Evaluating multiple regressors for the yield of orange orchards

Kleber X. S. de Souza[1], **Sônia Ternes**[1], **João Camargo Neto**[1], **Thiago T. Santos**[1], **Alécio Souza Moreira**[2], **Luciano V. Koenigkan**[1], **Roberta de Souza**[1]

[1]Embrapa Agricultura Digital
Av. André Tosello, nº 209 Campus da Unicamp, Barão Geraldo
Caixa Postal: 6041 CEP: 13083-886 - Campinas - SP

[2]Embrapa Mandioca e Fruticultura
Rua Embrapa s/no – Caixa Postal 007 – 44380-000 – Cruz das Almas – BA – Brasil

{kleber.sampaio, sonia.ternes, joao.camargo, thiago.santos}@embrapa.br
{alecio.moreira, luciano.vieira}@embrapa.br, robplaceres@gmail.com

Abstract. *Accurate fruit yield estimation is crucial for making informed decisions about harvesting, storage and marketing. However, estimating fruit yield can be challenging. Currently, yield estimation relies on labor-intensive manual counting combined with statistical methods. However, computer vision has emerged as a potential alternative by enabling automatic fruit counting, thus simplifying the process. In this paper, we assess the effectiveness of various machine learning regressors for yield forecasting based on fruit detection in images captured within the orchard. Our results indicate that deep forward neural networks outperform the other regressors examined, making them the most effective choice for yield prediction.*

1. Introduction

Agriculture plays a vital role in Brazil's economic development, contributing significantly to the country's GDP and generating substantial revenue and employment opportunities. In agriculture, citrus cultivation is a key sector. Brazil is responsible for more than half the world's production of orange juice, making it a key player on the international market.

Predicting orange yield is crucial for making informed decisions about harvesting, storage, and marketing. However, the annual orange crop forecast process, conducted by Fundecitrus [Fundecitrus 2022], involves a labor-intensive operation of manually felling fruits from samples of approximately 1,500 trees spread across the citrus belt. These sampling and counting procedures are widely used to collect pre-harvest fruit yield data. Due to the high cost and labor-intensive nature of this process, there is a need to explore alternative methods to streamline the forecast.

One promising approach is the use of automatic fruit identification, tracking, and counting methods. These methods leverage computer vision and image analysis techniques to automate the process, reducing the reliance on manual labor and improving efficiency. By accurately identifying, tracking, and counting fruits, these automated methods offer a more cost-effective and time-efficient solution for forecasting orange yield. Therefore, counting the number of fruits is essential to estimate harvest.

Nonetheless, computer vision and image analysis are not sufficient for predicting orange yield. It has been observed that a portion of the fruits are located inside the inner part of the plant canopy, making them invisible in the images collected in the field, even considering multiple views of the same side of the plant. As a consequence, there is a need for a regressor that can fill the gap and estimate the actual number of fruits based on the identified and tracked fruits.

The main contribution of this work is the application and evaluation of different types of machine learning-based regressors to estimate the number of fruits. The performance of these regressors is assessed based on their coefficient of determination.

This is organized as follows: Section 2 provides background information and related literature on the subject. Section 3 presents the data set used and the regressors studied. Section 4 shows the results obtained for each regressor. Finally, Section 5 presents our concluding remarks.

2. Related work

The prediction of yield is the final step in a pipeline that utilizes computer vision techniques to identify and count oranges in images. The process begins with the application of convolutional neural networks (CNN) to detect and locate oranges in field images. The identified fruits are then tracked across adjacent images to ensure accurate counting. Finally, the number of fruits is combined with other variables such as plant height, variety, and additional factors described in Section 3.

For detailed information on the fruit identification process and the algorithms used, please refer to the following references: [Camargo Neto et al. 2019], [Cerqueira et al. 2020], and [Sousa et al. 2021]. These sources provide comprehensive insights into fruit identification methods, including the use of YOLO (You Only Look Once), Faster R-CNN (Region-based Convolutional Neural Networks), and SSD (Single Shot MultiBox Detector) algorithms. This paper specifically focuses on the yield estimation step of the pipeline.

The methods for yield estimation vary depending on the type of data collected or the crop. [Wulfsohn et al. 2012] sample trees from an orchard and select branches and segments of branches using a multi-stage systematic sampling approach. They then apply statistical methods to propose a yield estimator.

On the other hand, [Häni et al. 2020] used machine learning and tracking to provide a direct estimation of fruits. In their case, the structure of the plants makes the fruits visible on the surface of the canopy, allowing for direct counting. Direct estimation of fruits based on images was also performed in [Maldonado and Barbosa 2016].

In our problem, direct estimation is not viable because we identified a large gap between the visible fruits and the total number of fruits obtained through manual harvesting. This gap may result from the fact that some fruits are located inside the canopy.

To address this issue, we evaluate regressors for the yield that considers the number of fruits identified by computer vision algorithms and other relevant variables, as detailed in Section 3. A similar approach was used in estimating mango fruit load, where [Koirala et al. 2019] applied a correction factor per orchard to estimate the fruit load per orchard. However, unlike using a regressor, as we are proposing, this correction factor is calculated per orchard based on the average ratio of the number of fruits identified in images to the manual harvest count per tree in that orchard.

3. Materials and methods

The plant images, along with other relevant data, used to train the regressors, were obtained by Fundecitrus, our partner in the research project[1]. Although we received raw data for 1,543 trees, only 1,197 trees could be successfully processed in the previous stages of the pipeline, due to problems while processing the images. Table 1 shows the first four lines of the dataset. The previous stages of the pipeline generated the last two columns, *CbyT-A* and *CbyT-B*, which correspond respectively to the fruits automatically counted on sides A and B of the plant. These two columns were aggregated to other data about the plants and the complete information was provided to the regressor. However, not all 1,197 records could be used for training the regressor, as we had to exclude plants presenting missing data and those without fruit counting from tracking for either side, resulting on a final dataset of 1,139 trees.

ID	Sector	Region	VarG	Var.	AG	F1	F2	F3	F4	H	W	D	CbyT-A	CbyT-B
103	Norte	LIM	VT	*Val.*	1	0	5	26	1	280	280	300	15	
1031	Sul	LIM	NT	*Nat.*	3	0	5	54	10	290	360	295		19
106	Norte	DUA	PRME	*Pera*	2	0	0	58	36	340	300	335	10	10
104	Norte	DUA	PRME	*Pera*	3	400	0	11	0	310	460	495		32

Tabela 1. Sample containing the first four lines of the dataset used for training and testing of the regressor. ID corresponds to the plant identifier; Sector and Region are references for geographical location; VarG and Var refer to the variety group and variety of the plant, respectively; AG is a classification for the age of the plant; F1, F2, F3 and F4 are the number of fruits manually counted in sizes corresponding to the first to the fourth flowering; H, W and D are the dimensions of the plant; CbyT-A and CbyT-B are the number of visible fruits automatically counted using the vision-based pipeline for both sides of the plant.

3.1. Data preprocessing

Before submitting data to the regressors, it is necessary to perform some operations to adequate these data to the dynamics of the processing of those regressors. The following operations were executed over the dataset:

Transformation of categorical variables: most machine learning algorithms only accept numerical values as inputs to their neurons, which means that categorical variables must be transformed into numerical values [Géron 2019]. For example, the value *Hamlin*, which is one of the possible values for the variable VARIETY, has to be converted to a number. In this work we use One Hot Bit Encoding [Géron 2019] in this transformation.

[1]Projeto SEG 10.18.03.016.00.00 - "Estimativa de quantidade de frutos em pés de laranja por meio de inteligência computacional".

Standardization of numerical values: to facilitate convergence to the optimal training point, numerical values should be standardized. Variables with very discrepant scales, such as one ranging from [0–100,000] and another from [-1.0–1.0], can cause problems because the gradients calculated on model's parameters will have very different effects when applied to different scales. This work employs standardization of numerical values [Géron 2019].

3.2. Machine learning regression models

In this paper, we are evaluating several machine learning algorithms for the regression problem, such as Support Vector Regression [Cortes and Vapnik 1995, Drucker et al. 1997], Random Forest [Breiman 2001], Gradient Boosting [Friedman 2001] and multilayer feed-forward neural network [Goodfellow et al. 2017]. The models were implemented in the Python programming language, using SciKit-Learn[2] and Keras[3] libraries. Due to lack of space, we will not provide details of each algorithm, but only a brief description. The interested reader should refer to the literature.

The regression problem involves find the minimum of a *loss function*, which represents the error distance $\mathcal{L}(f(x), y)$ between the value projected by the regression function $f(x)$ and the true value y present on the dataset. This loss function is implemented using appropriate metrics such as mean absolute error (MAE), Root Mean Squared Error (RMSE), or coefficient of determination (R^2), depending on the specific requirements of the problem. In this work, we will be using these three measures to compare the performance of the models.

Support Vector Regression: Support Vector Regression (SVR) is a machine learning algorithm that combines regression techniques with Support Vector Machines (SVM). It is specifically designed for solving regression problems where the goal is to predict continuous numeric values rather than discrete class labels. SVR utilizes kernels, which allow it to work in high-dimensional feature spaces without explicitly calculating the coordinates of the data points. The core commands in Python to use RBF are presented in the box below. Among the kernel functions tested, the Radial Basis Function (RBF) presented better results compared to the polyline and linear kernels. The parameter C is a regularization parameter that controls the trade-off between the model's simplicity and its accuracy on the training data. By setting it to 100 we are weighting more the training data. Setting Gamma to 'auto' means that the kernel coefficient for the RBF is automatically determined based on the number of features (n_features). Additionally, Epsilon=0.1 limits the penalty margin accepted in the training loss function.

```
from sklearn.svm import SVR
model = SVR(kernel='rbf', C=100, gamma='auto', epsilon=.1) )
```

Random Forest: Random Forest is an ensemble learning technique used for both classification and regression tasks. For a regression task, each tree in the Random Forest predicts a continuous value for the input sample. The final predicted value is usually the average (or weighted average) of the individual tree predictions.

[2]https://scikit-learn.org/
[3]https://keras.io/

The Random Forest Regressor was trained with 10, 20 and 50 estimators, with a maximum depth of 10 for each tree. The stopping criterion was the Mean Square Error (MSE).

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=20, max_depth=10, criterion='mse')
```

Gradient Boosting: Gradient Boosting is also an ensemble learning technique used for both classification and regression tasks. It builds multiple weak learners (typically decision trees) sequentially, where each new learner corrects the errors made by the previous ones. The first step is to initialize the model with a base learner, which can be a simple model like a decision tree. The base learner makes predictions based on some initial rule, such as the mean value for regression. After the initial predictions are made by the base learner, the algorithm calculates the residuals, i.e. the differences between the true target values and the initial predictions made by the base learner. A new weak learner is trained to predict the residuals rather than the original target values, and its predictions are added to the ensemble with appropriate weighting. The final prediction for regression tasks is obtained by summing up the predictions from all the weak learners in the ensemble. The Gradient Boost Regressor was trained with 100, 200, 1.000 and 2.000 estimators, all of them starting with $random\_state = 0$ to allow for reproducible output across multiple runs.

```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor(random_state=0,n_estimators=1000)
```

Multilayer Feed-forward Neural Network: Neural networks are highly flexible in their assembly. This flexibility is also a weakness, as it necessitates searching through a wide range of configurations to find the most suitable one for a specific problem [Géron 2019]. A common heuristic approach is to gradually increase the number of layers and neurons and evaluate how the results converge towards the desired outcome. We conducted experiments with neural networks comprising one to six layers, and varying the number of neurons per layer from seven to 28. The last layer of the regressor consists of a single neuron, which is responsible for the calculation of the yield estimation. Among the tested architectures, the one presented in Table 2 demonstrated the best results.

Layer (type)	Output Shape	Num. Params
dense_1 (Dense)	$(B, 7)$	294
batch_normalization_1	$(B, 7)$	28
dense_2 (Dense)	$(B, 7)$	56
dense_3 (Dense)	$(B, 1)$	8
Total params: 386		
Trainable params: 372		
Non-trainable params: 14		

Tabela 2. Feed-forward neural network architecture employed on yield regression. B corresponds to the batch size, i.e., the number of data entries.

3.3. Cross validation

In our pursuit of finding the best parameter configuration, we relied on *cross validation*, a widely used technique in machine learning, to statistically assess whether one computational experiment performs better than another [Kohavi 1995]. Cross validation involves running the same algorithm multiple times, allowing for more reliable performance statistics and a certain level of confidence.

For our research, we divided the training dataset into ten parts, also known as folds. The process entails training the algorithm on nine folds while reserving the tenth fold for evaluation. This process repeats until all ten folds have been used as the test set. To determine the level of certainty regarding the superiority of a model, we employed a *statistical hypothesis test*. Since we had ten folds, the assumption that the data follow a normal distribution does not hold true. The normal distribution is typically reserved for datasets larger than 30 elements with known variance [King and Zisserman 2019]. Instead, we used the t-Student distribution for our set of ten performance measurements. We set a p-value of 0.05 to ensure that we select a model with 95% confidence.

4. Results and discussion

Using the dataset of 1,139 plants, the regressors were specifically designed to estimate the sum of fruits from the first to the third flowering, represented by the sum $F1 + F2 + F3$. The fruits of the fourth flowering are too small to be detected by the previous stages of the pipeline. Out of the 1,139 fruits, 911 were used for training of the neural network regressor, and the remaining 228 were reserved for testing.

As part of the cross-validation process, each model was trained 10 times for each parameter configuration, and the test set was kept separate for final evaluation, not belonging to any of the folds used in cross-validation. The results of the four models using the test set are presented in Table 3. Among the tested regressors, Support Vector Regression and Feed-forward Neural Networks showed similar performance, both achieving $R^2 = 0.61$. Random Forest yielded $R^2 = 0.53$, while Gradient Boosting had the lowest performance with $R^2 = 0.47$.

This tendency is confirmed in other performance measures as well. The Mean Absolute Error (MAE) is better in Feed-forward Neural Networks (185.216) when compared with Support Vector Regression ($MAE = 190.37$), Random Forest ($MAE = 201.66$) and Gradient Boosting ($MAE = 211.03$). As for the Root Mean Squared Error (RMSE), the results also favor Feed-forward Neural Networks (241.40) when compared with Support Vector Regression ($RMSE = 288.27$), Random Forest ($RMSE = 319.19$) and Gradient Boosting ($RMSE = 325.47$).

Searching for a cause of the low performance of the regressors, we noticed a considerable gap between the total number of fruits detected (CbyT-A + CbyT-B) and the manually counted fruits used for the regressor (F1+F2+F3). This gap might be attributed to the poor quality of the images, resulting in failures during the image processing phase. Notably, a significant number of samples contained fewer than 40% of the counted fruits. To address this issue, we introduced a threshold of acceptance for the images, where (CbyT-A + CbyT-B)/(F1+F2+F3) $\geq$ Threshold.

Implementing a threshold of 30%, the results in Table 3 demonstrate notable improvements in the performance of all regressors. The neural network showed the best performance, reaching $R^2 = 0.85$. The other regressors also improved, with results ranging from 0.77 to 0.79, and the most significant improvement was observed in Gradient Boosting, increasing from 0.47 to 0.78. These findings suggest that the introduction of a threshold for image acceptance effectively enhances the performance of the regressors in estimating orange yield.

The superiority of Feed-forward Neural Networks is again confirmed for the problem at hand. Even in the scenario of 30% threshold implementation. The Mean Absolute Error (MAE) is better in Feed-forward Neural Networks (117.56) when compared with Random Forest ($MAE = 128.46$), Gradient Boosting ($MAE = 134.03$) and Support Vector Regression ($MAE = 145.42$).

As for the Root Mean Squared Error (RMSE), the results also favor Feed-forward Neural Networks (164.33) when compared with Random Forest ($RMSE = 199.90$), Support Vector Regression ($RMSE = 205.93$) and Gradient Boosting ($RMSE = 211.08$).

Algorithm	Global R²	R² 30%	Global MAE	MAE 30%	Global RMSE	RMSE 30%
SVR	0.61	0.77	190.37	145.42	288.27	205.93
Random Forest	0.53	0.79	201.66	128.46	319.19	199.90
Gradient Boosting	0.50	0.78	211.03	134.03	325.47	211.08
FF Neural Network	0.61	0.85	185.21	117.56	241.40	164.33

Tabela 3. Results of the four models tested on the same dataset. The second column shows the global results for the whole dataset, and the third, the results under the restriction that at least 30% of the fruits are visible

5. Conclusion

In this paper, we conducted tests on four machine learning algorithms to estimate the actual number of fruits in a tree, given the number of fruits identified and counted by computer vision methods. This estimation is a crucial part in the harvest estimation process. Among the algorithms tested, Feed-forward Neural Networks exhibited the best performance, achieving $R^2 = 0.85$, provided that at least 30% of the fruits are visible and identified in the collected images. Other algorithms, namely Support Vector Machines, Random Forest and Gradient Boosting, also showed good performances, with R^2 values of 0.77, 0.79 and 0.78, respectively, under the same scenario. The superiority of Feed-forward Neural Networks is also confirmed for other evaluation metrics, namely MAE and RMSE, as one can see in the previous section. We also concluded that the performance of any of these regressors is strictly linked to the quality of the images acquired in the field.

Referências

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Camargo Neto, J., Ternes, S., de Souza, K. X. S., Yano, I. H., and Queiros, L. R. (2019). Uso de redes neurais convolucionais para detecção de laranjas verdes. In *ANAIS DO CONGRESSO BRASILEIRO DE AGROINFORMÁTICA*, Indaiatuba, SP.

Cerqueira, L. M., de Souza, K. X. S., Ternes, S., and Camargo Neto, J. (2020). Usando a rede neural faster-rcnn para identificar frutos verdes em pomares de laranja. In *CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA*, Campinas, SP. Embrapa Informática Agropecuária.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Drucker, H., Chris, Kaufman, B. L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, volume 9, pages 155–161.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5).

Fundecitrus (2022). Relatório de atividades: junho 2021/maio 2022. Technical report, Fundo de Defesa da Citricultura (Fund for Citrus Protection) – Fundecitrus.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated.

Goodfellow, I., Bengio, Y., and Courville, A. (2017). *Deep Learning*. MIT Press.

Häni, N., Roy, P., and Isler, V. (2020). A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, 37(2):263–282.

King, A. P. and Zisserman, A. (2019). *Statistics for Biomedical Engineers and Scientists*. Academic Press, first edition.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1137–1143.

Koirala, A., Walsh, K., Wang, Z., and McCarthy, C. (2019). Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precision Agriculture*, 20:1107–1135.

Maldonado, W. and Barbosa, J. C. (2016). Automatic green fruit counting in orange trees using digital images. *Computers and Electronics in Agriculture*, 127:572–581.

Sousa, M. A., de Souza, K. X. S., Camargo Neto, J., Ternes, S., and Yano, I. H. (2021). Usando a rede neural ssd para identificar frutos verdes em pomares de laranja. In *CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA*, Campinas, SP. Instituto de Zootecnia.

Wulfsohn, D., Zamora, F. A., Téllez, C. P., Lagos, I. Z., and García-Fiñana, M. (2012). Multilevel systematic sampling to estimate total fruit number for yield forecasts. *Precision Agriculture*, 13:256–275.