Plant Leaf Classification
Big Data Analytics

Kapil Sharma, Jinal Patel, Harsha Ramireddy, Alankrit Verma

***Abstract*—Leaf classification is a challenging task due to large number of different leaf species and there can be lots of similarities in features for some of those leaves making leaf identification a tough job. This project implements leaf classification using different models and then compared results and test accuracies among those classifiers. After comparing and analyzing their results, the group members find the good and bad models for leaf classification.**

I. Introduction

There are about one million species of plants in the world. Classification of leaf species has been problematic for many years now, especially when the same leaf species are counted and identified more than once, creating duplicates entries. Automating plant recognition helps biologists in tracking and preserving species population, research on plant based medicines and crop and food supply management. The leaf species are considered because of their volume, prevalence and unique characteristics that help differentiate leaves of one plant from another. These leaf species that form the dataset have been collected by James Cope, Thibaut Beghin, Paolo Remagnino and Sarah Barman of the Royal Botanic Gardens, Kew, UK.

Extracted features of binary leaf images to identify 99 species have been taken. These features include margin, shape and texture. The input consists of 193 attributes and a class which include ID, 64 margins, 64 shape and 64 texture. The output gives the accuracy of the classifier used. This performance metric is used to determine the best classifier amongst those used.

The goal of this project is to classify the species of plant by recognizing the features of the leaf using Machine Learning techniques.

II. Background and Related Works

Researchers tried to solve this problem over the last few years. The first reference we used is called “Plant Leaf Classification Using Probabilistic Integration of Shape, Texture and Margin Features” [4] written by Charles Mallah. This paper introduces a new data set of sixteen samples each of one hundred plant species and describes a method designed to work in conditions of small training set size. They processed each of three features in separate ways: they used histogram accumulation for margin and texture features and normalized description of contour for shape feature. For each feature using the K-NN algorithm, they generated a separate posterior probability vector and then combined posterior estimates to give the final classification. The best result they got is 96% mean accuracy when combining all three features. The result of their approach is very impressive considering how small their data set is. That is also what motivates us to use a relatively small data set for our project [4].

Another related work we referred is called “Plant discrimination by Support Vector Machine Classifier based on spectral reflectance” [5] written by Saman Akbarzadeh. They used SVM classifier which is proposed to classify broad leaf and narrow leaf plants. The strength of this model is high speed, while still achieving 97% accuracy which is improved using raw reflected intensities and kernel tricks [5]. There are some other references we used for our project but above two are our favorite because of their clear demonstration and high accuracy. Some of the other references used different approaches, for example naive bayes [6], which are attached in the reference section.

III. Dataset

The dataset consists of 1584 images of leaf specimen. These images have been converted to white colored leaves of binary format against a black background (Fig. 1). Each image has 3 sets of features which are shape contiguous descriptor for shape feature, interior texture histogram for texture feature and fine-scale margin histogram for margin feature. Each of these features consists of a 64- attribute vector per leaf sample.

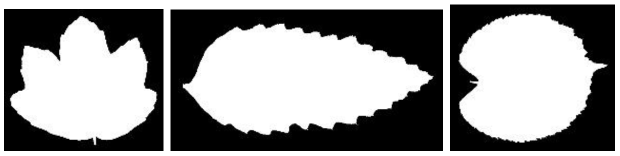
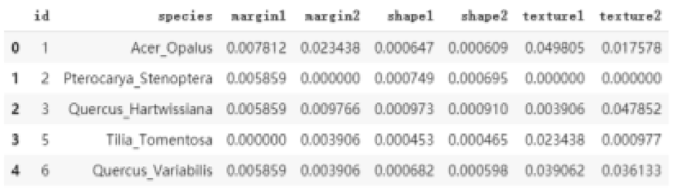


Fig. 1: Binary white leaves against black background

.

Each instance in the dataset has a unique ID. The number of features are 3 sets * 64 attribute vectors, a total of 192 features. The number of total instances are 16 samples of 99 species, a total of 1584 species. The number of training instances are 990 instances and the number of test instances are 594 instances.
The margin feature is divided into 64 attribute vectors, namely, margin1, margin2, …, margin64. Similarly, the shape feature is represented by shape1, shape2, …, shape64. This is followed by texture feature which is denoted by texture1, texture2, …, texture64.



	id	species	margin1	margin2	shape1	shape2	texture1	texture2
0	1	Acer_Opalus	0.007812	0.023438	0.000647	0.000609	0.049805	0.017578
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.000749	0.000695	0.000000	0.000000
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.000973	0.000910	0.003906	0.047852
3	5	Tilia_Tomentosa	0.000000	0.003906	0.000453	0.000465	0.023438	0.000977
4	6	Quercus_Variabilis	0.005859	0.003906	0.000682	0.000598	0.039062	0.036133

Fig. 2: Sample Data Structure

Figure 2 here shows five samples as examples. Actually, there are 64 margins, 64 shapes and 64 textures for each sample. They are all features extracted from images. As we can't show them completely here. We select two data points of each three features here(2x3 out of 64x3). As the features of leaves have been given by the dataset, we do not have to extract them from images by ourselves, which saves us a lot of work. There are three kinds of features which are important to recognize the species of leaves. They are margin, shape and texture. Each of them has a 64-attribute vector. As the results shown in [4], using all three features combined, we will get the best result. Therefore, we will use all three given features to train and test our model.

IV. PREPROCESSING

The original dataset consisted of raw data that represent the extracted features of binary leaves.
Data was split into training and testing data set that contained data in the ratio 80:20. This split was done to use the testing data to determine accuracy.

V. METHODS

4.1 Support Vector Machine

SVM can be well applied to high-dimensional data, avoiding the problem of dimensional disaster. And even if the data is linearly inseparable in the original feature space, as long as a suitable kernel function is given, it will run well. As the number of samples is greater than the number of features here, we use RBF kernel in our project.

4.2 K-Nearest Neighbours

K-NN algorithm is a non-parametric method that can be used in pattern recognition for classification and regression. Here we use it for classification. The input is k closest training samples in the feature space and output is a class membership. To use the K-NN algorithm, we first computed the Euclidean distance and then ordered the labeled samples by increasing distance and determined value of k by cross validation. Finally, we could classify each leaf using the result of K-NN.

4.3 Convolutional Neural Network

CNN is so far the most popular used model to analyze images. CNN consists of fully connected layers, convolutional layers and polling layers, where convolutional layers are essential to work. For regularization part, L2 regularization is use which would add term to loss as following. We chose 0.3 for dropout rate to avoid overfit.

4.4 Random Forest

Random Forest is an ensemble learning method used for classification and regression. Random Forest train on different parts of the same training sets. They aim at reducing the variance. Random Forest differs from Bagging by picking a random subset of features. This algorithm is applied to our dataset by using the "train" function that is provided by the "caret" package in R.

4.5 XGBoost

XGBoost stands for "Extreme Gradient Boosting". XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.

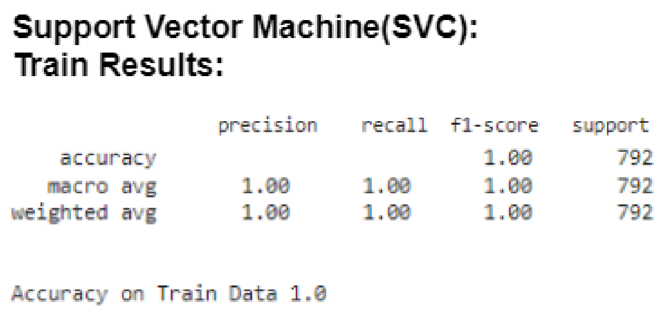
4.6 Decision Tree

A decision tree is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

4.7 Stacking Classifier

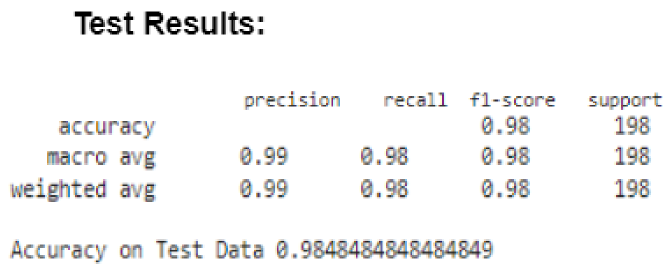
Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs -- meta-features -- of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.

VI. Results



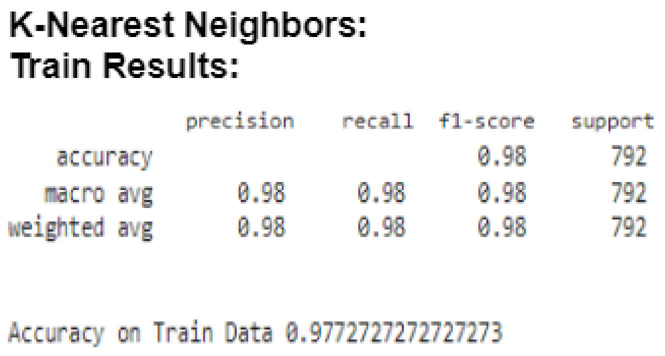
Support Vector Machine(SVC):
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


Accuracy on Train Data 1.0
```



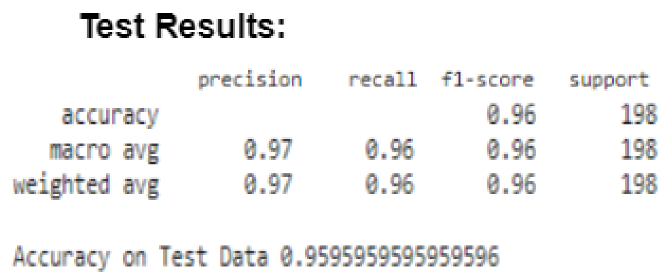
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.98       198
   macro avg       0.99      0.98      0.98       198
weighted avg       0.99      0.98      0.98       198

Accuracy on Test Data 0.9848484848484849
```



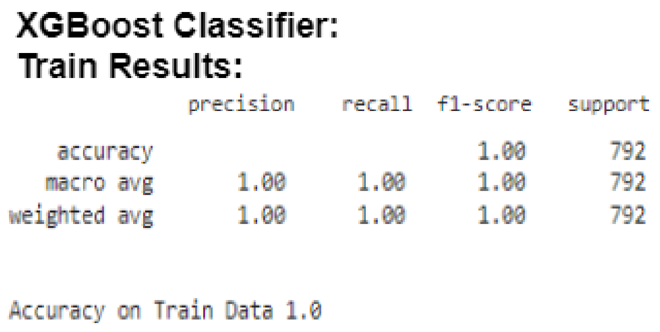
K-Nearest Neighbors:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           0.98       792
   macro avg       0.98      0.98      0.98       792
weighted avg       0.98      0.98      0.98       792


Accuracy on Train Data 0.9772727272727273
```



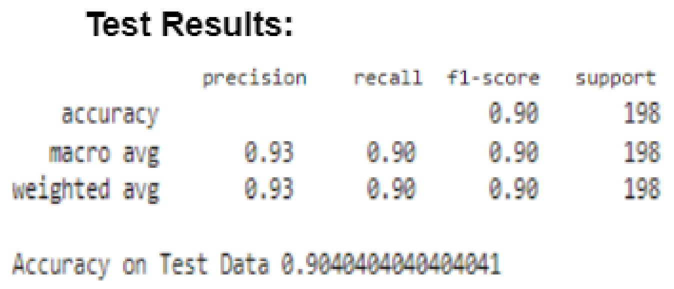
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.96       198
   macro avg       0.97      0.96      0.96       198
weighted avg       0.97      0.96      0.96       198

Accuracy on Test Data 0.9595959595959596
```



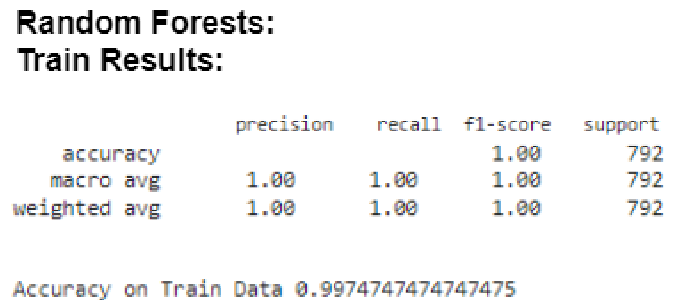
XGBoost Classifier:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


Accuracy on Train Data 1.0
```



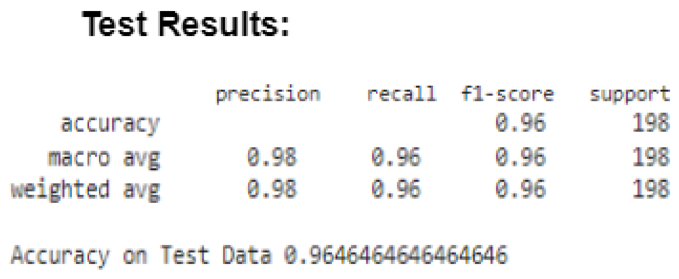
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.90       198
   macro avg       0.93      0.90      0.90       198
weighted avg       0.93      0.90      0.90       198

Accuracy on Test Data 0.9040404040404041
```



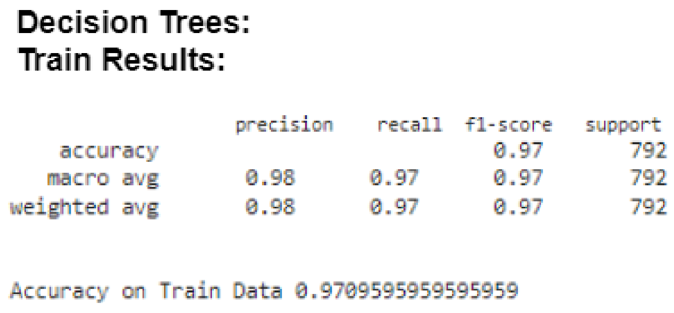
Random Forests:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792

Accuracy on Train Data 0.9974747474747475
```



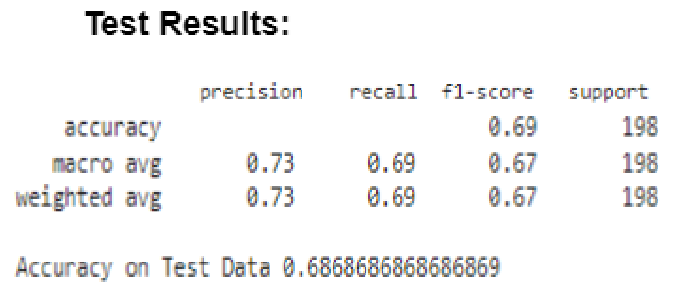
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.96       198
   macro avg       0.98      0.96      0.96       198
weighted avg       0.98      0.96      0.96       198

Accuracy on Test Data 0.9646464646464646
```



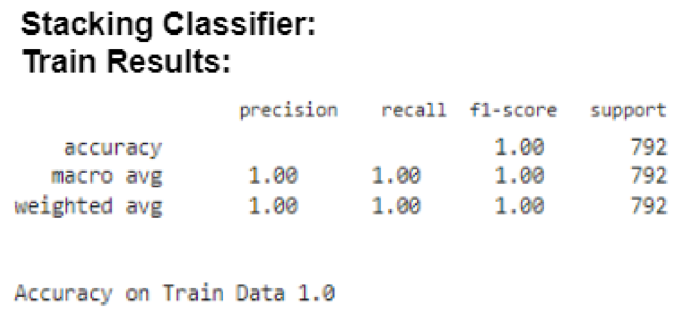
Decision Trees:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           0.97       792
   macro avg       0.98      0.97      0.97       792
weighted avg       0.98      0.97      0.97       792


Accuracy on Train Data 0.9709595959595959
```



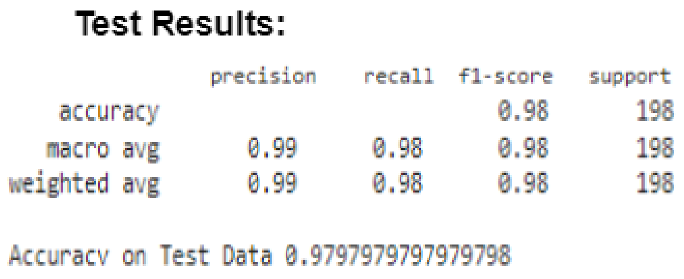
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.69       198
   macro avg       0.73      0.69      0.67       198
weighted avg       0.73      0.69      0.67       198

Accuracy on Test Data 0.6868686868686869
```



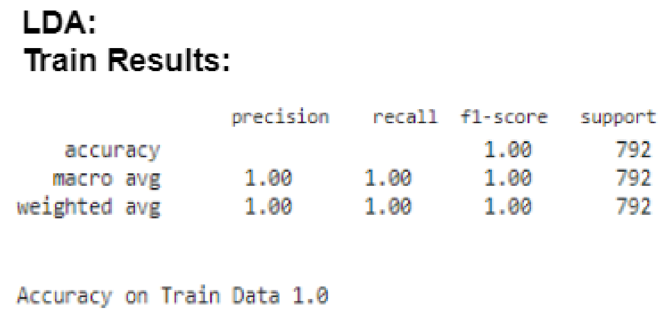
Stacking Classifier:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


Accuracy on Train Data 1.0
```



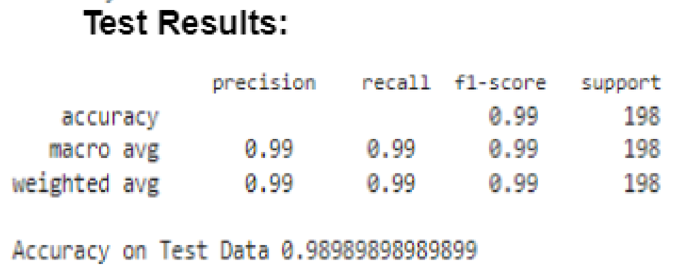
Test Results:

```
              precision    recall  f1-score   support
    accuracy                           0.98       198
   macro avg       0.99      0.98      0.98       198
weighted avg       0.99      0.98      0.98       198

Accuracy on Test Data 0.9797979797979798
```



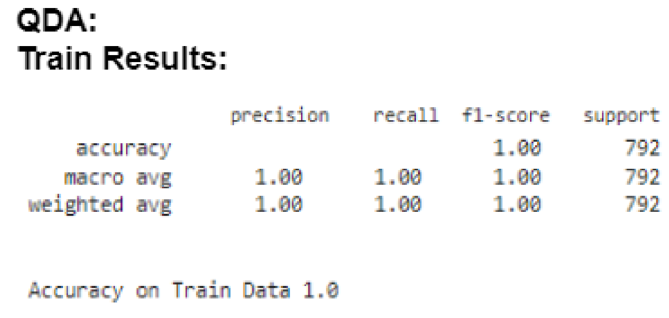
LDA:
Train Results:

```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


Accuracy on Train Data 1.0
```



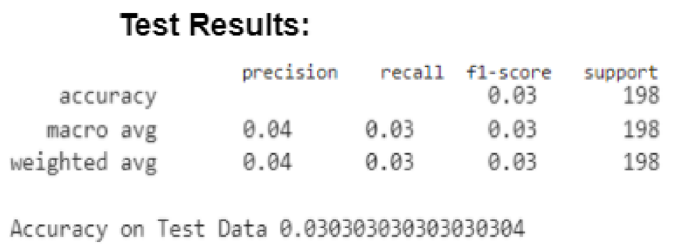
Test Results:

```
              precision    recall  f1-score   support

    accuracy                           0.99       198
   macro avg       0.99      0.99      0.99       198
weighted avg       0.99      0.99      0.99       198

Accuracy on Test Data 0.98989898989899
```



QDA:
Train Results:

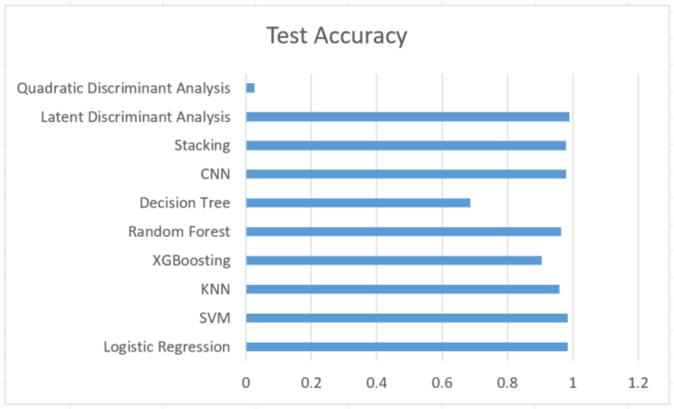
```
              precision    recall  f1-score   support

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


Accuracy on Train Data 1.0
```



Test Results:

```
              precision    recall  f1-score   support
    accuracy                           0.03       198
   macro avg       0.04      0.03      0.03       198
weighted avg       0.04      0.03      0.03       198

Accuracy on Test Data 0.030303030303030304
```

Barplot comparing the Model Accuracies:



Based on the Results, We can see that the below mentioned models are the best performing models comparatively.

- Latent Discriminant Analysis
- Logistic Regression
- SVM
- CNN

VII. Conclusion and Future Outlook

In this project, we successfully implemented multiple models namely multi-layered Convolutional Neural Network, Logistic Regression, Support Vector Machines, K-Nearest Neighbors, Ensemble Models(Random Forest, Decision Trees, XGBoost, Stacking Classifier), Linear Discriminant Analysis and Quadratic Discriminant Analysis to identify leaf species. Compared multiple Model Performances and determined from the results that Latent Discriminant Analysis, Logistic Regression, SVM and CNN performed well.

In the future there is a plausibility of feeding the networks with RGB colored images instead of preprocessed monochrome input images. Future areas for enhancements would be determining the anomalies and abnormalities in the leaf based on the image. Species population tracking and preservation, Plant-based medicinal research, Crop and food supply management are few noted applications. Expand the dataset and improve the model to improve predictions on noisy images as well. Explore state-of-the-art methods to detect and locate leaves from the background.

VIII. References

[1] “Leaf.” Wikipedia, Wikimedia Foundation, 27 May 2020,
en.wikipedia.org/wiki/Leaf.
[2] “How Many Plant Species Are There in the World? Scientists Now Have
an Answer.” Mongabay Environmental News, 12 May 2016,
news.mongabay.com/2016/05/many-plants-world-scientists-may-nowanswer/.
[3] “WenjinTao/Leaf-Classification --Kaggle.” GitHub, github.com/Wen
jinTao/Leaf-Classification--
Kaggle/blob/master/Leaf_Classification_using_Machine_Learning.ipynb.
[4] Mallah, Charles, et al. “Plant Leaf Classification Using Probabilistic
Integration of Shape, Texture and Margin Features.” Computer Graphics and
Imaging / 798: Signal Processing, Pattern Recognition and Applications,
2013, doi:10.2316/p.2013.798-098.
[5] Akbarzadeh, Saman, et al. “Plant Discrimination by Support Vector
Machine Classifier Based on Spectral Reflectance.” Computers and
Electronics in Agriculture, vol. 148, 2018, pp. 250–258.,
doi:10.1016/j.compag.2018.03.026.
[6] Padao, Francis Rey F., and Elmer A. Maravillas. “Using Naïve Bayesian
Method for Plant Leaf Classification Based on Shape and Texture Features.”
2015 International Conference on Humanoid, Nanotechnology, Information
Technology,Communication and Control, Environment and Management
(HNICEM), 2015, doi:10.1109/hnicem.2015.7393179.
[7] “Leaf Classification.” Kaggle,
www.kaggle.com/c/leafclassification/data.
[8] Gerstoft, Peter. “ECE228 Lecture 4.” 20 Apr. 2020, La Jolla.
[9]Gerstoft, Peter. “ECE228 Lecture 6.” 4 May 2020, La Jolla.
[10] “K-Nearest Neighbors Algorithm.” Wikipedia, Wikimedia Foundation,
28 May 2020, en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.
[11] “Linear Discriminant Analysis.” Wikipedia, Wikimedia Foundation, 3
June 2020,
en.wikipedia.org/wiki/Linear_discriminant_analysis.
[12] Gerstoft, Peter. “ECE228 Lecture 3.” 13 Apr. 2020, La Jolla.
[13] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual.
Scotts Valley, CA: CreateSpace.
[14] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B.,
Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12(Oct), 2825–2830.
[15] Abadi, Mart&#39;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean,
J., … others. (2016). Tensorflow: A system for large-scale machine learning.
In 12th $USENIX$ Symposium on Operating Systems Design and
Implementation ($OSDI$ 16) (pp. 265–283).
[16] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from
https://github.com/fchollet/keras
[17] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing
USA.
[18] McKinney, W., & others. (2010). Data structures for statistical
computing in python. In Proceedings of the 9th Python in Science
Conference (Vol. 445, pp. 51–56).