Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)
Article DOI:10.21474/IJAR01/9157
DOI URL: http://dx.doi.org/10.21474/IJAR01/9157
ISSN NO. 2320-5407

*RESEARCH ARTICLE*

# IMAGE CLASSIFICATION USING MACHINE LEARNING TECHNIQUES.

**M. Prasanna Lakshmi[1], M. Venkata Rao[2] And V. Esther Jyothi[3].**

1. Department of Computer Applications, V.R.Siddhartha Engineering College, Vijayawada, India.
2. Department of Mathematics, V.R.Siddhartha Engineering College, Vijayawada India.

## *Manuscript Info*

***Manuscript History***
Received: 24 March 2019
Final Accepted: 26 April 2019
Published: May 2019

***Key words:-***
probabilistic framework, histogram accumulation, accuracy, small training set size.

## *Abstract*

Plant species classification using leaf samples is a challenging and important problem to solve. This paper introduces a new dataset of sixteen samples each of one-hundred plant species; and describes a method designed to work in conditions of small training set size and possibly incomplete extraction of features. This motivates a separate processing of three feature types: shape, texture, and margin; combined using a probabilistic framework. The texture and margin features use histogram accumulation, while a normalized description of contour is used for the shape. In this paper we are using different Machine Learning algorithms to classify images based on different parameters to identify and compare the accuracy. Python supported open source modules are used here and loading the sample datasets collecting from internet and implement different neural network approaches to preprocess the data and analyses the output results.



## Introduction:-

Image processing is the recent growing technique in the world. The main objective of the project is to develop a leaf classifier which works on the principle of extracting information based on its architecture using image processing to classify correctly in the plant kingdom. Plants play an important role for the development of human society. The urgent situation is that due to environmental degradation, many rare plant species on the earth are still unknown and are at the margin of extinction, so it is necessary to keep record for plant protection.

Plant species are not only vast in number, but also in their use; estimates of the number of species of flowering plants range from 220,000 to 420,000. Automatic recognition of plants is useful for a variety of industries such as foodstuff and medicine, reduction of chemical wastage during crop spraying, and also for species identification and preservation.

Plant taxonomy suggests that a species can be successfully inferred from the leaves; the leaves are more readily available, easily found, and collected than other parts of the plant. The literature suggests a variety of work has been documented in the machine vision field using data sets of leaves and image processing techniques. The investigation is conducted using an existing and new data set. The new data set consists of 1,600 images of leaf specimens (16 samples each of one-hundred species). Estimation of the leaf class (species) uses three features,

**Corresponding Author:-M. Prasanna Lakshmi.**
Address**:-**Department of Computer Applications, V.R.Siddhartha Engineering College, Vijayawada, India.

which are analyzed separately: a shape descriptor, an interior texture histogram, and a fine- scale margin histogram. These are then combined to provide an overall indication of the species (and associated probability). The 'k-Nearest Neighbor'(K-NN) classifier is a fundamental tool in pattern analysis, providing a straightforward means of determining class membership for an unseen sample vector, given a finite training set. For a set of unseen vectors, the resulting precision and recall provide an insight into the separability of the class labels, and easily generalizes to problems with multiple classes.

We propose that the density estimates given from the standard and the recent extension are used in a framework to enable multiple features to be combined in a straightforward approach. A separate K-NN density estimator is generated for each feature, separately. Then,a simple product of the three different density estimations will provide the final estimate. Evaluation of discrete classifier performance is reasonably straightforward, combining the precision and recall into a single 'mean accuracy' characteristic. This approach can also be used to indirectly evaluate the per channel probabilistic outputs, which are integrated into a final discrete output. The direct evaluation of probabilistic output uses an information theoretic metric, i.e. the 'information gain', to measure the mutual information between the actual posterior and the estimates of this distribution provided by the classifier. The results show that, using all evaluation metrics, the proposed approach of combined density estimates from multiple features provides improved performance. The robustness of the approach is demonstrated on the challenging new one-hundred leaves data set, where the classification performance remains relatively accurate given a drastic reduction in training size.

This motivates a separate processing of three feature types: shape, texture, and margin; combined using a probabilistic framework. The texture and margin features use histogram accumulation, while a normalized description of contour is used for the shape. The shape feature was addressed using a hyper sphere classifier. The next phase in the plant leaf identification is the feature extraction phase. The main advantage of this stage is that it removes redundancy from the image and the leaf images are represented by a set of numerical features. The classifier used these features to classify the data. The Texture Feature Extraction is one of the main subjects in pattern recognition.

**Image Classification Techniques**

Image Classification is a branch of computer vision where images are classified into categories. This is a very important topic in today's context as large databases of images are becoming very common. Images can be classified as supervised or unsupervised techniques.

**Supervised learning:**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labelled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyses the training data andProduces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

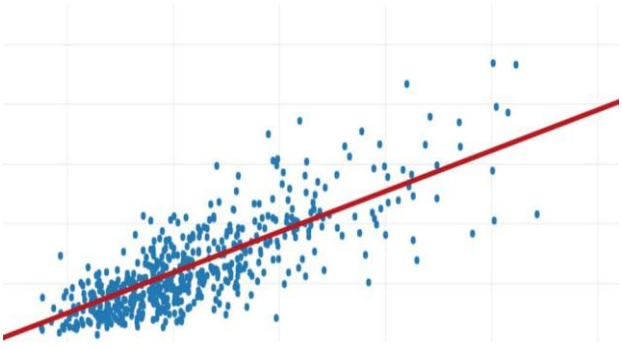

**Figure1:-**Supervised learning

In supervised learning, we start with importing dataset containing training attributes and the target attributes. The Supervised Learning algorithm will learn the relation between training examples and their associated target variables and apply that learned relationship to classify entirely new inputs (without targets).

To illustrate how supervised learning works, let's consider an example of predicting the marks of a student based on the number of hours she studied
Mathematically,
$Y = f(X)+ C$
Here,
F will be the relation between the marks and number of hours the student prepared for an exam.
X is the INPUT (Number of hours she prepared).
Y is the output (Marks the student scored in the exam).
C will be random error

The ultimate goal of the supervised learning algorithm is to predict Y with the maximum accuracy for a given new input X. There are several ways to implement Supervised Learning; we'll explore some of the most commonly used approaches.

Based on the given datasets the machine learning problem is categorized into two types, Classification, and Regression. If the given data has both input (training) values and output (target) values, then it is a Classification problem. If the dataset has continuous numerical values of attributes without any target labels, then it comes under Regression problem.

The most widely used learning algorithms are:
1. Support Vector Machines
2. linear regression
3. logistic regression
4. naive Bayes
5. linear discriminant analysis
6. decision trees
7. k-nearest neighbor algorithm
8. Neural Networks (Multilayer perceptron)
9. Similarity learning

**Unsupervised learning:**
Unsupervised Learning is a class of Machine Learning techniques to find the patterns in data. The data given to unsupervised algorithm are not labelled, which means only the input variables(X) are given with no corresponding output variables. In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data.

Compared to supervised learning where training data is labelled with the appropriate classifications, models using unsupervised learning must learn relationships between elements in a data set and classify the raw data without "help." This hunt for relationships can take many different algorithmic forms, but all models have the same goal of mimicking human logic by searching for indirect hidden structures, patterns or features to analyse new data.

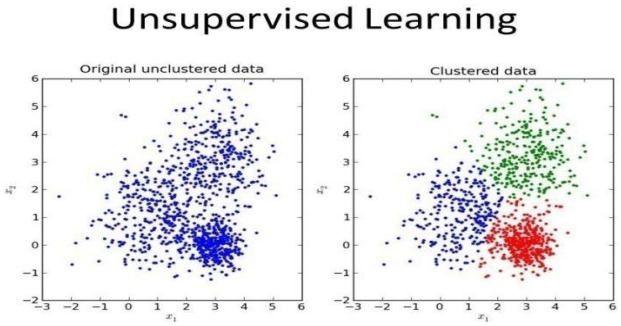


**Figure 2:-**Unsupervised learning

The most common algorithms used in unsupervised learning include:
1. Clustering
2. hierarchical clustering
3. k-means
4. mixture models
5. DBSCAN
6. OPTICS algorithm
7. Anomaly detection
8. Local Outlier Factor
9. Neural Networks
10. Autoencoders
11. Deep Belief Nets
12. Hebbian Learning
13. Generative Adversarial Networks

**Supervised Vs Unsupervised Learning:**
In supervised learning, the system tries to learn from the previous examples that are given. (On the other hand, in unsupervised learning, the system attempts to find the patterns directly from the example given.) So, if the dataset is labeled it comes under a supervised problem, it the dataset is unlabelled then it is an unsupervised problem.

**Image Classification Model**
Before we deep dive into the Python code, let's take a moment to understand how an image classification model is typically designed. We can divide this process broadly into 4 stages. Each stage requires a certain amount of time to execute:
1. Loading and pre-processing Data – 30% time
2. Defining Model architecture – 10% time
3. Training the model – 50% time
4. Estimation of performance – 10% time

**Loading and pre-processing data:**
First the data will be converted into two sets such as train set and test set. That data must be saved in a CSV(Comma Separated Value) format inorder to work in Anaconda IDE. The file must be loaded into the working directory by using Pandas library. After loading the data, we need to pre-process the data. We need to follow some pre-processing steps. Data pre-processing consists of importing the needful libraries, taking care of missing data such as replacing the missing data in different methods, encoding the categorical data which means encoding the strings into numerics, and feature scaling. This step of loading and pre-processing data takes 30% of project time.

**Defining model architecture:**
For defining the model architecture, it will take just 10% of project time. According to the problem definition and the data we contain, we need to decide the model. If our data contains one dependent variable and one independent variable, then it will be easy to build a model like Simple Linear Regression. If there are number of independent variables, all those are not used to predict dependent variable. Some of them must be thrown out. In this way the model will be defined.

**Training the model:**
This is the major part of the project. Basically, we need to train the model by giving the train data as input. While splitting the data, nearly 75-80 % of data will be given into traindata and the remaining 20% of data will become test data. For example, if we have 100 images of cat as data, then 80 images are used to train the model by telling it, it's a cat. And the last 20 images are used for prediction. After the prediction done, then comparison between the original result and the predicted result will takes place.

**Estimation Of Performance:**
The prediction of the output will be done by different models. But all those models are not same efficient. We will choose the efficient algorithm which gives more accurate results. This estimation of performance will take 10% of time.

**K Nearest Neighbor Classification**
K Nearest Neighbor (KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. In Credit ratings, financial institutes will predict the credit rating of customers. Inloan disbursement, banking institutes will predict whether the loan is safe or risky. In political science, classifying potential voters in two classes will vote or won't vote. KNN algorithm used for both classification and regression problems. KNN algorithm based on feature similarity approach.

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

**KNN has the following basic steps:**
1. Calculate distance
2. Find closest neighbors
3. Vote for label
4. KNN can be summarized as below:
5. Computes the distance between the new data point with every training example.
6. For computing the distance measures such as Euclidean distance, Hamming distance or Manhattan distance will be used.
7. Model picks K entries in the database which are closest to the new data point.
8. Then it does the majority vote i.e the most common class/label among those K entries will be the class of the new data point.

**Pros**
The training phase of K-nearest neighbor classification is much faster compared to other classification algorithms. There is no need to train a model for generalization, That is why KNN is known as the simple and instance-based learning algorithm. KNN can be useful in case of nonlinear data. It can be used with the regression problem. Output value for the object is computed by the average of k closest neighbors value.

**Cons**
The testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory. It requires large memory for storing the entire training dataset for prediction. KNN requires scaling of data because KNN uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weigh more than features with low magnitudes. KNN also not suitable for large dimensional data.

**Naïve Bayes Algorithm**
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

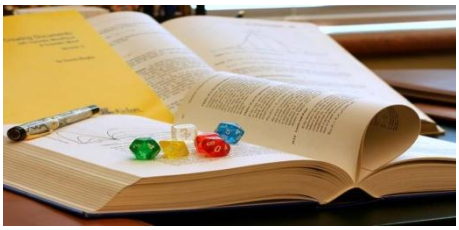

**Figure 3:-**Naïve Bayes

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as: P(h|d) = (P(d|h) * P(h)) / P(d) Where

1. P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
2. P(d|h) is the probability of data d given that the hypothesis h was true.
3. P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
4. P(d) is the probability of the data (regardless of the hypothesis).
5. You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).
6. After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

This can be written as:
MAP(h) = max(P(h|d))
` or
MAP(h) = max((P(d|h) * P(h)) / P(d))
or
MAP(h) = max(P(d|h) * P(h))

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.
Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation and we could drop it so that we end up with:
MAP(h) = max(P(d|h))
Best Prepare Your Data for Naive Bayes
Categorical Inputs: Naive Bayes assumes label attributes such as binary, categorical or nominal.

**Gaussian Inputs:**
If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).

**Classification Problems**:
Naive Bayes is a classification algorithm suitable for binary and multiclass classification.

**Log Probabilities:**
The calculation of the likelihood of different class values involves multiplying a lot of small numbers together. This can lead to an underflow of numerical precision. As such it is good practice to use a log transform of the probabilities to avoid this underflow.

**Kernel Functions:**
Rather than assuming a Gaussian distribution for numerical input values, more complex distributions can be used such as a variety of kernel density functions.

**Update Probabilities:**
When new data becomes available, you can simply update the probabilities of your model. This can be helpful if the data changes frequently.

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.
It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on.

**Gaussian Naive Bayes**
Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.
This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

**Representation for Gaussian Naive Bayes**
Above, we calculated the probabilities for input values for each class using a frequency. With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution. This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

**Pros**
Data set is small than high bias low variance classifier like NB will work well.

**Cons**
Data set is small than generative class will work well. Super simple, you're just doing a bunch of counts. It will converge quickly. No curse of dimensionality. Data set is large don't use it. Don't use when features are correlated.

Can perform poorly if independence assumptions do not hold.

**Linear discriminant analysis:**
Logistic regression is a classification algorithm traditionally limited to only two-class classification problems.

If you have more than two classes then Linear Discriminant Analysis is the preferred linear classification technique.

**Representation of LDA Models**
The representation of LDA is straight forward.
1. It consists of statistical properties of your data, calculated for each class. For a single input variable (x) this is the mean and the variance of the variable for each class. For multiple variables, this is the same properties calculated over the multivariate Gaussian, namely the means and the covariance matrix.
2. These statistical properties are estimated from your data and plug into the LDA equation to make predictions. These are the model values that you would save to file for your model.

**Prepare Data for LDA**
**Classification Problems:**
This might go without saying, but LDA is intended for classification problems where the output variable is categorical. LDA supports both binary and multi-class classification.

**Gaussian Distribution:**
The standard implementation of the model assumes a Gaussian distribution of the input variables. Consider reviewing the univariate distributions of each attribute and using transforms to make them more Gaussian-looking (e.g. log and root for exponential distributions and Box-Cox for skewed distributions).

**Remove Outliers:**
Consider removing outliers from your data. These can skew the basic statistics used to separate classes in LDA such the mean and the standard deviation.

**Same Variance:**
LDA assumes that each input variable has the same variance. It is almost always a good idea to standardize your data before using LDA so that it has a mean of 0 and a standard deviation of 1.

**Pros**
LDA in the binary-class case has been shown to be equivalent to linear regression with the class label as the output. This implies that LDA for binary-class classifications can be formulated as a least squares problem.

**Cons**
One disadvantage when compared to logistic regression is that the former can generate predicted probabilities outside the range 0-1.

## Conclusion:-

The paper has presented a K-NN density estimation framework with three features for species classification. It has been shown to be effective in this particular task, coping with low training samples and a relatively large variety of categories. Confirmation of results were obtained from testing the framework with the well known iris dataset and a ten-fold cross- validation evaluation. With regards to the new one-hundred species leaf data set presented in this paper, both test density estimators performed equally well for mean classification accuracy. With regards to the density performance metric (Expected Log Likelihood), the top performing method is the PROP density estimator. The results show that, using all evaluation metrics, the proposed approach of combined density estimates from multiple features provides improved performance.

We consider that future work on the density estimation framework using K-NN classifiers may include adapting the framework to cope with data loss and noisy feature vectors e.g. object occlusion and missing parts of the sample leaf. We believe that many other features pertaining to shape, texture or fine-scale edge could be implanted in the suggested framework. Given that each channel is individually processed, the evaluation of the various combinations does indeed reveal which features are useful for describing the leaves. It is entirely feasible to utilize a variety of conventional morphological shape features as additional input vectors into the density estimation framework. Further work on density estimations and various other methods in this domain could be explored, in particular in attempting to maximize the performance metric of the posterior estimates.

## References:-

1. https://www.researchgate.net/publication/266632357_Plant_Leaf_Classification_using_Probabilistic_Integration_of_Shape_Texture_and_Margin_Features
2. https://ieeexplore.ieee.org/abstract/document/4458016/
3. https://www.researchgate.net/publication/4156909_Machine_learning_techniques_for_ontologybased_leaf_classification
4. http://www.ijirae.com/volumes/Vol2/iss6/15.JNAE10092.pdf
5. https://www.researchgate.net/publication/220558722_Leaf_shape_based_plant_species_recog nition
6. https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/
7. https://www.dezyre.com/project/hackerday-project/project-title/identify-plant-species-with-imagebenchmarkingclassifiers#sub-hackerday
8. Introduction to Machine Learning with Python: A Guide for Data Scientists
9. Book by Andreas C. Müller and Sarah Guido
10. Image Recognition and Classification: Algorithms, Systems, and Applications Book by Bahram Javidi.