ANALYSIS OF THE EFFECT THAT ENTRY DATA SIZE HAS IN AN ALGORITHM

INSTITUTO TECNOLÓGICO DE COSTA RICA

PROFESSOR: EFRÉN JIMÉNEZ DELGADO II SEMESTER 2022 GROUP: 2

Isaac David Araya Solano
Computer Engineering
Carnet: 2018151703

Alexia Denisse Cerdas Aguilar
Computer Engineering
Carnet: 2019026961

***Abstract*—The motivation behind this project consists of the desire to apply the knowledge acquired in this subject to understand, by comparing the results from different tests, the reasons why is the efficiency of an algorithm and the way it is implemented so important. The purpose of the previously mentioned tests is to analyze the behavior of the chosen algorithms depending on the entry data variability, either in quantity or type. We proceeded by analyzing the Decision Tree and the K-means algorithms in phases with an entry data from a range of 10 to 100000. After running the tests, an evaluation of the results was performed and it was concluded that both algorithms have a linear time complexity and that the amount of entry data affects completely how efficient an algorithm is.**

***Keywords*—** efficiency, k-means, decision tree, time complexity, big-O, machine learning, python

I. Introduction

As the technology evolves and software and computers become more common in every area of the present world a very important aspect is to be considered: Efficiency. This world is ruled by efficiency, in both time and resources. As any other product on the market, software is also ruled by this principle; in this case, the resources that are needed to be focus on are time and memory.

In this project two algorithms will be studied, the K-means Unsupervised Algorithm and Decision Tree Supervised Algorithm; mainly focused on their performance. Both of them are Machine Learning algorithms implemented in completely different ways and purposes. This investigation aims to answer this question: How does the amount of entry data affects the performance of the algorithms?

In order to answer the question a variety of measurements will be performed to each algorithm, considering the execution time, the assignments on memory, the comparisons performed and the amount of lines of code executed. Then, an exhaustive analysis of the results will also be done with the intention of determining the behavior of the mentioned algorithms and classify them correctly.

As a result, the project is structured with the following sections: Related Projects, Solution, Methodology, Evaluation, Conclusions, Annexes and finally the References used throughout the project.

A. Objectives

- Implement flow-net algorithms efficiently in Python programming language.
- Analyze the algorithms by empirical, analytical, graphic and size factor measurements.
- Identify aspects that affect the efficiency of an algorithm.

II. Related Projects

To understand the concept of each algorithm an exhaustive research was done on how they work and their purposes, as noted in [1] Machine Learning (ML) can be broadly classified into two: supervised and unsupervised learning. In the case of supervised algorithm, maps a given input to an output based on the available input–output; otherwise, the unsupervised algorithm is used when this input data is not given nor available.

According to Ghosh and Kumar in their investigation [2] data clustering is considered as an unsupervised learning process which does not require any labelled dataset as training data and the performance of data clustering algorithm is generally considered as much poorer. Although data classification is better performance oriented but it requires a labelled dataset as training data and practically classification of labelled data is generally very difficult as well as expensive.

Therefore, the Decision Tree algorithm is a supervised machine learning algorithm that is used for classification and regression tasks. It is trained with an amount of data that is already classified for the algorithm to learn how to classify the future entry data [3], as opposed to the K-Means algorithm is an unsupervised machine learning used to find out the clusters of data in a given dataset, it depends on the value of k where clustering with different k values will eventually produce different results [4].

Finally, the algorithm complexity plays an important role when analyzing them since it allows to classify algorithms according to their behavior. As described by Abdiansah and Wardoyo [5] it is calculated using Big-O notation and it can be divided into two kinds of complexity: time complexity (how long the algorithm is executed) and space complexity (how much memory is used by its algorithm).

III. Solution

Within each code of each algorithm there are calculations carried out to make the tests more analytical and informative. They are both programmed in Python programming language and all of those calculations were estimated with some counters to analyze in the best way the results of them, every method executed is explained and more detailed in the Methodology of this project.

A. Supervised Algorithm: Decision Tree

This algorithm, shown in [3], works with both regression and classification tasks. It is considered one of the most intuitive and easy machine learning algorithms. The logic behind them is based on nodes and branches. There a different types of nodes in this algorithm:

- Root Node: contains a feature that best splits the data.
- Decision Nodes: nodes with conditions to classify this data.

- Leaf Nodes: final nodes at which prediction is made.

The first thing to do is to determine a root node. For this purpose the algorithm must choose a feature or condition to use it as a tool to classify the target variable. Then there must be a criteria to determine how pure the features are and which is the purest. This time the Entropy metric is being used, choosing the feature with the lowest entropy as the best and the one with the highest as the worst. The one with lowest entropy is used as root node. Then, each decision node will be determined by how much information is gained by each split, using an specific formula to determine which decision node is better. This will continue until all training data is finished and at the end we will have a decision tree able to perform predictions based on the data provided as the entry data.

The algorithm was modified by adding a random entry data and classification using the NumPy and Pandas modules [6] [7]. There are 3 columns and they are classified randomly between 3 possible classifications.

B. Unsupervised Algorithm: K-Means

The approach and structure of this algorithm, seen in [4], was used to analyze its behavior with the chosen methodology. Moreover, some modules are required to run the whole program, such as: NumPy, Pandas, Time and SciPy.

Its logic consists of five steps as mentioned in [4]: pick k data points as the initial centroids, then find the distance between each data points in the training set with the k centroids, assign each data point to the closest centroid according to the distance found, update centroid location by taking the average of the points in each cluster group and finally repeat fourth and fifth steps till the centroids don't change.

The modifications made to the algorithm itself was how data is selected and loaded with Pandas and NumPy modules [6] [7], the type of dataset the algorithm receives as parameter and use it inside and the k value is a constant equal to 3 that could be changed; however, for project purposes we will keep it at the same value for all tests.

IV. Methodology

The methodology of the present project is based on obtaining accurate information about these algorithms with the different tests that were run. This section includes empirical and analytical method, graphic measurement and size factor, the following tests are composed of: assignments count, comparisons count, executed code lines count and execution time. Each method and measurement let us did an analysis of the complexity based on the code of the algorithms and classify them according to their Big O notation.

A. Empirical

The first step is to modify the code of the algorithms and add the necessary instruction to count each operation and store them in variables. When finished, the tests are executed with the following amounts of data: 10, 50, 7000, 10000, 25000, 50000 and 100000, with the goal to observe the behavior of the algorithm. The purpose for acquiring this information is being able to analyze and identify the impact that the amount of items has in the performance. The next tables present the data acquired for both algorithms:

As we can observe, in Table I and Table II, each result is increasing depending on two variables shown in both figures which are: Operations and Array size. Furthermore, both algorithms work differently but still indicate a linear complexity whereby in this scenario they could be classified as Linear complexity or more specifically notation: **O(n)** where n is the amount of entry data we tested.

TABLE I
EMPIRICAL MEASUREMENT OF THE DECISION TREE ALGORITHM

Operations	Array Size						
	10	50	7000	10000	25000	50000	100000
Assignments	2883	42333	15219882	23266004	58266752	102976407	191163057
Comparisons	1201	24323	15014396	23069456	58008219	102608528	190669306
Executed Lines	1880	20081	264884	251063	359891	678870	958952
Execution Time (s)	0.0019	0.019	5.058	7.811	19.617	34.999	64.728
Code Lines	182						

TABLE II
EMPIRICAL MEASUREMENT OF THE K-MEANS ALGORITHM

Operations	Array Size						
	10	50	7000	10000	25000	50000	100000
Assignments	1320	6160	847110	1210110	3025110	6050110	12100110
Comparisons	81	401	56001	80001	200001	400001	800001
Executed Lines	116	556	77006	110006	275006	550006	1100006
Execution Time (s)	0.015	0.07	11.67	15.79	43.89	85.52	212.58
Code Lines	47						

B. Analytical

The following step is to complete a full analysis of the time complexity of the algorithms to obtain the polynomial expression and representation of them.

Although, there are a few nested loops in both programs where we recognized that there was a constant k, seen in Table IV, which is the number of centroids established in the code and it doesn't affect the complexity of the K-Means algorithm, while in the Decision Tree algorithm, showed in Table III, there is a variable called m that is referring to the amount of features that the data presents and the n represents the number of rows the data presents, also it shows the expression of the important functions in the program.

Additionally, it is significant for us to point it out and evidence that it could happen most of the time analyzing other algorithms, for that reason the polynomial expression of each one is Linear complexity, classified as **O(n)** notation. Here are the results of the analysis:

TABLE III
ANALYTICAL TABLE OF DECISION TREE ALGORITHM

Lines	Source Code	Measurement Executed Lines for Worst Case
90 -153	bestSplit(X, y)	24mn + m + 6
158 - 210	build(X, y, depth)	27
212 - 226	fit(X, y)	1
Total steps		**24mn + m + 34**

TABLE IV
ANALYTICAL TABLE OF K-MEANS ALGORITHM

Line	Source Code	Measurement Executed Lines for Worst Case
18	idx = np.random.choice(x.shape[0], k, replace=False)	4
22	centroids = x.iloc[idx, :]	1
26	distances = cdist(x, centroids ,'euclidean')	4
30	points = np.array([np.argmin(i) for i in distances])	3
34	for _ in range(no_of_iterations):	n+1
35	centroids = []	n+1
38	for idx in range(k):	n(k+1)
40	temp_cent = x[points==idx].mean(axis=0)	n(k+2)
41	centroids.append(temp_cent)	n(k+2)
46	centroids = np.vstack(centroids)	n+2
49	distances = cdist(x, centroids ,'euclidean')	n+4
52	points = np.array([np.argmin(i) for i in distances])	n+3
60	return points	1
Total steps		**10n+3nk+24**

C. Graphic Measurement

The next procedure is to run the tests with fewer data aiming to create a graphic representation of the assignment results and the comparison results to visualize the changes in performance and how the complexity of the algorithm affect these aspects. The quantities used were: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

The Decision Tree shows in Figure 1 and Figure 2 a stable growing behavior that resembles a linear function but with some imperfections. While K-Means algorithm shows almost the same time complexity as the Decision Tree algorithm, affected by the two important factors mentioned, the small difference is shown in Figure 3 and Figure 4 where the results are more straight and directed upwards.

Performing this method allows to understand and visualize the behavior of this algorithm and how the data is represented, also it helps to detect patterns, trends, relationships and structures in the entry data tested for this project. Both algorithms have Linear complexity as a consequence of what is demonstrated in the four figures already stated, and it is represented as **O(n)** where *n* is the amount of entry data we tested.

Here are the graphic representation of the results of both algorithms and their visual differences:

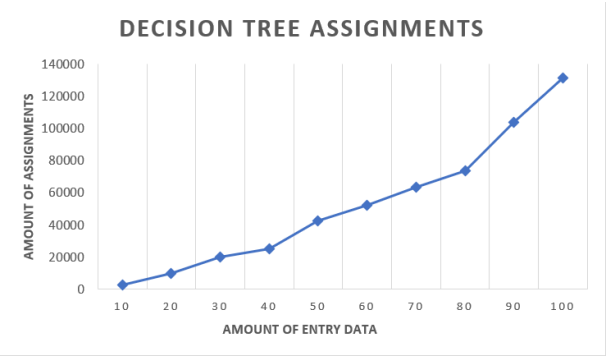

Fig. 1. Assignments Graph of Algorithm Decision Tree

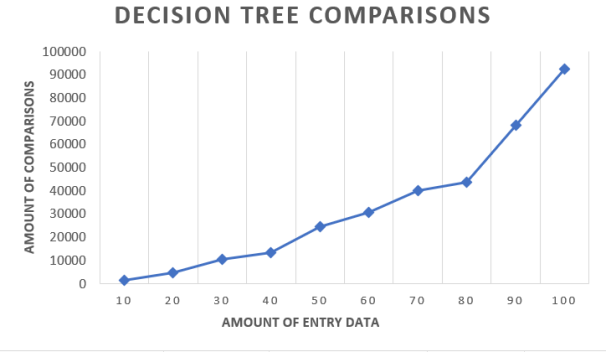

Fig. 2. Comparisons Graph of Algorithm Decision Tree

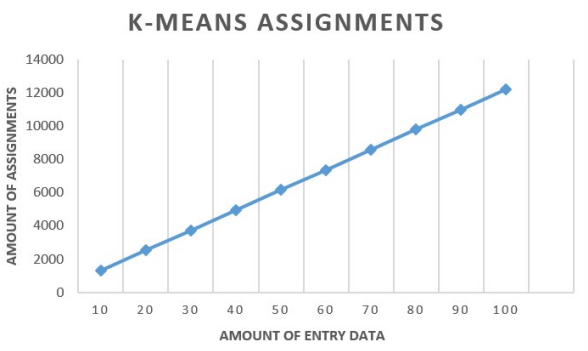

Fig. 3. Assignments Graph of K-Means Algorithm

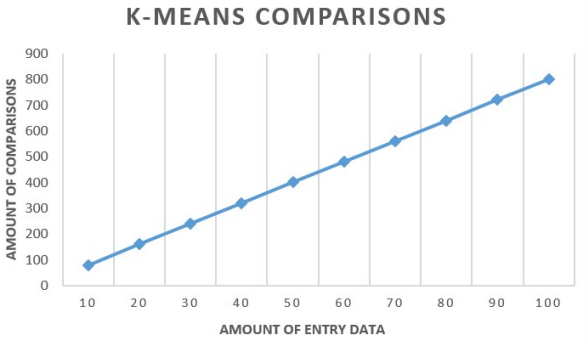

Fig. 4. Comparisons Graph of K-Means Algorithm

D. Size Factor

Finally, the last process to be executed is to calculate the Size Factor. The size factor is the way in which we can compare the size ratio of two entry data and the other size ratio results obtained through performing tests to those entry data amounts. This factor will show if the increase of the data is proportional or not to the changes in performance and will give a clue or a solid guide towards how the algorithm behaves. In this case the following amounts of entry data are used: 10, 50, 100, 200, 500, 1000, 10000, 50000, 100000. Here are the resulting tables with the Size factors in each variable studied:

TABLE V
FACTOR TABLE OF DECISION TREE ALGORITHM

Size	Size Factor	Assingments Factor	Comparisons Factor	Executed Lines Factor	Execution Time Factor
10 a 50	5	42333/2883 = 14.68	24323/1201 = 20.25	20081/1880 = 10.68	0.019/0.0019 = 10
50 a 100	2	108805/42333 = 2.57	72875/24323 = 3	40088/20081 = 2	0.0651/0.019 = 3.42
100 a 200	2	279924/108805 = 2.57	213796/72875 = 2.93	73653/40088 = 1.84	0.1147/0.0651 = 1.76
500 a 1000	2	2318196/996779 = 2.33	2201713/878413 = 2.51	131846/115397 = 1.14	0.808/0.364 = 2.22
10000 a 100000	10	191163057/23266004 = 8.22	190669306/23069456 = 8.27	958952/251063 = 3.82	64.728/7.811 = 8.29
50000 a 100000	2	191163057/102976407 = 1.86	190669306/102608528 = 1.86	958952/678870 = 1.41	64.728/34.999 = 1.85

TABLE VI
FACTOR TABLE OF K-MEANS ALGORITHM

Size	Size Factor	Assignments Factor	Comparisons Factor	Executed Lines Factor	Execution Time Factor
10 a 50	5	6160/1320 = **4.67**	401/81 = **4.95**	556/116 = **4.79**	0.07 /0.015 = **4.67**
50 a 100	2	12210/ 6160 = **1.98**	801/401 = **2.00**	1106/556 = **1.99**	0.15/0.07 = **2.14**
100 a 200	2	24310/12210 = **1.99**	1601/801 = **2.00**	2206/1106 = **1.99**	0.30/0.15 = **2.00**
500 a 1000	2	121110/60610 = **2.00**	8001/4001 = **2.00**	11006/5506 = **2.00**	1.69/0.92 = **1.84**
10000 a 100000	10	12100110/1210110 = **10.00**	800001/80001 = **10.00**	1100006/110006 = **10.00**	212.58/18.12 = **11.73**
50000 a 100000	2	12100110/6050110 = **2.00**	800001/400001 = **2.00**	1100006/550006 = **2.00**	212.58/85.52 = **2.49**

In Table V we can see the conduct of the Decision Tree algorithm and in Table VI is projected the behavior of the K-Means algorithm. Evidently the columns right after the size factor are numbers close to the size factor of each size calculated. It is increasing linearly and their behavior of how much it increases from the first entry data to the other one is very similar. There may be a confusion with the Decision Tree as it is closely similar to a Logarithmic algorithm; however, relating Table V with the behavior presented in Figure 1 and Figure 2 in the previous section we can infer that this result is linear as well. Therefore, both algorithms have a Linear complexity **O(n)** in every aspect of the table.

V. EVALUATION

After all the tests performed in both algorithms the initial question proposed *How does the amount of entry data affects the performance of the algorithms?*: The amount of entry data can affect the performance of an algorithm and the way it affects depends on how the algorithm is programmed. If the complexity of the algorithm is quadratic then increasing the amount of data slightly may decrease the efficiency significantly but in other kinds of complexity it may not be that significant. Using the information gathered by all the tests run, as seen in Table VII, we can conclude that both algorithms present a Linear complexity which is **O(n)** notation.

TABLE VII
COMPLEXITY CLASSIFICATION OF THE ALGORITHMS

Algorithm	Method	Big-O Classification
K-Means	Empirical	Linear
	Analytical	Linear
	Graphic	Linear
	Size Factor	Linear
Decision Tree	Empirical	Linear
	Analytical	Linear
	Graphic	Linear
	Size Factor	Linear

These algorithms are designed for similar but different tasks. The Decision Tree is better to classify information when a previous classification was provided since it works as a Supervised Machine Learning algorithm and it would be trained to perform predictions based on the previous training. On the other hand, the K-Means algorithm is better when you only have data without any classification as it would create clusters or "groups" of data and classify them by the similarity of their characteristics. Since both of them have a Linear complexity we cannot say any of them is significantly better than the other; however, the Decision Tree is slightly faster doing its task than the K-Means, even though they are not really comparable considering they are not designed for the same purposes.

VI. CONCLUSIONS

During the making of this project we learned how important the efficiency of an algorithm can be. Even computational resources are limited and the necessity to optimize them is a reality we have to face daily. Our purpose with this paper was to provide an analytical approach to this problem in order to help people understand how the time complexity of the algorithms work and how the structure and design of the code can be crucial when optimizing an algorithm.

We determined that both K-means algorithm and Decision Tree algorithm have a linear complexity, meaning that the growth in time and memory will be constant depending on the amount of data provided. This shows how the only fact that affects the time and memory that these algorithms use is the size of the entry data, answering the question we proposed at the beginning of the project.

In the future, to expand this project and give even more evidence of the effects that the amount of entry data has on an algorithm we could analyze different algorithms besides the ones already studied in this paper or even perform tests on different computers since the computational power is one of the main factors of how fast a program runs and it is not being considered in this investigation.

Every section of the project is completed as requested. Although, the analytical test could be improved by adding an specification of all the instructions that are being included in the analysis and providing the exact amount of steps that are being assigned to that specific line of code to fulfill the purpose of teaching how the analysis is performed.

VII. ANNEXES

A. Figures

B. Tables

C. Link repository

- https://github.com/aleedca/PrimerProyectoAnalisis

REFERENCES

[1] S. Mohammed, M. Ahmed, and R. Seraj, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/8/1295/htm

[2] S. Ghosh and S. Kumar, "Comparative analysis of k-means and fuzzy cmeans algorithms," 2013. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.683.5131&rep=rep1&type=pdf#page=46

[3] D. Radečić, "Master machine learning: Decision trees from scratch with python," *Better Data Science*, 2021. [Online]. Available: https://betterdatascience.com/mml-decision-trees/

[4] AskPython, "K-means clustering from scratch in python [algorithm explained]," 2020. [Online]. Available: https://www.askpython.com/python/examples/k-means-clustering-from-scratch.

[5] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (svm) in libsvm," 2015. [Online]. Available: https://www.ijcaonline.org/research/volume128/number3/abdiansah-2015-ijca-906480.pdf

[6] Pandas, "Dataframe," 2022. [Online]. Available: https://pandas.pydata.org/docs/reference/frame.html

[7] Stackoverflow, "How to create a dataframe of random integers with pandas?" 2015. [Online]. Available: https://stackoverflow.com/questions/32752292/how-to-create-a-dataframe-of-random-integers-with-pandas