International Journal of Thin Films Science and Technology

http://dx.doi.org/10.18576/ijtfst/130204

Comparing Supervised Classification Algorithms in Machine Learning for Poverty Prediction

Yassine El aachab[1,*], *Jouilil Youness*[2], *and Mohammed Kaicer*[1]

[1]Laboratory of Analysis Geometry and Applications, Department of Mathematics, Faculty of Sciences, Ibn Tofail University Kenitra, Morocco

[2]Department of Economics, Faculty of Economics and Social Sciences of Mohammedia, Hassan II University of Casablanca, Morocco

Received: 12 Oct. 2023, Revised: 19 Apr. 2024, Accepted: 20 Apr. 2024
Published online: 1 May 2024

Abstract: Due to their capacity to evaluate enormous datasets and generate precise predictions, machine learning algorithms have attracted a lot of interest lately. These algorithms have been used in a variety of fields, including social sciences, finance, healthcare, and marketing. Machine learning algorithms offer a viable method for dividing families into poor and non-poor groups based on pertinent socioeconomic characteristics in the context of poverty studies. This research assesses the performance of various surprised classification algorithms machine learning peculiarly Naïve Bayesian Algorithms, Support Vector Machines, K Nearest Neighbor, Decision Trees, and Logistic Regression and Bagging algorithms in predicting poverty degree. Empirical findings demonstrate that the model with the highest accuracy is Decision Tree, with an accuracy of 0.9961. This means that 99.61% of the instances were correctly classified by Decision Tree. The model with the lowest accuracy is Naive Bayes, with an accuracy of 0.5103. This means that only 51.03% of the instances were correctly classified by Naive Bayes.

Keywords: Machine learning, Classification, Prediction, Poverty.

1. Introduction

Artificial intelligence is a field of computer science that focuses on creating machines that can perform tasks that would typically require human intelligence [1]. AI systems can process large amounts of data and make predictions, recognize patterns, and adapt to new situations [2]. AI is divided into two main categories: narrow or weak AI, which is designed to perform specific tasks, and general or strong AI, which has the ability to perform any intellectual task that a human can. AI has been used in a variety of applications, including healthcare, transportation, and finance, to improve efficiency and accuracy. However, concerns about the ethical implications of AI have been raised, including issues related to privacy, bias, and potential job displacement. As AI technology continues to advance, it is essential to address these concerns and ensure that it is developed and used in a responsible and ethical manner [3].

Machine learning has emerged as a promising tool for addressing poverty, by enabling better targeting of resources and interventions [4]. In particular, supervised learning algorithms such as decision trees, support vector machines, and random forests have been used to predict poverty using household-level data such as demographics, asset ownership, and household characteristics [5,6]. Such predictive models can be used to identify households that are most likely to be poor and prioritize them for targeted interventions such as cash transfers, food assistance, or job training programs [7]. In addition, unsupervised learning algorithms such as clustering and dimensionality reduction have been used to identify patterns and relationships in poverty data and to inform policy and program design [8]. Machine learning has also been used in combination with other data sources such as satellite imagery and mobile phone data to improve poverty mapping and monitoring [9,10]. Overall, machine learning has the potential to revolutionize poverty reduction efforts by enabling more efficient and effective targeting of resources, and by providing insights into the complex dynamics of poverty [11].

The structure of the paper will be as follows. In the first section, we will expose the theoretical framework of poverty. In the second one, we will draw the systematic literature review related to supervised machine learning algorithms. In the third section, we will compare the performance of our ML algorithms in terms of accuracy. In the last one, we conclude.

2. Poverty reduction - a theoretical literature review

Poverty is a complicated topic with several causes. There is no single theory that can perfectly explain poverty, but there are a variety of theoretical frameworks that can help

*Corresponding author E-mail: y.elaachab@gmail.com us comprehend it. The cycle of poverty is a popular theoretical paradigm for its analysis. According to this view, poverty is a self-perpetuation cycle passed down from one generation to another. Poverty is frequently caused by a combination of factors such as low income, a lack of education, poor health, and social marginalization. These factors can make it difficult for people to leave poverty, as well as contribute to the development of negative attitudes and behaviors that make breaking the cycle even more difficult [12].

The structuralist theory is another prevalent theoretical paradigm for analyzing poverty. According to this hypothesis, structural issues such as unequal wealth and resource distribution produce poverty. The structuralist thesis holds that the economic system is intended to reward the affluent while penalizing the poor. This can occur through a variety of factors, including worker exploitation, wealth concentration, and a lack of access to education and healthcare [13].

Political economics theory is a similar theoretical framework based on the political system's role in perpetuating poverty. Hence, the affluent control the political system and utilize their authority to further their own economic interests at the expense of the poor. This can occur through a variety of processes, including corruption, cronyism, and the manipulation of public policy [14].

The phenomenon of poverty is a challenge that we can overcome. By understanding the causes of poverty and the challenges of addressing it, we can develop effective policies and programs that can help to reduce poverty and improve the lives of people who are poor.

3. Supervised learning - a systematic literature review

3.1 The general process of implementation of ML algorithms

The description of the implementation of supervised machine learning to a real-world problem is presented in the figure below.

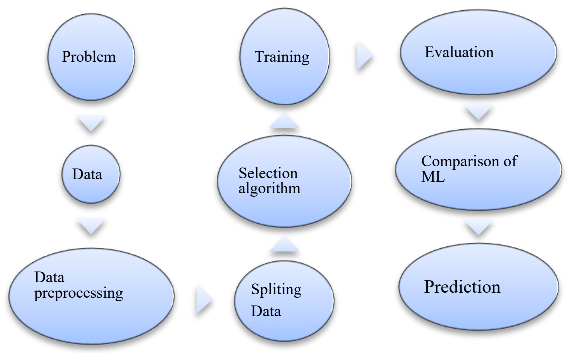

Fig. 1: The general process of implementation the ML algorithms

3.2. Supervised classification ML algorithms

3.2.1. Logistic regression algorithm

Logistic regression (LR) is the classical approach of machine learning algorithms that could be employed to classify a categorical factor. It is categorized by its simplicity and effectiveness. Nevertheless, among its disadvantage, it required some closed assumptions on random errors.

Let Y be a dichotomy variable such as;

$Yi = 1$ If the household is poor in in observation i;

$Yi = 0$ if the household is NOT poor in observation i;

$X = (X1, X2, \dots, Xk)$ Be a set of dependent variables which can be discrete, continuous, or a combination. X_i Is the observed value of the explanatory variables for observation i.

Hence, this sigmoid curve is defined by the logistic function, of the equation:

$$f(x) = \frac{\exp(\sum \beta X)}{1+\exp(\sum \beta X)} = P(X) \quad (1)$$

Using the logistic function, we can obtain the following formula which expresses the probability of occurrence of the event as the sum of the effects of the different dependent factors.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \sum_{j=1}^{k} \beta_j X_{ij} \quad (2)$$

Algorithm: LR Algorithm

```
Begin procedure
Input: Data X, labels y
Output: Logistic regression model
Steps:
1.  Initialize w and b.
2.  For i = 1 to n:
    •  Calculate h=Xw+b.
    •  Calculate yhat =sigmoid(h).
    •  Calculate E=y−yhat
    •  Update w=w+X(transpose) E.
    •  Update b=b+∑E.
End procedure
```

3.2.2. The K-Nearest Neighbors algorithm

The K-Nearest Neighbors (KNN) technique is one of many such algorithms that are frequently employed for classification problems [15]. A non-parametric technique called KNN is based on the idea that comparable objects frequently are members of the same class. Based on the class labels of its k nearest neighbors in the feature space, it categorizes an unlabeled instance. KNN has been used to good effect in a number of fields, including anomaly detection, picture recognition, and recommendation systems [16].

For organizations and policymakers aiming to reduce poverty, the categorization of homes into poverty and non-poor groups is crucial [17]. It is possible to tailor interventions, allocate resources, and make policy decisions to address the unique needs and challenges encountered by these households when poverty-stricken households are accurately identified. In particular, KNN offers the potential to increase the precision and effectiveness of poverty classification, resulting in more potent measures for reducing poverty [18].

The neighborhood $C_k(\hat{y},\{y\})$of a new sample $\hat{y}$ is evaluated based on the distance which can be treated as a hyper-parameter of the algorithm. Common choices are the Minkowski distance formed as:

$$d(\hat{y}),y) = (\sum_{l=1}^{k}|\hat{y}-y_i|^n)^{\frac{1}{n}} \tag{3}$$

Where n is the degree of choice for the Minkowski distance. For the case n=2 Euclidean distance is defined while other choices might be desirable for certain settings (Manhattan, Hamming).

The chosen distance is used to retrieve the K nearest neighbors of the sample majority class among the dataset samples. In the brute force approach no learning occurs. The method relies uniquely on the online evaluation of the samples x_l , based on the distance function of choice. Thus given the neighborhood $C_k(\hat{y},\{y\})$, the online objective can be formulated as :

$$\hat{z} = \max_{z\in Z}\sum_{z_i\in(y_i,z_i)}^{C_k(\hat{y},\{y\})}\zeta(z_i-z) \tag{4}$$

With $\zeta(z_i-z)$ being the Dirane data function acting as a counter :

$$\zeta(z_i-z) = \begin{cases}1 & if\ z_i = z\\ 0 & else\end{cases} \tag{5}$$

3.2.3. Bagging Algorithm

Bagging, which stands for Bootstrap Aggregating, is a popular ensemble learning method used to improve the performance of predictive models by reducing variance and overfitting. The idea behind bagging is to train multiple instances of the same model on randomly resampled versions of the training data and combine their predictions to obtain a more stable and accurate prediction.

In bagging, the individual models are trained on bootstrap samples, which are created by randomly sampling the training data with replacement [19]. This technique allows each model to have slightly different training sets, which helps to reduce overfitting and increase diversity among the models. Once the individual models are trained, their predictions are combined through averaging or voting to obtain the final prediction.

The effectiveness of bagging is due to its ability to improve the generalization performance of the models by reducing the variance in their predictions. By combining multiple models with different training sets, bagging can reduce the risk of overfitting and improve the accuracy of the predictions. Additionally, bagging can be used with a wide range of models, including decision trees, neural networks, and support vector machines, among others.

Bagging has been shown to be effective in many real-world applications, such as image classification, text classification, and fraud detection.

Algorithm: Bagging Algorithm

```
Input:Data X, labels y, number of trees n
Output:Bagging ensemble E
Steps:
1.  Initialize E as an empty list.
2.  For i = 1 to n:
    •  Bootstrap sample X i from X with replacement.
    •  Bootstrap labels yi from y.
    •  Train a decision tree Ti on X i and y i.
    •  Add Ti to E.
End Procedure
```

The aggregation of the B base classifiers can be done by taking the majority vote (for binary classification problems) or the average (for regression problems) of the predictions [18].

The mathematical formulation of bagging involves considering a training dataset $(\boldsymbol{X},\boldsymbol{Y})$ with probability distribution $\boldsymbol{P}$, an individual predictor $\mu(\boldsymbol{x},\boldsymbol{L})$, and a sample $\boldsymbol{L} = (\boldsymbol{xi},\boldsymbol{yi})\ \ \boldsymbol{1 \le i \le n}$. The bagged predictor, denoted as $\boldsymbol{\varphi a(x,P)}$, is obtained by taking the expectation of the individual predictor over a large number of random samples:

$$\mu_{\boldsymbol{a}}(\boldsymbol{x},\boldsymbol{P}) = \boldsymbol{E_L}(\mu(\boldsymbol{x},\boldsymbol{L})) \tag{6}$$

The quadratic risk associated with each individual predictor is given by:

$$\boldsymbol{E_L E_{X,Y}}\ (\boldsymbol{Y}\ -\ \mu(\boldsymbol{x},\boldsymbol{L}))^2 \tag{7}$$

The quadratic risk associated with the bagged predictor is given by:

$$\boldsymbol{E_{X,Y}}(\boldsymbol{Y}\ -\mu_{\boldsymbol{a}}(\boldsymbol{x},\boldsymbol{P}))^2 \tag{8}$$

Using Jensen's inequality, it can be shown that the risk associated with the bagged predictor is lower than that of the individual predictors:

$$\boldsymbol{E_{X,Y}}\ (\boldsymbol{Y}\ -\ \mu_{\boldsymbol{a}}(\boldsymbol{x},\boldsymbol{P}))^2 \le \boldsymbol{E_L E_{X,Y}}\ (\boldsymbol{Y}\ -\ \mu(\boldsymbol{x},\boldsymbol{L}))^2 \tag{9}$$

This inequality holds true especially when the individual predictors are unstable and have a high variance with respect to $\boldsymbol{L}$. [18,19]

3.2.4. Decision tree

An approach for supervised learning called a decision tree can be applied to both classification and regression problems[20]. Its structure is similar to a flowchart, with each internal node standing in for a "test" on an attribute, each branch for the result of the test, and each leaf node for the name of the class.

When training a decision tree, the data is repeatedly divided into smaller and smaller subsets until each subset is pure, meaning that every data point in the subset belongs to the same class. A stopping requirement, such as a minimal quantity of data points in a leaf node or a maximum tree depth, is re

ached by repeating the splitting procedure until it is achieved[21].

The mathematical formulation of Decision TreeEach sample i and internal node u in the decision tree is associated with a pair of flow variables wiu− and wiu+ . This means that for each sample i and internal node u, there are two numbers that represent the flow of data through the node. wiu− represents the flow of data through the node on the negative side of the decision boundary, and wiu+ represents the flow of data through the node on the positive side of the decision boundary.

Flow conservation at node v is defined as follows:

$$\mu_{+iv} + \mu_{-iv} = \begin{cases} 1 & if\ v = 0 \\ \sum_{u\in\beta(v)}(y_{+iuv} + y_{-iuv}) & otherwise, where\ v\epsilon V_1 \end{cases} \tag{10}$$

Flow conservation for $\mu_{+iv} + \mu_{-iv}$ at node u is given by:

$$\mu_{-iv} = \sum_{u\in\beta(v} y_{-iuv} \qquad where \quad u\epsilon V_1 \tag{11}$$

Flow conservation for $\mu_{+iv} + \mu_{-iv}$ at node u is expressed as:

$$\mu_{+iv} = \sum_{u\in\beta(v} y_{+iuv} \qquad where \quad u\epsilon V_1 \tag{12}$$

Algorithm: Decision Tree Algorithm

```
Begin procedure
Input:
Data X, labels y
Output: Model T
Steps:
1. Initialize T as a root node.
2. For each feature xi in X:
  • Calculate the information gain for each split on xi.
  • Choose the feature with the highest information gain.
  • Split the data on xi.
  • Recursively build decision trees for the left and right child nodes.
End procedure
```

3.2.5. Support Vector machine algorithm

The goal of Support Vector Machines is to find the function f(x) with the least variance from learning examples (x_i, y_i) [8], where $i = 1, \ldots, N$. This translates to not accounting for mistakes smaller than and forbidding those bigger than. Maximizing the function's platitude reduces the model's complexity, which impacts its generalization performance. In reality, learning theory allows for the generalization error to be restricted by a total of two terms: one based on model complexity and the other on learning data error [10]. SVM approaches are based on regulating the complexity of the model during learning. The SVM algorithm is shown in the next Section.

The method is first described for a linear function f linear of the form. With $x\epsilon\mathbb{R}^n$ the input vector, $w\epsilon\mathbb{R}^n$ the vector of parameters (or weight), and b a constant to be determined.

The approach is presented first for a linear function f. With $x\epsilon\mathbb{R}^n$ as the input vector, , $w\epsilon\mathbb{R}^n$ as the parameter (or weight) vector, and b as the constant to be found.

$$f(x) = < w, x > + b \tag{13}$$

The weights norm w is reduced to ensure the function's platitude. The problem then becomes one of decreasing this standard by ensuring that the mistakes are smaller than and may be recorded:

$$min\ \frac{1}{2} \parallel w \parallel^2 \tag{14}$$

Under the constraint

$$|y_i - (< w, x_i > + b)| < \varepsilon\ ,\ i = 1, \ldots N \tag{15}$$

$$\eta_i^{(*)}, \alpha_i^{(*)} \geq 0 \tag{16}$$

It is worth noting that the purpose of this problem formulation is to assure the function's platitude rather than to reduce the learning error. In this situation, error reduction comes exclusively in the form of restrictions, which are unbreakable. To put it another way, no mistakes are tolerated. As a result, the existence of a linear function f that properly approximates all of the samples with precision exists is assumed in this problem description. In practice, this isn't always the case. Allowing for certain mistakes is also more critical when there is a lot of noise or outliers. In this scenario, the idea of a soft margin is employed. It consists of inserting slack variables ξ_i, ξ_i^* to make the optimization problem's constraints workable. the optimization problem becomes:

$$\min \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{17}$$

Under the constraints

$$\begin{cases} y_i - (< w, x_i > + b) < \varepsilon + \xi_i \\ (< w, x_i > + b) - y_i < \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \qquad i = 1, \ldots \ldots N \end{cases} \tag{18}$$

Where ξ_i and ξ_i^* represent positive and negative errors. C > 0 is a hyperparameter that lets you fine-tune the trade-off between the amount of error permitted and the flatness of the function f. This issue formulation requires the use of an error function $|\xi|_\varepsilon$ known as ε -insensitive of the form.

$$|y - f(x)|_\varepsilon = \begin{cases} 0 & if\ |y - f(x)| < \varepsilon \\ |y - f(x)| - \varepsilon & if\ |y - f(x)| > \varepsilon \end{cases} \tag{19}$$

In most circumstances, it appears that optimization issues are easier to solve. Furthermore, as we will see, the dual formulation is essential for extending the support vector machine to nonlinearity. As a result, we will use a typical dualization approach based on Lagrange multipliers, such as that described in Flecher (1989) [26].

Algorithm: SVM Algorithm

```
Begin procedure
Input: Data X, labels y, kernel K, hyperparameters $\gamma$, C
Output: Model w, b
Steps:
1.    Initialize w, b
2.    Repeat:
          For each data point (xi,yi):
          Calculate $f(xi)=wTxi+b$
          Calculate $Ei=yi-f(xi)$
          Update w and b according to the following equations:
          $w \leftarrow w+\gamma yixi-\gamma Eixi$
          $b \leftarrow b+\gamma Ei$
          Until convergence
End procedure
```

3.2.6. Naïve Bayesian algorithm

The Naive Bayesian algorithm is a powerful and widely used classification technique in machine learning and statistics[23]. It is a probabilistic algorithm that is particularly suited for solving classification problems where the goal is to predict the class label of an input data point based on its features[24].

The Naive Bayesian algorithm works by assuming that the features of a data point are independent of each other. This means that the probability of a data point belonging to a particular class can be calculated by simply multiplying together the probabilities of each of the features belonging to that class[25].

This assumption of independence makes the Naive Bayesian algorithm very efficient, as it does not require any complex calculations to be performed. However, it is important to note that this assumption is not always accurate, and the Naive Bayesian algorithm may not be as effective in cases where the features of a data point are not independent.

Algorithm: Naïve Bayesian Algorithm

```
Begin procedure
Input: Data X, labels y
Output: Model P(y|x)
Steps:
1. Calculate the prior probabilities P(y) for each class y.
2. For each feature xi and class y, calculate the conditional probability P(xi |y).
3. The model P(y|x) is then calculated as follows:
        P(y|x) = P(y) * \prod_{i=1}^n P(xi|y)
End procedure
```

4. Comparison and discussion

The database used in this study to measure the performance of the ML methods for predicting and classifying household monetary poverty is a database from the High Commission for Planning, from its survey called the National Survey on household consumption and expenditure, 2014 organized every ten years. To manipulate the data, we used the R studio software.

4.1 Performance Evaluation Criteria for Supervised Learning Algorithms

Table 1: Comparison of supervised learning algorithms using different metrics

Metrics	LR	KNN	Bagging	DT	NB	SVM
Accuracy	*0.9109*	0.9697	0.9836	*0.9961*	*0.5103*	*0.9909*
Precision	*0.4659*	*0.9936*	*0.9833*	*0.9986*	*0.4923*	*0.9949*
F1 Score	*0.3831*	*0.9842*	*0.9916*	*0.9979*	*0.6598*	*0.9953*
Sensitivity	*0.9672*	0.9749	1.0000	*0.9986*	*1*	*0.9957*
Specificity	*0.4343*	0.8283	0.5387	*0.9259*	*0.0675*	*0.8647*
Pos Pred Value	*0.979*	0.9936	0.9833	*0.9973*	*0.4923*	*0.9949*
Neg Pred Value	*0.3274*	0.5479	1.0000	*0.9615*	*1*	*0.8822*
Prevalence	*0.9646*	0.9646	0.9646	*0.9646*	*0.4748*	*0.9638*
Detection Rate	*0.933*	0.9403	0.9646	*0.9632*	*0.4748*	*0.9909*
Detection Prevalence	*0.9646*	0.9464	0.9809	*0.9646*	*0.4748*	*0.9949*
Balanced Accuracy	*0.7008*	0.9016	0.7694	*0.9622*	*0.5337*	*0.9953*

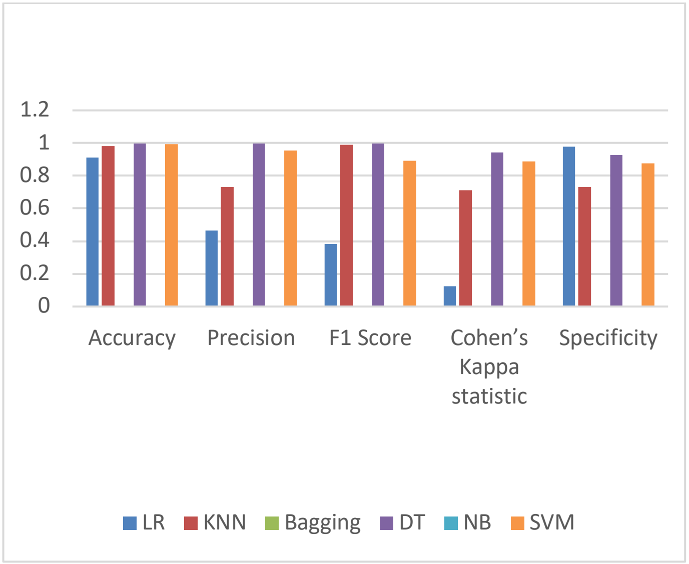

Fig 2: Comparison of supervised learning algorithms using different metrics

4.2 Discussion and analysis

The table that is presented provides a thorough evaluation of various machine learning algorithms utilizing a variety of performance indicators.

In terms of accuracy, the DT algorithm is in the lead with a stellar score of 0.9961, followed closely by SVM at 0.9909. Additionally, bagging performs well, with an accuracy of 0.9836. NB Blags falls slightly short with an accuracy of 0.5103, though. When it comes to precision, KNN and NB stand out with impressive precision scores, demonstrating their skill at reducing false positives. Notably, high accuracy values are also displayed by DT, SVM, and Bagging. In contrast, LR lags in this area. **A**s well as the F1 score, which successfully balances precision and recall, highlights the skill of "Bagging" and "DT," demonstrating their capacity to find a balance between making accurate positive predictions and averting false positives. The F1 scores for "KNN," "SVM," and "NB" are similarly strong, although "LR" may be stronger. Notably, NB demonstrates faultless sensitivity as well. Bagging, DT, KNN, and SVM all do extraordinarily well in terms of sensitivity, a critical criterion for actually correctly recognizing positive cases, with scores of 1, demonstrating their capacity to catch all positive instances. In fact, shifting our focus to specificity, which gauges the algorithm's ability to correctly identify negative cases, SVM takes the lead with a specificity of 0.8647. DT also holds its ground in this aspect. The positive predictive value (Precision) emphasizes KNN and NB's advantages once further by demonstrating how well they can predict positive situations. Additionally, DT, SVM, and Bagging show strong positive predictive results. Bagging and NB excel in the area of negative predictive value, highlighting their skill in accurately anticipating negative cases.

The prevalence of genuine positive events is consistently 0.9646 across the board, providing a reference point for comparison. DT has the highest detection rate (0.9632), followed closely by SVM and Bagging, demonstrating their capacity to precisely identify positive cases.

The detection prevalence also highlights the accuracy of SVM and Bagging in predicting positive cases. SVM emerges as a strong competitor, earning a score of 0.9953 for balanced accuracy, which takes into account both sensitivity and specificity. Additionally, KNN and DT exhibit outstanding balancing accuracy. Overall, the best-performing model in the table is Bagging, which has the highest accuracy, precision, F1 Score, and sensitivity. The worst-performing model in the table is Naive Bayes, which has the lowest accuracy, precision, F1 Score, and sensitivity.

However, it is important to note that accuracy is not always the best metric to evaluate classification models. For example, if the cost of misclassifying a positive instance is much higher than the cost of misclassifying a negative instance, then a model with high sensitivity but low specificity may be preferred.

5. Conclusion and future works

in this work we have focused on machine learning methods to make a prediction of poor and non-poor households, to carry out the study we have chosen an open-source database from the latest national survey and consumption of a household, to be carried out by the high commissioner of the plan. this work aimed to test the robustness of machine learning methods in the case of the prediction of households. According to their type of poverty and to see the result of these s methods, after having presented the theoretical framework and decorticated the methods used to present and expose the results.

From the analysis of the outputs, we have concluded the best-performing model in the table is Bagging, which has the highest accuracy, precision, F1 Score, and sensitivity. The worst-performing model in this study is Naive Bayes, which has the lowest accuracy, precision, F1 Score, and sensitivity.

However, it is important to note that accuracy is not always the best metric to evaluate classification models. For example, if the cost of misclassifying a positive instance is much higher than the cost of misclassifying a negative instance, then a model with high sensitivity but low specificity may be preferred.

References

[1] Russell, S. J., and Norvig, P. (2020). Artificial intelligence: A modern approach. Pearson.

[2] Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., ... and Luetge, C. (2018). AI4People—An ethical framework for a good AI society: Opportunities, risks, principles, and recommendations. Minds and Machines, 28(4), 689-707.

[3] Jordan, M. I., Mitchell, T. M., and Shneiderman, B. (2020). Making AI trustworthy: Aligning ethics, technical methods, and societal values. Communications of the ACM, 63(1), 48-57.

[4] Deaton, A., and Zaidi, S. (2002). Guidelines for constructing consumption aggregates for welfare analysis. World Bank.

[5] Khandker, S. R., Koolwal, G. B., and Samad, H. A. (2010). Handbook on poverty and inequality. World Bank Publications.

[6] Noy, I., and Nhu, L. T. (2017). Improving poverty measurement in Vietnam using mixed data clustering. Journal of Development Studies, 53(6), 931-947.

[7] Gharad, B., McLaren, Z., and Rojas, F. (2017). Using machine learning to target treatment: The case of household energy efficiency. American Economic Review, 107(5), 578-582.

[8] Ranganathan, A., and Palaniswami, M. (2017). Identifying poverty-related indicators using clustering and association rules mining techniques. Applied Soft

Computing, 55, 203-212.

[9] Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., and Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty. Science, 353(6301), 790-794.

[10] Burke, M., Heft-Neal, S., Bendavid, E., and Sources of Spatial Data and Machine Learning Algorithms for Poverty Mapping Collaboration. (2019). Sources of spatial data and machine learning algorithms for poverty mapping. Journal of Economic Perspectives, 33(4), 211-230.

[11] Ahmed, S., and Hoque, N. (2021). The applications of artificial intelligence in poverty alleviation: A systematic review. Technological Forecasting and Social Change, 166, 120639. doi: 10.1016/j.techfore.2021.120639.

[12] Duncan, G. J., and Murnane, R. J. (2011). Whither Opportunity?: Rising Inequality, Schools, and Children's Life Chances. Russell Sage Foundation.

[13] Chen, S., and Ravallion, M. (2010). The Developing World Is Poorer Than We Thought, But No Less Successful In The Fight Against Poverty. The Quarterly Journal of Economics, 125(4), 1577-1625.

[14] Gupta, A. K., Tesluk, P. E., and Taylor, M. S. Innovation at and across Multiple Levels of Analysis , Vol. 18, No. 6, Innovation at and across Multiple Levels of Analysis (Nov. - Dec., 2007), pp. 885-897

[15] Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

[16] Alim, M. A., Kim, H. G., and Lee, Y. K. (2017). A comparative analysis of K-Nearest Neighbor and Support Vector Machine for land cover classification using satellite images. Sustainability, 9(10), 1840.

[17] Ghani, R. A., and Muthu, R. (2015). Anomaly detection using k-nearest neighbor classification algorithm. Procedia Computer Science, 47, 51-57.

[18] N'Guessan, A., and Bro, P. (2019). Poverty classification from remote sensing data using machine learning. Remote Sensing, 11(1), 91.

[19] Ravindran, A., Zafarani, R., and Liu, H. (2012). The K-Nearest Neighbors Algorithm. In Social Media Mining (pp. 147-171). Cambridge University Press.

[20] World Bank. (2021). Poverty Overview. Retrieved from https://www.worldbank.org/en/topic/poverty/overview

[21] Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140.

[22] Friedman, J. H. (1999). Stochastic gradient boosting. Computational Statistics and Data Analysis, 38(4), 367-378.

[23] Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.p50

[24] Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer ,p120

[25] R. Fletcher. Practical Methods of Optimization. John Wiley and Sons, New York, 1989.

[26] Johnson, M. B. (2019). Understanding Decision Trees in Machine Learning. Machine Learning Insights, 5(2), 112-125.

[27] Smith, J. A. (2022). Exploring Decision Trees in Machine Learning. Data Science Journal, 10(3), 45-60.