A Comparative Analysis of MATLAB and Python Neural Networks for Diabetes Prediction

Gabriel Oliveira Pimentel[1,4][0009−0008−8697−7942], Augusto Luvisa Dessanti[1,3][0009−0002−6100−303X] and João Paulo Teixeira[1,2][0000−0002−6679−5702]

[1] Research Centre in Digitalization and Intelligent Robotics CeDRI, Instituto Politécnico de Bragança, 5300-253 Bragança, Portugal
[2] Laboratório para a Sustentabilidade e Tecnologia em Regiões de Montanha SusTEC, Instituto Politécnico de Bragança, 5300-253 Bragança, Portugal
[3] Federal Institute of Rio Grande do Sul, Farroupilha , Brazil
[4] Federal Center for Technological Education of Minas Gerais, Leopoldina, Brazil

Abstract. In recent years, artificial intelligence (AI) has become an integral part of many everyday applications. The growth of AI is bringing intellectual benefits to humans. It is revolutionizing discovery, learning, communication, and work. Machine learning, especially deep learning, is critical to this progress, providing complex models to process data more effectively. Neural networks, which emulate biological processes, find applications across diverse fields, notably in medicine, where they automate disease diagnosis and prediction tasks, such as diabetes. This paper proposes a comparative analysis between Python and MATLAB for diabetes prediction using a dataset with 100,000 individuals. The study conducts simulations on both platforms and validates the results using metrics such as precision, specificity, accuracy and F-measure. Additionally, the study emphasizes the importance of platform selection based on considerations of functionality and cost, offering insights into optimizing outcomes in healthcare applications.

Keywords: Neural Network; Feed-forward in Matlab and Python; MLP in Matlab and Python; MATLAB; Python.

1 Introduction

In recent years, the use of Artificial Intelligence (AI) has grown exponentially, playing a crucial role in a variety of applications in our daily lives. Previously described as a technology of futuristic societies, Artificial Intelligence is now a reality in high-tech society [8]. There are several definitions of AI, most of which currently state thatAI is intended to solve complex cognitive problems associated with human intelligence, facilite access to services through mobile devices or healthcare, or recognize problems and create solutions for the benefit of technology, people, and society. However, the central concept of AI remains the development of machines capable of emulating human thought [5,6,15,3,16,12].

The growth of AI brings intellectual benefits to humans. It has the potential to revolutionize the way we discover, learn, live, communicate, and work due to current technological advances [11]. With the application of artificial intelligence technology, computer information processing speed continues to accelerate, information security levels continues to improve, and the development of related technologies continues to advance. This is of great significance to the progress of human society and the solution of problems related to human intelligence [9].

Informally, AI is when a machine is able to perform functions that humans associate with other human minds, such as learning and problem solving. Therefore, learning is an essential sub-field of this area and a vital element in the functioning of machines. Over the decades, computer scientists have worked hard in the field of machine learning, making significant efforts to advance this area. In this context, deep learning has emerged as a promising endeavor, being a subcategory of machine learning that aims to create complex, hierarchical models to understand and process data more effectively. The continuous evolution in the field of machine learning opens up space for the exploration and development of new techniques, such as neural networks, which play a crucial role in the advancement of artificial intelligence [18].

Neural networks, a machine learning approach that mimics the architecture of the biological nervous system with a large number of interconnected neurons, have become a popular approach due to their methodology that models the processes of systems after the human brain. In general, the network is usually an approximation of some kind of algorithm or function in nature, or it can be an expression of a logical strategy [22].

With a wide range of applications in different fields, such as agriculture, science, medicine, education, finance, management, security, engineering and commerce, neural networks play an essential role in accomplishing tasks [1]. That includes making automatic diagnoses, solving problems, recognizing patterns, processing signals and supporting strategic decisions [19].

Moreover, neural networks began to be used in medical processes and problems when it was asserted that these processes and problems can be complex in nature and would be better dealt with by non-linear approaches [17]. Neural networks have been widely used in disease recognition and prediction processes. Disease prediction is one of the important and highly attentive biomedical areas where artificial neural networks have been used on a large scale.

In this context, disease prediction has emerged as an important tool that is capable of improving medical diagnoses and making important health care decisions, especially for diseases such as diabetes, which have a growing incidence and global impact [2]. By integrating machine learning techniques into the analysis of large volumes of healthcare data, it is possible not only to predict future trends and events, but also to optimize resources and improve clinical outcomes for the benefit of patients and healthcare professionals by identifying patterns and trends that may elude human observation [13].

The state of the art in predicting diabetes using AI techniques involves combining advanced machine learning algorithms with explicable AI approaches to provide accurate and interpretable diagnostics. A large number of research efforts are dealing with different architectures of neural networks and different strategies to increase the effectiveness of the models on different datasets [20,4,10,14,21,6,15]. Additionally, diagnostic results using the same dataset demonstrated a prediction accuracy of 91.10% using a balanced dataset, as presented in [7].

With this background, this paper will compare how neural networks developed using Python and MATLAB are used, developed, and trained to predict individuals with diabetes from data sets. This paper is organized as follows: Section 1 deals with the introduction described here; Section 2 describes the methodology used in this work; Section 3 deals with the analysis of the results and discussions. Section 4 deals with the conclusions drawn from this paper.

2 Methodology

The proposed methodology is detailed in this section. Based on structured data obtained from a dataset containing information such as gender, age, and smoking history, multilayer neural networks were implemented in this work. First, a preprocessing step was applied to the dataset. Then, the data was normalized to ensure more accurate results. Subsequently, The data was split into training, validation and testing. After that, the data was processed with neural networks, resulting in an output. Finally, to assess the performance of the model, the data was validated using metrics such as precision, specificity, accuracy and F-measure. Figure 1 illustrates the flowchart of the methodology used.

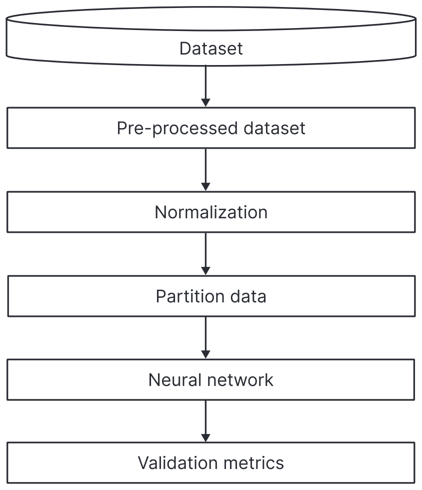


Fig. 1: Flowchart of the methodology used.

2.1 Procedures

Dataset: The dataset requires visual analysis to identify characteristics and behaviors that may affect the accuracy of future predictions or cause performance loss. Although some trends were observed that could potentially bias the neural network, it was decided not to address them in this study. These aspects will be addressed during dataset preprocessing in the next step.

In predicting diabetes using an artificial neural network, all nine attributes are considered. These attributes include gender, age, presence of hypertension, presence of heart disease, smoking history, body mass index (BMI), HbA1c level, blood glucose level, and diabetes status.

For this study, the dataset [5]. used contains information on various attributes of 100,000 individuals. For a more detailed understanding of these attributes, a complete description is provided in Table 1.

Table 1: Dataset attributes.

Feature	Description
Gender	Gender of the individual (male, female, other)
Age	Age of the individual (years)
Hypertension	Presence of hypertension (1 - yes, 0 - no)
Heart Disease	Presence of heart disease (1 - yes, 0 - no)
Smoking History	Smoking history of the individual (ever, current, former,...)
BMI	Body Mass Index (kg/m^2)
HbA1c Level	HbA1c level of the individual (%)
Blood Glucose Level	Blood glucose level of the individual (mg/dL)

Moreover, the dataset includes the attribute diabetes. This attribute indicates the presence or absence of the condition, which is the variable of interest in this study to be predicted. The corresponding value is 1 for diabetics and 0 for non-diabetics.

Furthermore, it is crucial to acknowledge the imbalance in the dataset between individuals with diabetes and individuals without diabetes. The imbalance is significant, with only 8,500 cases of diabetes out of 100,000 records. This imbalance may impact the performance of the predictive models developed from this data and the interpretation of the resulting outcomes.

Preprocessed dataset: Data may be inconsistent and incomplete when used in experiments conducted with machine learning techniques. Data preprocessing is the first phase of data treatment and is a technique used to transform this data into meaningful and clean data. In this case, modifications are made to the dataset to achieve better performance, shorter processing time, and other advantages.

[5] Public dataset available here.

In the case of the analyzed dataset, the OHE (One Hot Encoding) standard was adopted for columns with qualitative categorical data to convert variables into distinct binary representations. This procedure was applied to the gender attribute, where each category (”male,” ”female,” and ”other”) was converted into exclusive binary vectors.

Additionally, a numerical scale mapping method was applied, ranging from 0 to a value determined by the quantity and weight of the data. This method, applied to the ”smoking history” attribute, consisted of categorizing the different smoking statuses by assigning specific weights. In particular, ”ever” was given the highest weight (5), followed by ”current” (4), ”not current” (3), ”former” (2), ”never” (1), and ”no information” (0).

Normalization: Data normalization is a very important step in data preprocessing in machine learning techniques, where it plays a vital role in improving results. In the dataset, attributes often present different scales or ranges, so a normalization process was applied to a range of values between 0 and 1.

This process prevents possible biases in the network and prevents problems with variable resolution by having very high values in certain columns and very low values in others. It also helps improve processing efficiency and ensures that each column has equal weight in relation to its inputs.

Partition data: Data partitioning is an essential practice in machine learning projects, involving the division of the original dataset into different subsets. This division serves to separate the data and ensure the model’s ability to generalize to new data and prevent problems such as overfitting, thus contributing to the more effective development of the neural network. In general, the data is divided into three distinct sets: the training set, the validation set, and the test set, in proportions of 70%, 15%, and 15%, respectively.

Training: In neural networks, training is an essential stage in the development of machine learning models. During training, the model is exposed to the data to learn patterns between the input variables and the desired output variable. In this work, training is divided into two parts: Python and MATLAB. Both parts use the same neural network structure and loss function. Other factors, such as the training function or optimization algorithm, were kept as default in their respective languages.

The network structure consists of an input layer, a hidden layer, and an output layer. The input layer has a number of neurons equal to the number of variables of a sample, excluding the last column of the dataset, which represents the value to be classified by the neural network, resulting in 10 input neurons, as seen in Figure 2.

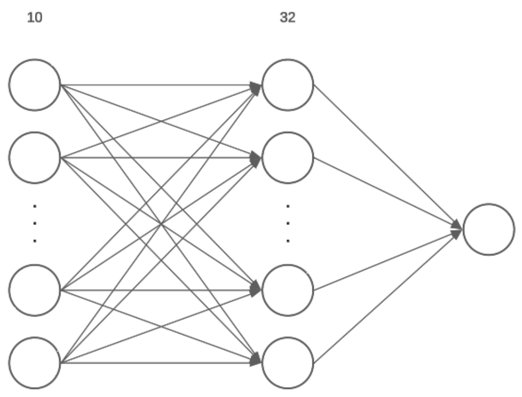
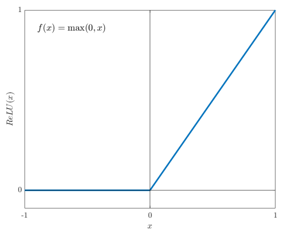


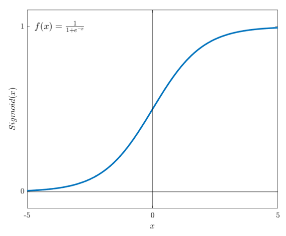
Fig. 2: Neural network structure.

The hidden layer has 32 neurons and uses the Rectified Linear Unit (ReLU) activation function, chosen because it is one of the most popular in hidden layers. Figure 3a shows the activation function.

The output layer has only one neuron with the Sigmoid activation function, responsible for assigning a continuous value between 0 and 1. This value is then processed to become 0 or 1, based on the difference with the smallest absolute value. As observed in Figure 3b illustrating the activation function.



(a) Rectified Linear Unit



(b) Sigmoid

Fig. 3: Activation functions of the neural network layers.

The loss function used is the Mean Squared Error (MSE), which calculates the mean square of the difference between the estimated value and the actual value, allowing to evaluate the accuracy of the prediction model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

Python: In Python, the model construction process begins with building the architecture of the neural network. Using the Keras API, the model can be built sequentially, as shown below.

```
model = Sequential([
    keras.Input(shape=(10,)),
    Dense(units=32, activation='relu'),
    Dense(units=1, activation='sigmoid')])
```

The keras.Input() function is used to create the first input layer of the network, defining the dimension of the input vector per sample. Next, the Dense() function is used to create subsequent layers of the network, specifying the number of neurons in each layer through the 'units' parameter and the activation function to be used through the 'activation' parameter. Sequential models automatically assign the output layer function to the last layer added to the network.

After building the model, it must be compiled. During this stage, the loss function to be used by the network is assigned. Compilation is performed as follows:

```
model.compile(loss='mean_squared_error',metrics['accuracy'])
```

Where the compile() method receives the name of the loss function to be used in the 'loss' parameter. The 'metrics' parameter can be used as a visualization tool. The values passed to this parameter are displayed on the user's screen during the network training.

With the network prepared, the next step is training. This is done through the fit() method, which receives as parameters the variables x, y, and other parameters such as epochs and validation_split. This can be observed below.

```
model.fit(x=train_samples, y=train_labels,
          validation_split=15/85,epochs=20)
```

The input x corresponds to the samples matrix (matrix with the samples and each of their input variables); y corresponds to the labels matrix (matrix with the target results for each sample); 'epochs' is the number of epochs chosen for training, an epoch is a complete pass through all the training data. And 'validation_split' is the proportion of the training data to be used as a validation dataset. The value of the validation split was chosen so that out of 85,000 samples, 15,000 were used as the validation dataset.

Finally, after training the network, the predict() method is employed to make predictions about the values of the categories of new samples introduced into the network. This method takes 'x' as input, which is the samples matrix to be classified by the network, and returns a matrix of labels corresponding to each of the samples introduced into the samples matrix. The usage of the method is as follows:

```
predict = model.predict(x=test_samples)
```

MATLAB: In MATLAB, the first step is to read the CSV file containing the relevant data. These data are extracted, separated into vectors, and prepared for use in a neural network.

Data preparation, as described in previous sections, includes the assignment of weights to different categories of smoking history and the application of one-hot coding to the gender variable. The data are then normalized for more accurate results and organized into input and output matrices.

The process of building the neural model begins with defining the architecture of the network using the feedforwardnet() function. This function creates a feed-forward neural network in the following manner:

```
net = feedforwardnet([32, 1]);
```

Here, the function argument is a neural network with a 32-neuron hidden layer and a 1-neuron output layer.

Once the network architecture is defined, the training and activation functions for each layer are configured. In this case, the functions ”poslin” and ”logsin” are used, where the ”poslin” function ensures positive or zero output for each neuron, while the ”logsin” function ranges input values between -1 and 1. These functions are employed as defined for this work:

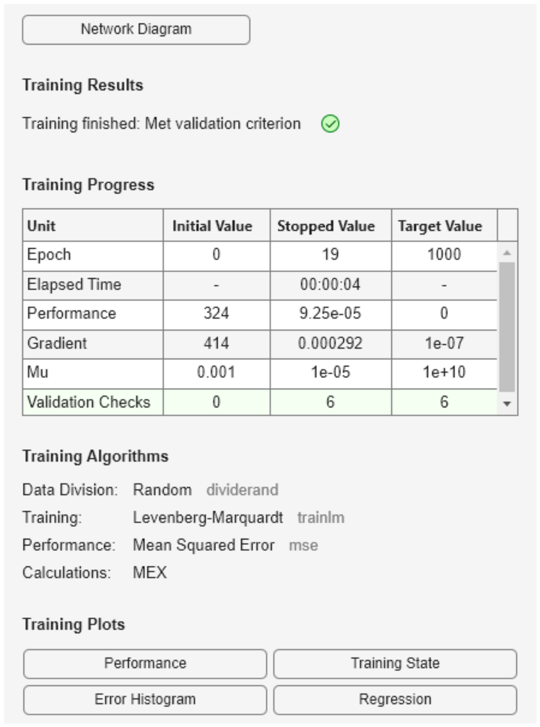
```
net.layers{1}.transferFcn = 'poslin';
net.layers{2}.transferFcn = 'logsin';
```

The network is then trained using the train() method. By default, the data is divided into training, validation, and test sets, and the neural network is trained using these sets with the following function:

```
[net, tr] = train(net, P, T);
```

By default, this MATLAB procedure accepts as input the neural network, the input data matrix (P), and the target output data matrix (T) with data from people with and without diabetes.

During training, multiple evaluation metrics are monitored, such as mean square error and model performance. These, described in Figure 4, are useful for evaluating the performance of the model.



Training Results

Training finished: Met validation criterion ✓

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	19	1000
Elapsed Time	-	00:00:04	-
Performance	324	9.25e-05	0
Gradient	414	0.000292	1e-07
Mu	0.001	1e-05	1e+10
Validation Checks	0	6	6

Training Algorithms

Data Division:	Random	dividerand
Training:	Levenberg-Marquardt	trainlm
Performance:	Mean Squared Error	mse
Calculations:	MEX	

Training Plots

Fig. 4: Neural network training toolbox in MATLAB
.

After training, the neural network starts predicting. The results, which will be presented in the next section, are evaluated using calculated metrics such as mean square error, model precision, accuracy, specificity, and F-measure.

Evaluation Factors: To assess the performance of the neural network, the following measures were used:

1. **True Positive (TP)**: It is the number of positive cases correctly identified by the neural network.
2. **False Positive (FP)**: It is the number of negative cases incorrectly identified as positive by the neural network.
3. **False Negative (FN)**: It is the number of positive cases incorrectly identified as negative by the neural network.
4. **True Negative (TN)**: It is the number of negative cases correctly identified by the neural network.
5. **Precision**: It is the proportion of positive cases correctly identified relative to the total cases identified as positive. The equation for precision is:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

6. **Sensitivity (Recall)**: It is the proportion of positive cases correctly identified relative to the total truly positive cases. The equation for sensitivity is:
$$Sensitivity = \frac{TP}{TP + FN} \tag{3}$$
7. **Specificity**: It is the proportion of negative cases correctly identified relative to the total truly negative cases. The equation for specificity is:
$$Specificity = \frac{TN}{TN + FP} \tag{4}$$
8. **Accuracy**: It is the proportion of cases (positive and negative) correctly identified relative to the total cases. The equation for accuracy is:
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$
9. **F-measure (F1 Score)**: It is the harmonic mean between precision and sensitivity. It provides a measure of the balance between precision and sensitivity. The equation for the F-measure is:
$$F - measure = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{6}$$

2.2 System development:

The study used the Python programming language, version 3.11. Libraries used included Keras, TensorFlow, scikit-learn, NumPy, and pandas. Additionally, MATLAB version R2020b was used with the functions described earlier. The development environment has been Visual Studio Code.

The specifications of the computer system used in the analysis are detailed in the Table 2 that describes the component specifications:

Table 2: Computer settings used.

Component	Description
Processor Unit (CPU)	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Graphics Unit (GPU)	Intel(R) Iris(R) Xe Graphics
RAM Memory	16 GB
Storage (SSD)	256 GB
Operating System	Windows 11 64-bit

3 Results and discussions

This section presents the results and discussion of the predictions in both MATLAB and Python for the identification of diabetes cases. For this work, it was decided to use the full dataset and partitions of 75%, 50%, 25%, and 10% of the data to apply the neural network. Simulations were then performed in Python and MATLAB to evaluate the applied neural networks.

3.1 Results

The tables 3, 4, 5, 6, and 7 detail the performance of the predictions made in MATLAB and Python during the simulations for the respective divisions of the dataset in order to obtain better accuracy in the values obtained.

Table 3: Metrics of simulations performed for complete dataset.

Dataset	Name	MATLAB (%)	Python (%)
Complete	true positive	13681 (99.92%)	13669 (99.54%)
	false positive	402 (30.75%)	473 (37.30%)
	false negative	12 (0.08%)	63 (0.46%)
	true negative	905 (69.25%)	795 (62.70%)
	precision	0.97145	0.966550
	sensitivity	0.99912	0.995412
	specificity	0.69243	0.626972
	accuracy	0.97240	0.964267
	F-measure	0.98510	0.980771

Table 4: Metrics of simulations performed for 75% dataset.

Dataset	Name	MATLAB (%)	Python (%)
75%	true positive	10270 (99.83%)	10251 (99.76%)
	false positive	318 (33.02%)	386 (39.63%)
	false negative	17 (0.17%)	25 (0.24%)
	true negative	645 (66.98%)	588 (60.37%)
	precision	0.96997	0.963712
	sensitivity	0.99835	0.997567
	specificity	0.66978	0.603696
	accuracy	0.97022	0.963467
	F-measure	0.98395	0.980347

Table 5: Metrics of simulations performed for 50% dataset.

Dataset	Name	MATLAB (%)	Python (%)
50%	true positive	6841 (99.87%)	6865 (99.73%)
	false positive	189 (29.08%)	216 (35.06%)
	false negative	9 (0.13%)	19 (0.27%)
	true negative	461 (70.92%)	400 (64.94%)
	precision	0.97312	0.969496
	sensitivity	0.99869	0.997240
	specificity	0.70923	0.649351
	accuracy	0.97360	0.968667
	F-measure	0.98573	0.983172

Table 6: Metrics of simulations performed for 25% dataset.

Dataset	Name	MATLAB (%)	Python (%)
25%	true positive	3432 (99.68%)	3422 (99.10%)
	false positive	100 (32.57%)	99 (33.33%)
	false negative	11 (0.32%)	31 (0.90%)
	true negative	207 (67.43%)	198 (66.67%)
	precision	0.97169	0.971883
	sensitivity	0.99681	0.991022
	specificity	0.67427	0.666667
	accuracy	0.97040	0.965333
	F-measure	0.98409	0.981359

Table 7: Metrics of simulations performed for 10% dataset.

Dataset	Name	MATLAB (%)	Python (%)
10%	true positive	1362 (99.78%)	1354 (94.48%)
	false positive	48 (35.55%)	55 (40.15%)
	false negative	3 (0.22%)	9 (5.52%)
	true negative	87 (64.45%)	82 (59.85%)
	precision	0.96596	0.960965
	sensitivity	0.99780	0.993397
	specificity	0.64444	0.598540
	accuracy	0.96600	0.957333
	F-measure	0.98162	0.976912

Additionally, Table 8 describes the computational performance times for the training. The table presents the performance times, in seconds, for both MATLAB and Python implementations for the respective divisions of the dataset.

Table 8: Performance time for train.

Dataset	MATLAB (s)	Python (s)
Complete	20.15	49.44
75%	15.12	35.26
50%	10.50	24.13
25%	6.00	12.55
10%	3.10	6.24

3.2 Discussions

As observed, both the MATLAB and Python implementations show consistent trends in all aspects analyzed. Accuracy and sensitivity remain above 0.96 and 0.99, respectively, corroborating to [7] with prediction accuracy of 0.91 using a balanced dataset and indicating the effectiveness of the models in identifying positive diabetes cases. However, as the size of the dataset decreases, specificity decreases, with Python showing a more pronounced decrease. This suggests that the reduction in training data affects the ability of the models to accurately identify negative cases. Moreover, the increase in false negatives is greater in Python. These differences highlight the greater sensitivity of the Python model to changes in data size, which has a greater impact on its overall specificity and error rates compared to the MATLAB model.

In additional, accuracy remained relatively stable across different dataset sizes for both tools, indicating a consistent balance between true positive predictions and false positive errors. The F-measure, which combines precision and recall, also shows consistent performance, highlighting the ability of the models to maintain a balance between precision and recall, as observed in Figure 5, which shows the relationship between F-measure and dataset size.

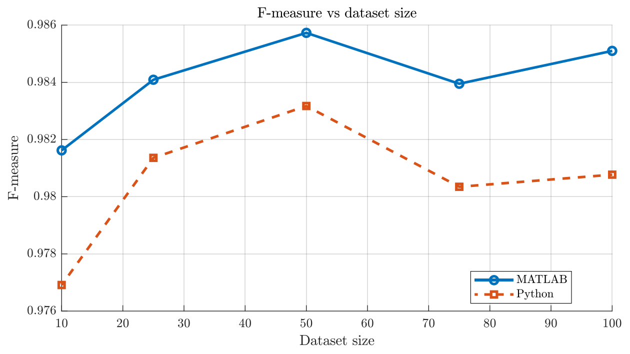


Fig. 5: F-measure vs dataset size.

Furthermore, for both tools, specificity and precision decreased as the dataset became smaller. The size of the dataset has an expected impact on the results presented, since the amount of training data available to teach the model to recognize patterns associated with negative cases is reduced. This trend is consistent for both the MATLAB and Python implementations.

Finally, a notable difference between the MATLAB and Python implementations is how they handle the interruption of training with the validation set. While MATLAB uses cross-validation to interrupt training, Python integrates validation into the model training process, reserving a portion of the training set for validation during training. This difference can affect the ability of models to generalize and deal with unseen data. This also has an impact on computational performance. As shown in Table 8, MATLAB generally has shorter execution times than Python across a range of dataset sizes. This discrepancy is due to differences in how the two platforms handle validation and training interruptions.

4 Conclusion

With this, we observed that MATLAB and Python presented consistent performance metrics across all divisions of the dataset, reflecting the solidity of their implementations. When we analyze the results, we see a balanced performance in the metrics presented. This consistency suggests that the differences between the implementations do not significantly affect the ability of the models to identify positive cases of diabetes between the different partitions of the dataset.

As a result, this study provides valuable perspectives for the application of these tools in diabetes prediction, demonstrating their usefulness to health professionals and researchers in the effective identification and prediction of this health condition. With the results presented, the importance of these tools is remarkable, contributing to the improvement of medical care and quality of life of patients.

Moreover, it should be noted that although the differences between MATLAB and Python were evident, these differences should not be interpreted as superiority of one tool over the other. Both have their advantages and limitations, and the choice between them should be based on broader considerations, including ease of use, cost, and availability of resources. Additionally, it is important to note that Python is a free, open-source programming language, while MATLAB is commercial software that requires a paid license, or free online use with some processing limitations. This distinction can be crucial for projects with budget constraints or for those looking for an affordable solution without compromising the quality or effectiveness of the analysis.

5 Acknowledgments

The authors are grateful to the Foundation for Science and Technology (FCT, Portugal) for financial support through national funds FCT/MCTES (PIDDAC) to CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and

UIDP/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020).

References

1. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. *Heliyon* **4**(11) (2018). Elsevier
2. Aggarwal, S., Gupta, S., Alhudhaif, A., Koundal, D., Gupta, R., Polat, K.: Automated COVID-19 detection in chest X-ray images using fine-tuned deep learning architectures. *Expert Systems* **39**(3) (2021).
3. Araújo, T.; Teixeira, J. P.; Rodrigues, P. M.: Smart-data-driven system for Alzheimer disease detection through electroencephalographic signals. *Bioengineering*, **9**(4), 1-16, ISSN 2306-5354. (2022) https://doi.org/10.3390/bioengineering9040141
4. Ashisha, G. R., and Anitha Mary, X: Prediction of Blood Pressure and Diabetes with AI Techniques—A Review. *International Conference on Information, Communication and Computing Technology*, pp. 749-760 (2023).
5. Goralski, M.A., Tan, T.K.: Artificial intelligence and sustainable development. *The International Journal of Management Education* **18**(1), 100330 (2020). https://doi.org/10.1016/j.ijme.2019.100330.
6. Guedes, V., Junior, A., Teixeira, F., Fernandes, J., and Teixeira, J. P.: Long Short Term Memory on Chronic Laryngitis Classification. *CENTERIS/ProjMAN/HCist 2018 - Procedia Computer Science*, Elsevier, **138**, pp. 250-257. (2018) https://doi.org/10.1016/j.procs.2018.10.036
7. Guerreiro, N., Nijo, R., and Teixeira, J. P.: Comparison of Neural Network Architectures for Diabetes Prediction. *CENTERIS/ProjMAN/HCist 2024 - Procedia Computer Science*, Elsevier. (2025)
8. Hamet, P., Tremblay, J.: Artificial intelligence in medicine. *Insights Into the Future of Medicine: Technologies, Concepts, and Integration* **69**, S36–S40 (2017). https://doi.org/https://doi.org/10.1016/j.metabol.2017.01.011
9. Liu, Z., Huang, Y., Zhou, T., Wang, Y., Yan, H., Zhang, W., Wang, Y.: Discussion on the Application of Artificial Intelligence in Computer Network Technology. In: *2023 2nd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, pp. 51–55 (2023). https://doi.org/10.1109/AIARS59518.2023.00017
10. Modak, Sandip Kumar Singh, and Vijay Kumar Jha: Diabetes prediction model using machine learning techniques. *Multimedia Tools and Applications*, 83.13 (2024).
11. National Science and Technology Council and Networking and Information Technology Research and Development Subcommittee: *National Artificial Intelligence Research and Development Strategic Plan.* Executive Office of the President of the United States (2016).
12. Nguyen, L. Q., Fernandes, P. O., and Teixeira, J. P.: Analyzing and Forecasting Tourism Demand in Vietnam with Artificial Neural Networks. *Forecasting*, **4**(1), 36-50 (2022). https://doi.org/10.3390/forecast4010003
13. Ragab, M., AL-Malaise AL-Ghamdi, A.S., Fakieh, B., Choudhry, H., Mansour, R.F., Koundal, D.: Prediction of Diabetes through Retinal Images Using Deep Neural Network. *Computational Intelligence and Neuroscience* **2022**, Article ID 7887908, 6 pages (2022). https://doi.org/10.1155/2022/7887908

14. Rajendra, Priyanka, and Shahram Latifi: Prediction of diabetes using logistic regression and ensemble techniques. *Computer Methods and Programs in Biomedicine* (2021).
15. Rodrigues, P. M., Teixeira, João Paulo.: Classification of Electroencephalogram Signals Using Artificial Neural Networks. *Proceedings of 3rd International Conference on BioMedical Engineering and Informatics (BMEI'10).* Yantai, China, (2010). https://doi.org/10.1109/BMEI.2010.5639941
16. Rodrigues, P., Teixeira, J. P.: Artificial Neural Networks in the Discrimination of Alzheimer's Disease. *ENTERprise Information Systems Communications in Computer and Information Science*, **221**, pp. 272-281. Springer. (2011) https://doi.org/10.1007/978-3-642-24352-3_29
17. Silva, A.L.R. da: Seleção de atributos para apoio ao diagnóstico do câncer de mama usando imagens termograficas, algoritmos geneticos e otimização por enxame de partículas. Master's thesis, Universidade Federal de Pernambuco (2019).
18. Shinde, P.P., Shah, S.: A Review of Machine Learning and Deep Learning Applications. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6 (2018). https://doi.org/10.1109/ICCUBEA.2018.8697857
19. Soni, B., Mathur, P., Bora, A.: In Depth Analysis, Applications and Future Issues of Artificial Neural Network. *Enabling AI Applications in Data Science Studies in Computational Intelligence*, pp. 149–183 (2020).
20. Tasin, I., Nabil, T.U., Islam, S., Khan, R.: Diabetes prediction using machine learning and explainable AI techniques. *Healthcare Technology Letters*, 10(1-2), pp. 1-10 (2023). https://doi.org/10.1049/htl2.12039.
21. Thaiyalnayaki, S., et al: Classification system on diabetes prediction using deep learning approach. *AIP Conference Proceedings*, Vol. 2523 (2023).
22. Wu, Y.C., Feng, J.W.: Development and Application of Artificial Neural Network. *Wireless Pers Commun* **102**, 1645–1656 (2018). https://doi.org/10.1007/s11277-017-5224-x