Leaf Classification using Machine Learning

Group 3: Chen Ruo, Lechong Zhou, Ruoyu Xiang

Problem Description

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 project 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.

Datasets

The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species in total, 10 samples each of 99 species are labeled, the other 6 samples are used to evaluate the model via Kaggle), 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.

The train dataset has the shape of [990x194]. 990 represents the number of species, and 194 represents individual features in columns, i.e., 'id', 'species', 'margin 1~64', 'shape 1~64', 'texture 1~64'. More in detail for 194:

- id - an anonymous id unique to an image
- species - 99 species, not given in test.csv
- margin_1, margin_2, margin_3, ..., margin_64 - each of the 64 attribute vectors for the margin feature
- shape_1, shape_2, shape_3, ..., shape_64 - each of the 64 attribute vectors for the shape feature
- texture_1, texture_2, texture_3, ..., texture_64 - each of the 64 attribute vectors for the texture feature

Solutions

Python and *Scikit-learn* were deployed in this project. Scikit-learn provides us all kinds of powerful tools to solve this classification problem. What we need to do is pick a proper model and use the optimal parameters to maximize the performance of this classifier.

Cross Validation is a statistical method of evaluating generalization performance that is more stable and thorough than using a test set. In cross-validation, the data is instead split repeatedly and multiple models are trained. [Ref.Müller] The most commonly used cross-validation methods are k-fold cross-validation and leave-one-out.

Grid Search is the most commonly used method to find the values of the important parameters of a model. It basically means trying all possible combinations of the parameters of interest. [Ref. Müller]

Scikit-learn provides the *GridSearchCV* class to implement grid-search with cross-validation, which is a commonly used method to find the optimal parameters. The overview of *Grid-search* is shown below.

The pipeline of the machine learning process for this project is listed as follows:

- Preprocess the data
    - Split for cross-validation
    - Standard-scaler (optional)
    - Principal Component Analysis
- Feed the preprocessed data into the selected classifiers
    - Naive Bayes
    - Support Vector Machine (SVM)
    - Logistic Regression
    - k-nearest neighbours (k-NN)
    - Linear Discriminant Analysis (LDA)
    - Quadratic Discriminant Analysis (QDA)
    - Clustering (K-means and agglomerate)
- Evaluate the classifiers

Analysis and Evaluation

Principal Component Analysis

Principal Component Analysis (PCA) is the most important way of dimension reduction in our project. There are 192 features in our train set. Additionally, because of the characteristic of our dataset, the interpretability of the features is very low. PCA becomes the first choice in our project. For each model that PCA makes sense, we performed PCA to reduce the number of features. For some models, such as logistic regression, Naive Bayes and K-NN, PCA makes the

accuracy higher in classification. However, in the complicated models such as SVM, PCA did not effective improve the accuracy in the classification.

Since the number of principal components is very hard to decide intuitively, we performed PCA with different number of components, from 1 to 100. We find that the number between 25 to 45 is the range that highest accuracy always appear.

In the models that we have the best accuracy, such as LDA and SVM, PCA did not improve the classification accuracy effectively.

Naive Bayes

In the Naive Bayes model that without PCA performed, we got accuracy of only 45%. Then we performed PCA to the train data, and we got accuracy of 85%. The best accuracy appears when we set the number of principal components as 44. However, among all the models, Naive Bayes model has relatively low accuracy.

Support Vector Machine

To achieve the best accuracy for the SVM model, we have tried several parameters of kernel, gamma, and Regularization/penalty parameter. For kernel, we tried rbf, sigmoid, and linear. For gamma, we set gamma as [0.001, 0.1, 1, 10, 1000] for each kernel. We also tries regularization of [0.001, 0.1, 1, 10, 1000] for each kernel and gamma.

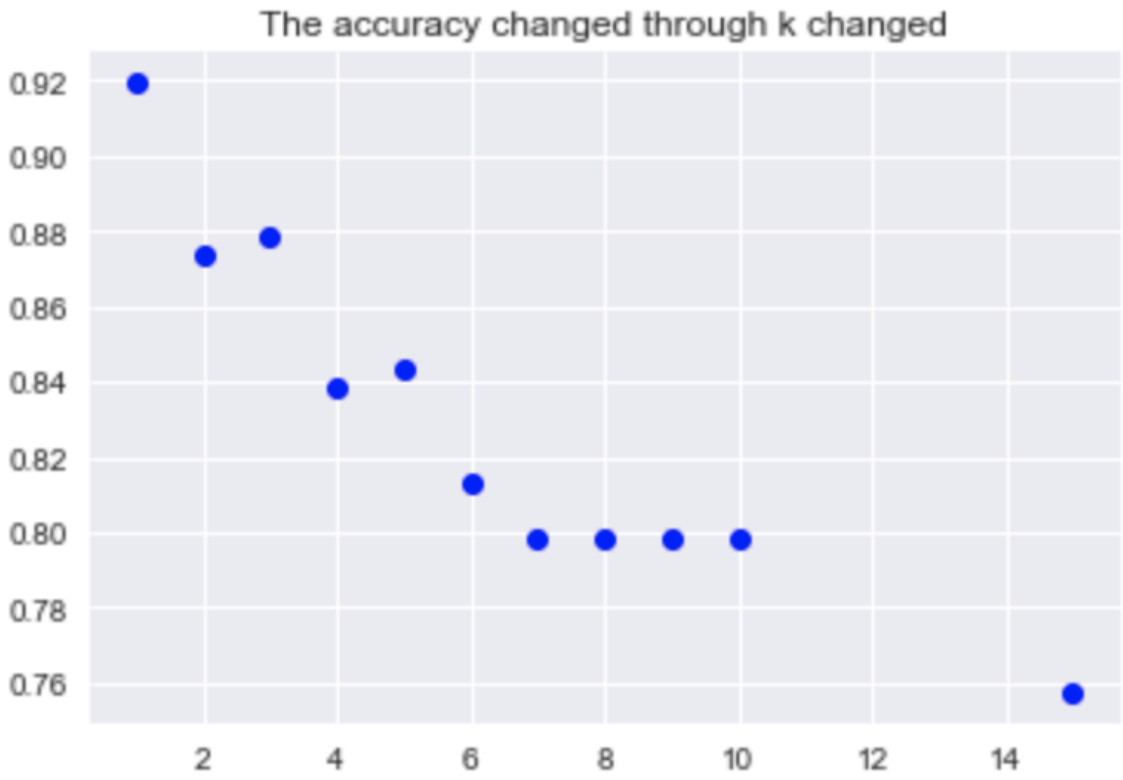
When we run SVM model with PCA performed, we find that PCA did not improve the accuracy of classification at all. Among all the kernels, parameter of gamma and regularization, we find that sigmoid kernel works the best for our data, which has the highest accuracy of 91%, with C = 1000 Gamma = 1.

Logistic Regression

We performed Logistic Regression with principal components and with the regularization of Lasso and Ridge. For each model, we tried Regularization Range of [0.1, 1, 10, 100, 1000]. As the result, we find that Logistic Regression that is performed with PCA and Ridge regularization(by default) has the best accuracy of 92%, with the number of principal components is 40 and the regularization strength is 1000.

k-NN

K-Nearest Neighbors algorithm is also one of the algorithms used in our study. In this algorithm, we tried different values of k and see how the accuracy changed through this process.



As we can see in this chart, the accruary is generally going down with increase of k. And also, we think when k=1 may cause overfitting, so we choose k = 3 as our KNN model value which the accuracy is 87.8788%.

Linear Discriminant Analysis

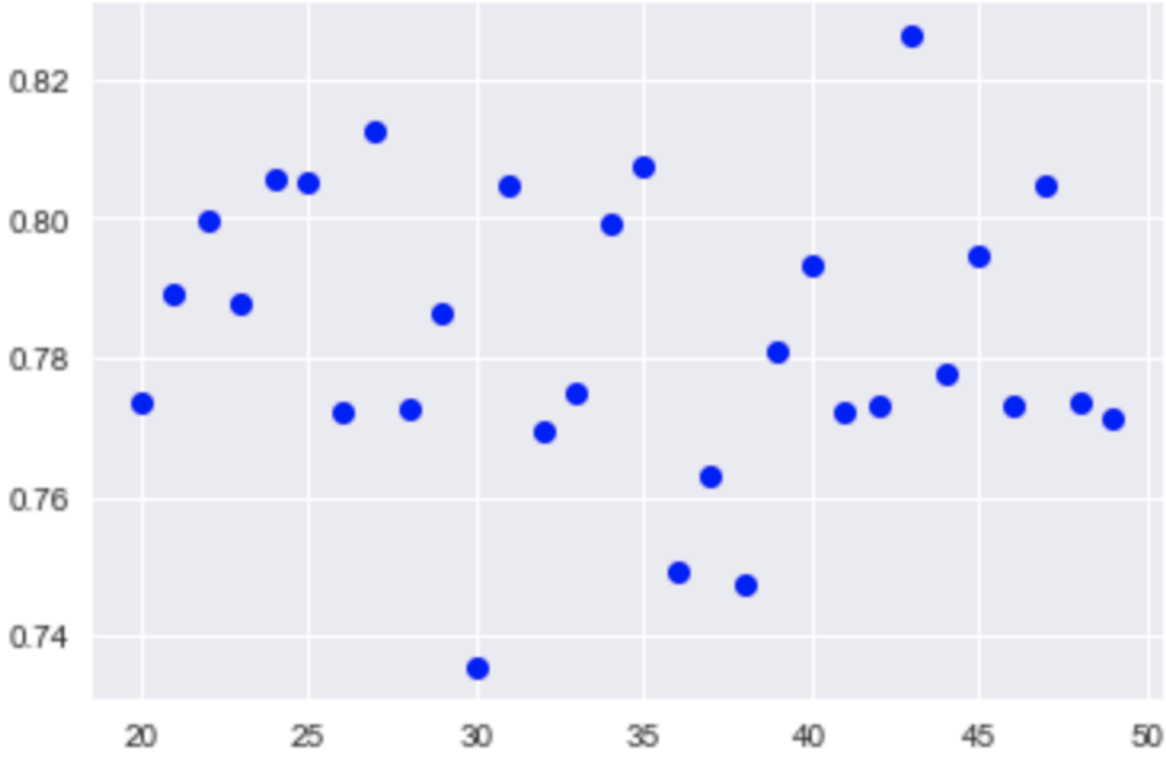
This classifier is attractive because it has closed-form solutions that can be easily computed, is inherently multiclass, have proven to work well in practice and have no hyperparameters to tune. For LDA, we first try to use it in combination with PCA, and set the number of components in the range of 1-100. Seems there is an overlap, since LDA has its own dimensionality reduction parameter. And then we tried to only use LDA, and found out that the best result keeps all of the dimensions. That is to say, the highest accuracy of 0.98 with the number of components equaling 99.

Quadratic Discriminant Analysis

Same underlying principle as LDA, but Quadratic Discriminant Analysis can learn quadratic boundaries and is therefore more flexible. However, the feature of the dataset determines that linear boundaries are more suitable here. We finally got a very low accuracy of 0.38.

Clustering
(K-means)

As we mentioned before, there are 192 features in our dataset and we decide to perform PCA method to reduce the numbers of feature.
One of disadvantages of K-means is that you have to decide clusters number before training the model, In this study, we already knew that there are 99 species of leaf. Therefore we just defined cluster number equal to 99. Then we try different PCA components number and looking for the best accuracy.

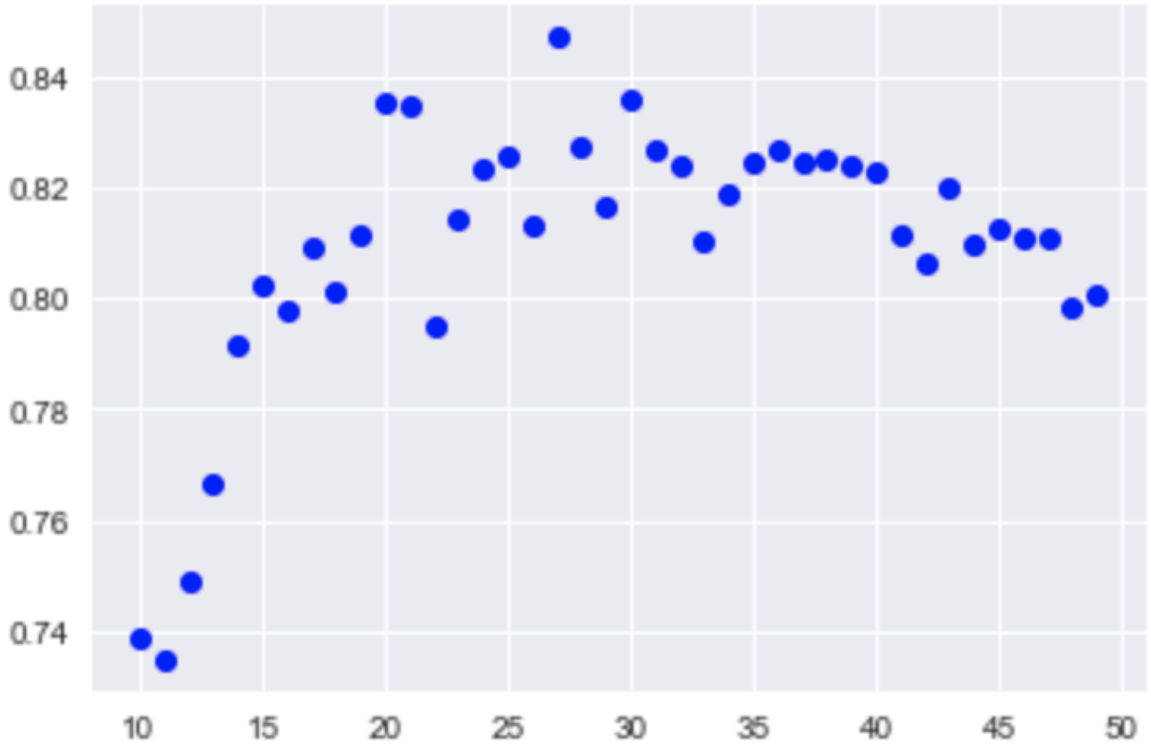


We chose the number of components from 20 to 50 which the result is little bit hard to interpreted. However, we can see that when components number between 20 and 30 the model performs generally good.

There is one node that we can't ignore is the 43, when components number equal to 43, the accuracy is surprisingly high. To explain this, I think this may cause by another disadvantage of K-means algorithm which is the performance of K-means depends on initialization. In our study, we set each K-means to run 30 times and choose best accuracy as the final accuracy. So, that surprisingly high accuracy may because of there is a initialization which is perfectly fit to the real situation.

Clustering
(Agglomerative)

We use the same method in Agglomerative algorithm. And we perform the PCA components number from 10 to 50. Also we set the clustering number as 99 which is the number of species.



From the chart we also can conclude that when components number between 20 and 40 the model shows good accuracy. And we find that when components number equal to 27 the model has the best accuracy which is 83.9%.

Results and Conclusion

Machine Learning Model	Accuracy
Gaussian NB	0.85
Support Vector Machine	0.91
Logistic Regression	0.92
k-NN	0.88
Linear Discriminant Analysis	0.98
Quadratic Discriminant Analysis	0.38
KMeans Clustering	0.81
Agglomerative Clustering	0.84

The results of the eight classifiers are listed in the above table. We can see that QDA didn't perform well in this classification task, since quadratic boundaries are not suitable for this dataset. The other seven classifiers have relatively good performance. Logistic Regression and Linear Discriminant Analysis both have very high predicting accuracy.

Future Works

- Keep optimizing the logistic regression and LDA classifier to improve their performance
- Try other machine learning model to tackle this problem
- Develop a pipeline to classify leaf from its raw colorful image

References

[1] Andreas C. Müller and Sarah Guido, Introduction to Machine Learning with Python. GitHub

[2] Mallah, Charles, James Cope, and James Orwell. "Plant leaf classification using probabilistic integration of shape, texture and margin features." Signal Processing, Pattern Recognition and Applications 5 (2013): 1.

[3] Mallah, CharlesD. "Probabilistic classification from a k-nearest-neighbour classifier." Computational Research 1.1 (2013): 1-9.