VOL.5 NO.2 September 2016 - ISSN 2233 – 1859

UOIuBIH
ORSinBIH
Operations Research Society in Bosnia and Herzegovina

Southeast Europe Journal of Soft Computing

Available online: http://scjournal.ius.edu.ba

IUS Soft Computing Research Group

Classification of Leaf Type Using Multilayer Perceptron, Naive Bayes and Support Vector Machine Classifiers

AzraMusić, SadinaGagula-Palalić

International University of Sarajevo,
Faculty of Engineering and Natural Sciences,
Hrasničkacesta 15, Ilidža 71210 Sarajevo, Bosnia and Herzegovina

amusic@student.ius.edu.ba; sadina@ius.edu.ba

Article Info

Article history:
Article received on 1 Jul. 2016
Received in revised form 10 Aug.2016

Keywords:

Multiclass Classification, Neural Networks, Multilayer Perceptron, Naive Bayes, Leaf Dataset

Abstract

Multiclass classification has always been challenging in the area of machine learning algorithms. Different publicly available software applications offer various learning algorithms' implementations. This paper uses leaf dataset with 30 different plant species with simple leaf types prepared by Silva et al (2014), and classification is performed using Multilayer Perceptron, Naive Bayes and Support Vector Machine classifiers. Performance of classifiers is compared based on time needed for building the model and classification accuracy.

1. INTRODUCTION

Plants play an important role in our environment. Without plants there will be no existence of the earth's ecology [1]. In order to protect plants that are at the risk of extinction, a plant database should be developed to help people recognize endangered plants and act according to that. The development of a plant recognition system requires a set of discriminating variables representing plants' features, and a structured database to train statistical models [2].

Dataset prepared by Silva et al. consists of 40 different plants and a total of 340 data samples. Dataset is based on the analysis of the leaf shape features and provides 15 attributes. Table 1 details each plant's scientific name and the number of leaf specimens available by species. In this paper, 30 different plants that exhibit simple leaves will be used in classification process. The classifiers used are Multilayer Perceptron, Naive Bayes and Support Vector Machine classifiers implemented in Weka. The organization of the paper is as follows: section 2 provides a description of data and methods, section 3 the experimental results obtained for each classifier, and section 4 provides the overall conclusion and the scope for future research.

Table 1. Leaf dataset class list

C	Scientific Name	C	Scientific Name
1	Quercussuber	22	Primula vulgaris
2	Salix atrocinera	23	Erodium sp.
3	Populusnigra	24	Bougainvillea sp.
4	Alnus sp.	25	Arisarum vulgare
5	Quercusrobur	26	Euonymus japonicas
6	Crataegusmonogyna	27	Ilex perado ssp. Azorica
7	Ilex aquifolium	28	Magnolia soulangeana
8	Nerium oleander	29	Buxussempervirens
9	Betulapubescens	30	Urticadioica
10	Tiliatomentosa	31	Podocarpus sp.
11	Acer palmatum	32	Accasellowiana

12	Celtis sp.	33	Hydrangea sp.
13	Corylusavellana	34	Pseudosasa japonica
14	Castaneasative	35	Magnolia grandiflora
15	Populus alba	36	Geranium sp.

2. MATERIALS AND METHODS

2.1. Data Description

Each leaf specimen was photographed over a colored background using an Apple iPad 2 device, and a short description of common shape features is done in following manner [3]. Let I denote the object of interest in an image, ∂I its border, $D(I)$ the diameter defined as the maximum distance between any two points in ∂I and $A(I)$ its area. Let $A(H(I))$ denote the area of the object's convex hull and $L(\partial I)$ the object's contour length. The operator $d(.)$ stands for the Euclidean distance [2,4].

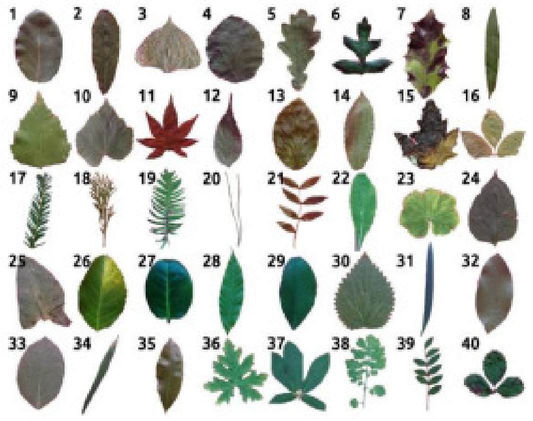


Figure 1. Leaf database overview

The provided dataset by Silva et al. contains 15 attributes out of which eight are shape features and other six texture features. One attribute is the percent of specimens available with the similar leaf structure. Attributes are [4]:

1. *Eccentricity* – The eccentricity of the ellipse with identical second moments to I is computed. This value ranges from 0 to 1.

2. *Aspect Ratio* – Consider any $X,Y \in \partial I$. Choose X and Y such that $d(X,Y) = D(I)$. Find $Z,W \in \partial I$ maximizing $D^{\perp} = d(Z,W)$ on the set of all pairs of ∂I that define a segment orthogonal to $[XY]$. The aspect ratio is defined as the quotient $D(I)/D^{\perp}$. Values close to 0 indicate an elongated shape.

3. *Elongation* – Compute the maximum escape distance $d_{\max} = \max_{X \in I} d(X,\partial I)$. Elongation is obtained as $1 - 2d_{\max}/D(I)$ and ranges from 0 to 1.The minimum is achieved for a circular region.

4. *Solidity* – The ratio $A(I)/ A(H(I))$ is computed and it measures how well I fits a convex shape of leaf.

5. *Stochastic convexity* – This variable extends the usual notion of convexity in topological sense, using sampling to perform the calculation. The aim is to estimate the probability of a random segment $[XY]$, X, $Y \in I$, to be fully contained in I.

6. *Isoperimetric Factor* –The ratio $4\pi A(I)/L(\partial I)^2$ is calculated. The maximum value of 1 is reached for a circular region. Curvy intertwined contours yield low values.

7. *Maximal Indentation Depth* – Let $C_{H(I)}$and $L(H(I))$ denote the centroid and arc length of $H(I)$. The distances $d(X,C_{H(I)})$ and $d(Y, C_{H(I)})$ are computed $\forall X \in H(I)$ and $\forall Y \in \partial I$. The indentation function can be defined as $[d(X,C_{H(I)}) - d(Y,C_{H(I)})]/ L(H(I))$, which is sampled at one degree intervals. The maximum indentation depth D is the maximum of this function.

8. *Lobedness*– The Fourier Transform of indentation function above is computed after mean removal. The resulting spectrum is normalized by the total energy. Calculate lobedness as F $\times$ D^2, where F stands for the smallest frequency at which the cumulated energy exceeds 80%. This feature characterizes how lobed a leaf is.

Following six attributes are based on statistical properties of the intensity histograms of grayscale transformations of the original RGB images. If Z is random variable indicating image intensity, its nth moment around the mean is$\mu_n = \sum_{i=0}^{L-1}(z_i - m)^n p(z_i)$, where m is the mean of Z, p(.) its histogram and L is the number of intensity levels [2,4].

9. *Average Intensity* –It is defined as the mean of the intensity image, m.

10. *Average Contrast* – The standard deviation of the intensity image, $\sigma = \sqrt{\mu_2(z)}$.

11. *Smoothness* – Defined as $R = 1 - 1/(1+\sigma^2)$ and measures the relative smoothness of the intensities in a given region. For a region of constant intensity, R takes the value 0 and R approaches 1 as regions exhibit larger disparities in intensity values.σ^2 is generally normalized by $(L - 1)^2$ to ensure that $R \in [0,1]$.

12. *Third moment* – μ_3 is a measure of the intensity histogram's skewness. This measure is generally normalized by $(L - 1)^2$like smoothness.

13. *Uniformity*–Defined as $U = \sum_{i=0}^{L-1} p^2(z_i)$, its max value is reached when all intensity levels are equal.

14. *Entropy* – A measure of intensity randomness. [4]

2.2. Multilayer Perceptron in Weka 3.8

Waikato Environment for Knowledge Analysis is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License. Weka's implementation of multilayer perceptron uses backpropagation to classify inputs into target classes. Users can control properties of algorithm including learning rate, number of neurons in hidden layer, stopping momentum, and others [5].

At the heart of backpropagation algorithm is an expression for the partial derivative ∂C/∂w of the cost function C with respect to any weight w (or bias b) in the network. The expression tells us how quickly the cost changes when we change the weights and biases. The quadratic cost function has the form

$$C = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2 \qquad (1)$$

where: n is the total number of training examples; the sum is over individual training examples, x; y=y(x) is the corresponding desired output; L denotes the number of layers in the network; and $a^L=a^L(x)$ is the vector of activations output from the network when x is input.

After first algorithm iteration, the error in the output layerδ^L needs to be calculated using equation:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \qquad (2)$$

After calculating error in the output layer, errors in hidden layers should be calculated as well. Equation for the error δ^l in terms of the error in the next layer δ^{l+1} is

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \cdot \ \sigma'(z') \qquad (3)$$

After determining all the errors, gradient descent is performed in order to minimize the cost function and find optimal solution [6].In this paper, the number of neurons in the hidden layer is set to the number produced when summation of number of attributes and classes is divided by 2 (23). The cross-validation is performed using 10-folds to test the network created.

2.3. Naive Bayes in Weka 3.8

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. Bayes' theorem is given by:

$$P(y_i|x) = \frac{P(y_i)P(x|y_i)}{P(x)} \qquad (4)$$

Using Bayesian probability terminology, the above equation can be interpreted as that probability of posterior is directly proportional to the multiplication of likelihood$P(x|y_i)$ and prior probability $P(y_i)$, and inversely proportional to the evidence probability$P(x)$ [7].

Weka suit offers this algorithm as well in their selection of machine learning algorithms. It can be applied to the multiclass classification problems and numeric estimator precision values are chosen based on analysis of the training data. Naive Bayes is an example of simple and fast performance algorithm that does not require high computational space.

2.4. Support Vector Machine in Weka 3.8

Support Vector Machine classifier in Weka, represented as SMO, implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. In that case, the coefficients in the output are based on the normalized data, not the original data. This is important for interpreting the classifier. Multi-class problems are solved using pairwise classification by Hastie and Tibshirani [8].

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class, since in general the larger the margin the lower the generalization error of the classifier. The main aim is to maximize profit or decision function given by

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + \rho \qquad (5)$$

where K is kernel of support vectors derived from the input training set of vectors and ρ is an independent term like bias is in multilayer perceptron [9].

3. EXPERIMENTS AND RESULTS

3.1.Multilayer Perceptron in Weka 3.8

Multilayer perceptron is Weka did not require a lot of data preprocess before classification itself. Using preprocessing filters did not produce higher accuracy rates and therefore no filter is applied in the final results presented in this paper. The number of neurons in hidden layer is 23 and 10-fold cross-validation is used. The neural network has structured given in Fig.2.

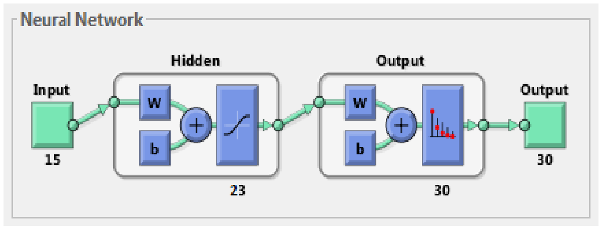
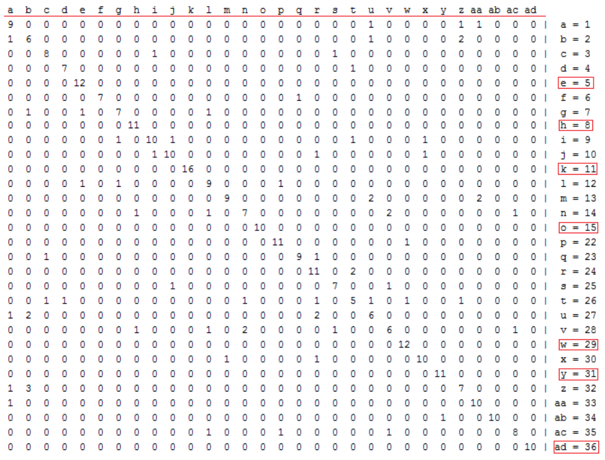


Figure 2. Neural network structure

Time taken for building model is 7.77 seconds and accuracy rate is 79.71% which means that out of 360 samples, 271 samples are correctly classified. Some of the classes such as 5, 8, 11, 15, 29, 31, 36 have 100% accuracy rate. Confusion matrix can be seen in Fig.3, and the performance plot in Fig.4.



```
  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab ac ad
  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  1  0  0  0 |   a = 1
  1  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0  0  0 |   b = 2
  0  0  8  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 |   c = 3
  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0 |   d = 4
  0  0  0  0 12  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   e = 5
  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0 |   f = 6
  0  1  0  0  1  0  7  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   g = 7
  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   h = 8
  0  0  0  0  0  0  1  0 10  1  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0 |   i = 9
  0  0  0  0  0  0  0  0  1 10  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0 |   j = 10
  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   k = 11
  0  0  0  0  1  0  1  0  0  0  0  9  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   l = 12
  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0  2  0  0  0  0  0  2  0  0  0 |   m = 13
  0  0  0  0  0  0  0  1  0  0  0  1  0  7  0  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0 |   n = 14
  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   o = 15
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0  0  1  0  0  0  0  0  0  0 |   p = 22
  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  9  1  0  0  0  0  0  0  0  0  0  0  0  0 |   q = 23
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  0  2  0  0  0  0  0  0  0  0  0  0 |   r = 24
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  7  0  0  1  0  0  0  0  0  0  0  0 |   s = 25
  0  0  1  1  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  5  1  0  1  0  0  1  0  0  0  0 |   t = 26
  1  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  6  0  0  0  0  0  0  0  0  0 |   u = 27
  0  0  0  0  0  0  0  1  0  0  0  1  0  2  0  0  0  0  1  0  0  6  0  0  0  0  0  0  1  0 |   v = 28
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 12  0  0  0  0  0  0  0 |   w = 29
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0 10  0  0  0  0  0  0 |   x = 30
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0 |   y = 31
  1  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0 |   z = 32
  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0 |  aa = 33
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0 10  0  0 |  ab = 34
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  8  0 |  ac = 35
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10 |  ad = 36
```

Figure 3. Confusion matrix – Multilayer perceptron

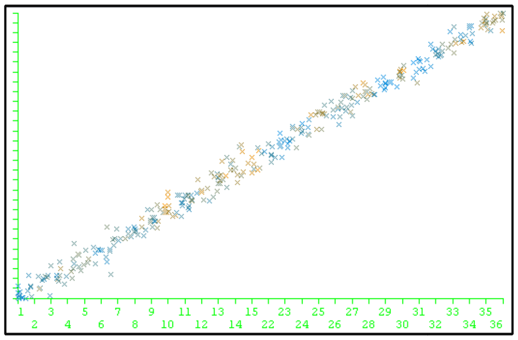


Figure 4. Performance plot – Multilayer perceptron

3.2. Naive Bayes in Weka 3.8

Same as the previous method, Naive Bayes did not require a lot of data preprocess, and 10-fold cross-validation is used. Due to algorithm's simplicity, time needed for building model is only 0.01 seconds. Unfortunately, even though computational time gave it advantage, accuracy rate hit only 74.12% which means that out of 360 samples, 252 are correctly classified. Three classes 8, 23 and 36 have 100% accuracy rate. Confusion matrix can be seen in Fig.6, and the performance plot in Fig.5.

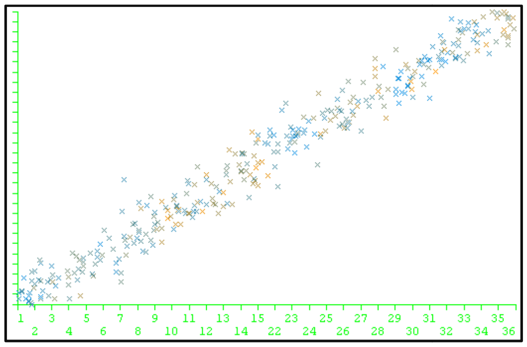
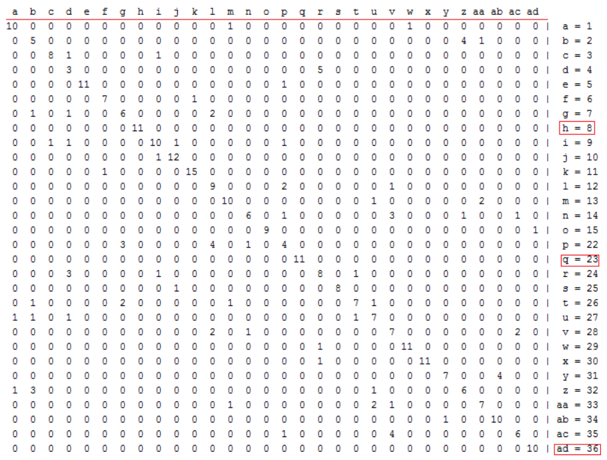


Figure 5. Classification performance plot – Naive Bayes

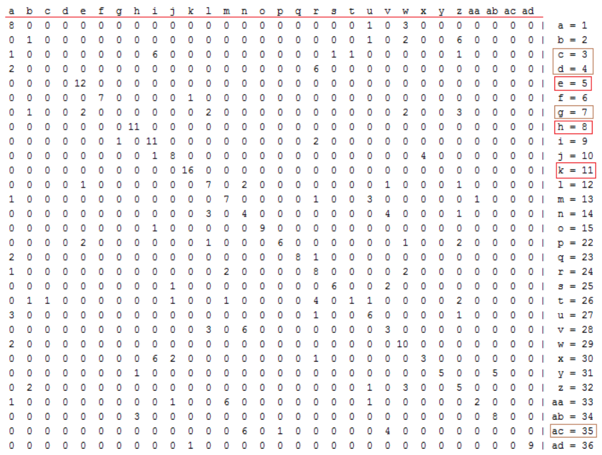


```
  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab ac ad
 10  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 |   a = 1
  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  1  0  0  0 |   b = 2
  0  0  8  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   c = 3
  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0 |   d = 4
  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   e = 5
  0  0  0  0  0  7  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   f = 6
  0  1  0  1  0  0  6  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   g = 7
  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   h = 8
  0  0  1  1  0  0  0  0 10  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   i = 9
  0  0  0  0  0  0  0  0  1 12  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   j = 10
  0  0  0  0  0  1  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   k = 11
  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  2  0  0  0  0  0  1  0  0  0  0  0  0  0  0 |   l = 12
  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  1  0  0  0  0  0  2  0  0  0 |   m = 13
  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  1  0  0  0  0  0  3  0  0  0  1  0  0  1  0 |   n = 14
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 |   o = 15
  0  0  0  0  0  0  3  0  0  0  0  4  0  1  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   p = 22
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0 |   q = 23
  0  0  0  3  0  0  0  0  1  0  0  0  0  0  0  0  0  8  0  1  0  0  0  0  0  0  0  0  0  0 |   r = 24
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  8  0  0  0  0  0  0  0  0  0  0  0 |   s = 25
  0  1  0  0  0  0  2  0  0  0  0  0  1  0  0  0  0  0  0  7  1  0  0  0  0  0  0  0  0  0 |   t = 26
  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  7  0  0  0  0  0  0  0  0  0 |   u = 27
  0  0  0  0  0  0  0  0  0  0  0  2  0  1  0  0  0  0  0  0  0  7  0  0  0  0  0  0  2  0 |   v = 28
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 11  0  0  0  0  0  0  0 |   w = 29
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 11  0  0  0  0  0  0 |   x = 30
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  4  0  0 |   y = 31
  1  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  6  0  0  0  0 |   z = 32
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  2  1  0  0  0  0  7  0  0  0 |  aa = 33
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0 10  0  0 |  ab = 34
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  4  0  0  0  0  0  0  6  0 |  ac = 35
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10 |  ad = 36
```

Figure 6. Confusion matrix – Naive Bayes

3.3. Support Vector Machine in Weka 3.8

Support Vector Machine learning algorithm used on the data is based on the structure of polynomial kernel and with normalization filter applied on the training inputs before the learning process. Same as in previous experiments, 10-fold cross-validation is used. Time needed for building the model is 3.73 seconds, but accuracy rate is only 53.2353% which is the lowest rate compared to results of other two methods. Out of 360 samples, 181 samples are correctly classified. Three classes 5, 8, and 11 have 100% accuracy rate, but there are four classes 3, 4, 7, and 35 where accuracy rate is 0%. Confusion matrix can be seen in Fig.7, and the performance plot in Fig.8.



```
  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab ac ad
  8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  3  0  0  0  0  0  0  0 |   a = 1
  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  2  0  0  6  0  0  0  0 |   b = 2
  1  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  1  0  0  0  0 |   c = 3
  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0 |   d = 4
  0  0  0  0 12  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   e = 5
  0  0  0  0  0  7  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   f = 6
  0  1  0  0  2  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  2  0  0  3  0  0  0  0 |   g = 7
  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   h = 8
  0  0  0  0  0  0  1  0 11  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0 |   i = 9
  0  0  0  0  0  0  0  0  1  8  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0 |   j = 10
  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   k = 11
  0  0  0  0  1  0  0  0  0  0  0  7  0  2  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0 |   l = 12
  1  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  1  0  0  3  0  0  0  0  0  1  0  0  0 |   m = 13
  0  0  0  0  0  0  0  0  0  0  0  3  0  4  0  0  0  0  0  0  0  4  0  0  0  1  0  0  0  0 |   n = 14
  0  0  0  0  0  0  0  0  1  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |   o = 15
  0  0  0  0  2  0  0  0  0  0  0  1  0  0  0  6  0  0  0  0  0  0  1  0  0  2  0  0  0  0 |   p = 22
  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  8  1  0  0  0  0  0  0  0  0  0  0  0  0 |   q = 23
  1  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  8  0  0  0  0  2  0  0  0  0  0  0  0 |   r = 24
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  6  0  0  2  0  0  0  0  0  0  0  0 |   s = 25
  0  1  1  0  0  0  0  0  0  1  0  0  1  0  0  0  0  4  0  1  1  0  0  0  0  2  0  0  0  0 |   t = 26
  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  6  0  0  0  0  1  0  0  0  0 |   u = 27
  0  0  0  0  0  0  0  0  0  0  0  3  0  6  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0 |   v = 28
  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0 |   w = 29
  0  0  0  0  0  0  0  0  6  2  0  0  0  0  0  0  0  1  0  0  0  0  0  3  0  0  0  0  0  0 |   x = 30
  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  5  0  0 |   y = 31
  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  3  0  0  5  0  0  0  0 |   z = 32
  1  0  0  0  0  0  0  0  0  1  0  0  6  0  0  0  0  0  0  0  1  0  0  0  0  0  2  0  0  0 |  aa = 33
  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  8  0  0 |  ab = 34
  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  1  0  0  0  0  0  4  0  0  0  0  0  0  0  0 |  ac = 35
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9 |  ad = 36
```

Figure 7. Confusion matrix – Support Vector Machine

The reason why accuracy rate is low is the large number of output classes due to which training time is increased, and it does not perform very well when the data set has more noise i.e. target classes are overlapping. Therefore, multilayer perceptron is keener to the multiclass problems rather than support vector machine.

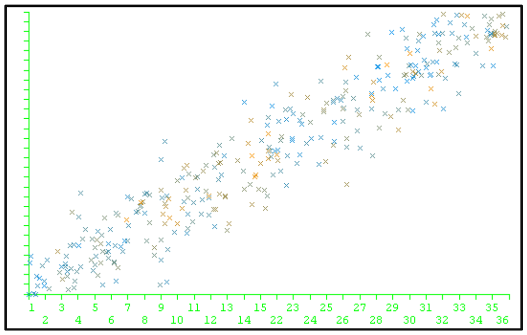


Figure 8. Classification performance plot – Support Vector Machine

4. CONCLUSION AND FUTURE WORK

The evaluation of algorithms used for leaf type multiclass classification has been presented in Table 2.

Table 2. Performance evaluation of algorithms

	Accuracy rate	Training time
Multilayer Perceptron	79.71%	7.77 s
Naive Bayes	74.12%	0.01 s
Support Vector Machine	53.235%	3.73 s
Best result:	*Multilayer P.*	*Naive Bayes*

Accuracy rates obtained are not better than the accuracy rate achieved while using linear discriminant analysis by Silva et al. On the other hand, the paper by Yasar et al. used linear regression in classification problem and stated 92% accuracy rate [10], but it is to be further discussed whether that method is appropriate for the classification problems of this kind.

Future work on this topic is to be based on improvement of algorithms' accuracy rates in the processes of multiclass classification in order to produce efficient and effective plant recognition system.

REFERENCES

[1] J. Chaki and R. Parekh, "Plant Leaf Recognition using Shape based Features and Neural Network classifiers", International Journal of Advanced Computer Science and Applications, Vol. 2, No. 10, 41-47. 2011.

[2] P.F.B. Silva, A.R.S. Marcal, R.M.A da Silva, "Evaluation of Features for Leaf Discrimination", Springer Lecture Notes in Computer Science, Vol. 7950, 197-204. 2013.

[3] E.J. Pauwels, P.M. de Zeeuw, E. Ranguelova, "Computer-assisted tree taxonomy by automated image recognition". Eng. Appl. of AI Vol. 22, 26–31. 2009.

[4] P.F.B. Silva, "Development of a System for Automatic Plant Species Recognition"(Master's Thesis), Faculdade de Ciˆencias da Universidade do Porto. Available for download or online reading at http://hdl.handle.net/10216/6773. 2013.

[5] "Multilayer Perceptron", *Weka.sourceforge.net*, 2016. [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html. [Accessed: 17-May- 2016].

[6] M. Nielsen, *Neural Networks and Deep Learning*, "Chapter 2: How the backpropagation algorithm works", Determination Press, 2016.

[7] C. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

[8] "SMO", *Weka.sourceforge.net*, 2016. [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMO.html. [Accessed: 28- May- 2016].

[9] L. Wang, *Support vector machines*. Berlin: Springer, 2005.

[10] A. Yasar, I. Saritas, M. Sahman and A. Dundar, "Classification Of Leaf Type Using Artificial Neural Networks", International Journal of Intelligent Systems and Applications in Engineering, vol. 3, no. 4, pp. 136-139, 2015.