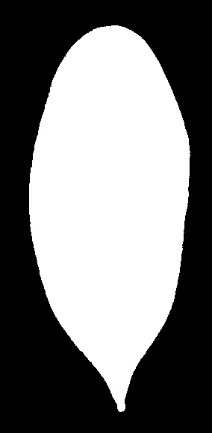
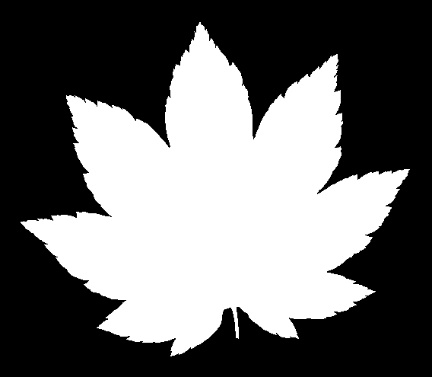
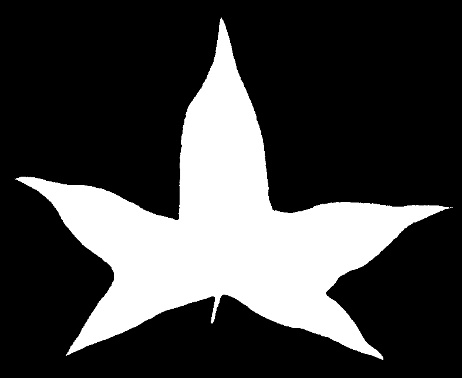
**Introduction to machine Learning – Final Project**

**Leaf Classification**

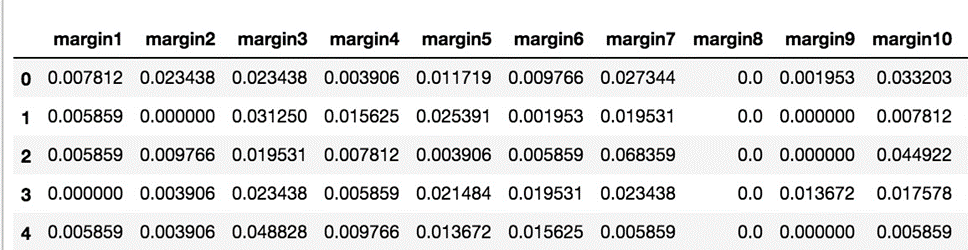
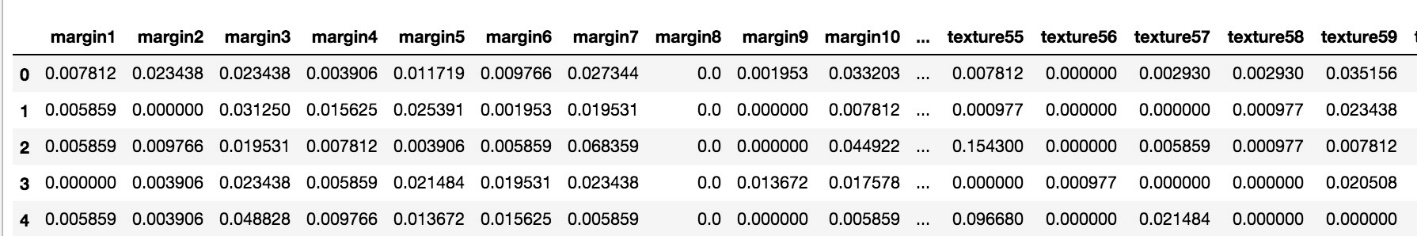
**Dataset**

Our dataset is downloaded from Kaggle(https://www.kaggle.com/c/leaf-classification), and it’s related to leaf classification. The dataset is made up of **1,584 images** of leaf specimens (**16 samples each of 99 species**), which have been converted to binary black leaves against white backgrounds. Two types of data are put on Kaggle: The **raw pictures and the numeric numbers** describing leaf’s features.

Raw Data(gray picture):

Numeric Data(192 features):

There are three sets of features: Margin, Shape and Texture, each of them consists of 64 attributes. Therefore, the leaf dataset has 192 features in all.

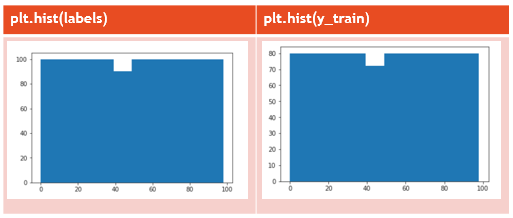
**Machine Learning**

* **Data preparation – Stratified training and testing data：**

StratifiedShuffleSplit(labels, test\_size=0.2)

Although we have 1584 data, they belong to 99 species, which means that **each species has only 16 data in the dataset**. In order to make sure every kind of species is put in our training model, we use StratifiedShuffleSplit to spilt training/testing data. **Stratified ShuffleSplit cross-validator** and provides train/test indices to split data into train/test sets.

Diagram below is the histogram of all the original data and after spiliting data, we can see that all species are concluded in our testing data. In addition, their labels **ratio is equal**.



Stratification is necessary for this dataset because there is a relatively large number of classes (99 classes for 1584 samples). This will ensure us to have all classes represented in both the train and test indices.

* **Preprocessing – Standard Normalization：**

scaler = **StandardScaler()**

Standardize features by removing the mean(to 0) and scaling to unit variance(to 1). It is helpful when we want to compare data that are to different units.

This funstion makes all the features represented in a consisent way.

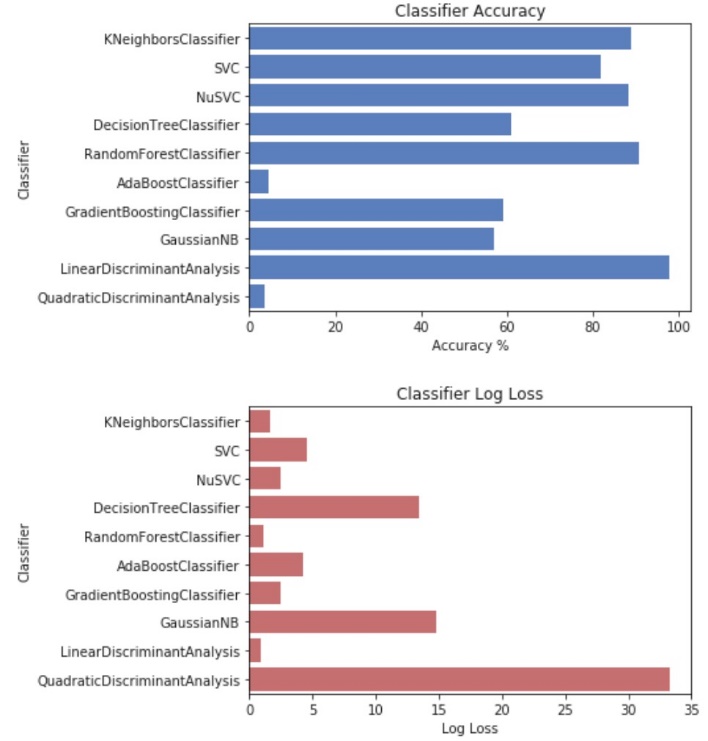
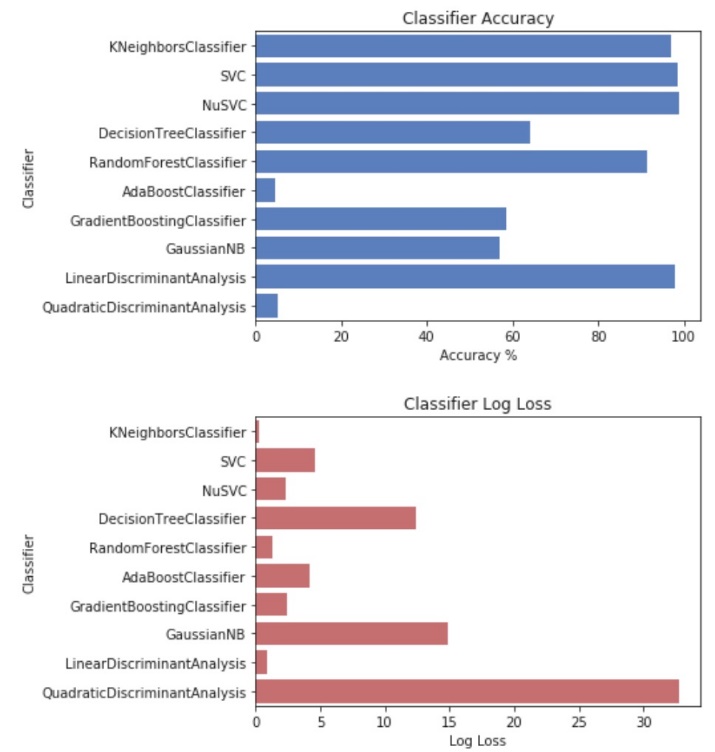
* **Implementation：**

We run 10 common classifiers(**KNeighborsClassifier, SVC, NuSVC, DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, GaussianNB, LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis**), and the default outcome are shown below(left graph).

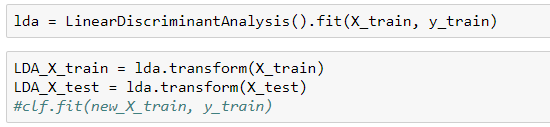
The graph shows that **LinearDiscriminantAnalysis** classifier has the best accuracy, which can make it to 97%(REALLY HIGH!). And another classifier such as kneighbor and decisiontree behaves well, too. However, **Gaussian Naive Bayes**’ accuracy is quite low, less than 60%.



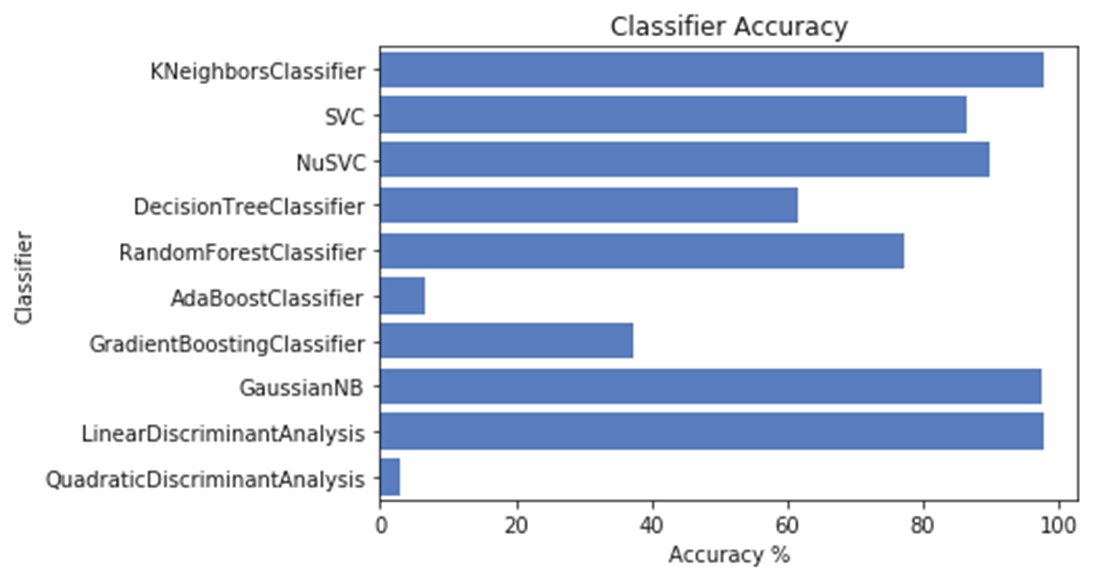
After completing **standard normalize**, accuracy and log loss change(right graph). Feature normalization makes accuracy increases and log loss drops a little bit:

**Reduce feature dimension - LDA method 192 -> 92 dim**

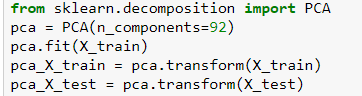


LDA works as a dimensionality reduction algorithm, we use LDA to drop features numbers from 192 to 92, and take the new 92 features to build train model again.



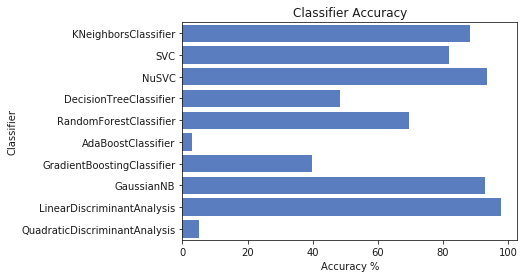
Most of the classifiers become better. There is also a few classifier don’t appreciate LDA, for instance, Randomforest’s accuracy drops. Surprisingly, **GaussianNB**’s accuracy progress a lot!

**Reduce feature dimension - PCA method 192 -> 92 dim**



We also use PCA to implement feature reduction, Decreasing features numbers to 92 also, and the new accuracy are shown below:

Take the new 92 features to build training model again.



Behaving like PCA’s result, most of the classifiers becomes better. There is also a few classifier don’t appreciate LDA, for instance, Randomforest’s accuracy drops. Surprisingly, **GaussianNB**’s accuracy progress a lot!

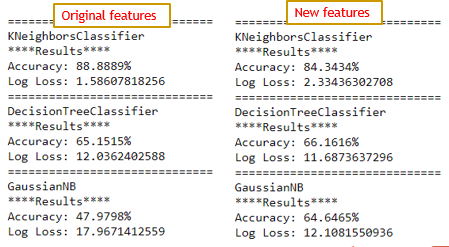
**Why LDA > PCA?**

LDA is a **supervised** whereas PCA ignores class labels, is **unsupervised**. The leaf dataset provides us answers. LDA cares about class separability, while LDA makes assumptions about normally distributed classes and equal class covariances. During model building, LDA takes both features and targets in, so it can tune out the parameters that are close to real data appearance. On the contrary, PCA only focuses on the separability of data features, the more the better.

We are testing leaf classification today, it is a supervised learning, so LDA performs better!

**Reduce feature dimension – Feature importance(Decision Tree) method 192 -> 92 dim**

The third method we use for dimension reduction is calculating the **feature importance, basing on Decision Tree**. When spilting left/right child, decision tree **calculate Gini to get information gain**, and the information gains enables us to find feature importance.

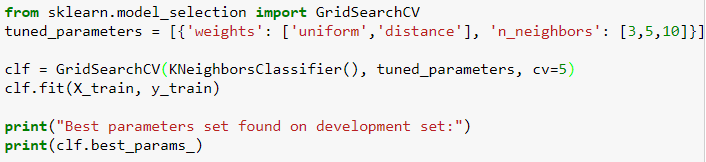


We can find that **GaussianNB improves again**! In our three different kinds of reduction, GaussianNB’s accuracy all increase a lot.

**Why?**

Originally, we have 192 features, GaussianNB is **probability based,** so too many features will impose noises, lowering the influences of important features. Therefore, in GaussianNB, adequate features selecting is important, we should take the useful and crucial ones, rather than put all features in to build models.

**KNN Fine-tuning**



The parameters of the estimator used to apply these methods are optimized by **cross-validated grid-search over a parameter grid**.



The best\_params\_ it returns are n\_neighbors=3, weights=distance(closer neighbors of a query point will have greater influences).

**Deep Learning Approach**

We use three different models: MLP、CNN、MLP+CNN.

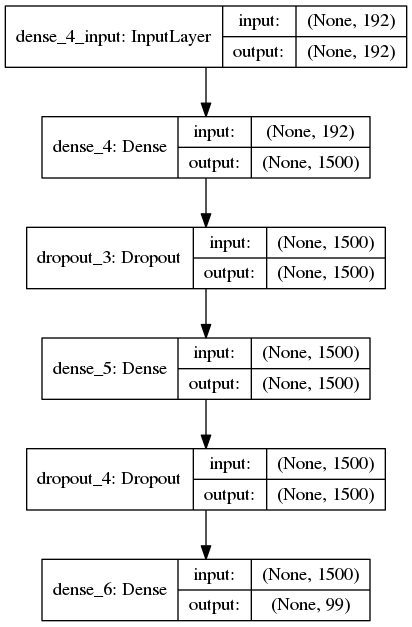
* **MLP**
* **Input:**

Three feature vectors with 64 dimensions

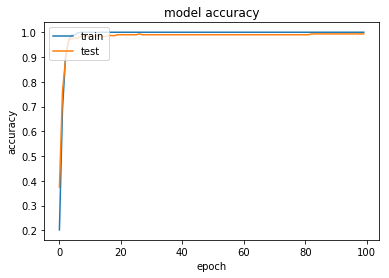
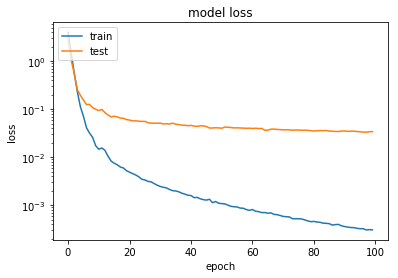
* **Output:**

The probabilities of 99 classes

* **Architecture:**



* **Plot:**



* **Description:**

我使用一個有兩層hidden layer的MLP來做預測，其中第一層的activation function是Relu，第二層為Sigmoid，output layer的activation function則是softmax。

在訓練時，loss function為Cross Entropy，優化演算法為Adam，訓練100個epoch後，選用在validation data表現最好的模型最為最後的模型。

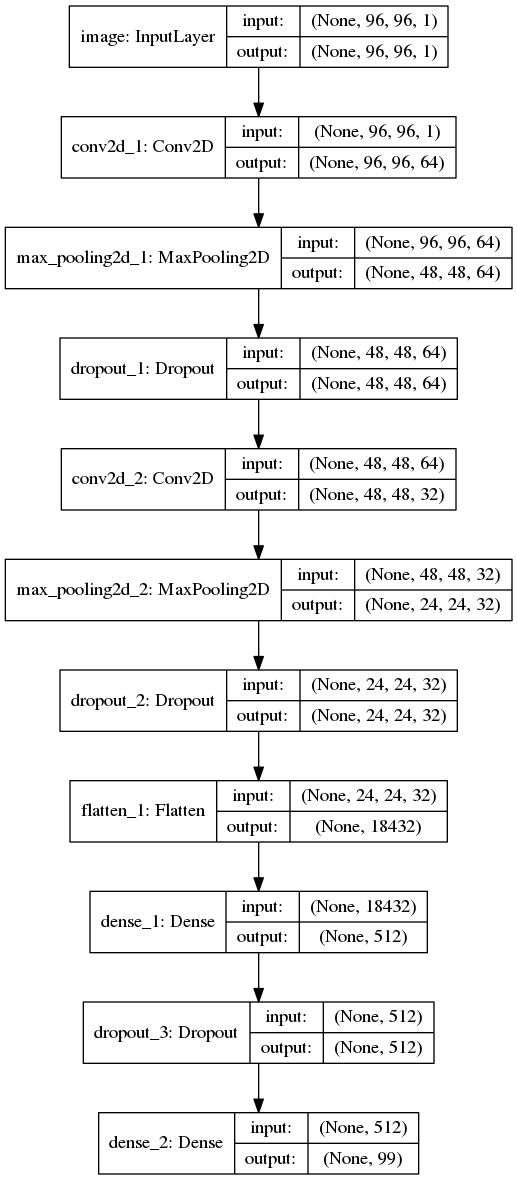
* **CNN**
* **Input:**

96 x 96 Images with data augmentation

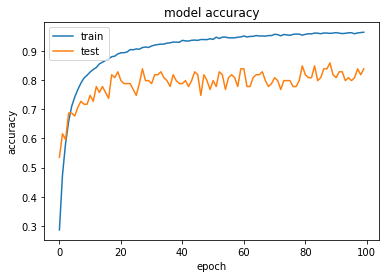
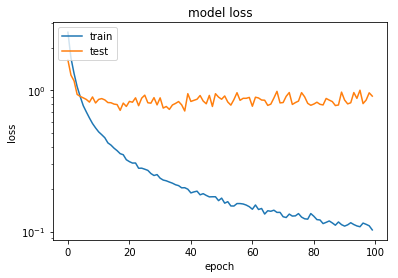
* **Output:**

The probabilities of 99 classes

* **Architecture:**



* **Plot:**



* **Description:**

在這個架構中，我打算使用CNN來直接抽取圖片中的特徵來取代用來dataset中的特徵。我使用了兩層Convolution + Maxpool，其activation function皆為Relu，在將由CNN所產生的18432個特徵輸入有1層hidden layer的MLP中，最後的output layer同樣使用softmax作為activation function。

由loss及accuracy變化圖來看，這個架構即使在加上dropou仍有**overfit**的情形發生。我認為可能原因有三：

1. 只有1層hidden layer的MLP可能無法學出正確的分類方法

2. 只有兩層的CNN所抽取的特徵無法分辨出99類的葉子

3. 即使經過data augmentation後，資料仍然不夠多或不夠一般化。

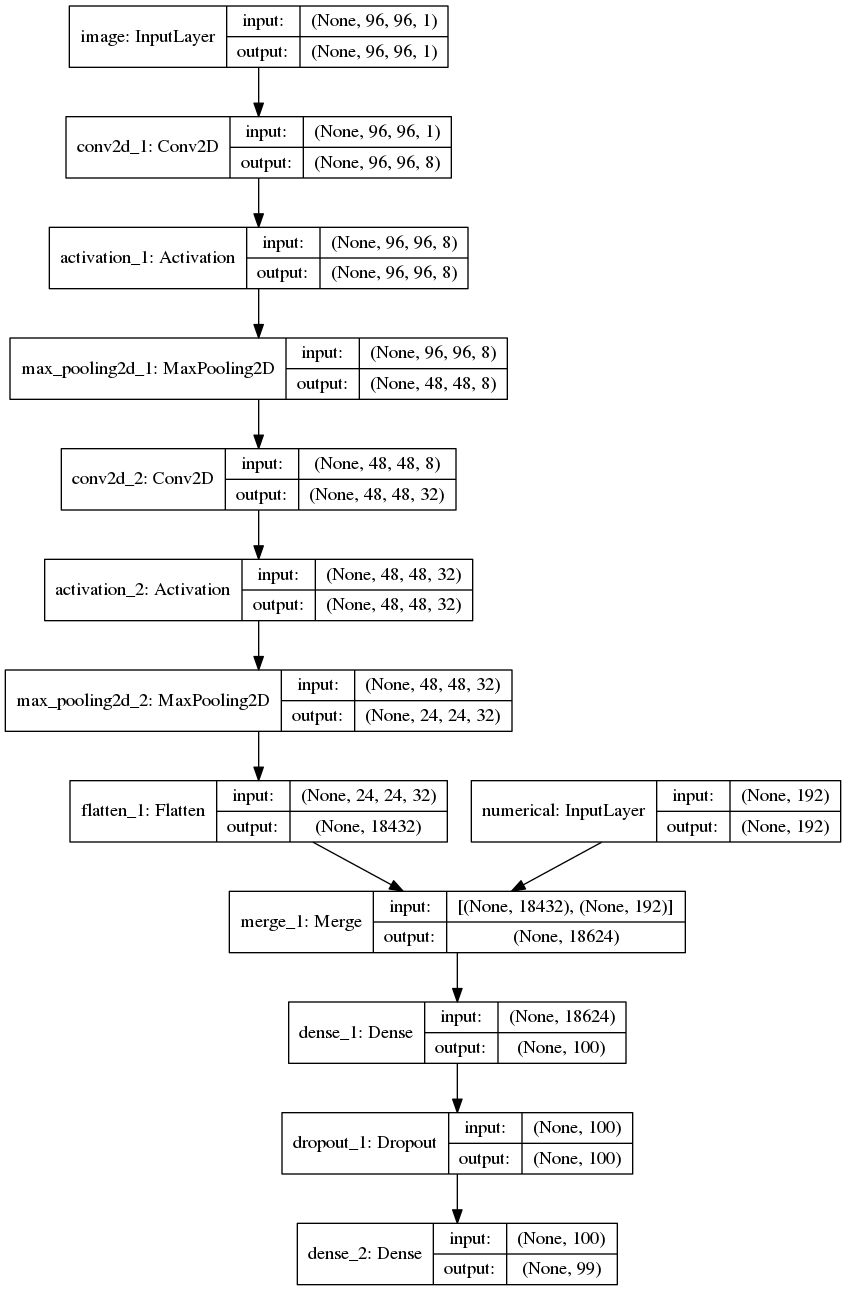
* **MLP + CNN**
* **Input:**

Three feature vectors with 64 dimensions + Three feature vectors with 64 dimensions

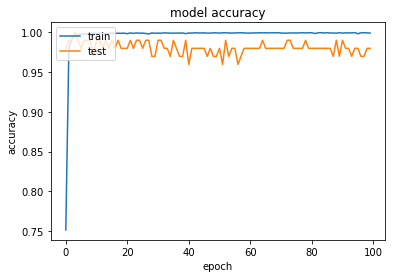
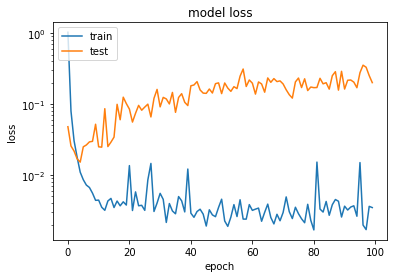
* **Output:**

The probabilities of 99 classes

* **Architecture**:



* **Plot:**



* **Description:**

在這個架構中，我**結合前面兩個架構**，使用64\*3再加上從CNN抽取出來的特徵去訓練模型，這個模型是**三個模型中表現最好的**，有最高的Val accuaracy跟最低的test loss。

這個模型的架構很簡單，就是將前述CNN的架構再加上原來的64\*3個特徵，輸入一個一層hidden layer的MLP中。

* **Result Table**

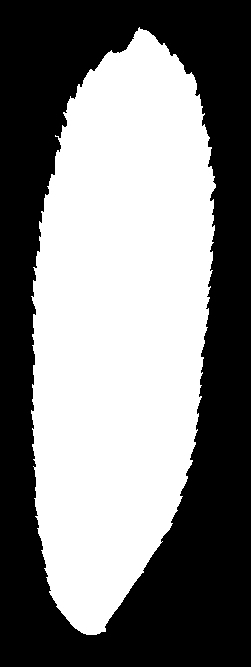
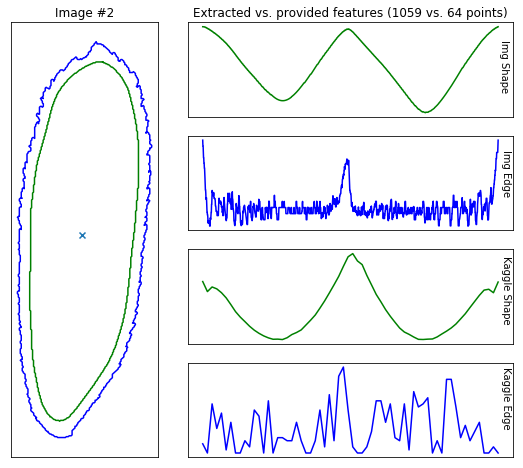
|  |  |  |  |
| --- | --- | --- | --- |
|  | Train Accuracy | Validation Accuracy | Test Loss |
| MLP | 1.00 | 0.99 | 0.02954 |
| CNN | 0.96 | 0.86 | 0.89944 |
| MLP+CNN | 0.99 | 1.00 | 0.01778 |

**Feature Engineering**

* **Motivation：**

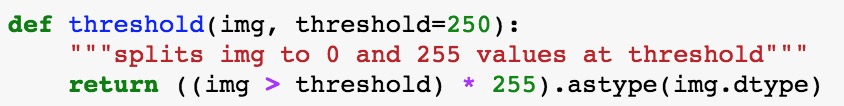
Kaggle provides three type of features (shape, margin, texture); however, the dataset **already had further feature extraction**. Therefore, we cannot use the true ‘**raw data**’ for meaningful feature engineering.

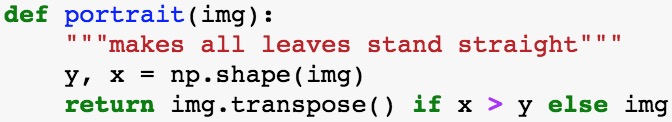
In order to solve this problem, we try to reconstruct original shape, edge features from each given image.



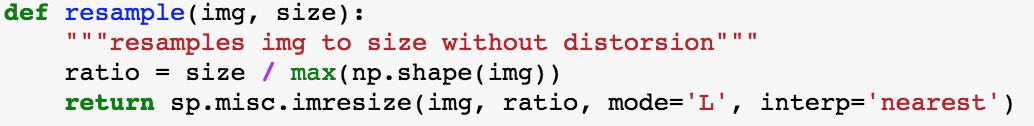
* **Implementation：**

**Step 1 - Preprocessing**

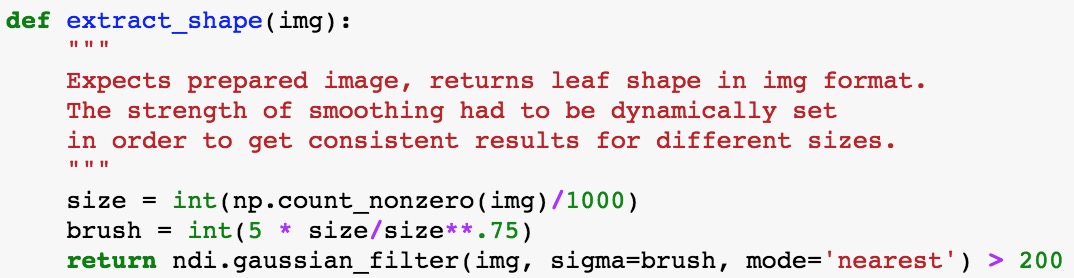
1. threshold 🡪 leaf image not simply 0 or 255, is 0 ~ 255

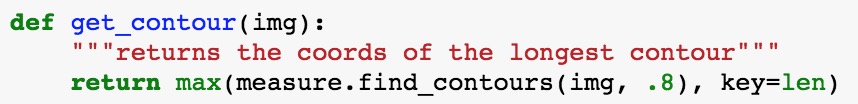
2. portrait 🡪 make every leaf stand up

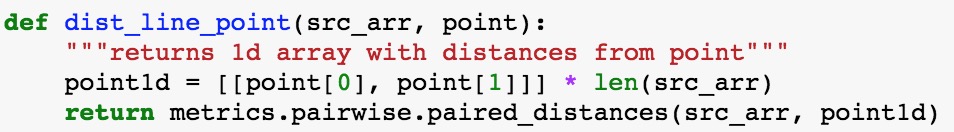
3. resample 🡪 resize image to a same size

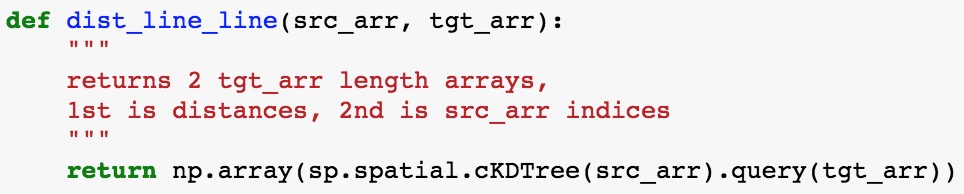


**Step 2 – Adding Features**

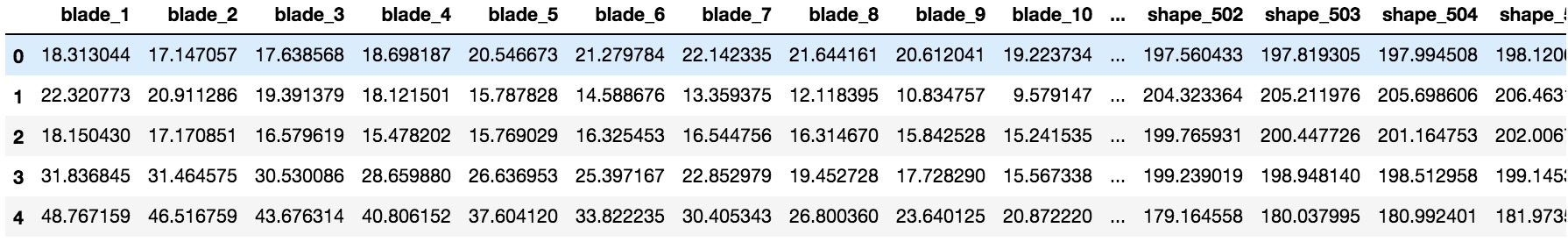
1. Make a gaussian blur image for shape

2. Get contours from original / blur image

3. Compute shape distance (line to center) 

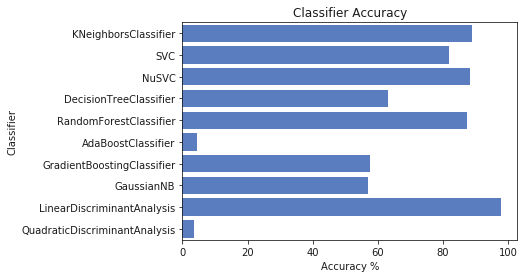
4. Compute blade distance (line to line) 

5. Combine 3. 4. into original feature

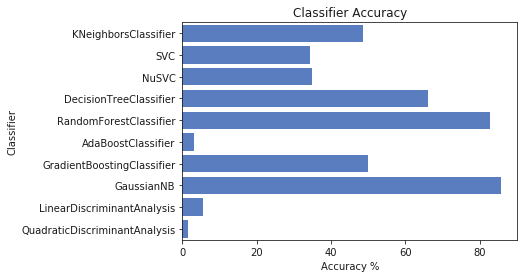


* **Result**

(Default 192 features)



(New 1214 features)

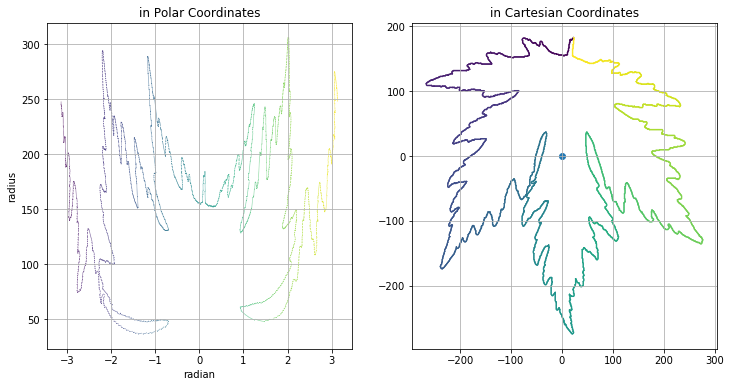


Obviously, adding new features don’t increase the accuracy of each classifier. We conclude that the new features are not meaningful at all currently. Further general feature extraction is needed. For example, comparing leaf shape ratio, smoothness etc.

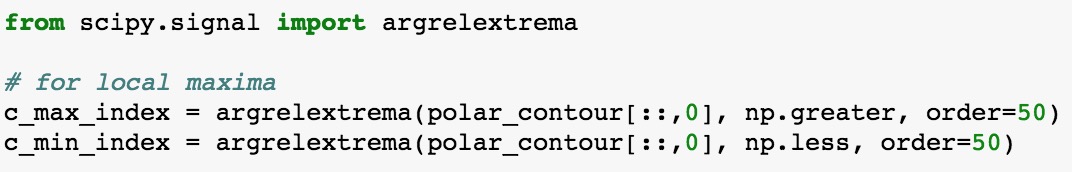
**Future Plan**

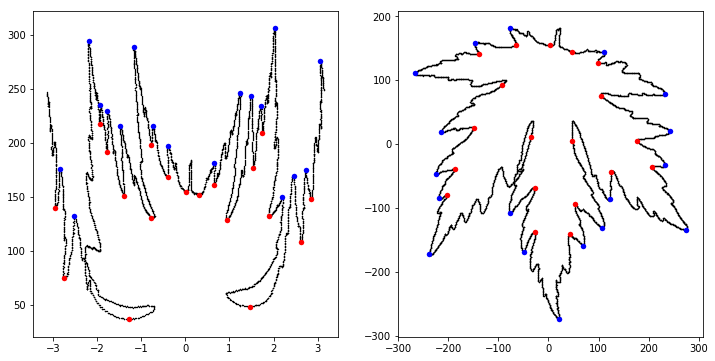
* **Use Polar coordinates instead of Cartesian coordinates**

****

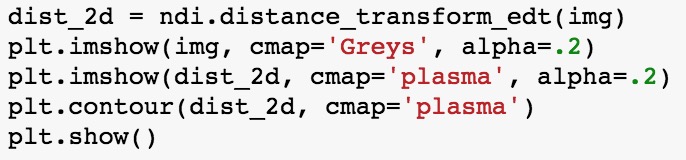
****

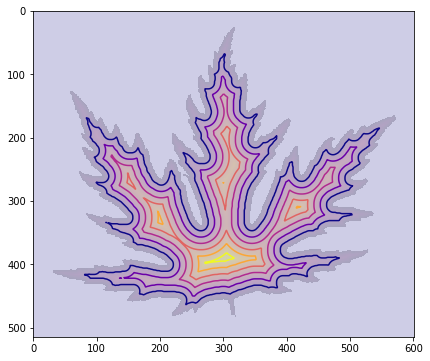
* **Add local maximum, local minimum features**

****

****

* **Use heat map**

****

****