ANALYZING SUPER TITULO

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 we have 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. Because of this, we pretend to analyze each algorithm in phases which help our project to know if they are treatable or non-treatable, with an entry data from a range of 10 to 100000.**

I. Introduction

As the technology evolves and software and computers become more common in every area of the present world we have to consider a very important aspect: Efficiency. The world we live in 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 we have to focus on are time and memory.

In this project we will study 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 these questions: How does the amount of entry data affects the performance of the algorithms? and also, could these algorithms be treatable or non-treatable depending on the amount of entry data?

In order to answer these questions we will perform a variety of analytical and empirical measurements and then do an analysis of the results with the intention of determining the behavior of the mentioned algorithms.

plantea el problema en relación con los desafíos del mundo real, el objetivo de la investigación debe aparecer literal al planteado en el resumen, la propuesta para resolver el problema debe ser incluida junto a las preguntas de investigación que se desea abordar, se debe agregar una vista general de la evaluación para probar/validar la propuesta y por último un párrafo que describa la estructura del documento.

A. Objectives(????)

- Implement flow-net algorithms efficiently in Python programming language.
- Analyze the algorithms by empirical and analytical measurements.

II. Related Projects

es una revisión de literatura que permita aportar conocimiento para desarrollar el proyecto de software. Se debe hacer una búsqueda de artículos que aborden temáticas que aporten información para plantear una propuesta de solución al problema que se aborda.

To understand the concept of each algorithm we did an exhaustive research 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.

aca el de decision tree

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 [2].

III. Solution

Within each code of each algorithm there are calculations made by us 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 we are using the Entropy metric, using 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.

acordate de poner lo de numpy y pandaspara lo de random, no importa si se repite igual a lo que yo puse abajo

B. Unsupervised Algorithm: K-Means

The approach and structure of this algorithm coded in Python programming language, seen in [2], 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 [2]: 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 [4] [5], 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 and if some of the entry amount could even make one of these algorithms become non-treatable. 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. Here are the results of the analysis:

hacer tabla

TABLE III
FACTOR TABLE OF ALGORITHM K-MEANS

Line	Source Code	Measurement Executed Lines for Worst Case
18	idx = np.random.choice(x.shape[0], k, replace=False)	1
22	centroids = x.iloc[idx, :]	1
26	distances = cdist(x, centroids ,'euclidean')	1
30	points = np.array([np.argmin(i) for i in distances])	1
34	for _ in range(no_of_iterations):	n+1
35	centroids = []	n+1
38	for idx in range(k):	n(n+1)
40	temp_cent = x[points==idx].mean(axis=0)	n(n+1)
41	centroids.append(temp_cent)	n(n+1)
46	centroids = np.vstack(centroids)	n+1
49	distances = cdist(x, centroids ,'euclidean')	n+1
52	points = np.array([np.argmin(i) for i in distances])	n+1
60	return points	1
	Total steps	$3n^2+8n+10$

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 bla bla.. Here are the graphic representation of the results:

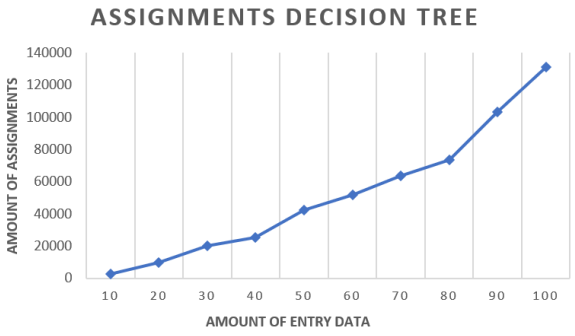

Fig. 1. Assignments Graph of Algorithm Decision Tree

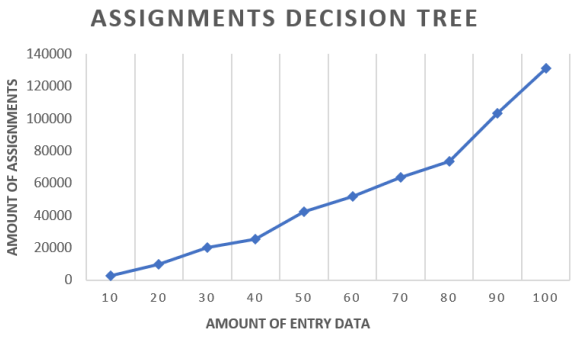

Fig. 2. Comparisons Graph of Algorithm Decision Tre

While K-Means algorithm shows, in Figure 3 and Figure 4, almost the same complexity time as the Decision Tree algorithm, affected by the two important factors mentioned, the tiny difference is that both figures are more straight and directed upwards. Here are the graphic representation of the results:

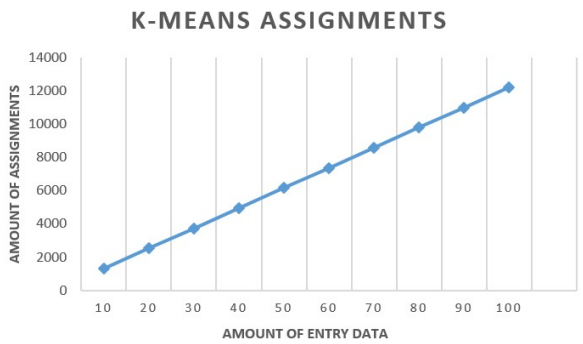

Fig. 3. Assignments Graph of Algorithm K-Means

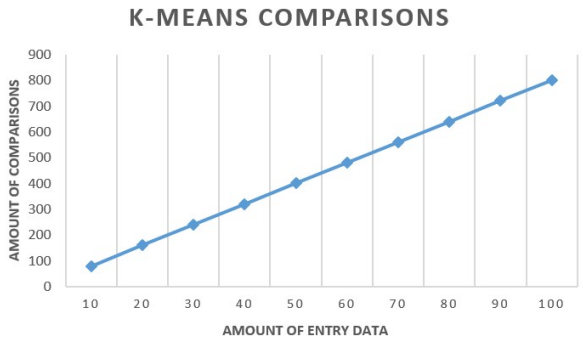

Fig. 4. Comparisons Graph of Algorithm K-Means

Doing this method allow us to understand and visualize the behavior of this algorithm and how the data is represented, also it helps us 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 mentioned, and it is represented as **O(n)** where *n* is the amount of entry data we tested.

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 IV
FACTOR TABLE OF ALGORITHM K-MEANS

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 IV 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. Therefore, K-Means algorithm has a Linear complexity **O(n)** in every aspect of the table.

faltan una tabla del final que no se que es pero es parte del factor de talla

V. EVALUATION

debe incluir la evaluación para probar/validar la propuesta planteada, analice los datos obtenidos, a que se debe que un algoritmo sea mejor que otro. Indique las características o ventajas de cada algoritmo sobre el otro.

podemos hacer una tabla de la clasificacion en cada metodo de cada algoritmo, como un resumen y explicamos lo que pide el profe

VI. CONCLUSIONS

indica los impactos de su propuesta y marcar el territorio de su trabajo futuro., resultados finales, indique que partes están completas, cuales defectuosos, y cuáles no se realizaron y el porqué, que aspectos se pueden mejorar.

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

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

[5] 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