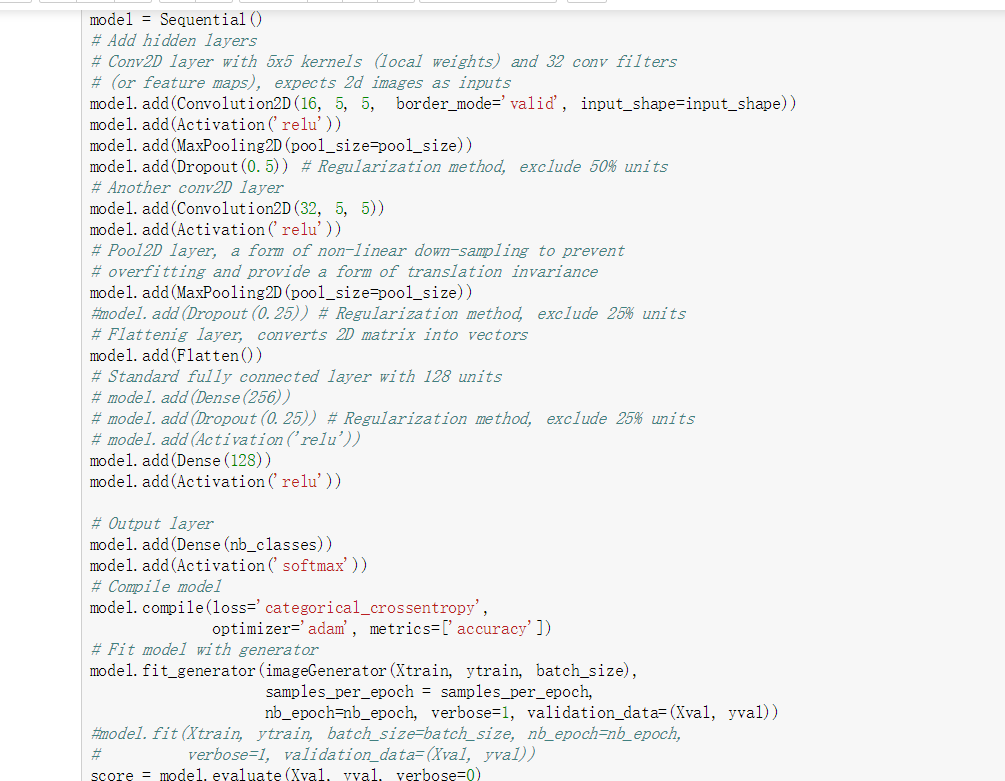
**Summary:**

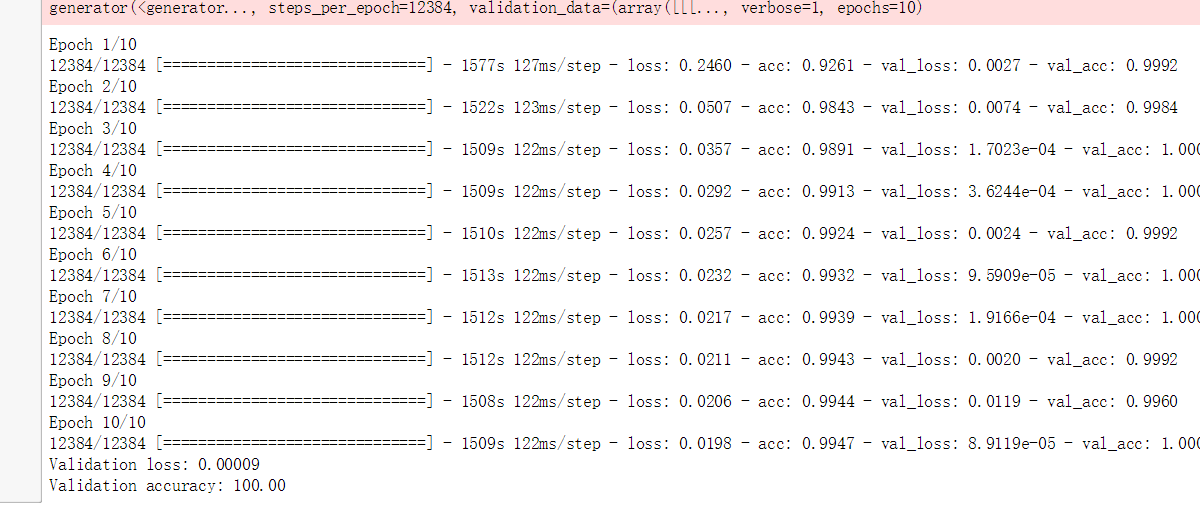
when we use the epoch 15, softmax , sgd categorical\_crossentropy we can get the best accuracy. Binary\_crossentropy may not suitable for our dataset because we can’t make out data binary.

**Part5:using CNN to train csv file and image at the same time**

Train set: image and 1240 csv

Test set: 357 csv



 Then we also do the same work as part 4, just change some methods, parameters activation and the size of the image. But the process is too long and we forget to get the screen shot, so we only post the highest one.

**Summary:**

we found that this one is the best one. When we use the image and the csv file at the same time, our accuracy is almost 1.

**Results:**

First, we use PCA and K-mean to prove that the eigenvalues have some rules and we can do some train using models.

Then we use CNN to train the csv file and the image folder. After comparing many kinds of parameters, we get the result that when we use the both files and use 4 Convolution layers. The activation=’softmax’ loss='categorical\_crossentropy', optimizer='adam', we can get almost 100percent accuracy.

**Discussion:**

1. Whether it is okay for us to reduce some parameters and then we can still get such a high accuracy.
2. If we only use the image part, can we get the similar accuracy.
3. Is there any other model except we can use to train the dataset, and how is the accuracy?
4. Is there any other method we can use to prove the feasibility of the machine learning?

**Reference:**

Dataset:<https://archive.ics.uci.edu/ml/datasets/One-hundred+plant+species+leaves+data+set>

Course: <https://onlinecourses.science.psu.edu/stat505/node/49>

Researchpaper: <https://www.researchgate.net/publication/266632357_Plant_Leaf_Classification_using_Probabilistic_Integration_of_Shape_Texture_and_Margin_Features>

T-SNE: http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Gaussianmodel:https://blog.dominodatalab.com/fitting-gaussian-process-models-python/

PCA example: <https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60>

PCA-KNN: <https://www.kaggle.com/heibankeli/pca-knn>

KNN:https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm

[k-means script](https://www.kaggle.com/selfishgene/leaf-classification/visualizing-k-means-with-leaf-dataset/notebook): <https://mubaris.com/2017/10/01/kmeans-clustering-in-python/>

k-means exmaple: https://www.kaggle.com/naivecharles/k-means-neighborhood-clustering

CNN example:https://www.kaggle.com/tobikaggle/nn-through-keras-copied-mod-shuffle

CNN: <https://github.com/keras-team/keras/blob/master/examples/imdb_cnn.py>

CNN: <https://www.kaggle.com/tonypoe/keras-cnn-example?scriptVersionId=589403>

Tensorflow CNN：<https://www.kaggle.com/jiexus/cnn-with-tensorflow/notebook>

[feature extraction from images](https://www.kaggle.com/lorinc/feature-extraction-from-images):https://www.python.org/dev/peps/pep-0008/