NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering

CZ4041: Machine Learning

Project report: Leaf Classification

Group 18

CAO Liu (U1622132B)

HENG Weiliang (U1620593L)

LI Shanlan (U1622186B)

MA Yuqian (U1622117B)

Content

1. Introduction

CZ4041 - Machine Learning project aims to equip students with a bunch of machine learning techniques in real life application. The main scope of this project is to explore, analyze, and discuss the machine learning methods used by our team in Kaggle Data Science Competition - Leaf Classification.

1.1 Team Members & Roles

The team consists of four students, all of us highly contributed to our project and discussions:

- CAO Liu (Data preprocessing, experimenting on classifiers ,ensemble, ANN)
- HENG Weiliang (Feature engineering, feature extraction, CNN, final model tuning)
- LI Shanlan (Experimenting on ensemble, CNN, final model tuning, data augmentation)
- MA Yuqian (Experimenting on classifiers, collecting information, writing report)

1.2 Problem Statement

1.2.1 Challenge

There are estimated to be nearly half a million species of plant in the world. Classification of species has been historically problematic and often results in duplicate identifications. Automating plant recognition might have many applications, including:

- Species population tracking and preservation
- Plant-based medicinal research
- Crop and food supply management

The objective of this playground competition is to use binary leaf images and extracted features, including shape, margin & texture, to accurately identify 99 species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species. They also provide a fun introduction to applying techniques that involve image-based features.

As a first step, try building a classifier that uses the provided pre-extracted features. Next, try creating a set of new features. Finally, examine the errors and see what to improve.

1.2.2 Dataset

The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species) which have been converted to binary black leaves against white backgrounds. Three sets of features are also provided per image: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample.

Note that of the original 100 species, we have eliminated one on account of incomplete associated data in the original dataset.

Dataset we used:

- train.csv - the training set
- test.csv - the test set
- images - the image files (each image is named with its corresponding id)

1.3 Evaluation Score

We have attempted to solve this problem with a few models, including K Nearest Neighbours, SVM, random forest, MLP, ANN. Out of all these models, ANN with feature engineering triumphes. The table below shows the score and ranking of our project.

	Score	Ranking
Public	0.01018	117/1598 (top 7.3%)

Table 1: Performance of our best model

2. Data Preprocessing

For the data from Kaggle website, we did some preprocessing with the raw data.

2.1 Preprocessing Methods

2.1.1 Standardization

Standardization will rescale the features such that they will have the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 0$ where μ will be the mean and σ will be the standard deviation of the features. All the resulting features will be calculated as

$$z = \frac{x-\mu}{\sigma}$$

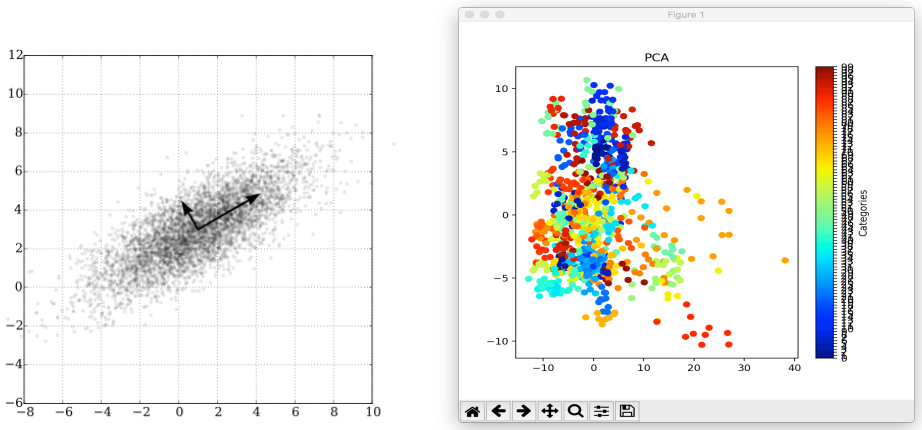
2.1.2 Min-Max scaling

Min-Max scaling will rescale the features such that they all have a fixed range of 0 to 1. Since, we scale the features to a fixed range, they will have a relatively smaller standard deviation comparing to standardization.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.2 PCA (Principal component analysis)

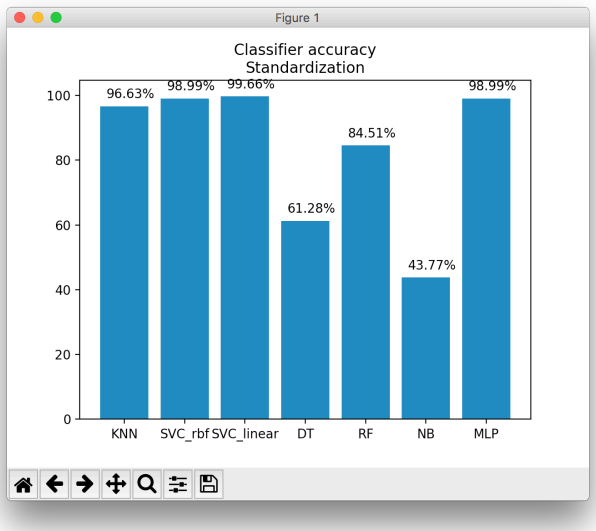
PCA convert a set of features of possibly correlated variables into a set of values of linearly uncorrelated variable called principal components. In the Leaf Classification problems, we have 192 features for each training data. It is natural to apply PCA to do dimensionality reduction to help eliminate irrelevant features and reduce noise, and thus improve the performance of the classifier.

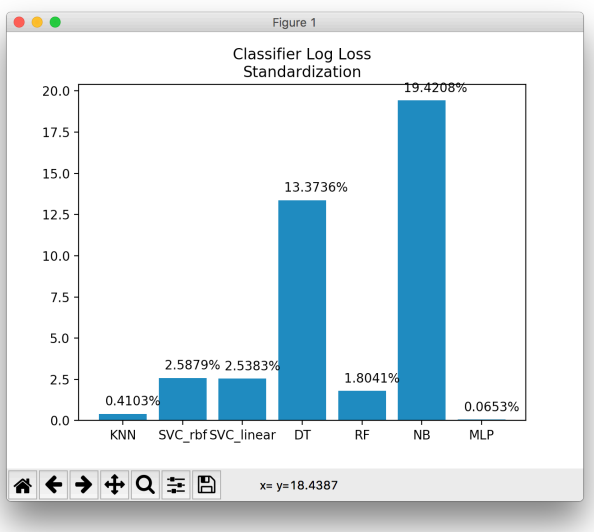


We first apply PCA on the whole dataset to reduce the feature dimension to 2. Based on the 2 features we can plot a picture of the whole dataset. The picture of the dataset provides us some hits that leaves belonging to the same categorial may be close to each other.

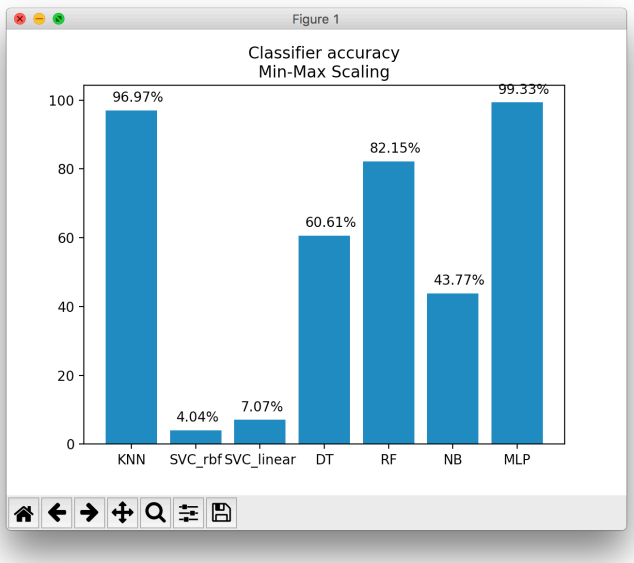
2.3 Preprocessing Method Comparison

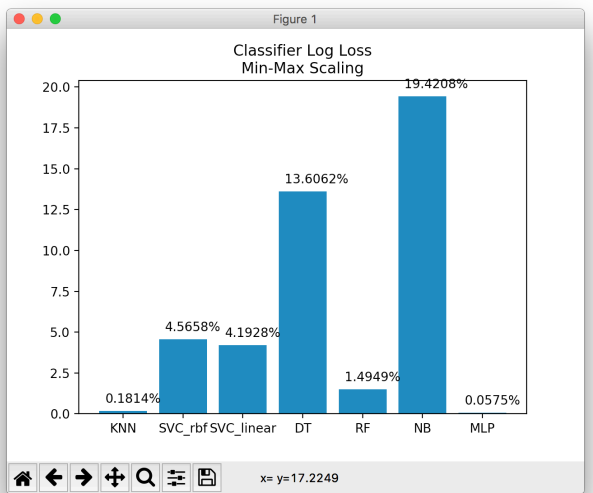
We test the impact of different preprocessing methods on a set of different classifiers. The numbers of principal components of PCA is 80. We plot one graph of the accuracy of the classifiers, and another graph of the log loss of the classifiers. Based on the graph, standardization on the dataset makes our classifiers have an overall higher accuracy and lower log loss. We will choose standardization as the preprocessing method.



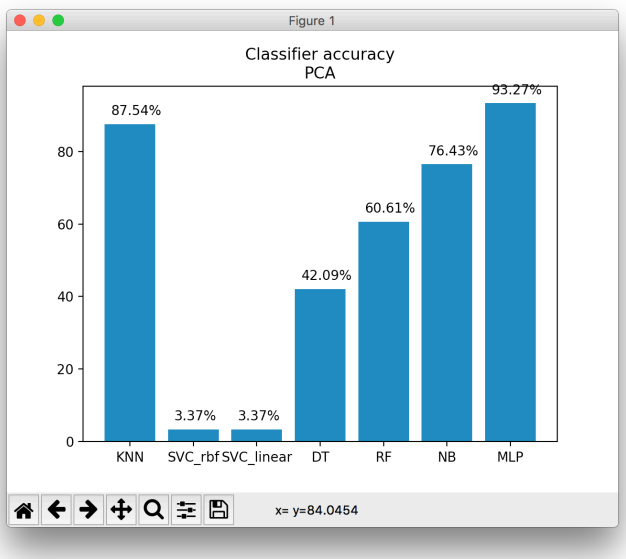


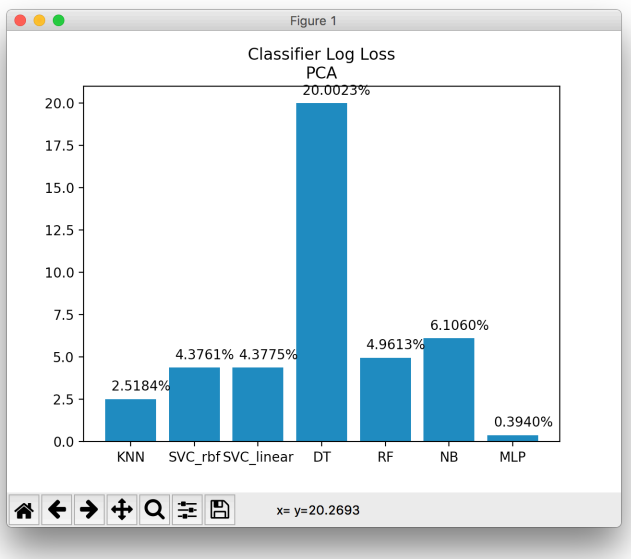
Accuracy and Log Loss by Standardization





Accuracy and Log Loss by Min-Max Scaling





Accuracy and Log Loss by PCA(n_components=80)

2.4 Feature Engineering

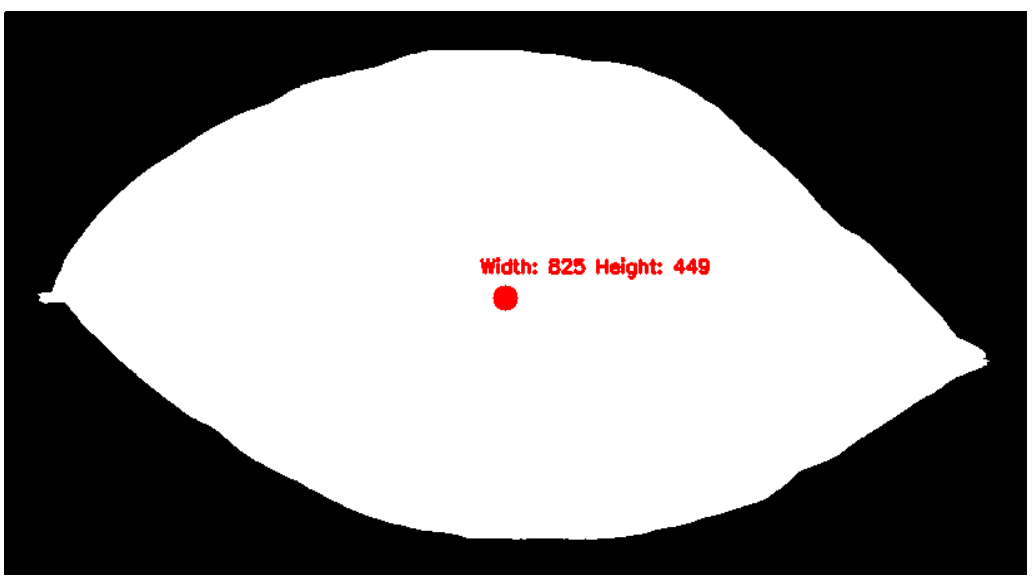
A part from the csv file provided, images files are also provided to us. There are many more features can be extracted from images which can be used to improve the accuracy of our model. We start with the physical properties of the images and moved on to features that are not so obvious.

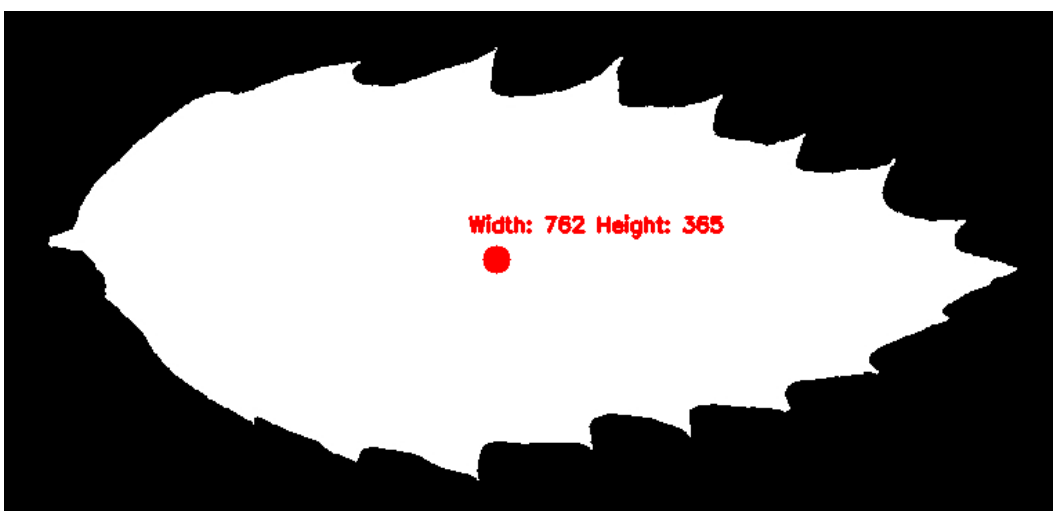
Physical Properties

Physical properties such as width and height are different across classes. The general shape of an images can also be used to classify leaf. Rectangular shape and square shape image generally classify leaf into different classes.

Width and Height

From the images, we can immediate classify them into some classes less than 99 by simply looking at the orientation of the images.



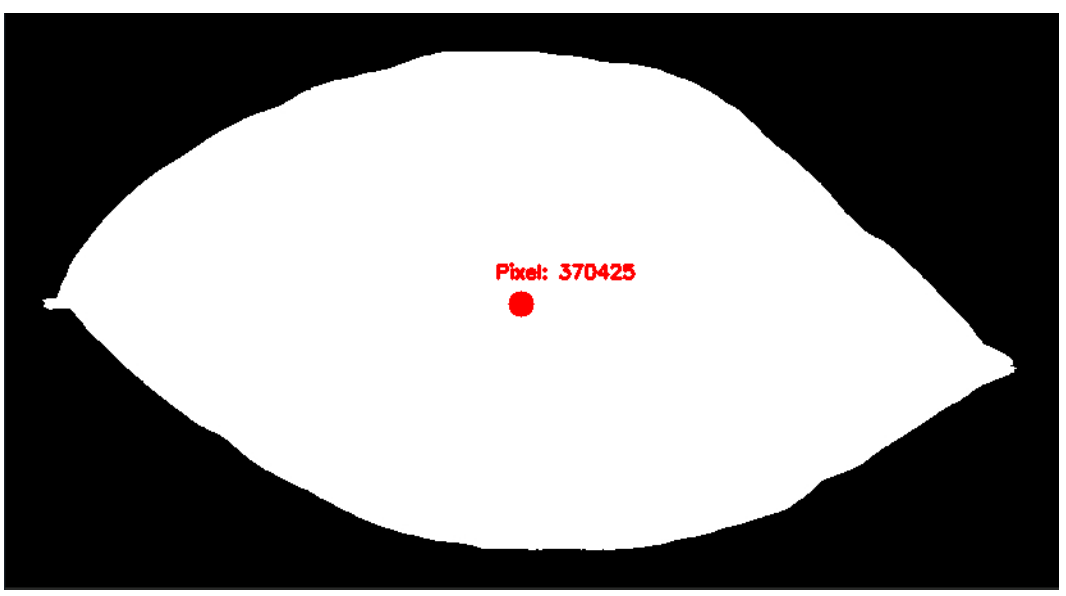


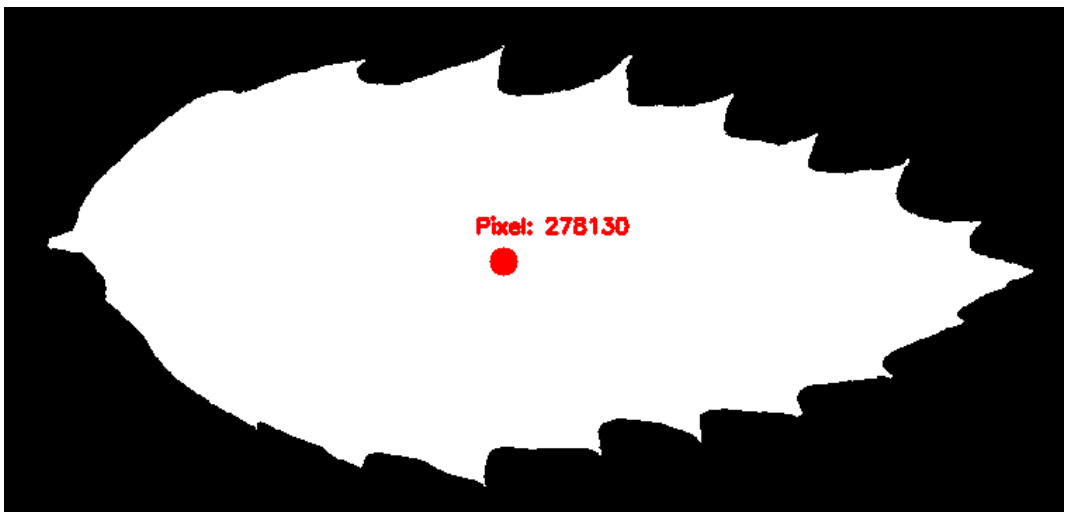
Since we do not know that is the features provided, we assume that this piece of information is not provided in the csv.

```
from PIL import Image
image = Image.open('./images/{}.jpg'.format(i))
w, h = image.size
```

Pixel

The number of pixel in an images is define as the width times the height sometimes also known as the resolution. The width and height of images can be used to classify images into different classes however with this piece of information alone. The classification is definitely not accurate. To increase the accuracy, We take the total pixel of an image. Which acts as an extra piece of information.



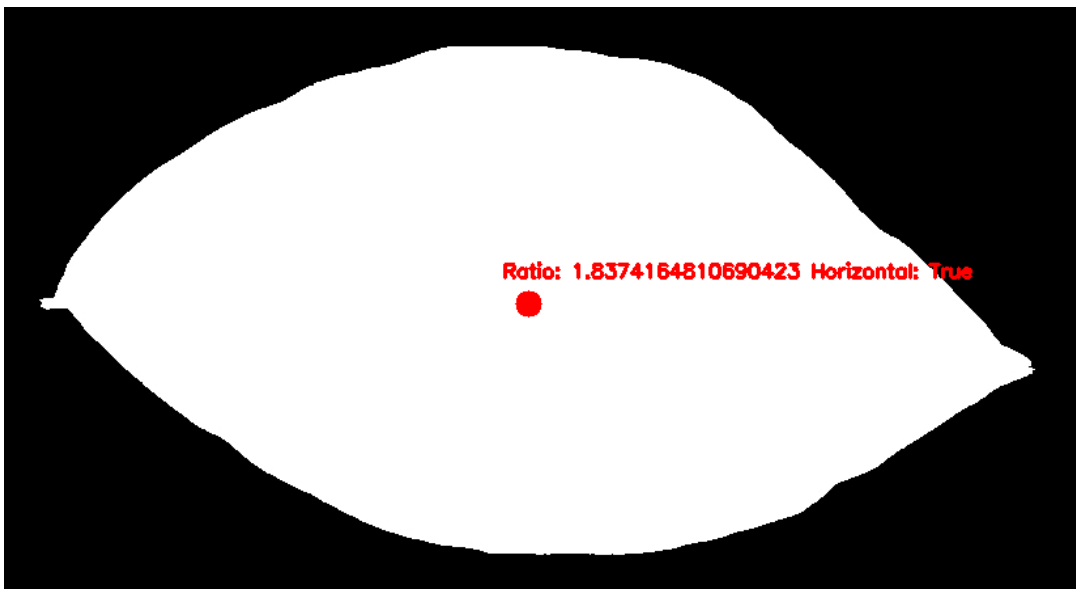


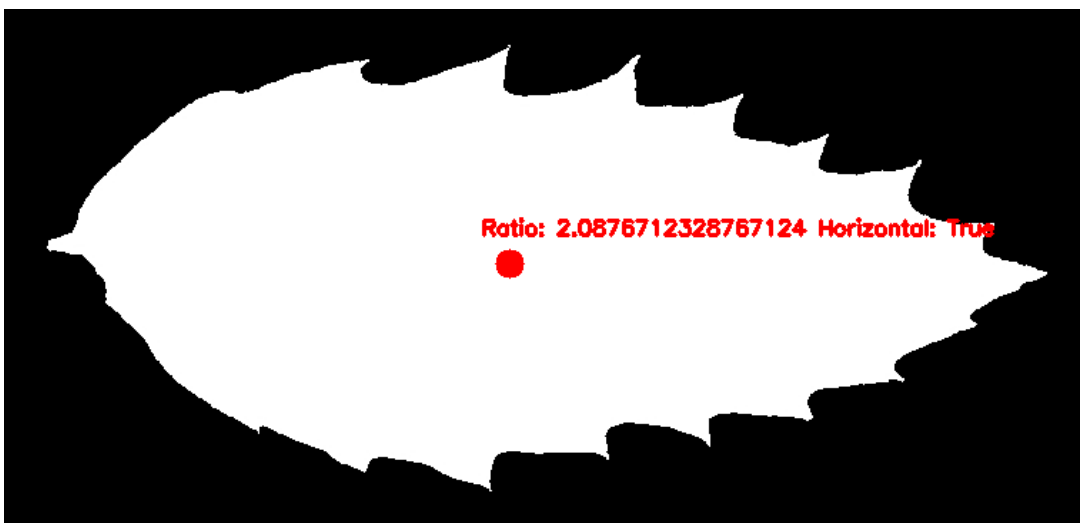
Both image's width and height are very similar however once we look at the number of pixel. The differences are huge.

Ratio and Horizontal

There are however some problem with number of pixels in an image. That is by a little difference in width and height the difference in number of pixel are huge. Hence, we go on by looking at the

width over height ratio and test the orientation of the image that is whether an image is horizontal or not.



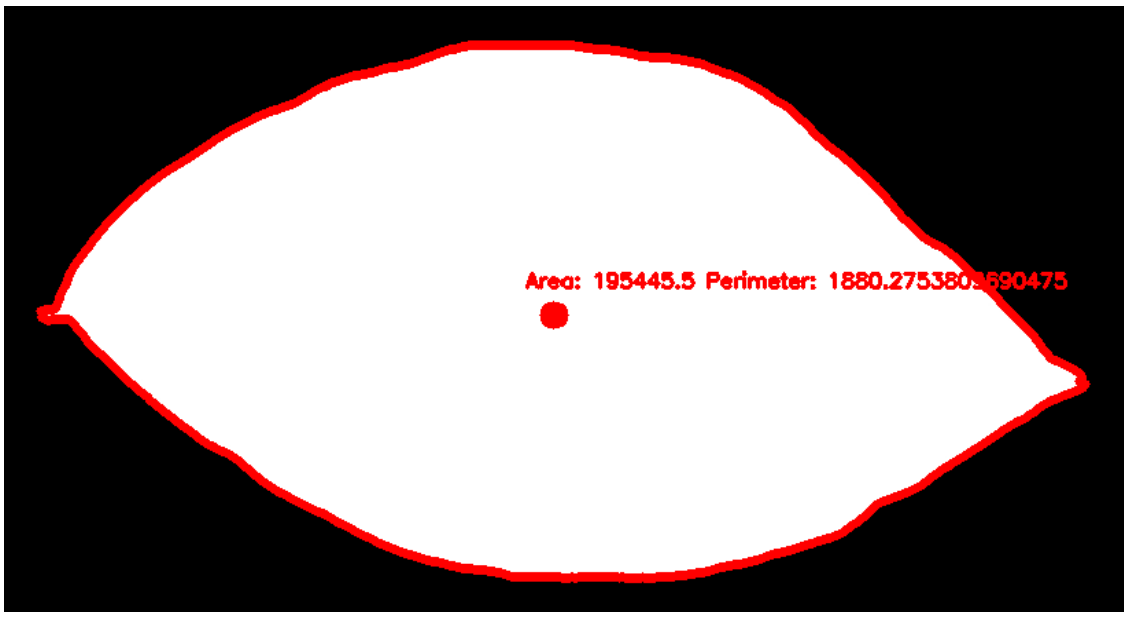


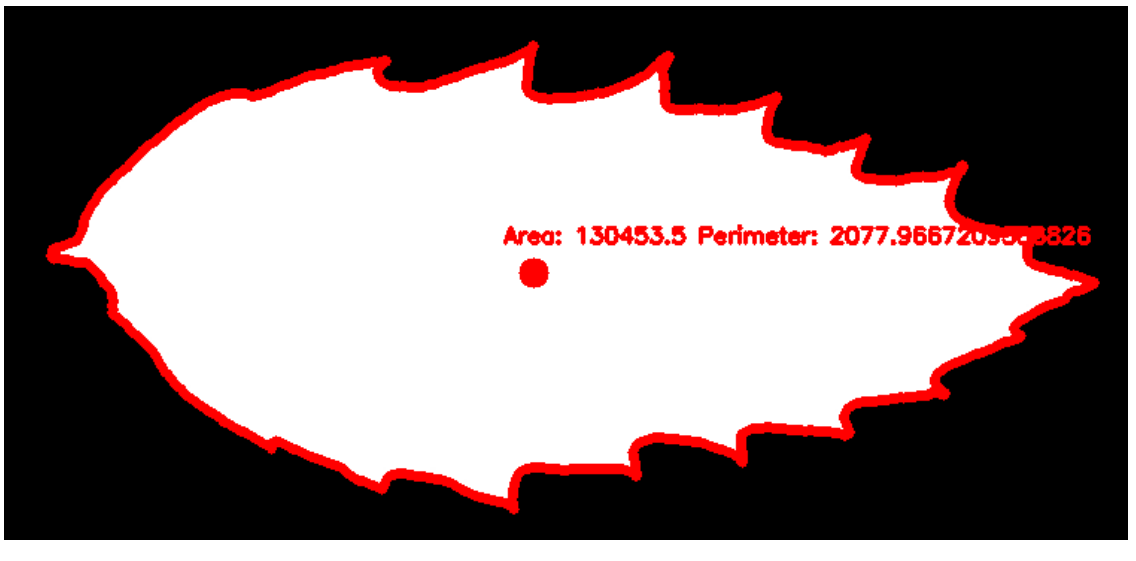
The width over height ratio tell us to some degree how different the image is within a similar image. The horizontal property give an definite answer between at least two classes of leaf.

Non-physical Properties

Area and Perimeter

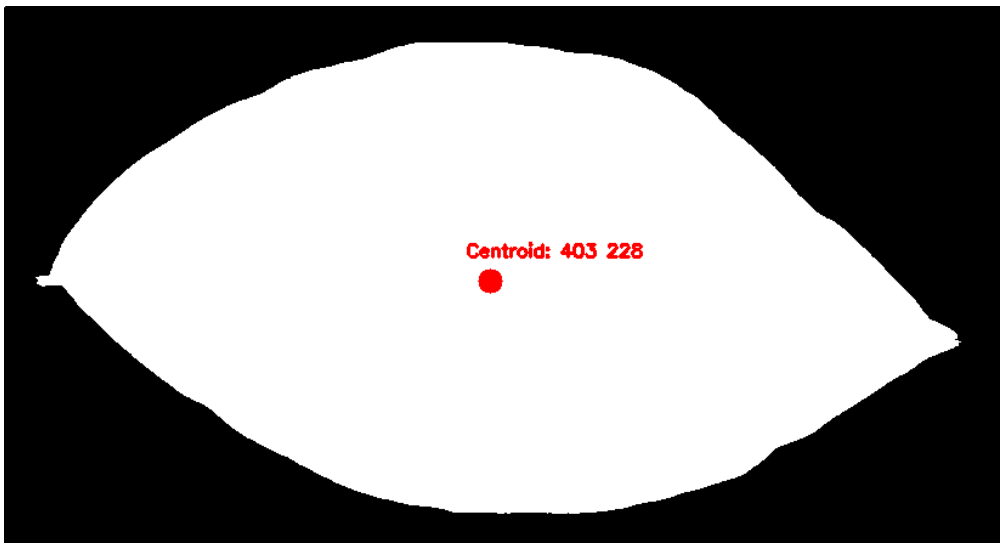
Similar to pixel and ratio, area and perimeter give extra information of and image this piece of information help us to ‘fine tune’ our feature sets. Unlike width and height which is determine by the physical property of and image. Area and perimeter is actually determine by the outline of the leaf it self. Given two image with opposite width and height, area and perimeter can be used to make the difference. For example, image 1 have width equal to 400 and height equal to 500. Image 2 have width equal to 400 and height equal to 500. Their width, height and pixel will be the same but not area and perimeter.

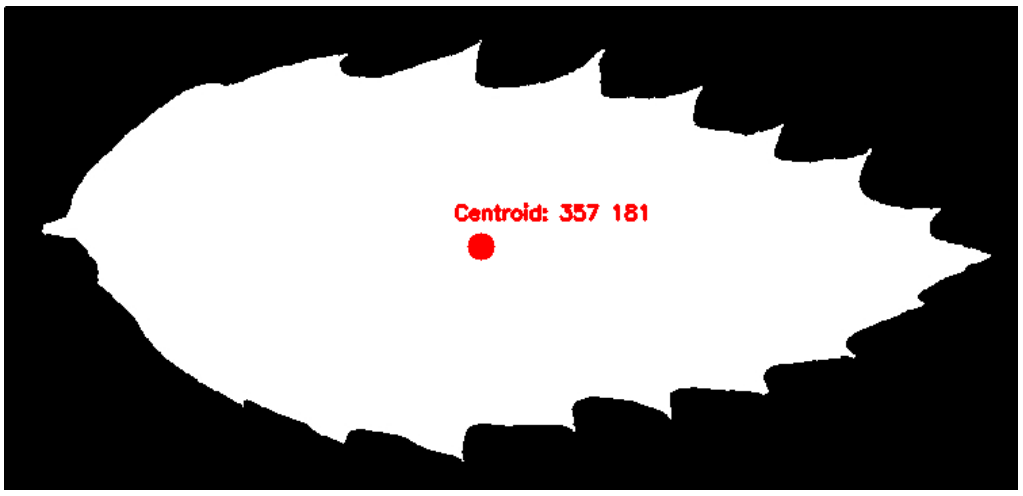




Centroid

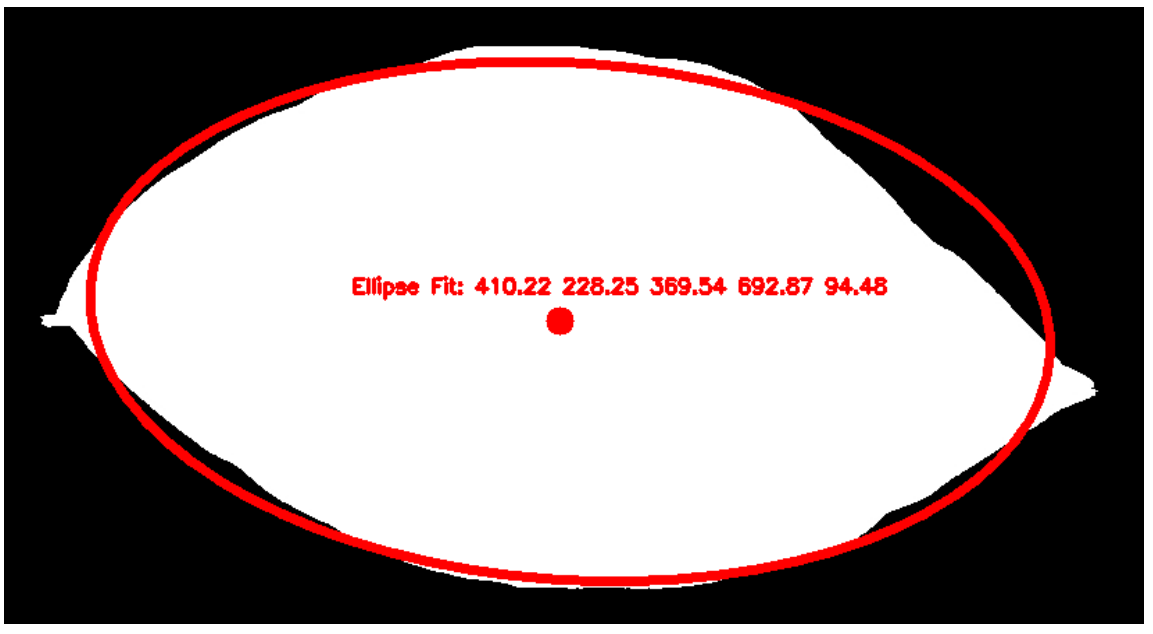
Another piece of information is the centroid point of image. Browsing through the images, we can see that the leaf filled up most the the horizontal and vertical space of the place. The centroid point of the leaf are quite different across images. In fact, with only centroid point added to our feature set, the accuracy improve.





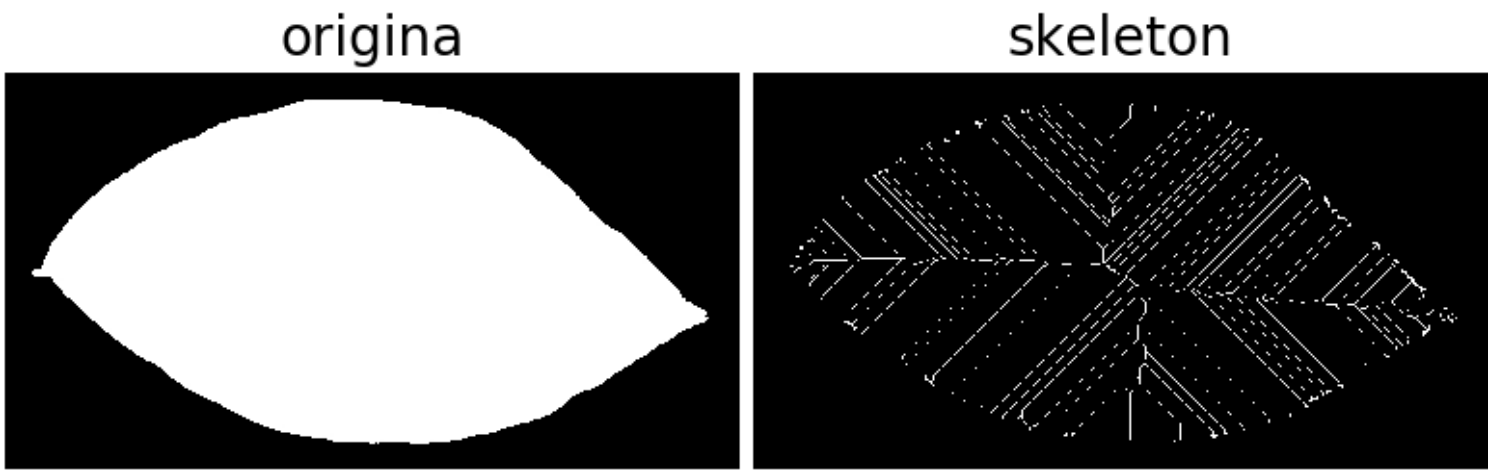
Ellipse Fit

The best ellipse fit is another piece of important information. It give us the coordinate a rotated rectangle in which a ellipse is inscribed. This various from image to image and of course is very different between different type of leaf. Together with the centroid point of an leaf. It help us to improve our accuracy.



Skeletonize

The last piece of information we tried to add to our feature set without success is the veins and vessels of the leaf. We start of thinking that if we can accurately count the number of veins and vessels of the leaf. This can be a great feature use to improve the accuracy. However that are two major problem we had encountered. First, we are not able to flatten the information since this piece of information is given to use in various dimension. Second, by taking a closer look, it actually lose the information on the sharp edges of the leaf. As of this writing, I am still trying to flatten the information.



3. Classifiers Exploration

3.1 K-Nearest Neighbours Classifier

3.1.1 Background

the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification. The input consists of the *k* closest training examples in the feature space.The output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). An example is shown below

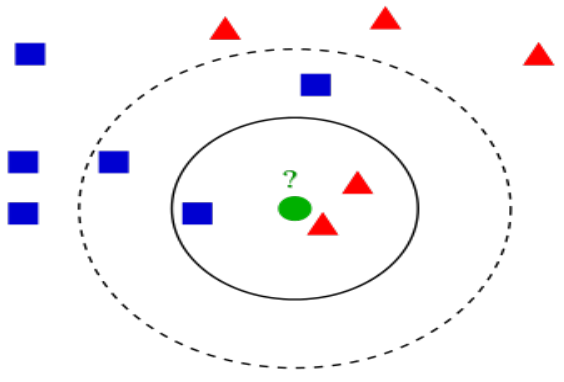


Figure . Example of KNN

3.1.2 Implementation

Based on the observation of the picture produced by PCA, it is natural to apply k-NN to do the classification. We implement k-NN by choosing k with the value 3 which has the best performance.

3.2 SVM (Support Vector Machine)

3.2.1 Background

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An example is shown below.

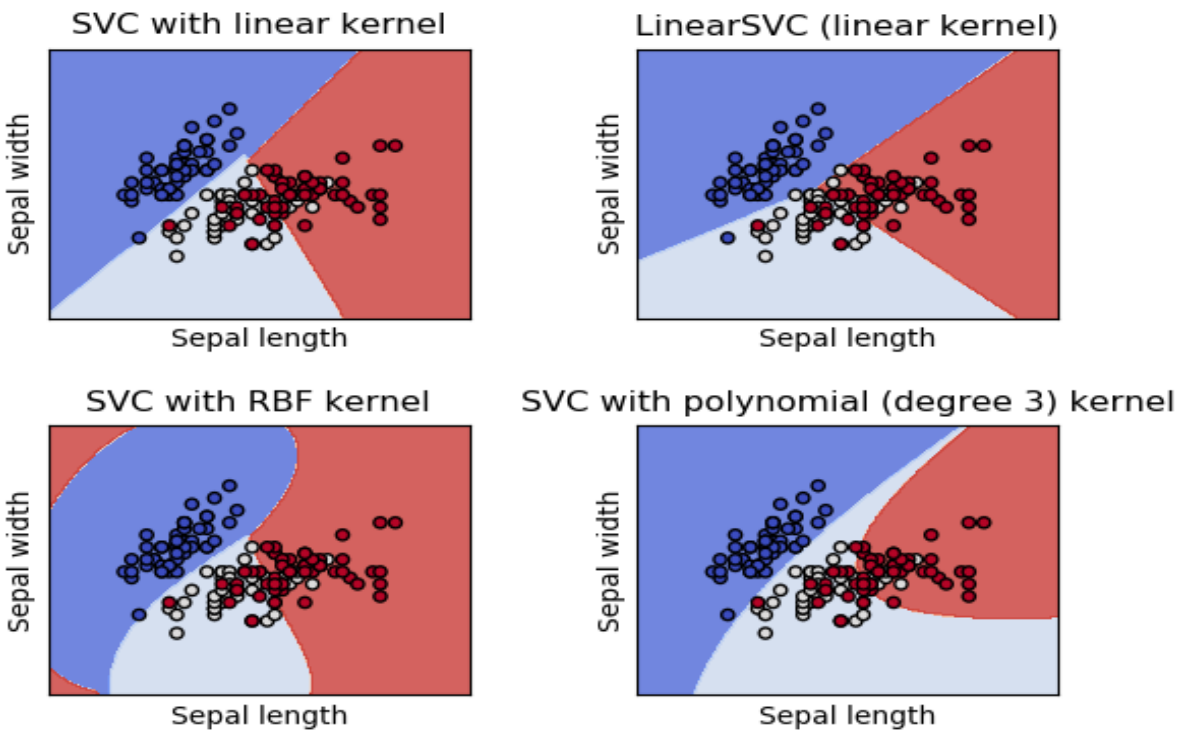


Figure . Example of SVC

3.2.2 Implementation

We implement two versions of Support Vector Machine classifier. One with an RBF kernel and the other is a linear Support Vector Machine. Based on the test on this two types of SVM, linear Support Vector Machine has a better performance.

(kernel="rbf", C=1, probability=True)

(kernel="linear", C=0.025, probability=True)

3.3 Random Forest Classifier

3.3.1 Background

Random Forest Classifier is a machine learning algorithm that constructs a number of decision trees and generates the mean prediction of each tree. The result of each tree will then be averaged over the number of trees. This averaged result is the final prediction of the classifier. An example is shown below.

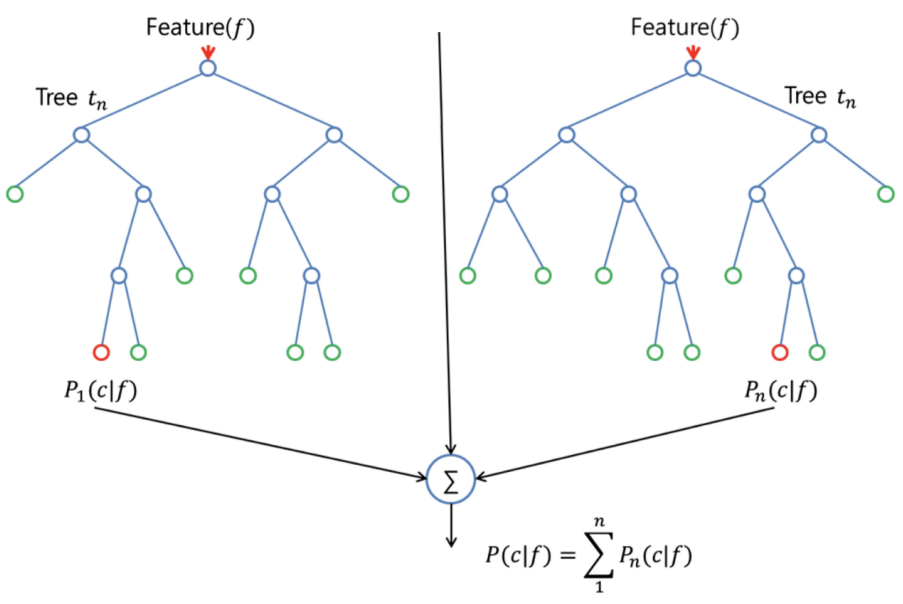


Figure . Example of Random Forest

3.3.2 Implementation

(max_depth=100)

3.4 MLP Classifier (Multilayer Perceptron)

3.4.1 Background

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.An example is shown below.

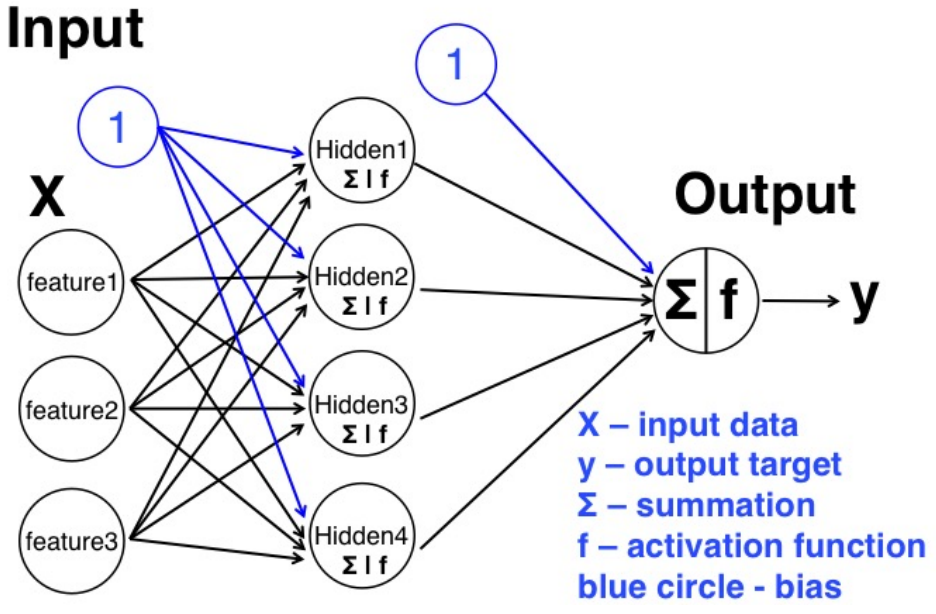


Figure . Example of MLP

3.4.2 Implementation

Here we implement a simplified version of MLP. The model uses ReLU function as the activation function and Adam as the optimizer.

```
MLPClassifier(solver='adam', activation="relu", alpha=1e-5,
        learning_rate="invscaling", hidden_layer_sizes=(512, ))}
```

3.5 Ensemble Model

Finally, we implemented the combine different conceptually different machine learning classifiers and use a soft voting technique to predict class labels. Based on the experiment we tested before, we choose the K-nearest neighbor classifier, linear Support Vector Machine, MLP and Random forest classifier with weights 2, 2, 2.5, 0.8 to form the soft VotingClassifier. We achieve an accuracy of 0.99 and a loss of 0.56 on our validation dataset.

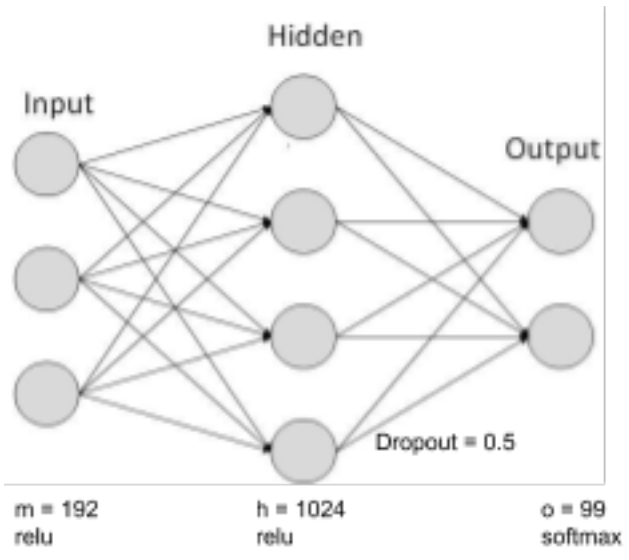
```
models = [
```

```
      ("KNN", KNeighborsClassifier(3)),
      ("SVC_linear", SVC(kernel="linear", C=0.025, probability=True)),
      ("MLP", MLPClassifier(solver='adam'activation="relu",alpha=1e-5,learning_rate="invscaling",
hidden_layer_sizes=(512, ))),
      ("RF", RandomForestClassifier(max_depth=100))
   ]

   eclf = VotingClassifier(
      estimators=models,
      voting='soft', weights=[2,1,2.5,0.8])
```

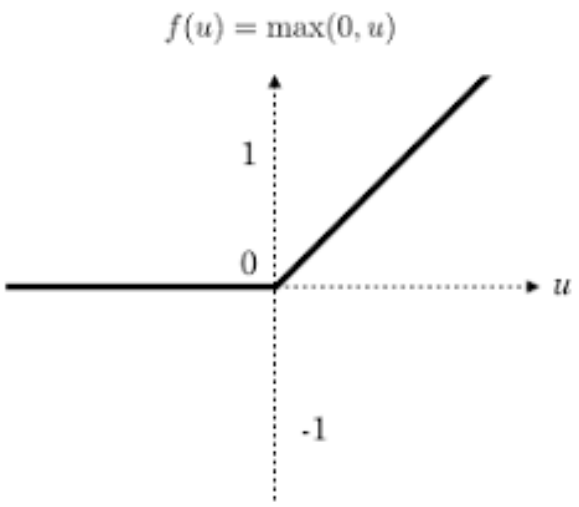
4. Neural Networks

4.1 ANN (Artificial Neural Networks)



After trying different numbers of hidden layers, the neural network architecture with one hidden layer has the best results. The hidden layer consists of 1024 nodes and have the ReLU function as the activation function. The ReLU function has an output equals 0 if the input is 0, otherwise the output equals input. Research has shown that applying ReLU function as the activation function results in much faster training for large dataset.

$$f(u) = max(0, u)$$



The output layer has 99 nodes which is the number of the different classes we want to predict and the softmax function is applied as the activation function. The softmax function squashes the output of each unit between 0 and 1, and its output is a categorical probability distribution.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

The categorical cross entropy function is chosen to be the loss function of our model, as in this Kaggle competition, our goal is to minimize the loss. After forward passing of the data, the model uses backpropagation to optimize the weight and the bias of each node. RMSprop is chosen to be the optimizer of out model. It divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weights, which helps speed up the process of backpropagation.

$$H(y) = \sum_i y_i \log \frac{1}{y_i} = -\sum_i y_i \log y_i$$

After training our model with the batch size equals 32 and epochs equals 150. Our model have a loss of 0.028 on the test dataset.

```
Epoch 150/150
693/693 [==============================] - 0s 273us/step - loss: 1.6419e-07 - acc: 1.0000
```

4.2 CNN (Convolutional Neural Networks)

To make more use of the information hidden in the image dataset given, we further explored convolutional neural network. CNN is a class of deep, feed-forward artificial neural networks that

has successfully been applied to analyzing visual imagery. We make use of “2D Convolutional layer”, “Maxpooling layer”, “Dropout layer” and “Flatten layer” to stack up a CNN.

Convolutional layer:

In Convolutional layer, matrices called “filter” or “kernel” or “feature detector” are trained, which slide over the matrix from upper layer, compute the dot product and form another matrix as a result. Different features will be detected by different filters, which are determined by the whole algorithm of backpropagation. Here we applied filters with size of (3,3) because the size of our image inputs is only (96,96) and differences between classes are very small in our dataset. It is aimed to sense very tiny but important features.

Maxpooling layer:

Maxpooling layer takes the max of every small square area in a matrix and forms a new matrix. Here we defined the moving area to be of size (2,2) to shrink the matrices we gained and filters the most significant features out of that.

Dropout layer:

Dropout layer is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. Technically it changes some of the weights between nodes to 0. For every 2D Convolutional layer, a Maxpooling layer and a Dropout layer is also combined to form a “block” which represents a layer of feature extraction.

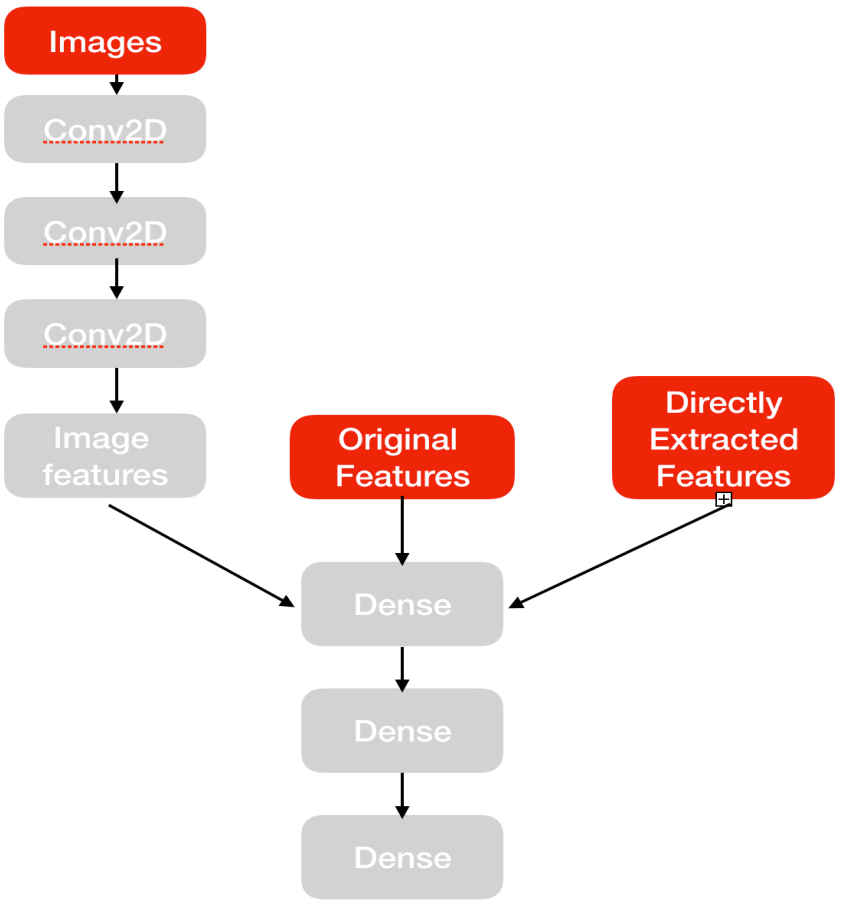
In all, we stack three sets of Convolutional layer, Maxpooling layer and Dropout layer because our training examples are very limited (16 examples per class). The model is built simple to prevent overfitting.

Flatten layer:

The features we extracted by layers above is a three dimensional matrix, the first two dimensions come from image, the last dimension is the number of filters applied. To combine the features we extracted from CNN with other features we obtained, Flattern layer is applied to reduce the dimension to the same as other features.

5. Final Model

1. Diagram display:



2. Model summary:

Layer (type)	Output Shape	Param #	Connected to
image (InputLayer)	(None, 96, 96, 1)	0	
conv2d_7 (Conv2D)	(None, 96, 96, 16)	160	image[0][0]
max_pooling2d_7 (MaxPooling2D)	(None, 48, 48, 16)	0	conv2d_7[0][0]
dropout_11 (Dropout)	(None, 48, 48, 16)	0	max_pooling2d_7[0][0]
conv2d_8 (Conv2D)	(None, 48, 48, 32)	4640	dropout_11[0][0]
max_pooling2d_8 (MaxPooling2D)	(None, 24, 24, 32)	0	conv2d_8[0][0]
dropout_12 (Dropout)	(None, 24, 24, 32)	0	max_pooling2d_8[0][0]
conv2d_9 (Conv2D)	(None, 24, 24, 64)	18496	dropout_12[0][0]
max_pooling2d_9 (MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_9[0][0]
dropout_13 (Dropout)	(None, 12, 12, 64)	0	max_pooling2d_9[0][0]
flatten_3 (Flatten)	(None, 9216)	0	dropout_13[0][0]
numerical (InputLayer)	(None, 201)	0	
concatenate_3 (Concatenate)	(None, 9417)	0	flatten_3[0][0] numerical[0][0]
dense_7 (Dense)	(None, 1024)	9644032	concatenate_3[0][0]
dropout_14 (Dropout)	(None, 1024)	0	dense_7[0][0]
dense_8 (Dense)	(None, 512)	524800	dropout_14[0][0]
dropout_15 (Dropout)	(None, 512)	0	dense_8[0][0]
dense_9 (Dense)	(None, 99)	50787	dropout_15[0][0]

Total params: 10,242,915
Trainable params: 10,242,915
Non-trainable params: 0

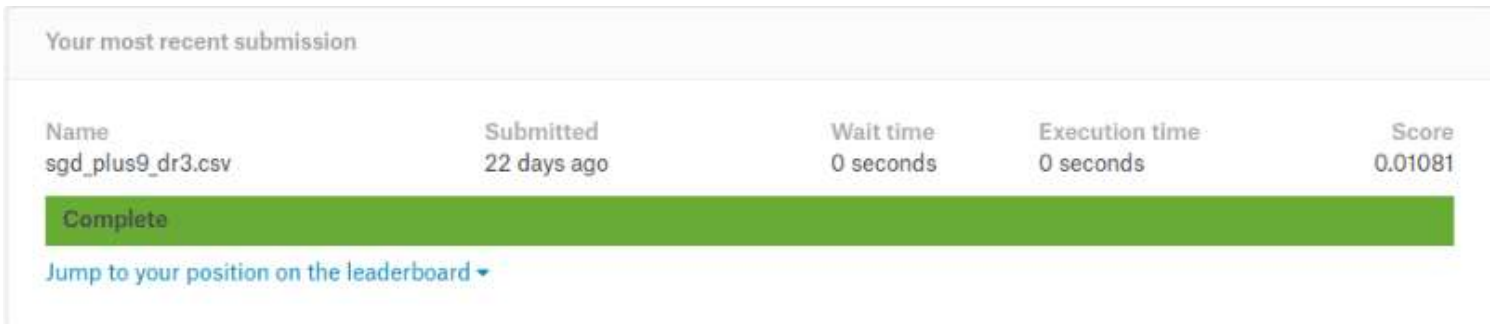
As a result of extensive experiments of model designing and parameter tuning, this model is chosen as our final model to use. We combine all the features we extracted as the feature inputs, feed them to the three-layer ANN and obtain the final softmax output. We discard the ensemble of KNN, SVM, RF and MLP approach described above, because this model gives us a far much better score than the traditional classifiers' ensemble (loss of less than 0.001 vs loss of 0.56, accuracy of 1 vs accuracy of 0.99 on validation set). The information obtained from the ensemble is not strong enough to compete with the information gained from this model.

We choose "SGD(lr=0.05, momentum=0.8, decay=0.0)" as the optimizer of the final model. The reason we choose Stochastic Gradient Descent is that it performs update for each trading example. Due to frequent updates, parameter updates have high variance and causes the loss function fluctuate to different intensities. It helps to jump out of a local minima and reach to the global one. Momentum was involved which accelerates SGD by navigating along the relevant direction and softens the oscillations in irrelevant directions.

Lastly, the reason we choose a combined model instead of a ensemble of separate Neural Network models is that the results gained separately from training CNN on images and training ANN on physically extracted features are not satisfying (0.04101), worse than training on original features only (0.028). The result gets better only when the additional features are assembled into the model as helping ingredient.

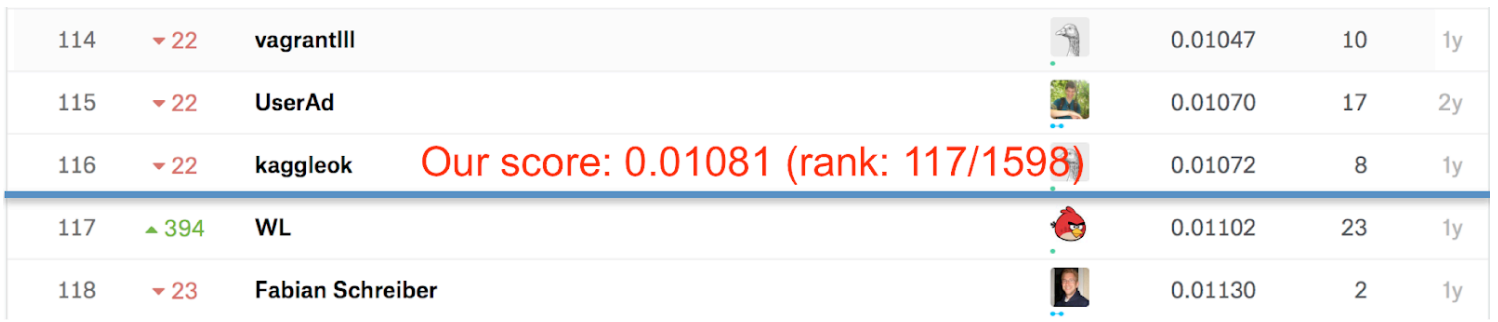
6. Result and Analysis

Score on Kaggle:



Position on leaderboard : (First 7.3 %)

(The competition is closed so we cannot place ourselves on the leaderboard)



This is result is satisfying but not perfect. We tried to threshold the entries with more than 0.95 confidence in the output csv file, change their confidences to 1, but that results in worse score, which indicates that our model is giving a wrong class a very high confidence in some of the test data. This may come from a dirty training data. In this case it is most likely that the training data are quite similar in some classes and our extracted features cannot differentiate them. This problem might be solved by more sophisticated feature extraction which, in spite of various experiments, we were not able to find out.

We also tried data augmentation by applying width_shift, height_shift, zoom, horizontal_flip, vertical_flip in different combination, none of them gives a better result. We think the reasons may be that the differences between some classes are actually tiny, the noise we put on cannot magnify those tiny differences but confuse them sometimes.

Another thing to note is that, Neural Networks perform better if given more training data. With only 1584 training data, we are not able to build the neural network deeper and stronger. But compared to other traditional machine learning method, this is already the one that performs the best.

7. Conclusion

For this Kaggle's competition -- leaf classification, the dataset is quite clean and well-organized. One can easily gain a pretty decent validation accuracy by applying any simple machine learning approach. But to lower the loss as low as possible, Neural Network is the most powerful and simple way. Feature extraction and feature engineering are also used in our approach to sharpen our model to a higher level. During this project, we have also explored much more machine learning classifiers and techniques, like KNN, SVM, Random forest, MLP, ensembling and data augmentation, though some of them do not performs well for this case. A more powerful feature extractor and more training data may help to give

the classification problem a better result. Above all, we achieved top 7.3% ranking on the public leaderboard with a loss of 0.01018.