

Python vs Matlab: Language Comparison for Machine Learning and Data Mining Techniques

Usman Kaisser Moghal
Computer Sciences
FAST University
Islamabad, Pakistan
usman.kaisser@nu.edu.pk

Yasmeen Afzal
Computer Sciences
FAST University
Islamabad, Pakistan
yasmeen.afzal@nu.edu.pk

Abstract—We are moving towards an age where computers are embedded with intelligence, and as organizations become aware of the importance of having this technology embedded in their products/services, they are scrambling to find experts of the field to get ahead of the curve.

Within the realm of Machine Learning, there are several different means to accomplish one's goal. One of the first areas where experts differ is their choice of programming language, often opting for one of the following: Python or Matlab.

This paper hopes to find a solution to the problem of which programming language is superior for machine learning and data mining purposes.

To compare the different languages, the common and well defined techniques ADASYN, fuzzy c-mean and multi-class SVM will be used. As such, we will be emulating the paper "Activity Recognition in Smart Homes using Clustering based Classification" on both Matlab and Python.

The paper concludes by specifying under which circumstances does the programming language outperform the other.

Index Terms—Python vs Matlab, Language Comparison, Machine Learning, Data Mining, Fuzzy c-mean, Clustering, Multi-class SVM, Classification, Oversampling, ADASYN, Activity Recognition, Smart Homes, Clustering based Classification

I. INTRODUCTION

The age of information has evolved in to the age of intelligence, the computers and devices are embedded with a built in ability for intelligence. As the trend for data intelligence takes it root, the organizations are in search for the experts that can take them ahead of the curve.

A branch called Machine Learning is at the heart of the domain and is used in multiple sub domains. The actual implementation of the idea is where the experts stand apart in the choice of programming language to be used for its implementation. The common languages used for its implementation are Python and Matlab.

Machine Learning is a superset of multiple fields such as mathematics, statistics and computer science. The core idea behind machine learning is to give the computer an ability to learn and predict in terms of a given scenario and data set. The core essence of machine learning lye in its ability of clustering and pattern identification. This ability allows the larger entities to employ machine learning in products such as recommendation engines, medical diagnosis, economic projections and self-driving cars. [1] [2]

The paper focus on the machine learning domain in general and the algorithms in specific that are employed for the actual process and the analysis of languages that these algorithms are implemented in. The clustering algorithms under review will be fuzzy C-mean as it is similar to another clustering algorithm soft K-means. [3]

The basic working of C-means and soft K-means is similar in the sense that the instance can belong to more than one cluster at a time therefore this technique is the most commonly used in pattern recognition applications. The nature of fuzzy C-means algorithm is also iterative as K-means so in this work the crucial analysis will be based on the language performance analysis on evaluation parameters. For language analysis the important factor is the ability of cluster formation from the points of dataset related to the cluster. The iterative nature of this algorithm will help to analyze the underlying behavior of the programming languages (matlab and python) on the same clustering algorithm (as this algorithm is unsupervised so the computation complexity also differs in these languages). The selected programming languages provide the library support and share some common features they can be judged by the iterative functioning of fuzzy C-means.

In this work for classification, we will be using multiclass-support vector machine. This is an unsupervised learning model in machine learning and has an extension of Binary SVM with two classifiers. The reason for the choice is related to the analysis parameters for the language analysis and for its flexibility to belong to each instance. For the language analysis, its importance can be viewed from the perspective of library support and computation cost/processing on specific data type. The flexibility can be viewed from the perspective of the allocation of multiple labels to multiple margin classifiers (this higher margin is referred by extension of SVM).

A common problem faced in such scenarios is that of having an imbalanced data for training, to mitigate this problem we will be working on ADASYN to have a balanced data for training purposes. ADASYN is inherited from smote (oversampling algorithm). The difference being that ADASYN bias the sample space towards a non-homogenous neighborhood (mostly based on distance)

The clustering and classification techniques identify the instances and their relations to the related classes and therefore

the purpose of proper programming language utilization is important. The parameters of evaluation for the analysis of the programming language under review will be that of Exception Handling, Type Checking, Library Support, Syntax Design, Coding Efficiency, Computation Speed, and Scalability. The selected parameters are equally important for the language evaluation as the effect of exception handling is as crucial for performance analysis while dealing with the algorithmic flow of unsupervised learning as the other parameters that will be considered for further analysis. The clustering techniques that we have chosen are quite basic but the nature of the algorithm implementation is different due to language dependence that is fine for the analysis of such parameters that we aim to evaluate on matlab and python.

II. LITERATURE REVIEW

A. Python

Python is an object-oriented programming language with integrated dynamic semantics, used primarily for web and app development. It is highly popular for Rapid Application Development due to its dynamic typing and dynamic binding options.

Its a relatively simple language, so it's easy to learn. Its unique syntax focuses on readability, allowing both novices and professional developers to read and translate python code much easier than other languages. This enables a reduction of program maintenance and development, as it allows teams to collaborate effectively without significant barriers.

Python also supports the use of modules and packages, allowing code to be reused across various projects. Modules can be scaled for use in other projects, even allowing easy import or export.

One of the primary benefits of python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, developers flock to this platform, as these settings create a hotbed of creativity.

Python is an interpreted language, also referred to as a "scripting language", meaning that written code is not translated to computer-readable format at runtime. As python is now commonly used for large scale applications, the concept of it being a "scripting language" is being reconsidered. [4] [5] [6]

B. Matlab

Matlab (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language, which primarily focuses on matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Even though Matlab is intended for numerical computation, it contains optional packages that allow various functionality, such as symbolic computing, graphical simulation, and model-based design.

Matlab is an interactive system whose basic data element is an array that does not require dimensioning. This enables several technical computations, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

Matlab contains a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

Matlab also contains the ability to write C and Fortran programs that interact with other sections of matlab on non-matlab code. [7] [8] [9]

C. Clustering Techniques

The process of dividing the data points into groups such that data points in each group are similar and give an appropriate representation of said group, is known as clustering. In other words, the process splits instances into groups with similar traits and assign them into clusters.

They are different types of clustering methods (classified either hard or soft clustering), including: [10] [11]

- Partitioning methods
- Hierarchical clustering
- Fuzzy clustering
- Density-based clustering
- Model-based clustering

D. Classification Techniques

Classification is a supervised learning approach in which the system generates a model based on the input data, and then uses this model to classify future observations. This model takes into the consideration the character traits of each group, and based on the classification technique, enhances its consideration as more data points are considered. We use a dataset labeled for training to improve boundary conditions, which would then be utilized to determine each target class. The whole process is known as classification.

Here we have few types of classification algorithms in machine learning: [12] [13] [14] [15]

- Linear Classifiers: Logistic Regression, Naive Bayes Classifier
- Nearest Neighbor
- Support Vector Machines
- Decision Trees
- Boosted Trees
- Random Forest
- Neural Networks

E. Oversampling Techniques

Because the data might be imbalanced, or because an imbalance might come about by taking a completely random

data sample, our results get skewed. Hence, we introduce a bias to give certain samples of a specific group more weight over the other. This is known as oversampling. Basically, a technique used when the data is insufficient. It attempts to balance a dataset by emphasizing the size of rare samples. This is done by generating new rare samples rather than removing abundant samples. [16] [17] [18]

Oversampling techniques for classification problems include:

- Random oversampling
- SMOTE
- ADASYN

F. Fuzzy c-mean

Fuzzy clustering (also referred to as soft clustering or soft k-means) is a clustering technique where each data point can belong to more than one cluster. [19] [20] [21] [22]

The fuzzy c-means algorithm is very similar to the k-means algorithm, and is as follows:

- Choose a number of clusters.
- Assign coefficients randomly to each data point for being in the clusters
- Repeat until the algorithm has converged (that is, the coefficients' change between two iterations is no more than $\mathcal{E}$, the given sensitivity threshold):
 - Compute the centroid for each cluster (shown below).
 - For each data point, compute its coefficients of being in the clusters.

G. Multi-class SVM

SVM inherently uses two classifiers, which require labelling the data and applying the models on both classes. However, in real world applications, we typically see multiple classes, where a simple SVM would prove insufficient.

Multiclass SVM solves this problem by forming multiples of paired classifiers based on the feature vectors that are derived from the input data. This can be achieved by distinguishing the classifiers based on particular labels against the remaining labels (one-vs-all), or between individual pairs of classes (one-vs-one). Multiclass SVM consists of two modules, one for learning and one for classification. [23] [24] [25] [26]

H. ADASYN

Adaptive synthetic sampling, or ADASYN, is an algorithm that generates synthetic data, boasting the ability to generate rare samples and not being limited to only copying minority data. ADASYN being derived from SMOTE, is similar to it, with one major difference. The difference being that ADASYN bias the sample space towards a non-homogenous neighborhood. [27] [28] [29]

I. Comparing Programming Languages

The following factors weigh heavily on software development, and hence should be considered during language comparison:[30] [31]

- Application requirements: Languages tend to be suited to particular applications.
- Platform requirements: The platform on which an application is to be executed may have a limited choice of language implementations available.
- Development time: How quickly could the application be developed in a particular language
- Portability: Execution platforms for programs tend to change over time

Criteria for comparing programming languages include:[32] [33]

- Simplicity of language constructs, which relates to ease of programming
- Readability, which relates to maintainability, an important factor as many programs greatly outlive their expected lifetimes (witness the Y2K software crisis)
- How tuned a language's features are for a particular application (e.g., Perl relates well to text processing)
- Compilation speed
- Runtime efficiency, in terms of speed and machine resources
- Library support
- Debugging help
- Language safety
- Longevity of language and compiler tools
- Portability across platforms and machine architectures

III. METHODOLOGY

The methodology that followed for the accomplishment of language analysis has been modeled in figure 1 where the first phase is the collection of data-set. This step is very important for accomplishment of main objective that is language analysis; the reason is that the size of data-set matters. As each language has its own specification for data loading similarly the time required for it.

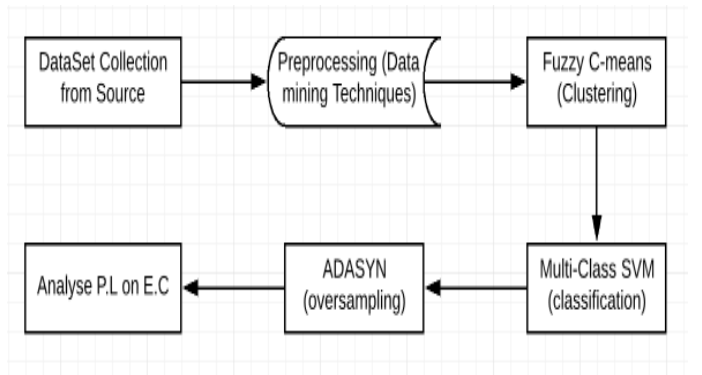


Fig. 1. Emulated Model: Each step(essentially algorithms) will be performed in two programming languages python and matlab and will be evaluated according to evaluation criteria.

For our intents and purposes, it was decided to use a well known and established dataset. For this, the *iris* dataset from the UCI repository was selected, shown in table I.

TABLE I
IRIS DATASET FROM UCI REPOSITORY

Data set characteristics	Multivariate
Number of instances	150
Area	Life
Number of attributes	4
Attribute characteristics	Real
Associated tasks	Classification
Missing values	Nil
Number of web hits	3366076
Date donated	1988-07-01

This dataset has already been pre-processed, hence allowing us to move onto the next stage directly.

The dataset could potentially be imbalanced. In this case several dataset balancing algorithms exist. For the sake of following the original paper, we will be using ADASYN algorithm, which involves generating new samples in the dataset in favour of the minority class. Doing so will allow our machine learning algorithm of the next stage to better assess the dataset, and potentially generate more accurate clusters.

The next stage involves clustering our data. For this, we will be using the Fuzzy c-means algorithm. Based on best practices, the dataset will be split into test and training group with a 90-10 ratio. The clustering algorithm will work on the training dataset, and generate clusters.

After generating appropriate clusters for the dataset, we can move onto the final stage of classification. Here, we will be using the multi-class SVM classifier. This will work on the test dataset and generate a confusion matrix, which will detail the amount of sample accurately classified.

TABLE II
EXPERIMENTAL SYSTEM DETAILS

System Details	
Model	i7-4600U
Cores	4
Threads	4
min MHz	800
max MHz	3300
Base Speed	2.10
Boost Speed	3.30
L1-I Cache	32 KB 8-way associative
L1-D Cache	32 KB 8-way associative
L2 Cache	256 KB 8-way associative
L3 Cache	4 MB 16-way associative (shared)
RAM	8 GB

IV. EXPERIMENTAL DETAILS

An environment was setup for both languages. Using the Anaconda distribution, python and all its dependencies were installed onto the operating system. For matlab, the R2015a version of matlab was installed onto the same operating system.

After setting up the environments, the *iris* dataset was downloaded from the UCI repository.

Using only inbuilt functionality, and basic coding constructs, the data set was first balanced using ADASYN, then after clustering with fuzzy c-means algorithm, the result was classified using a multi-class svm. This process was repeated in both languages.

For the python language, the internal optimized functions for fuzzy c-means and ADASYN were used, while the multi-class SVM was custom coded by a developer on GitHub, following best practices. The figures 2 and 3 show the initial random clusters of the dataset before any operations are applied, while the figures 4 and 5 show the final clusters of the dataset after applying the operations.

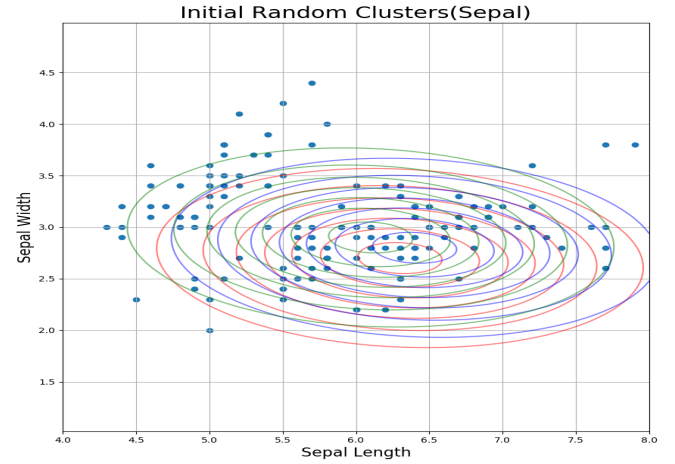


Fig. 2. Python: Initial Random Clusters (Sepal)

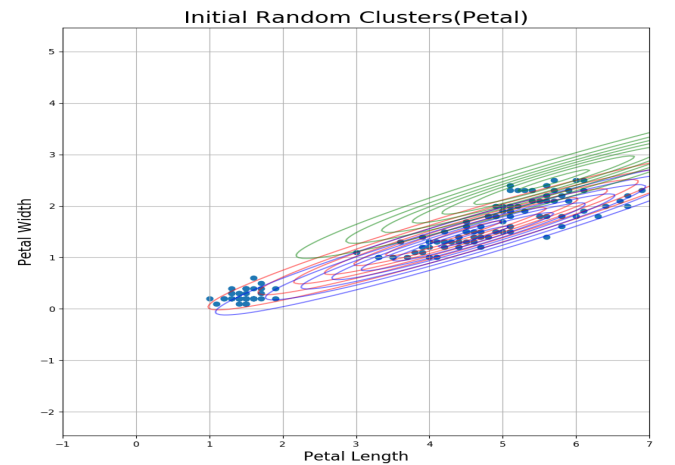


Fig. 3. Python: Initial Random Clusters (Petal)

For Matlab, each operation was applied in a different manner, For ADASYN, the code was supplied by a GitHub

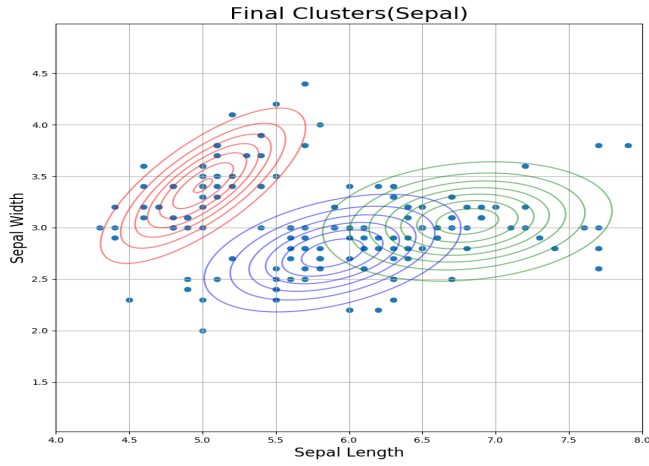


Fig. 4. Python: Final Clusters (Sepal)

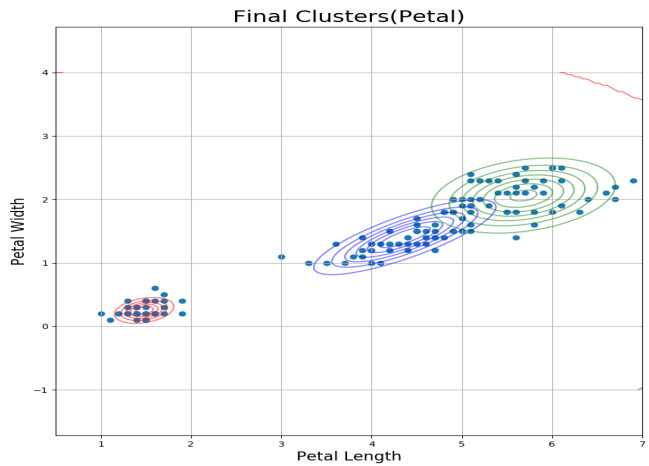


Fig. 5. Python: Final Clusters (Petal)

repoitory; for fuzzy c-means, the code was inbuilt; for the multi-class SVM, the Matlab learning applet was used. This applet would take the dataset previously worked on, and apply certain desired machine learning models on them, which in this case was SVM. After applying fuzzy c-means, we recieved the figure 6. After applying SVM, the learning applet was used to plot the figures 7 and 8.

In order to compare accuracy, a confusion matrix was generated at the end of all operations. After taking an average of 10 runs, the final confusion matrix can be seen in table III for python, and in table IV for Matlab. A 90-10 split was made in the dataset for python as per best practices, whereas in Matlab, the entire dataset was used for both training and testing.

In python, fuzzy c-means clustering resulted in a 93% accuracy, while SVM classification resulted in a 88.67% accuracy.

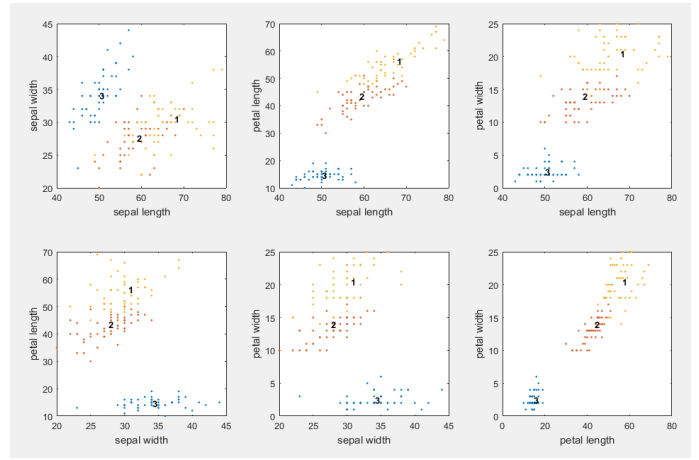


Fig. 6. Matlab: fuzzy c-means clustering

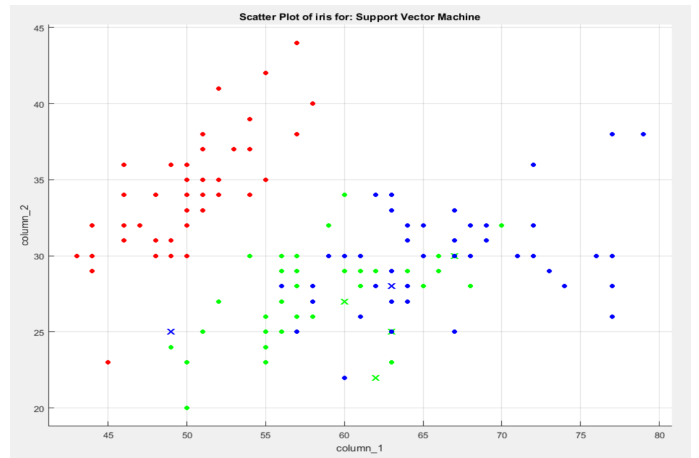


Fig. 7. Matlab: SVM clusters (Sepal)

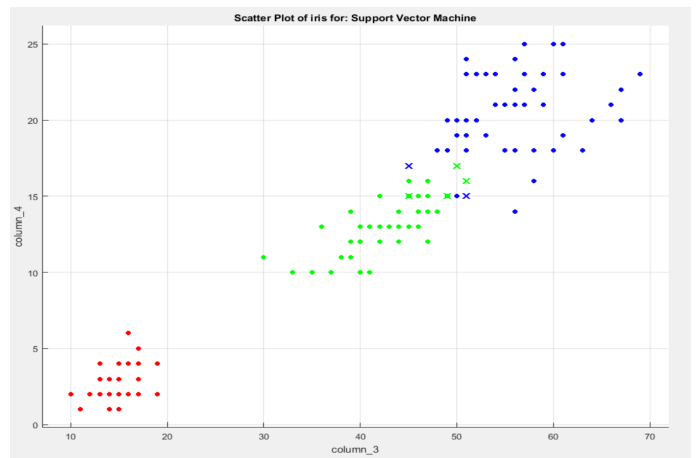


Fig. 8. Matlab: SVM clusters (Petal)

TABLE III
PYTHON: CONFUSION MATRIX

	Setosa	Versicolor	Virginica
Setosa	9	0	0
Versicolor	0	8	1
Virginica	0	1	11

TABLE IV
MATLAB: CONFUSION MATRIX

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	46	4
Virginica	0	2	48

In python, fuzzy c-means clustering resulted in a 92% accuracy, while SVM classification resulted in a 96% accuracy.

V. RESULTS AND DISCUSSION

After performing the classification techniques on the datasets, certain points of the programming languages are highlighted. The details are mentioned later on, however for simplification, the figure 9 shows the score each programming language achieved in its respective categories, based on experience gained from this study. The scoring is as follows:

- 0) Non Existent
- 1) Bad
- 2) Mediocre
- 3) Ok
- 4) Good
- 5) Excellent

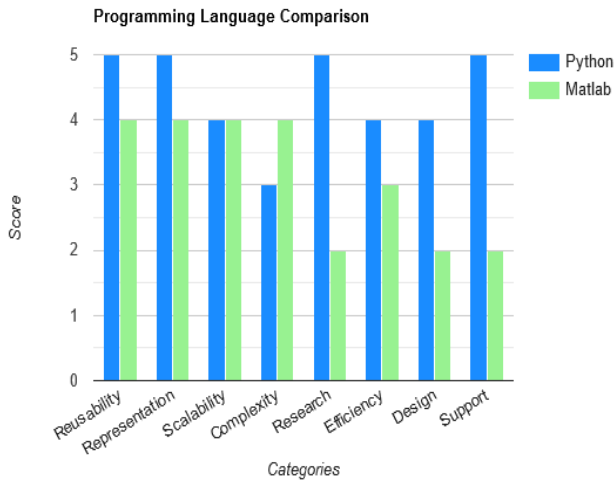


Fig. 9. Programming language comparison based on experience

A general comparison between python and matlab can be seen from table V.

To gauge more from experiences gained in this study, each category has been discussed in detail below.

TABLE V
GENERAL COMPARISON OF PROGRAMMING LANGUAGES

	Python	Matlab
Paradigm(s)	imperative, object-oriented, functional, aspect-oriented, reflective	procedural, imperative, array programming
Standardized	No, language reference included with each version's documentation	No, however the language is also implemented in Octave and FreeMat
Type Strength	strong	strong
Type Safety	safe	unsafe
Expression of Types	implicit	
Type Compatibility		
Type Checking	dynamic	dynamic
Parameter Passing Methods	by value (Call by object reference)	by value
Garbage Collection	Yes	Yes
Intended Use	Application, Education, Scripting	Numeric computation and visualization
Design Goals	Simplicity, Readability, Expressiveness, Modularity	At the beginning designed as interpreter for easy use of fortran libraries, nowadays high performance numerical analysis and visualization

A. Data loading /Representation

The data loading and presentation is a very basic feature that has been noticed in this experiment. For each dataset and classification technique data has been loaded and after classification and clustering the results has been visualized. For data loading panda library of python is used in both experiments while as in matlab the function call uimport has been used. The efficiency of a programming language for data loading feature also depends on the data format and the size of file. Matlab is better option for data loading as compare to python because of intended code complexity in terms of package requirement and time required for file read. Data representation is very important aspect for language analysis because this clarifies the programmer point of view for showing data patterns. Similarly, the results of the classification and clustering are easy to understand when visualized instances are referred. For data loading and representation matlab is very fine option as a whole considering the beneath details of library/package support.

B. Scalability

Scalability is the feature of a programming language where a software solution can handle increased loads of work. This can be larger data-sets, higher request rates etc. A solution that can scale out can usually grow to larger loads in a more cost effective way. Amdahl's law that states that the ability to scale out is limited by the sequential part of the software. Python is not highly scalable language like the data that range over than ram the performance is effected badly but there are certain plug-ins that improves this drawback. The scalability

feature of matlab is very prominent due to available support. The design of matlab provides proper scaling to the software level that assist the growing size loads.

C. Computation Complexity

The computation complexity of the codes has been analyzed with respect to the cpu clock cycles required for particular computations. For that purpose, perf Linux based tool is used. In Table1, there are essential features (parameters) of the test machine which plays a significant role in computation time (CPU clock cycles).

The runs on both programming languages showed negligible difference. This implies that the dataset is too small to push the hardware enough to make a significant impact on the performance time of the programming language.

D. Reusability/Maintainability

The reusability and maintainability are the features that make a language more suitable choice for the programmer. Python receives the highest score, followed by matlab. The reason is tied to the efficiency and availability of built-in and specific libraries for machine learning. More importantly, tied to the method of interoperability. In other words, depending on how the developer designed the libraries functions to handle data, the final code will be either highly reusable, on single use only. Based on the experience in this study, python offered the highest reusability, with often the only requirement being to load a new dataset. Matlab had the second highest score, with greater needs like also changing the dimensions and adjusting features of the dataset. Python got higher score because it is easy to understand and simple. It requires less coding effort because of higher maintainability. Matlab less score as compare to python. As matlab is very easy and simple but with machine learning perspective it has some inherited complexities. Matlab requires more coding effort and it also has some package dependencies issues that makes it less maintainable but if the classification learner is used for classification purposes and some of its built-in standard format supported tools makes the task more easy and but for customization this is not a good option at all.

E. Research

Python having such a large feature set, is constantly a first choice of developers for machine learning applications. As such, it is a constant in research papers, and developers are consistently improving algorithms in python. As such, python has the best score for 'Research'.

Finally, matlab has the lowest score due to its lack of popularity, and limited scope in machine learning. If it's used, its primary focus is on specific types of data processing.

F. Coding Efficiency

This category is a bit tricky. Matlab has more elaborate basic constructs, while python has simpler concrete constructs. Due to matlabs elaborate constructs and simple goals, it allows users to quickly setup basic tasks. But setting up a larger

project would be unorganized, and limit the users ability greatly.

Pytho on the other hand has simpler constructs, requiring a greater amount of effort form the user to build their code. However, dealing with larger projects becomes easier, and introducing new code into an already built project becomes simpler. As a whole, python provides a better long-term experience.

G. Syntax Design

Python once again receives the highest score, as it possesses a better syntax design. Python syntax is designed to be highly readable, and set into logical lines. The semantics allow both novice programmers, and experts of a different domain to easily grasp the syntax, and begin coding. Its part of the reason of the languages popularity.

Matlab syntax is a bit of a mix and match, often not maintain consistency. This lack of a consistent design stems from attempting to make a language friendly to code and compute mathematical equations. It is difficult for both novice and experts to grasp, though once completely understood, the language can achieve quite a many feats.

H. Library Support

Python receives the highest score for library support. As python is one of the most popular languages in use, there are numerous developers actively working on cutting edge research. As such, they make it easy for new researchers and developers by making their work public, and creating public repositories for their work. Beyond making their code public, they even properly document their work for others to interpret.

Matlab has a smaller following, primarily due to the application and its features being paid. Moreover, python being more flexible than matlab, can thoroughly implement any feature matlab has to offer. Hence, not many people work in matlab, and hence very few works become available for public use.

VI. CONCLUSION

In conclusion, python is a far better language for machine learning algorithms. However, this statement comes with a caveat, that the answer depends truly on the required task.

If the user requires a a tool that will allow quick results, with limited know-how and needs to work on small-scale projects focusing primarily on simple yet extensive computation, then matlab is the better choice.

However, if the user requires a tool for a potentially scaleable project, with robustness that will provide any and all features that the user could potentially want either now or later, with the primary goal being anything other than simple computation, then python is the better choice.

From a research perspective, python is the better option as well, as it has a larger user base, open-source libraries of work, and more recent publications and active projects.

Beyond these parameters, both languages exhibit limited differences.

REFERENCES

- [1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [2] Donald Michie, David J Spiegelhalter, CC Taylor, et al. “Machine learning”. In: *Neural and Statistical Classification* 13.1994 (1994), pp. 1–298.
- [3] Thomas G Dietterich. “Machine-learning research”. In: *AI magazine* 18.4 (1997), pp. 97–97.
- [4] Travis E Oliphant. “Python for scientific computing”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20.
- [5] Mark Lutz. *Programming python.* ” O’Reilly Media, Inc.”, 2001.
- [6] Guido Van Rossum et al. “Python Programming Language.” In: *USENIX annual technical conference*. Vol. 41. 2007, p. 36.
- [7] Holly Moore. *MATLAB for Engineers*. Pearson, 2017.
- [8] Michael Paluszczek and Stephanie Thomas. *MATLAB machine learning*. Apress, 2016.
- [9] Phil Kim. “Matlab deep learning”. In: *With Machine Learning, Neural Networks and Artificial Intelligence* 130 (2017).
- [10] Aastha Joshi and Rajneet Kaur. “A review: Comparative study of various clustering techniques in data mining”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 3.3 (2013).
- [11] Amit Saxena, Mukesh Prasad, Akshansh Gupta, et al. “A review of clustering techniques and developments”. In: *Neurocomputing* 267 (2017), pp. 664–681.
- [12] Thair Nu Phyu. “Survey of classification techniques in data mining”. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 1. 2009, pp. 18–20.
- [13] V Krishnaiah, G Narsimha, and N Subhash Chandra. “Survey of classification techniques in data mining”. In: *International Journal of Computer Sciences and Engineering* 2.9 (2014), pp. 65–74.
- [14] Miao Li, Shuying Zang, Bing Zhang, et al. “A review of remote sensing image classification techniques: The role of spatio-contextual information”. In: *European Journal of Remote Sensing* 47.1 (2014), pp. 389–411.
- [15] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. “Supervised machine learning: A review of classification techniques”. In: *Emerging artificial intelligence applications in computer engineering* 160 (2007), pp. 3–24.
- [16] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. “Handling imbalanced datasets: A review”. In: *GESTS International Transactions on Computer Science and Engineering* 30.1 (2006), pp. 25–36.
- [17] Anjana Gosain and Saanchi Sardana. “Handling class imbalance problem using oversampling techniques: A review”. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2017, pp. 79–85.
- [18] Rushi Longadge and Snehalata Dongre. “Class imbalance problem in data mining review”. In: *arXiv preprint arXiv:1305.1707* (2013).
- [19] James C Bezdek, Robert Ehrlich, and William Full. “FCM: The fuzzy c-means clustering algorithm”. In: *Computers & Geosciences* 10.2-3 (1984), pp. 191–203.
- [20] Nikhil R Pal, Kuhu Pal, James M Keller, et al. “A possibilistic fuzzy c-means clustering algorithm”. In: *IEEE transactions on fuzzy systems* 13.4 (2005), pp. 517–530.
- [21] Nikhil R Pal and James C Bezdek. “On cluster validity for the fuzzy c-means model”. In: *IEEE Transactions on Fuzzy systems* 3.3 (1995), pp. 370–379.
- [22] Robert L Cannon, Jitendra V Dave, and James C Bezdek. “Efficient implementation of the fuzzy c-means clustering algorithms”. In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1986), pp. 248–255.
- [23] Kai-Bo Duan and S Sathiya Keerthi. “Which is the best multiclass SVM method? An empirical study”. In: *International workshop on multiple classifier systems*. Springer. 2005, pp. 278–285.
- [24] Gjorgji Madzarov, Dejan Gjorgjevikj, and Ivan Chorbev. “A multi-class SVM classifier utilizing binary decision tree”. In: *Informatica* 33.2 (2009).
- [25] Chih-Wei Hsu and Chih-Jen Lin. “A comparison of methods for multiclass support vector machines”. In: *IEEE transactions on Neural Networks* 13.2 (2002), pp. 415–425.
- [26] Vojtech Franc and Václav Hlaváč. “Multi-class support vector machine”. In: *Object recognition supported by user interaction for service robots*. Vol. 2. IEEE. 2002, pp. 236–239.
- [27] Haibo He, Yang Bai, Eduardo A Garcia, et al. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE. 2008, pp. 1322–1328.
- [28] Annisa Aditsania, Aldo Lionel Saonard, et al. “Handling imbalanced data in churn prediction using ADASYN and backpropagation algorithm”. In: *2017 3rd International Conference on Science in Information Technology (ICSITech)*. IEEE. 2017, pp. 533–536.
- [29] Yulia Ery Kurniawati, Adhistya Erna Permanasari, and Silmi Fauziati. “Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data”. In: *2018 4th International Conference on Science and Technology (ICST)*. IEEE. 2018, pp. 1–6.
- [30] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, et al. “MatLab vs. Python vs. R”. In: *Journal of Data Science* 15.3 (2017), pp. 355–372.
- [31] Paul Miles, Isaac Michaudb, and Ralph Smitha. “MCMC Demonstrations in MATLAB, Python, and R”. In: *MH* 208.207 (2018), p. 205.

- [32] Martin Stein and Andreas Geyer-Schulz. “A Comparison of Five Programming Languages in a Graph Clustering Scenario.” In: *J. UCS* 19.3 (2013), pp. 428–456.
- [33] Guy Helmer Pradeep Bollineni Patrick Hartling. *Comparing Programming Languages*. 1998. URL: <http://www.eecs.ucf.edu/~leavens/ComS541Fall98/hw-pages/comparing/>.