Development and Implementation of a Python library for Rough Set Theory Data Analysis

Research Project undertaken by: Mukul C. Mahadik

B. Tech 3rd Year

Electronics and Communications Engineering,

National Institute of Technology, Hamirpur

Under the guidance of: Mrs. Vanita Agarwal

Assistant Professor

Electronics and Telecommunications

Engineering Department,

College of Engineering Pune

Internship Period: May 2019 –July 2019

Table of Contents

Sr. No.	Title	Page No.
1.	Introduction	3
2.	Aim and Objectives	3-4
3.	Algorithm Developed	4-8
A.	Type of Data Used	4
B.	Python Libraries Used	5
C.	Main Algorithm	6
i.	Parameter Calculation Algorithm	7
ii.	Time Complexity Algorithm	8
4.	Observations	8-12
A.	Parameter Calculation Plot	9
B.	Time Complexity Plot	10
5.	Conclusion	12
6.	Acknowledgement	13
7.	Bibliography	14

1. Introduction

This report essentially deals with the Rough Set Theory (RST) as well as an insight into the practical and real-time applications of RST. RST is a mathematical tool for imperfect data analysis or simply put, data that is inconsistent in many ways. *Approximations* are the fundamental concepts of RST. RST assumes that every object is associated with some information. Objects characterized by the same information are indiscernible (similar). The indiscernibility relation is the mathematical basis of RST. Any set of all indiscernible objects is called an *elementary set*. The set of indiscernible entries of the decision attribute only is called a *crisp set*. With any rough sets, a pair of sets called the lower and the upper approximation of the rough set, is associated. The *lower approximation* consists of all objects which *surely* belong to the set and the *upper approximation* contains all the objects which *possibly* belong to the set. The difference between the upper and lower approximation constitutes the *boundary region*.

The above-mentioned features are calculated for any input dataset by the developed Python code library, which are then used to obtain the Accuracy as well as the Stability Index (detailed definition of these two parameters is given in Section 3. C. 1.) The changes in the dataset are taken into account after an interval of 5.0 seconds each, which is continuously monitored and written to a log file. Manual analysis can then be performed using the output data from the log file which is then plotted to obtain a visual representation of the analysis over a specific time interval.

In addition to this, a time complexity analysis is done with two separate datasets – one with variable no. of conditional attributes and the other with variable no. of objects. The time taken (in seconds) to compute the requisite RST parameters is output to a log file. The obtained values are then plotted to obtain a visual representation of the analysis over a specific time interval.

GitHub Link for the code: https://github.com/MukuFlash03/coep-rough-set-theory

Google Drive Link for all research data:
https://drive.google.com/drive/folders/1Rlg13oJYzUfS7YKBzM6bUYjEx_hebY-f

2. Aim and Objectives

The aim of the research project was working towards the development and implementation of a python library module to compute certain Rough Set Theory parameters for data analysis of the given datasets.

The objectives of the research work are as follows:

i. To analyze the given datasets by Rough Set Theory analysis.
ii. To evaluate the two parameters – Accuracy and Stability Index for the given datasets.
iii. To plot these two parameters obtained in (ii.) following a real-time data analysis.
iv. To compute the Time complexities in two different categories of datasets –

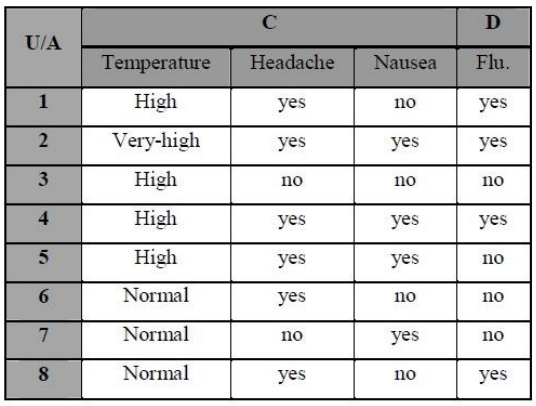
a. Variable U (No. of objects)
b. Variable A (No. of conditional attributes)

v. To plot the variations obtained in (iv.) and perform analysis by curve-fitting techniques.

3. Algorithm Developed

A. Type of Data Used

Rough Set based data analysis starts from a data table called a decision table, columns of which are labeled by attributes, rows – by objects of interest and entries of the table are attribute values. Attributes of the decision table are divided into two disjoint groups called condition and decision attributes, respectively. This system of objects and attributes is represented by (U, A) where:
U: A non-empty, finite universal set of objects.
A: A non-empty, finite set of attributes associated with each object.

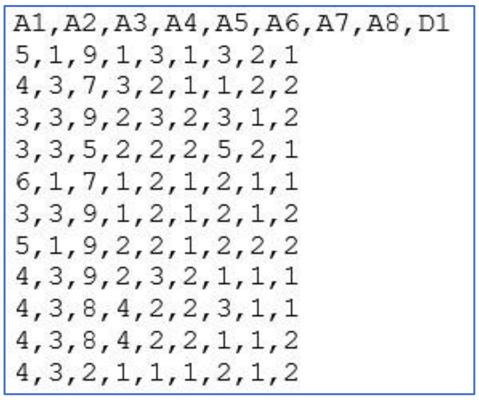


U/A	C			D
	Temperature	Headache	Nausea	Flu.
1	High	yes	no	yes
2	Very-high	yes	yes	yes
3	High	no	no	no
4	High	yes	yes	yes
5	High	yes	yes	no
6	Normal	yes	no	no
7	Normal	no	yes	no
8	Normal	yes	no	yes

Fig 1. Complete Decision Table with Objects and Attributes clearly indicated.

The adjacent Fig. 1 is one such example of a decision table where:
U represents the objects (here, labelled from 1 to 8);
A represents the two types of attributes – C for Conditional Attributes; D for Decision Attribute;
Both the types of attributes have been labelled as can be seen.

Now for the code to be able to understand the data present in the decision table, it needs to be input in a specific format in the form of a dataset. A dataset utilized by the code consists of various entries or values that are separated by commas in the same order and manner as the entries would appear in a tabular form in a decision table. The object labels have been omitted in the datasets while the attribute labels are mentioned at the top of the dataset. This entire dataset is then stored in a file format with the file extension ***.isf***.
This ***.isf*** is then stored as a spreadsheet document with the file extension ***.csv,*** that stands for "comma-separated values".

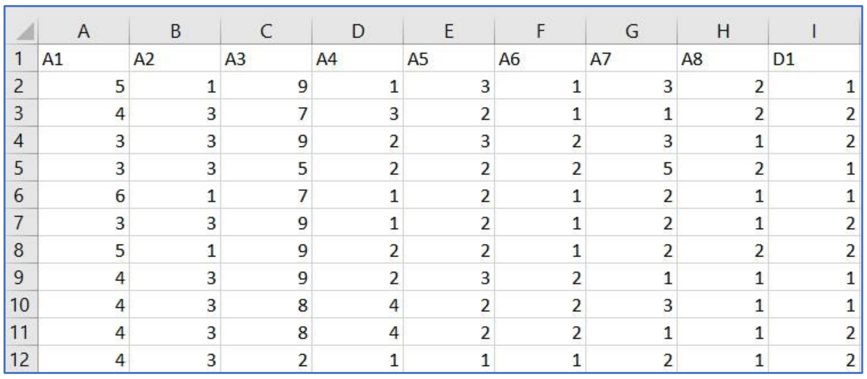


```
A1,A2,A3,A4,A5,A6,A7,A8,D1
5,1,9,1,3,1,3,2,1
4,3,7,3,2,1,1,2,2
3,3,9,2,3,2,3,1,2
3,3,5,2,2,2,5,2,1
6,1,7,1,2,1,2,1,1
3,3,9,1,2,1,2,1,2
5,1,9,2,2,1,2,2,2
4,3,9,2,3,2,1,1,1
4,3,8,4,2,2,3,1,1
4,3,8,4,2,2,1,1,2
4,3,2,1,1,1,2,1,2
```

Fig. 2 Dataset format when stored as a *.isf* file.

In Fig. 2 (left), the dataset is prepared by entering the data as elements separated by commas and mentioning the attributes labels as the first line of the dataset.

In Fig. 3 (below), the dataset stored as a ***.isf*** file is then imported into a ***.csv*** file to obtain a neat organized tabular dataset with great readability.



	A	B	C	D	E	F	G	H	I
1	A1	A2	A3	A4	A5	A6	A7	A8	D1
2	5	1	9	1	3	1	3	2	1
3	4	3	7	3	2	1	1	2	2
4	3	3	9	2	3	2	3	1	2
5	3	3	5	2	2	2	5	2	1
6	6	1	7	1	2	1	2	1	1
7	3	3	9	1	2	1	2	1	2
8	5	1	9	2	2	1	2	2	2
9	4	3	9	2	3	2	1	1	1
10	4	3	8	4	2	2	3	1	1
11	4	3	8	4	2	2	1	1	2
12	4	3	2	1	1	1	2	1	2

Fig. 3 Dataset format when stored as a *.csv* file.

Now that we are familiar with the type of data being utilized as input to the code, we can now learn about the working of the Python library code.

B. Python Libraries Used

The Python IDLE version utilized for developing the library is ***Python 3.7 64-bit***

The various Python libraries used in developing the Rough Set Theory Python Library along with their purpose in the program are mentioned below:

i. Pandas: It is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. The dataset input as a ***.csv*** file is stored a Python pandas dataframe.

ii. Itertools: The module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an "iterator algebra" making it possible to construct specialized tools succinctly and efficiently in pure Python. Here, I have made use of the combinatoric generator combinations() present in the library, to generate a tuple combination containing all the conditional attribute column labels.

iii. Matplotlib: It strives to produce publication quality 2D graphic for interactive graphing, scientific publishing, user interface development and web application servers targeting multiple user interfaces and hardcopy output formats.

iv. Datetime: The datetime module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic are supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. Here, I have made us of this module to acquire the current date and time stamp.

v. Time: Python has defined a module, “time” which allows us to handle various operations regarding time, its conversions and representations, which find its use in various applications in life. The beginning of time is started measuring from 1 January, 12:00 am, 1970 and this very time is termed as “epoch” in Python.

vi. Openpyxl: It is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files. It was born from lack of existing library to read/write natively from Python the Office Open XML format.

vii. FuncAni_2: This is a custom-made library that consists of all the functions that are used to calculate the various features of dataset with respect to Rough Set Theory (RST) such as obtaining the elementary set, lower approximation, upper approximation, boundary condition, accuracy, stability index. It is the main library that essentially performs all the tasks related to RST.

C. <u>Main Algorithm</u>

From Sec 2. pt. ii., the RST Parameters – Accuracy and Stability Index are to be evaluated while from Sec 2. pt. iv. Time Complexity Analysis is to be done for two different types of datasets.

The algorithms for each of the above-mentioned tasks are described in detail below:

1. **Parameter Calculation:**

This algorithm is utilized to complete the major task of analysis the dataset and finally computing all the required parameters with the help of the data analysis.
The main flow of the program is as follows:

a. Input the dataset to be analyzed by rough set approach.
b. Initiate an animation loop using animate() contained in the *matplotlib* library that repeats the loop body after an interval of 5.0 seconds each.
c. Read the file each time the loop refreshes so as to record any changes in the dataset and compute the parameters taking in consideration the updated dataset.
d. Obtain the elementary set (ES) and the crisp set (CS).
   The elementary set is for each conditional attribute is obtained by storing the index numbers (starting from 1 for the first object) of each unique entry in a column whenever it repeats.
   For instance, if an entry "high" occurs 3 times, a set is formed containing the index numbers of entries with value "high". Similarly, sets are calculated for each unique entry in a column. All these individually obtained sets are stored as the elementary set for that specific column or conditional attribute (CA).
   In the similar manner, the crisp set is obtained for the values of only the decision attribute (DA).
e. Next, for each of the unique values of the DA, the lower approximation (La) and the upper approximation (Ua) are calculated. The La and the Ua are calculated taking all CAs at a time.
   The La is then calculated by taking the ES (starting from the first CA) and adding only the elements common to both the ES and the CS to the La set.
   The Ua is then calculated by taking the ES (starting from the first CA) and checking whether the ES has at least one element common to the CS. If it does, the entire ES is added to the Ua set.
f. The boundary region is calculated as the difference of Ua and La i.e. $Ua - La$
g. Similarly, the outside region is calculated as the difference of U and Ua i.e. $U - Ua$
h. Now, the cardinal numbers of both the La and the Ua are calculated. Cardinal number refers to the total number of elements in a set.
   Let nLa be the cardinal number of the La while nUa be the cardinal number of the Ua.
i. The Accuracy of the current unique value of the DA is given by:
   Accuracy (DA value) = $\frac{nLa}{nUa}$
j. The Stability Index (SI) of the current unique value of the DA is given by:
   SI (DA value) = $\frac{(nUa+nLa+1)}{N+1} - 0.5$

where, N is the total number of objects.

k. Next, the calculated parameters – Accuracy, SI, as well as the current date and time stamp, nLa, nUa are written to a log file for manually plotting graphs for visual analysis.

2. Time Complexity

This algorithm consists of the same code as for the Parameter Calculation algorithm it simply outputs the time taken to complete all the tasks performed by the first algorithm after the file has been read.

a. The initial time is calculated using the *time* library and stored in time_start.
b. The final time is calculated using the *time* library and stored in time_end.
c. The difference of the two times is then output as:

$$time\_start - time\_end$$

This computation of the time taken is done for varying sizes of datasets as per the following two categories:

a. Variable U (No. of objects)
b. Variable A (No. of conditional attributes)

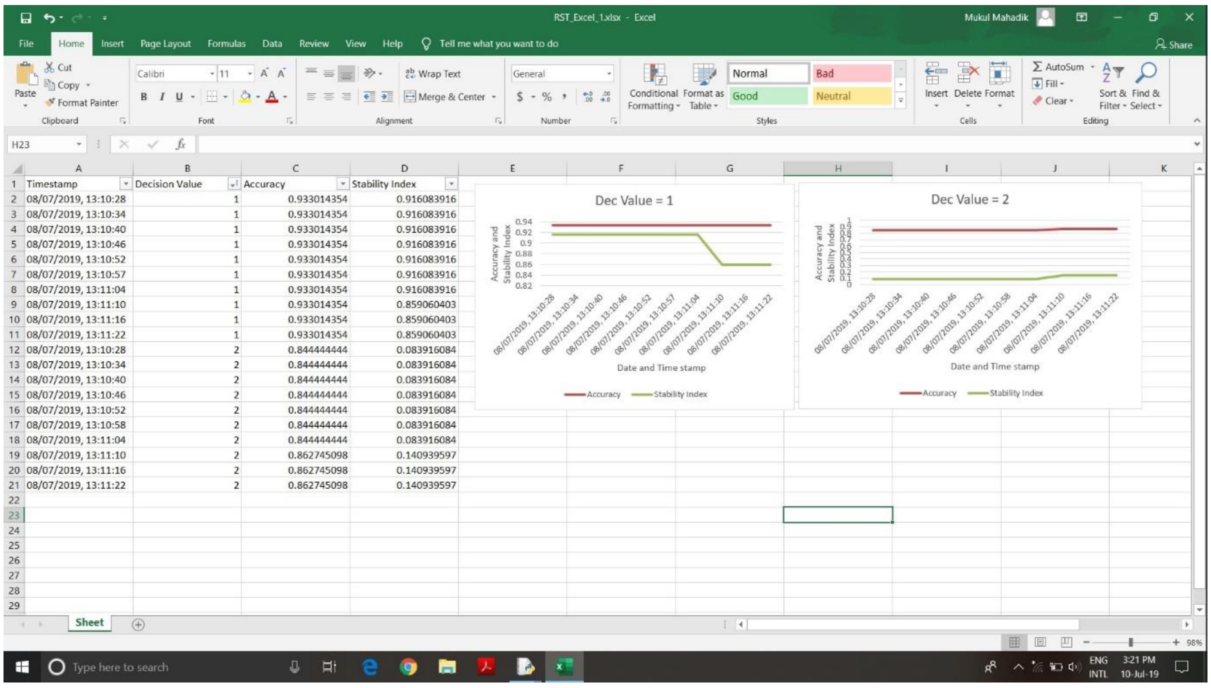
In category (a), the total no. of conditional attributes is kept fixed (say 4 CAs) while the no. of objects (U) is varied in powers of 2 as 32, 64, 128 and so on.

Similarly, in category (b), the total no. of objects is kept fixed (say 32 objects) while the no. of CAs is varied in powers of 2 as 1, 2, 4 and so on.

4. Observations

Now that we have successfully obtained the values of the RST parameters as well as the computation times, the data values are chosen and plots are generated between certain parameters for visual data analysis and representation.

i. Parameter Calculation Plot



Timestamp	Decision Value	Accuracy	Stability Index
08/07/2019, 13:10:28	1	0.933014354	0.916083916
08/07/2019, 13:10:34	1	0.933014354	0.916083916
08/07/2019, 13:10:40	1	0.933014354	0.916083916
08/07/2019, 13:10:46	1	0.933014354	0.916083916
08/07/2019, 13:10:52	1	0.933014354	0.916083916
08/07/2019, 13:10:57	1	0.933014354	0.916083916
08/07/2019, 13:11:04	1	0.933014354	0.916083916
08/07/2019, 13:11:10	1	0.933014354	0.859060403
08/07/2019, 13:11:16	1	0.933014354	0.859060403
08/07/2019, 13:11:22	1	0.933014354	0.859060403
08/07/2019, 13:10:28	2	0.844444444	0.083916084
08/07/2019, 13:10:34	2	0.844444444	0.083916084
08/07/2019, 13:10:40	2	0.844444444	0.083916084
08/07/2019, 13:10:46	2	0.844444444	0.083916084
08/07/2019, 13:10:52	2	0.844444444	0.083916084
08/07/2019, 13:10:58	2	0.844444444	0.083916084
08/07/2019, 13:11:04	2	0.844444444	0.083916084
08/07/2019, 13:11:10	2	0.862745098	0.140939597
08/07/2019, 13:11:16	2	0.862745098	0.140939597
08/07/2019, 13:11:22	2	0.862745098	0.140939597

Fig. 4 Shows the output log file with the various computed parameters – Date and Time stamp, Decision Value, nLa, nUa, Accuracy, Stability Index

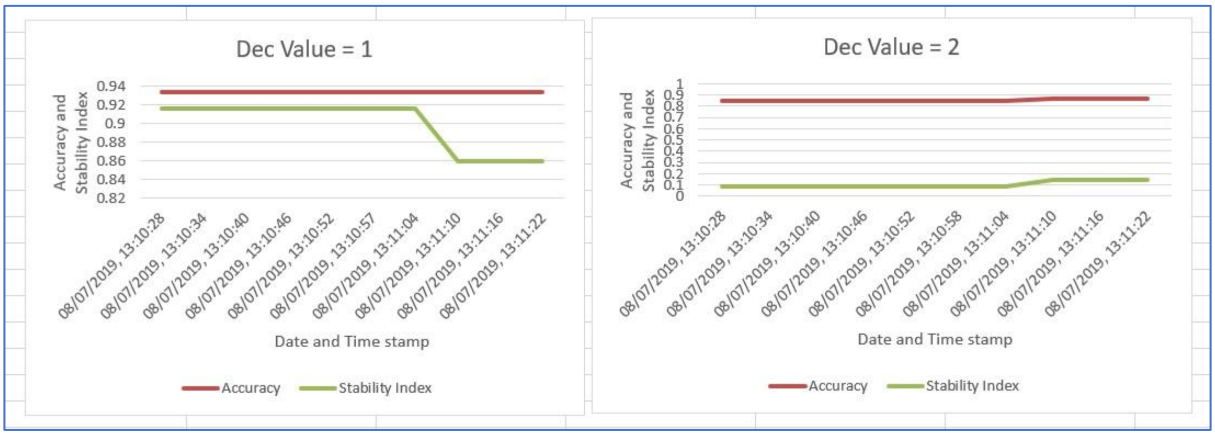


Fig. 5 Shows the plots between the RST parameters vs Datetime of record of the log file output in Fig. 4

From Fig. 5, we take a closer look at the plots obtained by plotting the values of Accuracy and Stability Index (Y-axis) against the Date and Time stamp values (X-axis). The two plots are one each for each of the unique DA values. Here we have two plots for two unique decision attribute values – ‘1’ and ‘2’.
The variation in the Accuracy values are indicated by the Red line while the Green line indicates the variation in the Stability Index. The changes in the parameters at different time intervals of 5.0 seconds each can be seen in the two plots.

ii. Time Complexity Plot

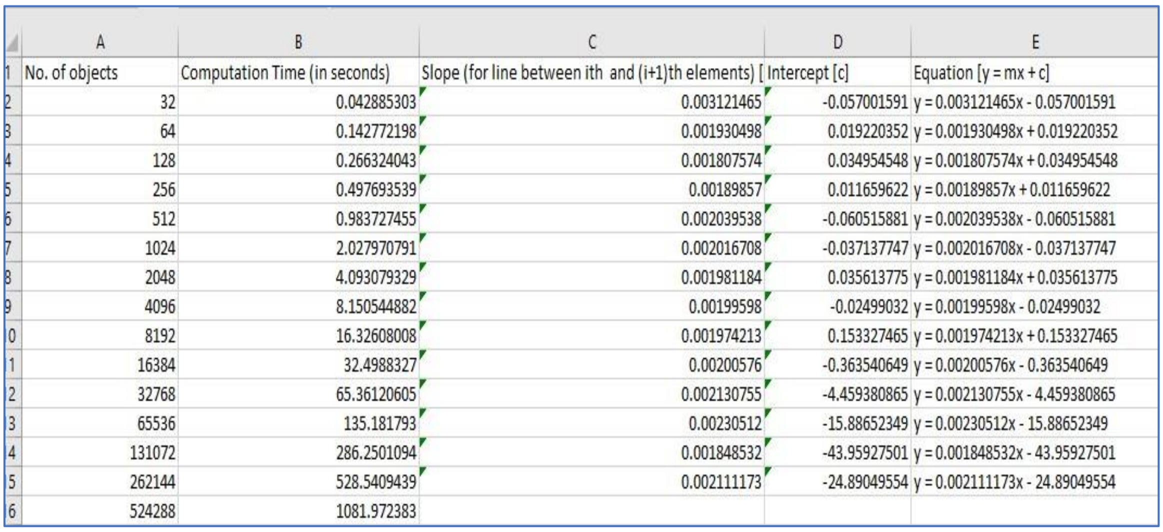
In general, the time taken to compute the RST parameters and run the program or code, there is said to be a square dependency on both the number of objects (U) and the number of attributes (A).

i.e. Time Complexity $\alpha\ U^2$ as well as Time Complexity $\alpha\ A^2$

But this is not the case when this general statement is tested in practical conditions. Instead, a piecewise linear curve plot is obtained in place of a parabolic curve expected by the square dependence of the time complexity on U and A.

Please note that the following observations have been made on a personal computer with an Intel Core i5 – 7200U CPU, 2.50 GHz Processor, having 8.00 GB RAM size with 2 Cores. The observations are bound to vary from system to system as well as taken on the same system but at different points of time.

a. Variable U, Fixed A



No. of objects	Computation Time (in seconds)	Slope (for line between ith and (i+1)th elements) [	Intercept [c]	Equation [y = mx + c]
32	0.042885303	0.003121465	-0.057001591	y = 0.003121465x - 0.057001591
64	0.142772198	0.001930498	0.019220352	y = 0.001930498x + 0.019220352
128	0.266324043	0.001807574	0.034954548	y = 0.001807574x + 0.034954548
256	0.497693539	0.00189857	0.011659622	y = 0.00189857x + 0.011659622
512	0.983727455	0.002039538	-0.060515881	y = 0.002039538x - 0.060515881
1024	2.027970791	0.002016708	-0.037137747	y = 0.002016708x - 0.037137747
2048	4.093079329	0.001981184	0.035613775	y = 0.001981184x + 0.035613775
4096	8.150544882	0.00199598	-0.02499032	y = 0.00199598x - 0.02499032
8192	16.32608008	0.001974213	0.153327465	y = 0.001974213x + 0.153327465
16384	32.4988327	0.00200576	-0.363540649	y = 0.00200576x - 0.363540649
32768	65.36120605	0.002130755	-4.459380865	y = 0.002130755x - 4.459380865
65536	135.181793	0.00230512	-15.88652349	y = 0.00230512x - 15.88652349
131072	286.2501094	0.001848532	-43.95927501	y = 0.001848532x - 43.95927501
262144	528.5409439	0.002111173	-24.89049554	y = 0.002111173x - 24.89049554
524288	1081.972383			

Fig. 6 This shows the computation time for datasets with variable number of objects

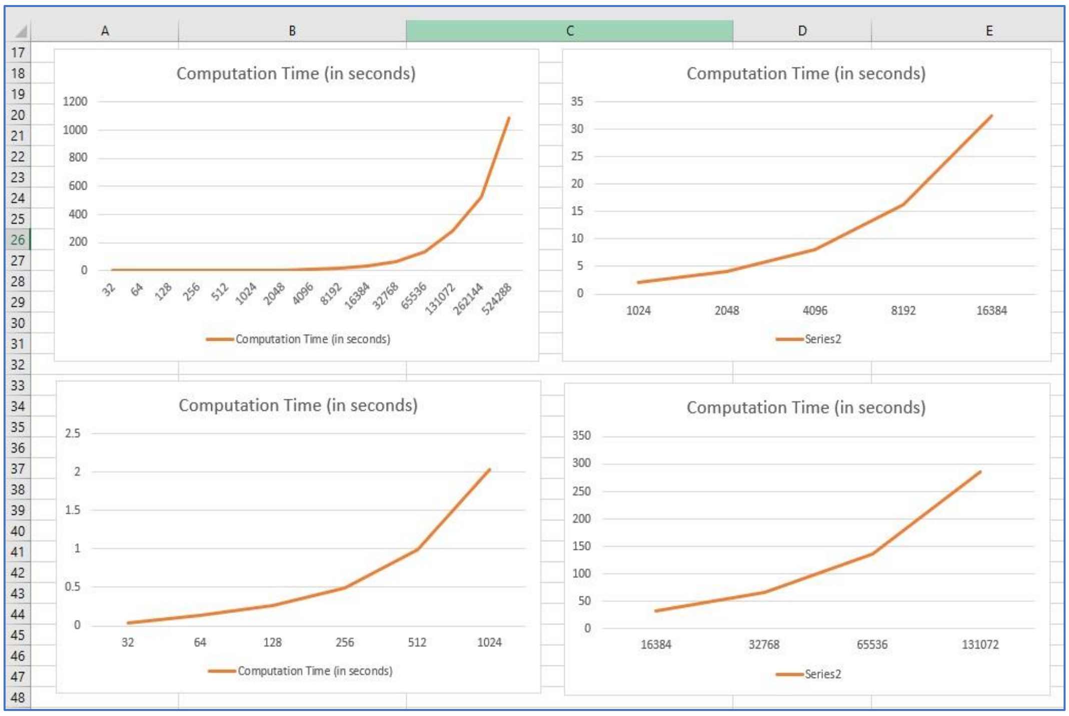
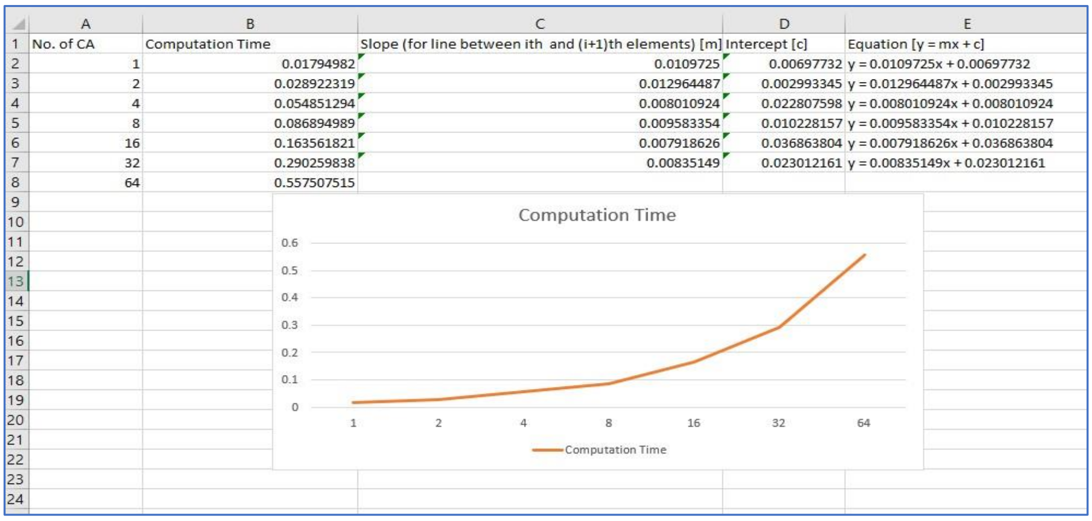


Fig. 7 This shows the plots obtained by plotting the computation time against the no. of objects

Examining Fig.6 and Fig.7, it can be observed that the variation in computation time (in seconds) (Y-axis) is plotted against the number of objects in the dataset (X- axis) for a fixed no. of CAs (here, 4 CAs). As expected, we can see from the first plot, that the graph isn't exactly parabolic but rather a piecewise linear curve. The other three plots are zoomed in versions of the first plot and confirm the linear nature of the plot.
The Column labelled "Equation" in Fig.6 represents the equation obtained by applying curve-fitting techniques to each piecewise linear part in the plot between each consecutive set of (x, y) values.

b. Variable A, Fixed U

Applying the similar analysis to the condition of varying the no. of attributes while keeping the no. of objects fixed, we obtain the plot shown in Fig. 8.
Here, the variation in computation time (in seconds) (Y-axis) is plotted against the number of conditional attributes in the dataset (X-axis) for a fixed no. of objects (here, 32 objects).
As expected, we can see from the first plot, that the graph isn't exactly parabolic but rather a piecewise linear curve.
The Column labelled "Equation" in Fig.6 represents the equation obtained by applying curve-fitting techniques to each piecewise linear part in the plot between each consecutive set of (x, y) values.



No. of CA	Computation Time	Slope (for line between ith and (i+1)th elements) [m]	Intercept [c]	Equation [y = mx + c]
1	0.01794982	0.0109725	0.00697732	y = 0.0109725x + 0.00697732
2	0.028922319	0.012964487	0.002993345	y = 0.012964487x + 0.002993345
4	0.054851294	0.008010924	0.022807598	y = 0.008010924x + 0.008010924
8	0.086894989	0.009583354	0.010228157	y = 0.009583354x + 0.010228157
16	0.163561821	0.007918626	0.036863804	y = 0.007918626x + 0.036863804
32	0.290259838	0.00835149	0.023012161	y = 0.00835149x + 0.023012161
64	0.557507515			

Fig. 10 This shows the computation time for datasets with variable number of conditional attributes. It also shows a plot of computation time (in seconds) against the number of CAs.

5. Conclusion

The algorithms mentioned in Section 3.C have been successfully implemented and their respective Python libraries have been developed on the Python IDE 3.7 version. The libraries have been tested with real-life datasets and their analysis being carried out, results in the observations elucidated in Section 4. Despite the limiting factor of the results having the surety of varying from system to system, the observations obtained are almost relatively precise with respect to the expected theoretical results before performing the data analysis. The Rough Set Theory approach for data analysis can now be applied to say a sensor data analysis system that measures the confidence level in the readings of each sensor as well as the probability of any specific event taking place which would again be dependent on the sensor readings. Thus, this makes the research work a successful endeavor. Nevertheless, it is always open to suggestions, modifications so as to facilitate future applications as well as updates when it comes to real-life scenarios which are constantly being updated. To be implemented as a real-time functioning library based on the Rough Set Theory approach for data analysis, the library is almost efficient but can surely be improved upon so as to be the best version of itself to be utilized in real-time applications.

6. Acknowledgments

The time I spent in the College of Engineering, Pune as an intern from May 2019 to July 2019 was truly a memorable one as it was rich in building experience over time and helped me discover my potential. I personally believe that the experience that I've gained as an intern by working under the guidance of my esteemed professors will forever shape and influence my professional life while fostering personal growth and development.

First of all, as always, I will always be grateful to my parents who have always been there by my side, to encourage me, to support me; being the constant source of motivation and guidance throughout my life. Thank you for being there.

Second, I would like to thank Prof. A.B. Patki, for his invaluable time and knowledge and sharing his wonderful experiences with him, which greatly motivated me to keep working towards the completion of this project.

Lastly, I would like to express my sincere thanks and gratitude for my guide, Asst. Prof. Mrs. Vanita Agarwal, for giving me this essential opportunity to work as an intern under her. Thank you, ma'am for being so patient throughout the period of my internship while also being available anytime for your constant guidance and support. I believe I have greatly benefitted from the great rapport that we shared and will definitely implement all that you have taught me. I will always be grateful to you for this opportunity.

To all those who have provided me with any sort of help that made this internship and the completion of the research work assigned to me possible, I extend a sincere thanks and gratitude for your contribution and support.

7. Bibliography

1. Missing Values Via Covering Rough Sets, T. Medhat, International Journal on Data Mining and Intelligent Information Technology Applications (IJMIA) Volume2, Number1, March 2012.

2. Rough set theory and its applications, Zdzisław Pawlak, Journal of Telecommunications and Information Technology, 3/2002.

3. Software Implementation Issues in RFD Analysis, Neha Verma, Niti Verma, Rishika Bansal, Bhagyashree Kulkarni, A. B. Patki, R C Meharde, 978-1-4244-7818-7/10/$26.00 © 2010 IEEE.

4. Design and Implementation of Rough Set Algorithms on FPGA: A Survey, Kanchan Shailendra Tiwari, Ashwin. G. Kothari.

5. Air Quality Monitoring with IoT Big Data - A Technical Guide for Data Processing and Analytics, July 2018, GSMA.

6. A Survey of Software Packages Used for Rough Set Analysis, Zain Abbas, Aqil Burney, accepted 18 July 2016; published 21 July 2016.

7. On the Stability of Coalitions: A Rough Set Approach, Jiri Mazurek, Elena Mielcova, Issue 3, Volume 6, 2012.

8. Python Libraries and Documentation – https://www.pypi.org/.

9. Dataframes in Python by Data Carpentry – https://www.datacarpentry.org/

10. RSL - The Rough Set Library version 2.0 by Michał Gawrys´, Jacek Sienkiewicz, ICS Research Report, Warsaw University of Technology, Warsaw, May 94.