Environmental Impact on Predicting Olive Fruit Fly Population Using Trap Measurements

Romanos Kalamatianos(✉), Katia Kermanidis, Markos Avlonitis, and Ioannis Karydis

Department of Informatics, Ionian University, 49100 Corfu, Greece
{cl4kala, kerman, avlon, karydis}@ionio.gr

Abstract. Olive fruit fly trap measurements are used as one of the indicators for olive grove infestation, and therefore, as a consultation tool on spraying parameters. In this paper, machine learning techniques are used to predict the next olive fruit fly trap measurement, given as input environmental parameters and knowledge of previous trap measurements. Various classification algorithms are employed and applied to different environmental settings, in extensive comparative experiments, in order to detect the impact of the latter on olive fruit fly population prediction.

Keywords: Olive fruit fly · Machine learning · Population prediction · Classification · Naive bayes · Nearest neighbors · Decision trees · Random forests · Support vector machines

1 Introduction

The olive fruit fly is a pest that has been recorded to infest solely the olive fruits since at least the third century BC [1]. Such infestations cause great damage to the production of both olive oil or table olives [2] in many olive oil producing countries, including Greece. The olive fruit fly is active during the summer and reaches its population peak during autumn, while during the winter and in the first months of spring it hibernates, until environmental conditions are favorable for it to reemerge [1].

The population growth of the olive fruit fly and, by extension, the level of infestation of an olive grove are affected by various environmental factors. However, the two primary factors that affect the activity of the olive fruit fly are temperature [3] and relative humidity [4, 5].

Population control of the olive fruit fly can be achieved through spraying of the olive trees, either with bait or universal [1, 6]. However, in order for the spraying to have any effect, it has to be applied when conditions are appropriate. Two factors indicate when spraying should commence [6]: (a) the ripeness level of the olive fruit, as the fruit needs to be ripe in order for it to be susceptible to the olive fruit fly and (b) the population of the fly, i.e. when a certain population threshold (recorded via sampling) is exceeded. Sampling is achieved through McPhail traps or yellow sticky traps [1, 6]. The threshold is set to seven olive fruit flies per trap per week during the summer and is decreased to five olive fruit flies per trap per week during autumn [6].

L. Iliadis and I. Maglogiannis (Eds.): AIAI 2016, IFIP AICT 475, pp. 180–190, 2016.
DOI: 10.1007/978-3-319-44944-9_16

Machine learning techniques have been used to detect oil spills on the surface of the sea by scanning radar images [7], to automatically identify species by sound [8] and to monitor flood protection systems [9]. Machine learning techniques have also been applied in numerous agriculture processes such as the prediction of when a cow should be culled in a dairy herd [10], the estimation of soil moisture [11], the estimation of a cow's oestrus [12] and the prediction of olive fruit fly infestation using information about olive tree health as well as trap measurements [13].

The aim of this paper is to predict future olive fruit fly trap measurements, and by extension olive fruit fly infestations/outbreaks, using machine learning algorithms. Our approach differentiates from previous work [13] by constructing a feature vector that consists of environmental factors e.g. temperature, instead of the olive tree health, as well as trap measurements.

2 Data Collection

2.1 Environmental Data

The data used in the experiments were collected from environmental sensors and olive fruit fly traps that were installed at 16 locations on the north-western side of the island of Corfu, Greece. Readings on the olive fruit fly traps show the total number of olive fruit flies caught by the trap. Each reading was conducted every five days for the period from 10th June 2015 to 29th September 2015, at all locations. All sensors at all locations logged temperature values at a 15 min interval, while a few of these also logged relative humidity values.

2.2 Feature Selection

In order to perform classification experiments, the aforementioned environmental data were transformed into the following set of attributes (in order to represent readings as feature-value learning vectors):

- Mean temperature of the last five days before next trap reading
- Mean maximum temperature of the last five days before next trap reading
- Mean minimum temperature of the last five days before next trap reading
- Day 1 Mean Temperature
- Day 1 Maximum Temperature
- Day 1 Minimum Temperature
- Day 2 Mean Temperature
- Day 2 Maximum Temperature
- Day 2 Minimum Temperature
- Day 3 Mean Temperature
- Day 3 Maximum Temperature
- Day 3 Minimum Temperature
- Day 4 Mean Temperature
- Day 4 Maximum Temperature

- Day 4 Minimum Temperature
- Day 5 Mean Temperature
- Day 5 Maximum Temperature
- Day 5 Minimum Temperature

Apart from the environmental attributes, one more attribute, namely the trap measurement of the last reading (number of flies caught), was used as input. All aforementioned attributes are numeric. Finally another attribute, denoting the next trap reading was used as the classification class.

2.3 Feature Vector Extraction

The process of extracting the attributes for the feature vectors from the sensor data was automated by use of a script. The script was written in Python that automatically computes the mean, mean maximum and mean minimum temperature for the five day period before the next trap reading, as well as the mean, maximum and minimum temperature for each day in the aforementioned five day period. The script exports all vectors in a CSV (Comma Separated Values) file. Finally trap readings are added manually at each corresponding vector instance.

The temperature-related attributes, initially numeric, were discretized into the following three bins:

- <15, temperature is lower than 15 °C,
- 15 to 32, temperature is between 15 °C and 32 °C,
- >32, temperature is greater than 32 °C.

The discretization of the temperature values was based on the temperature range (between 15 °C and 32 °C [14]), in which the olive fruit fly is active. If the temperature of the environment is below the lower or exceeds the upper threshold, then the olive fruit fly is motionless due to extreme cold or heat. Accordingly, herein we assume that outside the optimal temperature range of the olive fruit fly, the traps will not capture any olive fruit flies.

Trap reading related attributes have also been discretized into the following bins:

- 0 to 4, none or up to 4 olive fruit flies inside the trap,
- 5 to 6, five or six olive fruit flies inside the trap,
- >=7, greater than or equal to seven olive fruit flies inside the trap.

The use of a ternary quantisation of the number of olive fruit flies is based on trap measurements analysis, i.e. the infestation threshold depends on the season the measurements are made. Specifically, in the summer months the infestation threshold is set to seven olive fruit flies per trap per week. On the other hand, from September onwards the infestation threshold is decreased to five olive fruit flies per trap per week, due to cooler weather [6]. Therefore, although the last bin value would always indicate infestation, the second bin value would be depended on the season.

3 Machine Learning Algorithms

The WEKA machine learning workbench[1] was used for running the classification experiments. In the sequel, a number of classification algorithms that were selected for experimentation are shortly presented.

J48 [15] is a decision tree induction algorithm and it is a version of C4.5, an earlier algorithm developed by J. Ross Quinlan [16]. C4.5 generates a decision tree based on information gain of the attributes in the available training data. More specifically, the attribute whose values discriminate most clearly the training examples according to their class label is identified in each iteration. The algorithm stops when there are no further attributes to explore or when all the training examples are separated according to their class label. Additionally, J48 incorporates two tree pruning methodologies: The first one is known as subtree replacement and it replaces a node in a decision tree with the corresponding leaf, if the given subtree does not help classification accuracy. This pruning process starts from the leaves of the fully formed tree, and moves bottom up toward the root. The second methodology is known as subtree raising in which a node may replace other nodes while it is moved towards the root. This type of pruning most of the times has insignificant effect on decision tree models (Table 1).

Table 1. J48 parameter values

Binary splits	No
Confidence factor	0.25
Minimum instances per leaf	2
Reduced error pruning	No
Subtree raising	Yes
Pruned	Yes
Laplace smoothing	No

Sequential Minimal Optimization or SMO [17] is an ameliorated algorithm for training support vector machines. SMO cuts in pieces a large quadratic programming optimization problem converting it into smaller problems (sub-problems of quadratic programming). The sub-problems are solved quickly because they are solved analytically which means that SMO avoids to use extra time for arithmetical quadratic programming optimization as an inner loop. So SMO manages to reduce computation time significantly (Table 2).

Naïve Bayes [15] is a probabilistic classifier based on the assumption of conditional independence [18], which assumes that the appearance of a specific feature given the class value is unrelated to the appearance of any other feature in the dataset. Though not valid in reality, this assumption has been proven to cope well with several classification problems. Additionally, this algorithm needs a small amount of training data to determine the parameters necessary for classification. Due to the hypothesis of

[1] http://www.cs.waikato.ac.nz/ml/weka/.

Table 2. SMO parameter values

Complexity parameter	1.0
Round-off error	1.0E-12
Filter type	Normalize training data
Kernel	PolyKernel
Random seed for cross validation	1
Tolerance parameter	0.001

Table 3. Naive Bayes parameter values

Use kernel estimator	No
Use supervised discretization	No

independent variables; there is no need to estimate the entire covariance matrix but only the differentiations of the variables for each class (Table 3).

The RandomForest [19] is a meta-learning classification algorithm that runs iteratively. In each iteration a decision tree is induced from a randomly selected subset of the features. The number of iterations is pre-defined. The final classification error is the mean error over all iterations (Table 4).

Table 4. RandomForest parameter values

Maximum depth	Unlimited
Number of attributes	0
Number of trees to be generated	100
Seed	1

AdaBoost [20] is another meta-learning algorithm, that iteratively changes instance weights, based on whether they were classified correctly (or not) in a previous iteration. Thereby, the learner is forced to focus on instances that are hard to classify. The final classification is derived from the weighting of the models induced after every iteration (Table 5).

Table 5. AdaBoost parameter values

Classifier	SMO
Number of iterations	10
Seed	1
Use resampling	No
Weight threshold	100

The IBk algorithm [15] is an alternate version of the k-nearest neighbor algorithm. Using the Euclidean distance as a distance metric, it identifies the k training examples that are closest to a given test instance, and, via majority voting selects its class label.

The value of k can be explicitly pre-defined, or estimated optimally using cross validation. In our experiments, the number of nearest neighbors ranged from 1 to 33 neighbors, where only odd values were selected (Table 6).

Table 6. IBk parameter values

Cross validate	No
Distance weighting	No
Use of mean squared error	No
Nearest neighbor search algorithm	LinearNNSearch
Window size	0

The Multilayer Perceptron is an artificial neural network [21], where each of the multiple layers of nodes is fully connected to the following one. Multilayer perceptrons are able to distinguish data that are not linearly separable. Experiments were conducted for one hidden layer with the number of nodes ranging from 1 to 10 (Table 7).

Table 7. Multilayer Perceptron parameter values

Decrease learning rate	No
Hidden layers	1
Learning rate	0.3
Momentum	0.2
Nominal to binary filter	Yes
Normalize attributes	Yes
Normalize numeric class	Yes
Reset	Yes
Seed	0
Training time	500
Validation set size	0
Validation threshold	20

4 Experimental Process

184 training instances were supplied. Due to the small size of the training data, no test set could be supplied for the validation of the results. Therefore the 10-fold cross-validation method was used. The original sample is randomly partitioned into ten subsamples. One out of ten subsamples is kept as validation data for testing the model, and the remaining nine subsamples are used as training data. The cross-validation process is then repeated ten times, with each of the ten subsamples being used only once as validation data. Results are averaged across the ten experiments.

5 Evaluation

Figure 1 displays the classification results of the five aforementioned machine learning algorithms. The SMO algorithm produces the best results in both precision and recall, with the J48 algorithm being close by. AdaBoostM1 produces the same results with SMO using as a base learner the SMO algorithm. On the other hand, NaiveBayes produces the worst results in comparison with the other algorithms, with a significant decrease in precision and a quite noteworthy low recall.

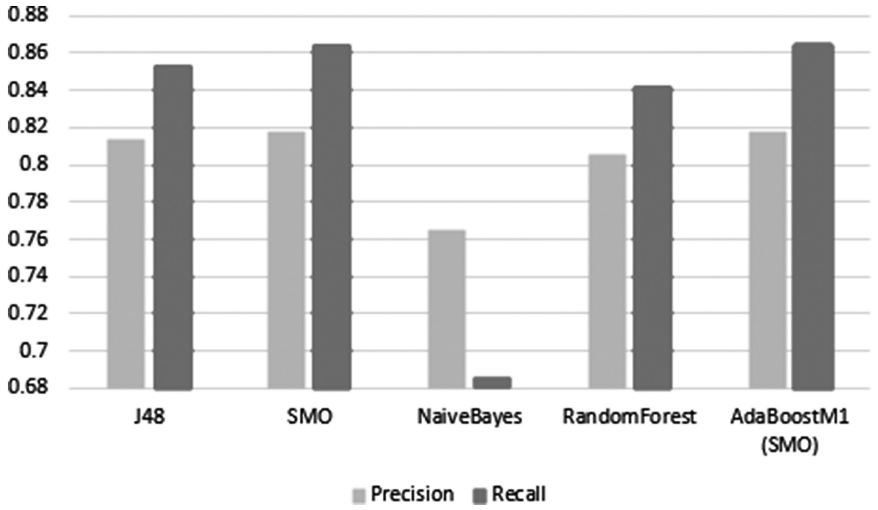

Fig. 1. Precision and recall comparison for five classification algorithms

Figure 2 presents the pruned tree constructed by the J48 algorithm. It is clear that the algorithm considers the "previous reading" as the most important attribute for classification. With the minimum temperature of day 3 being used when the previous reading has a value of "5 to 6".

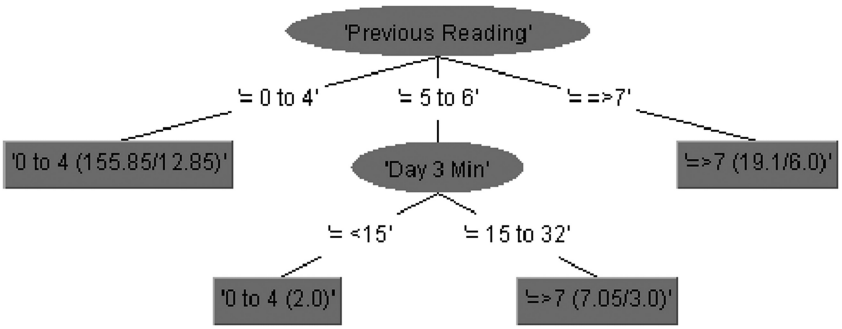

Fig. 2. Tree view constructed by J48

The next experiment is using the IBk algorithm, as shown in Fig. 3, with maximum recall for one and for five nearest neighbors. The best results in both recall and precision are achieved when using one nearest neighbor. From three nearest neighbors and onwards the precision of the algorithm decreases, with a significant drop after nine nearest neighbors. After eleven nearest neighbors, precision and recall stay stable.

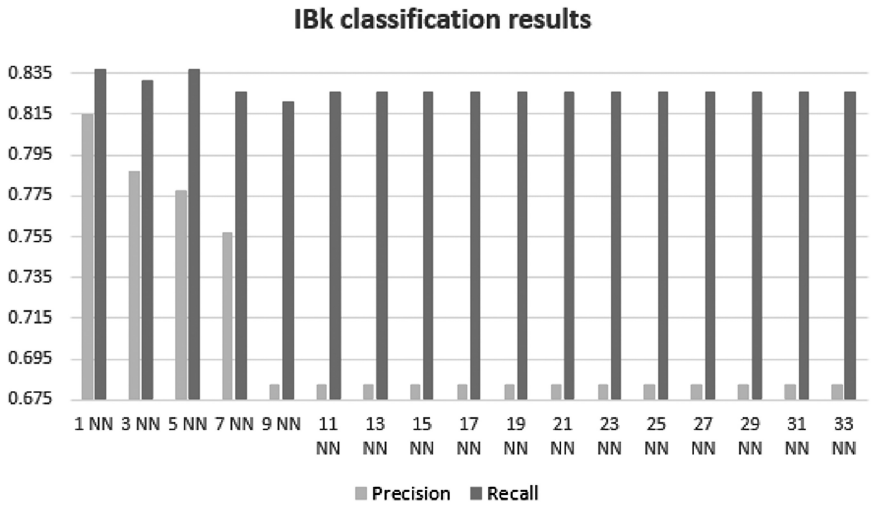

Fig. 3. IBk classification results for different number of k neighbours

Finally, the Multilayer Perceptron algorithm, shown in Fig. 4, exhibits a great difference between recall and precision values for any number of nodes in the

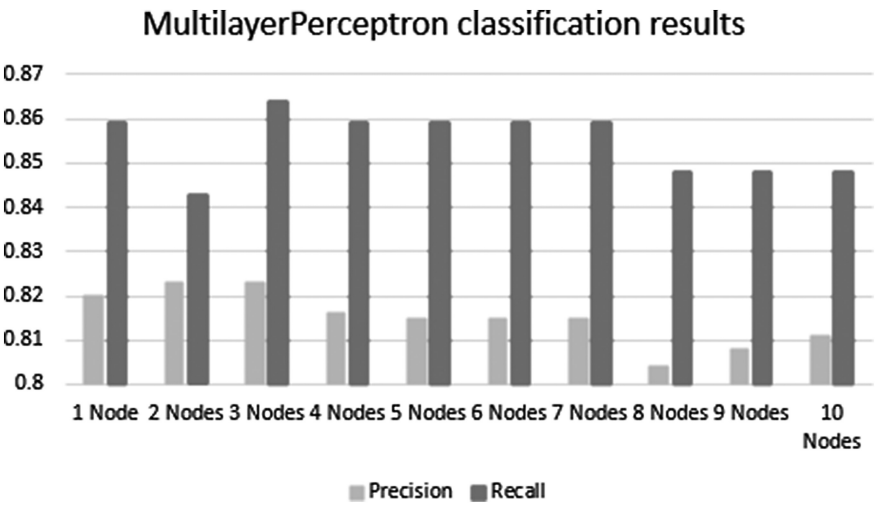

Fig. 4. Multilayer Perceptron classification results for different number of nodes

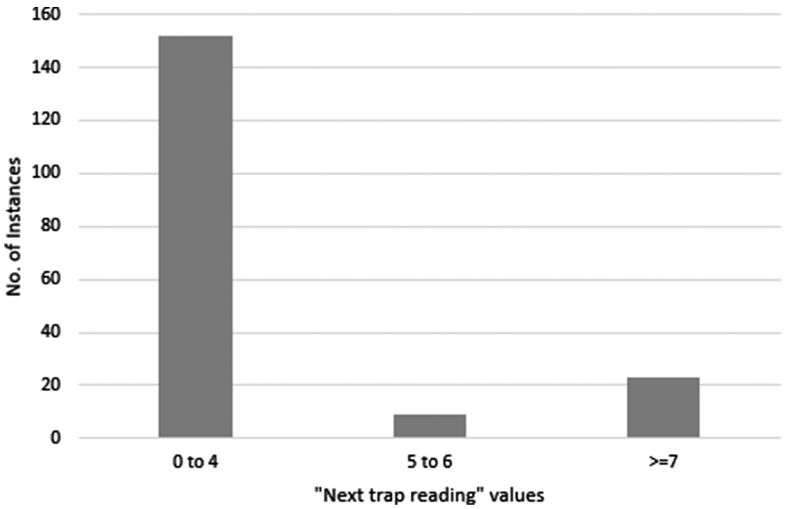

Fig. 5. Value distribution of the "next trap reading" classification attribute

perceptron. The best results for both precision and recall are reported for three nodes in the hidden layer.

Comparing all machine learning algorithms that were used in the experiments, the best performance was achieved by SMO and AdaBoostM1, using SMO as a base learner.

It is important to note that the values of the classification attribute are not balanced. Figure 5 depicts the distribution of the three values among the 184 instances. Over 80 % of the instances have a value of "0 to 4", while the value "5 to 6" is the least encountered

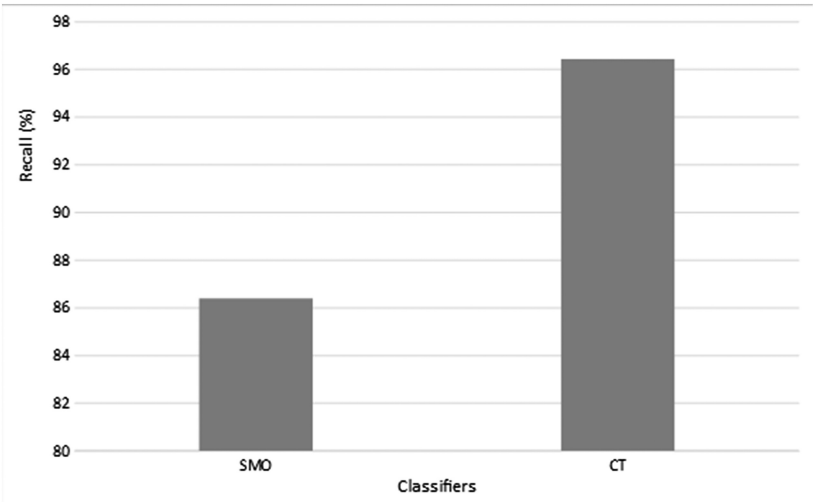

Fig. 6. Results comparison with previous work [13]

value, with nine instances. This fact explains why the J48 algorithm, as shown by the constructed tree in Fig. 2, can't classify any instance to the value "5 to 6".

Figure 6 shows the comparison of the best classifier presented herein against the best classifier of [13] in terms of recall. It is obvious that the Classification Trees (CT) have a far better recall compared to the SMO algorithm by almost 10 %. This difference is attributed to the significant imbalance in the distribution of the class value in the training data. Additionally, the results produced by the CT were achieved using only six attributes and the classification class had only two values, whether to treat or not. Finally, a number of the used attributes contained information about the phenological state of the olive tree.

6 Conclusion

In this work, supervised machine learning is used to predict future olive fruit fly trap's measurements. The proposed feature vector consists of environmental parameters, specifically temperature, and information about previous trap measurements. Results produced by the conducted experiments were promising with the support vector machine algorithm providing the best classification results, although the values of the classification attribute were unbalanced.

Our approach differentiates from previous work by using environmental parameters instead of information about the health of the olive trees. When results between the best classifiers of the proposed and existing work were compared, the proposed approach produced results with 10 % ameliorated recall.

Future research will take into account more environmental parameters such as relative humidity and the amount of light the olive fruit flies are exposed to. Furthermore, the experiments described are planed to be conducted again in more training instances as measurement data accumulate.

Acknowledgments. Financial support of the European Union and of National Funds of Greece and Albania under the IPA Cross-Border PROGRAMME "Greece - Albania 2007–2013", project title "Enhancing Olive Oil Production with the use of Innovative ICT" with the acronym "e-Olive", is gratefully acknowledged.

References

1. Vossen, P., Varel, L., Devarenne A.: Olive fruit fly. University of California Cooperative Extension, Sonoma County (2006)
2. Rice, R.: Bionomics of the olive fruit fly bactrocera (Dacus) olea. University California Plant Protection Quarterly **10**, 1–5 (2000)
3. Fletcher, B.S.: Temperature development rate relationships of the immature stages and adults of tephritid fruit flies. In: Robinson, A.S., Hooper, G. (eds.) Fruit Flies: Their Biology, Natural Enemies and Control, vol. 3A, pp. 273–289. Elsevier, Amsterdam (1989)

4. Yokoyama, V.Y., Rendon P., Sivinski, J.: Biological control of olive fruit fly (Diptera: Tephritidae) by reseases of Psyttalia cf. concolor (Hymenoptera: Braconidae) in California, parasitoid longevity in presence of the host, and host status of walnut husk fly. In: 7th International Symposium on Fruit Flies of Economic Importance, pp. 157–164. Salvador, Brazil (2006)
5. Broufas, G.D., Pappas, M.L., Koveos, D.S.: Effect of relative humidity on longevity, ovarian maturation, and egg production in the olive fruit fly (Diptera: Tephritidae). Ann. Entomol. Soc. Am. **102**(1), 70–75 (2009)
6. Patsias, A.: EE Katapolmeesee tou THkou tee Elee [The Fighting of Olive fruit fly]. Publicity Department of Agricultural Sector Applications and Publicity. Nicosia, Cyprus (2005)
7. Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Mach. Learn. **30**, 195–215 (1998)
8. Acevedo, M.A., Corrada-Bravo, C.J., Corrada-Bravo, H., Villanueva-Rivera, L.J., Aide, T.M.: Automated classification of bird and amphibian calls using machine learning: a comparison of methods. Ecol. Inf. **4**, 206–214 (2009)
9. Pyayt, A.L., Mokhov, I.I., Lang, B., Krzhizhanovskaya, V.V., Meijer, R.J.: Machine learning methods for environmental monitoring and flood protection. World Acad. Sci. Eng. Technol. **78**, 118–124 (2011)
10. McQueen, R.J., Garner, S.R., Nevill-Manning, C.G., Witten, I.H.: Applying machine learning to agricultural data. Comput. Electron. Agric. **12**(4), 275–293 (1995)
11. Ahmad, S., Kalra, A., Stephen, H.: Estimating soil moisture using remote sensing data: a machine learning approach. Adv. Water Res. **33**(1), 69–80 (2010)
12. Mitchell, R.S., Sherlock, R.A., Smith, L.A.: An investigation into the use of machine learning for determining oestrus in cows. Comput. Electron. Agric. **15**(3), 195–213 (1996)
13. del Sagrado, José, del Águila, I.M.: Olive fly infestation prediction using machine learning techniques. In: Borrajo, D., Castillo, L., Corchado, J.M. (eds.) CAEPIA 2007. LNCS (LNAI), vol. 4788, pp. 229–238. Springer, Heidelberg (2007)
14. Vossen, P.: Monitoring and Control of Olive Fruit Fly (OLF) for Olive Production in California. University of California Cooperative Extension (2014)
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
16. Quinlan, J. R.: Bagging, boosting and C4.5. In: 13th National Conference on Artificial Intelligence, pp. 725–730. AAAI Press, Portland (1996)
17. Platt, J.C., Sequential minimal optimization: a fast algorithm for training support vector machines. Technical report MSR-TR-98–14, Microsoft Research (1998)
18. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, Upper Saddle River (2009)
19. Breiman, L.: Random Forests. Mach. Learn. **45**, 5–32 (2001)
20. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156 (1996)
21. Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington, D.C. (1961)