2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)
Karnataka, India. Jun 28-29, 2024

Hybrid Model of Data Segregation for Enhanced OS Modularity

Biswajit Mishra
Amity Institute of Information Technology
Amity University Jharkhand
Ranchi, India
mishrabiswajit660@gmail.com

Roumo Kundu
Amity Institute of Information Technology
Amity University Jharkhand
Ranchi, India
rjroumo@gmail.com

Pooja Jha
Amity Institute of Information Technology
Amity University Jharkhand
Ranchi, India
pjha@rnc.amity.edu

Tannisha Kundu
Amity Institute of Information Technology
Amity University Jharkhand
Ranchi, India
tkundu@rnc.amity.edu

Mohan Kumar Dehury
Amity Institute of Information Technology
Amity University Jharkhand
Ranchi, India
mkdehury@rnc.amity.edu

***Abstract* - In the data-driven environment we live in today, where disorganization may inhibit productivity and workflow efficiency, effective digital file management is essential. This study and its suggested model present a novel mechanism concept a Python-based system created to address the widespread problem of disarrayed digital file systems. The "Group by" function of file managers available on many OSs currently does not address this problem because it groups files solely on the basis of one type of extension. Reducing the time & effort needed for manual organization, the suggested methodology automates the sorting & categorizing of files, including photos, documents, music, videos, archives, books, & code. Through its ability to optimize digital workflows & improve user experiences, this technology makes file management easier. To better understand the scalability, performance of the designed module, we conducted extensive performance testing on different devices & architectures . The results of this testing provided insightful information that can be applied to improve digital file organization in the real world. Additionally, the information revealed factors & relationships related to the organizing & sorting at the hardware level.**

Keywords - File Organization , Sorting , Grouping

I. Introduction

At this current era all the file managers available in the market sorts or groups the files according to their respective file types like java, c, cpp etc. This file grouping mechanism might not be sophisticated or appropriate in real life scenarios when a user wants to organize their files in the respective cluttered folder which has a mixture of different variants of file extensions. At that specific scenario the currently present file managing mechanism might lack some logical & efficient grouping structures. So, with an extension or an addon to this mechanism this proposed model takes this file grouping mechanism to a step further by organising the files according to their natures i.e., codes, videos, documents etc.

In the digital age, effective file management is crucial for personal & professional life. The exponential growth of digital content often leads to disorganized file systems, causing inefficiency for users. An effective solution is proposed to streamline file organization & enhance user productivity. This innovative software automates the sorting & categorization of files, providing a simple yet robust interface for decluttering directories. The motivation behind this research study is to offer a practical & efficient tool that simplifies the lives of individuals dealing with extensive digital content. By automating the process of categorising files into separate categories such as photographs, documents, movies, etc., the model seeks to reduce the laborious and prone to mistake manual sorting procedure. It not only optimizes file management but also contributes to the overall cleanliness & organization of computer systems. This research paper explores the suggested paradigm in detail, carefully analysing its applicability, functioning, & design. The study highlights the tool's great importance as a vital resource for digital file management & performs in-depth tests & observations to evaluate its functionality. These thorough assessments seek to identify important components & interdependencies ingrained in the model's structure & operation.

II. Litreature Review

A novel technique to content-based file type detection is presented by Irfan Ahmed et al. (2010), who combine a divide-and-conquer strategy with cosine similarity. The method efficiently accounts for byte order by calculating the cosine similarity between the content of a file & reference files. Subsequently, it employs a divide-and-conquer tactic, allocating clusters at first according to cosine similarity & splitting again & again until only one cluster is left, which gives the file type. The approach is computationally efficient for real-time application, & its superiority over conventional file-type recognition is demonstrated by evaluation on a dataset including over 1,000 files. Finally, the approach provides a new & reliable technique for identifying content-based file formats, with potential uses in data leak prevention, file forensics, malware detection, & system security enhancement. [1]

Using n-gram analysis, byte frequency distribution, & anomaly detection, Irfan Ahmed & Kyung-suk Lhee provide a unique method for efficiently detecting malware. In terms of detecting different kinds of malware without being aware of their signatures beforehand, it performs better than conventional intrusion detection systems (IDS) based on signatures. The authors provide an enhanced technique that makes use of byte frequency distribution & n-gram analysis to overcome difficulties in differentiating between code-type & data-type packets . It is appropriate for realtime detection of various malware kinds & signatures, & overall, it provides an efficient solution for malware detection, beyond the constraints of standard IDS. [2]

PAYL is a cutting-edge payload-based intrusion detector, automatically modeling network traffic application payloads using byte frequency distribution & Mahalanobis distance according to Ke Wang & Salvatore J. Stolfo. It surpasses current intrusion detection methods in both accuracy & efficiency. During training, PAYL calculates byte frequency distribution & standard deviation for application payloads to a specific host & port. In detection, it uses Mahalanobis distance to assess data similarity to the pre-computed profile, generating alerts if the distance exceeds a threshold. PAYL stands out for identifying new & evolving attacks, enhancing security for various networks & systems. The paper thoroughly reviews prior work, compares PAYL to other methods, & demonstrates its effectiveness with real-world datasets. In summary, this paper advances intrusion detection through the innovative PAYL payload-based anomaly detector. [3]

Wei- Jen Li et al. introduces an innovative n-gram analysis-based method for file classification, particularly suited for high-bandwidth network devices. It overcomes the vulnerabilities of traditional file identification relying on file headers or extensions, which are susceptible to cyberattacks. The approach examines a file's binary content to create a fileprint, summarizing n-gram distributions. File type is determined by comparing this fileprint to a recognized database. The study demonstrates its effectiveness by training on known malicious code & identifying it in email attachments using supervised Naive Bayes machine learning with n-gram distributions. This research advances file type recognition & holds promise for enhancing system & network security by detecting malicious files disguising as benign ones. It also paves the way for new file forensics & malware analysis tools. [4]

Traditional file recognition methods have limitations in identifying new file types & detecting harmful files posing as benign ones according to Mason McDaniel et al. The study proposes novel techniques for classifying computer files based on their content, rather than relying on predetermined file extensions or magic numbers. The recommended algorithms create file fingerprints by analyzing content, extracting features like byte frequency distributions & entropy. These fingerprints are compared to a database of known file types to accurately determine a file's true nature. Experimental results indicate that these algorithms are more reliable & accurate than conventional methods, capable of identifying novel file formats. They have potential applications in antivirus software & digital forensics. This research significantly advances file type recognition, providing a promising solution to the shortcomings of traditional methods. [5]

III. METHODOLOGY

Having effective file management skills is crucial in the current digital era. Files in a variety of forms, including papers, photos, video & etc , can overwhelm users. Various techniques for arranging & classifying files inside folders are used by contemporary file managers to address this issue. The various file managers employ various file organization techniques, which are examined in this section along with their unique characteristics.

File Organization in Contemporary File Managers

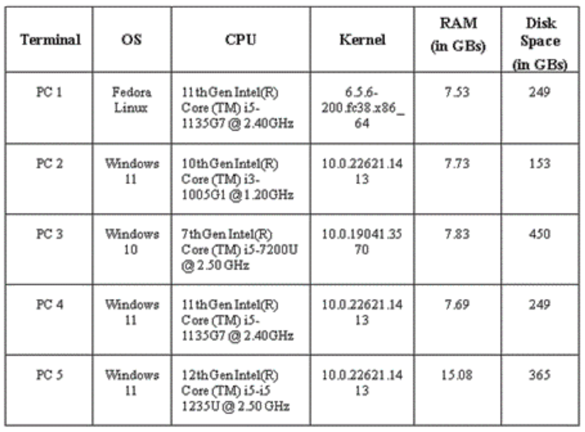
- **Manual File Organization** - Many users manually organize their files by creating folders & moving files into them. While this approach offers complete customization, it can be time consuming & prone to human error. This method involves a straightforward process of creating folders based on categories or projects & then manually dragging & dropping files into the corresponding folders.
- **Automatic Sorting by File Type** - Contemporary file managers, such as Windows Explorer & macOS Finder, incorporate automatic sorting mechanisms. These file managers classify files based on their file extensions, sorting them into predefined categories like "Images," "Documents," "Music," & "Videos." This simplifies file organization, making it easier for the users to locate specific types of files efficiently .
- **Tagging & Metadata** - Some advanced file managers, like macOS Finder, enable users to tag files with metadata, such as labels or keywords. This metadata can be used to classify files into user defined categories, enhancing the granularity of file organization. However, this approach relies on user's feed & may not be as intuitive for large volumes of files.

The Proposed Model

It is an innovative file management script designed to streamline the file organization process. The primary objective of this model is to automate file sorting while providing users with flexibility & customization options. It takes inspiration from the existing file managers while offering a unique approach to file organization & serves its task at a preferable O(n) time complexity. File classification based on file extensions is automated using this model. The application makes it easier for the user to identify & transfer files into the appropriate folders by using prepared lists of file extensions for various categories (such as photos, documents, audio, etc.). It also has a function that automatically recognizes & removes empty folders to help keep the file system organized . This feature makes sure that there is never anything messy in the file system. To conclude modern file managers use a variety of techniques to arrange files , ranging from mechanical sorting to computerized classification & sophisticated labelling programmes .

This proposed model introduces an efficient, automated, & highly customizable model for file organization. By automatically sorting files based on user-defined categories & allowing easy customization, it aims to revolutionize the file management experience. The inclusion of an empty folder deletion feature ensures the long-term maintenance of an

TABLE I. THE ATTRIBUTES & CONFIGURATION OF THE MACHINES TESTED



Terminal	OS	CPU	Kernel	RAM (in GBs)	Disk Space (in GBs)
PC 1	Fedora Linux	11thGen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz	6.5.6-200.fc38.x86_64	7.53	249
PC 2	Windows 11	10thGen Intel(R) Core (TM) i3-1005G1 @1.20GHz	10.0.22621.1413	7.73	153
PC 3	Windows 10	7thGen Intel(R) Core (TM) i5-7200U @2.50 GHz	10.0.19041.3570	7.83	450
PC 4	Windows 11	11thGen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz	10.0.22621.1413	7.69	249
PC 5	Windows 11	12thGen Intel(R) Core (TM) i5-i5 1235U @ 2.50 GHz	10.0.22621.1413	15.08	365

organized file system. The performance of the organizing of the files will totally depend on the files system architecture of the respective OS , in which the task is being performed on . That internal mechanism is not harnessed or tuned by the proposed mechanism; it is assumed that the architecture would handle the internals in an optimized manner. This proposed model has been tested on heterogenous machines running on various operating systems & hardware configurations. Testing cycles at an iteration of ten were conducted & a mean aggregate was inculcated to be used for analysis & inference, so the examined data doesn't directly proportionate to the fact that a same configuration machine would also undergo same time turbulence when tested again.

To ensure the efficiency & versatility of the model, a comprehensive testing phase was undertaken on diverse computing environments which is demonstrated at Table 1 . The model was executed & tested across multiple systems with varying hardware configurations & operating systems, reflecting the real world scenarios in which it would be

TABLE II. PROPERTIES & ATTRIBUTES OF SAMPLE FOLDERS FOR TEST 1

Samples	Attributes	Count	Collective Size	Extensions Involved				
sample 1	audio	455	5.7 gbs	mp3	flac	wav		
sample 2	codes	2450	120 mbs	rc	xml	py	bp	mkv
sample3	text file	5632	1.6 gbs	docx	pdf	ods	xlsx	pptx
sample 4	mix	162	17.9 gbs	png	pdf	jpg	docx	apk
				rpm	mp4	sh	zip	iso
				xlsx	dev	ico	mkv	pem
				rdp				
sample 5	images	287	1.1 gbs	jpg	png			
sample 6	motion	58	67.7 gbs	mkv	mp4			

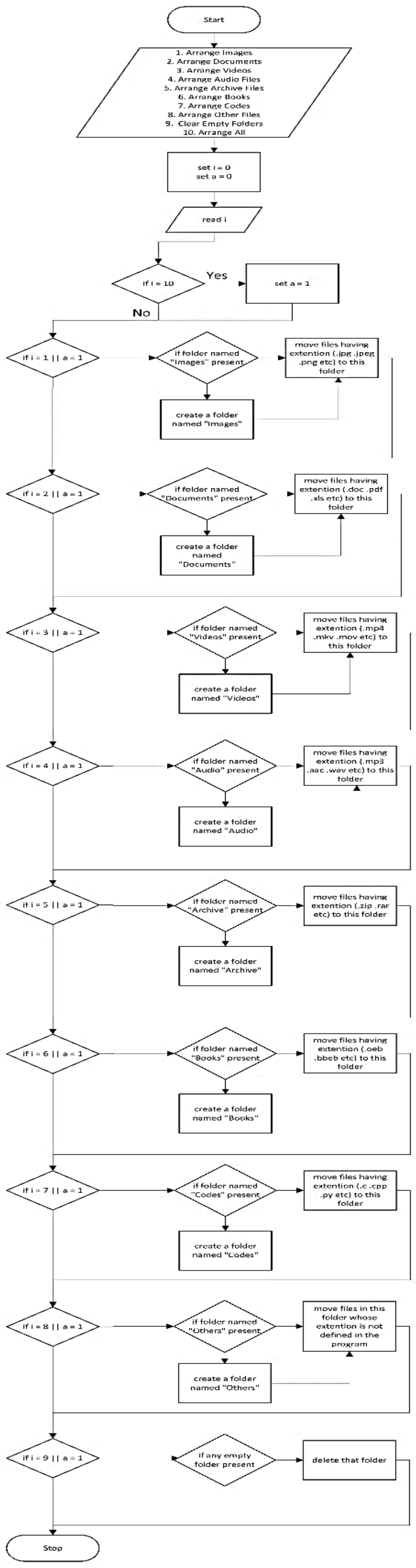

Start
1. Arrange Images
2. Arrange Documents
3. Arrange Videos
4. Arrange Audio Files
5. Arrange Archive Files
6. Arrange Books
7. Arrange Codes
8. Arrange Other Files
9. Clear Empty Folders
10. Arrange All
set i = 0
set a = 0
read i
if i = 10
Yes
set a = 1
No
if i = 1 || a = 1
if folder named "Images" present
move files having extention (.jpg .jpeg .png etc) to this folder
create a folder named "Images"
if i = 2 || a = 1
if folder named "Documents" present
move files having extention (.doc .pdf .xls etc) to this folder
create a folder named "Documents"
if i = 3 || a = 1
if folder named "Videos" present
move files having extention (.mp4 .mkv .mov etc) to this folder
create a folder named "Videos"
if i = 4 || a = 1
if folder named "Audio" present
move files having extention (.mp3 .aac .wav etc) to this folder
create a folder named "Audio"
if i = 5 || a = 1
if folder named "Archive" present
move files having extention (.zip .rar etc) to this folder
create a folder named "Archive"
if i = 6 || a = 1
if folder named "Books" present
move files having extention (.oeb .bbeb etc) to this folder
create a folder named "Books"
if i = 7 || a = 1
if folder named "Codes" present
move files having extention (.c .cpp .py etc) to this folder
create a folder named "Codes"
if i = 8 || a = 1
if folder named "Others" present
move files in this folder whose extention is not defined in the program
create a folder named "Others"
if i = 9 || a = 1
if any empty folder present
delete that folder
Stop
Fig. 1. Algorithm of the Proposed Model

TABLE III. PROPERTIES & ATTRIBUTES OF THE SAMPLED FOLDERS FOR TEST 2

Samples	File Count	Collective Size
mix1	120	300 mbs
mix2	29	466 mbs
mix3	57	680 mbs
mix4	58	933 mbs
mix5	60	1.4 gbs

deployed. The objective of this rigorous cross platform testing was to assess the program's performance, stability, & compatibility under different conditions. Systems included a wide & various range of hardware specifications, encompassing both older machines & more modern, high performance computers. The test environment featured a spectrum of operating systems, such as different editions of Windows & various Linux distributions.

The data samples tested in the methodology employed in this study were designed to provide a comprehensive & realistic representation of real-world file organization scenarios. To this end, a carefully crafted sample dataset was assembled, comprising multiple folders, each simulating distinct file categories demonstrated in Table 2 . To be more precise, files with the suitable extensions were placed in directories labelled with Audio , Video , Documents , Images , Archives , Books & Codes . The Audio folder incorporated an wide collection of audio file formats, while the Video folder encompassed a diverse range of video files. Similarly, the Documents folder housed various document types, & the Images folder contained image files. Archives, Books, & Codes folders were also included to represent distinct file categories & unique circumstances to the real-life activities which the user might end up doing. The model's algorithm is described in Figure 1 .

Furthermore, sample folders of mixed files were made by aggregating files from all categories into a single, unorganized collection which is demonstrated at Table 3 . This methodical approach ensured that the dataset captured the complexities & nuances of file organization, offering a robust foundation for the evaluation of the proposed model. This extensive testing approach was paramount in validating the program's robustness & its ability to provide seamless file organization solutions across heterogeneous computing ecosystems. The results & insights gained from this testing phase form a critical component of the evaluation, reflecting the adaptability & utility of the proposed model.

IV. RESULTS

The construction of the samples & hosting of the samples consumed nearly thirteen hours of effective work . The actual model would disintegrate the files into the respective folders but for the testing & analysis purpose the mechanism of the model was tweaked a bit , which is the samples were allocated at a universal position & the iterations of testing were done directly to the samples or in actuality the paths to it . After one testing cycle the files were again brought together in the sample form . With the cycling testing of ten rotations & then aggregating them to the average time the following results were obtained for both of the tests . First up are the results & inference of test 1 conducted.

TABLE IV. PERFORMANCE OF THE MODEL FOR TEST 1

Samples /terminals	PC 1	PC 2	PC 3	PC 4	PC 5
sample 1	0.0232	0.2212	0.3224	0.0894	0.1405
sample 2	0.0418	0.5785	1.2551	0.2153	0.2703
sample 3	0.1297	3.8198	2.0955	0.9641	1.3043
sample 4	0.0203	0.8029	5.1372	0.1486	0.2365
sample 5	0.0147	0.1278	0.2161	0.0538	0.0644
sample 6	0.0101	0.0331	0.3598	0.6672	0.0936

In Table 4 , demonstrates the performance in terms of speed that is the time consumed to organize the specific sample at an unit of seconds , whereas Figure 2 also demonstrates the same outcomes but in a standard graphical & visual manner which clearly outlines the all over performances of the architectures , in which the x-axis is defined for the samples & the y-axis is defined for the performance speed or the time consumed . The lines represent the different architectures i.e., the PCs.

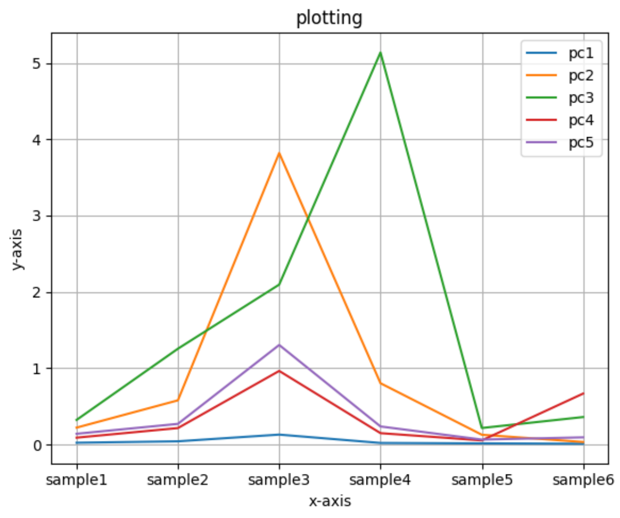


Fig. 2. Visualization Performance of the model for Test 1

With these visuals & results , it is determined that PC1 , PC2 , PC4 performed optimally well , with respect to their individual OS & architecture configuration where PC3 & PC5 consumed maximum of the time for the test. A relationship to the time consumption & total number of files were also inferred from these tests.

TABLE V. RELATIONSHIP WITH ARCHITECTURE & FILE'S ATTRIBUTES OF TEST 1

terminals	count	size
PC 1	0.9795943099778146	-0.3949313752375556
PC 2	0.9260731042314262	-0.33332117724563803
PC 3	0.0855387952587936	-0.10162519003789756
PC 4	0.6942440410907332	0.3354841530094178
PC 5	0.9470883551073287	-0.3134302361997474

In Table 5 , it is observed that the number of files that is to be tasked to the model does actually affect the time consumption & speed of the operation to be completed, for which its relationship metric of correlation was evaluated with which is described in equation 1 described by Asuero, A. G et al 2006 [25] ,

$$r = \frac{\sum (x_i - \bar{x})\ (y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \qquad (1)$$

Which as observed a positive correlation is observed for all of the architectures & the number of files involved in the samples. Whereas, an average of negative correlation is observed for the machines & the size of the files, so by this it is inferred that size of the files or samples for the model doesn't relate & affect that much & hence can be shadowed & neglected for over-view of the structure & operation. Next, test 2 was conducted focused on sample 4 with variate counts & sizes taken under consideration.

In Table 6 , demonstrates the performance in terms of speed that is the time consumed to organize the specific sample at an unit of seconds for the specific sample 4 with its nature of variation , whereas Figure 3 also demonstrates the same outcomes but in a standard graphical & visual manner which clearly outlines the all over performances of the architectures , in which the x-axis is defined for the mix's & the y-axis is defined for the performance speed or the time consumed . The lines represent the different architectures i.e., the PCs .

TABLE VI. PERFORMANCE OF THE MODEL FOR TEST 1

samples /terminals	pc 1	pc 2	pc 3	pc 4	pc 5
mix 1	0.0031	0.06595	0.1557	0.0634	0.0458
mix 2	0.0095	0.0587	0.0308	0.0148	0.0311
mix 3	0.0104	0.0977	0.3901	0.0228	0.0464
mix 4	0.0103	0.1082	0.3888	0.0055	0.0481
mix 5	0.0078	0.0959	0.06227	0.0231	0.058

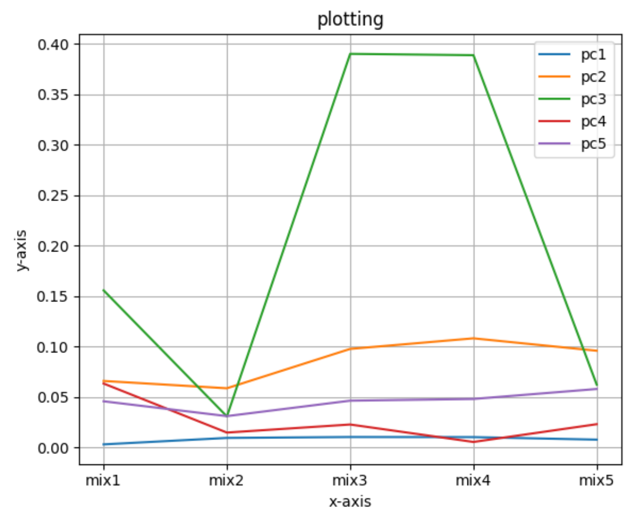


Fig. 3. Virtualization Performance of the Model for Test 2

With this visuals & results , it is determined that PC3 poorly in terms of time consumption as it is the only machine which has consumed most of the time which has resulted at an inefficient speed factor of the operation , with respect to their individual OS & architecture configuration , where the rest of the machines demonstrated optimally well performance ; out of which PC1 & PC4 performed exceptionally well . Along with these a relationship to the time consumption & total number of files were also inferred from these tests.

TABLE VII. RELATIONSHIP WITH ARCHITECTURE & FILE'S ATTRIBUTES FOR TEST 2

terminals	count	size
PC 1	-0.8790841502295974	0.34778649348872026
PC 2	-0.1450636266088838	0.6984163819207564
PC 3	0.06480676305222631	-0.017123456503946508
PC 4	0.893591004992368	-0.48685812287295127
PC 5	0.3480529605048086	0.7542839449497393

In Table 7 , it is observed that the number of files that is to be tasked to the model did not directly affect the time consumption & speed of the operation to be completed for both the parameters we get both negative & positive correlation, but a maximum of positive correlation is obtained but that cannot be used to infer a specific nature or property to this mechanism. Rather from this test it is inferred that for PC 5 we can see a positive correlation for both the attributes , as the PC 5 is strong in terms of configuration & build . So, all systems with configuration family of PC 5 should experience a smooth functionality in manner of task.

From all of these tests it is concluded that PC 1 is the common terminal system which has performed exceptionally followed by PC 5 & PC 3.

But in terms of the actual model performance evaluation of the system exhibited commendable performance during extensive testing. It efficiently categorized & sorted files across multiple formats, showcasing its versatility. The model's execution speed was found to be optimal, ensuring minimal processing time & resource utilization. It outperformed several existing file management tools in terms of speed & resource efficiency. As well as in point of user experience testing involved a diverse group of participants highlighted the tool's user-friendly interface. Participants reported that the program to be intuitive & easy to navigate. The incorporation of features like custom file path selection & directory creation received positive feedback for enhancing usability.

V. Limitation

The only feedback which was received to be modified was for the hidden files of the OS, i.e., log-based dump files (Dumstack.log.tmp) which was not able to be moved by the model so those extensions of the hidden files were hardcoded to be overlooked as well as won't be allowed for the user to access those files as it is secured by the OS itself.

For optimized screening, faster execution, & low size of the model, it is proposed at CLI level & not GUI based.

VI. Conclusion and Future Scope

The results of these tests & this study indicate that the proposed model is a capable & efficient tool for digital file organization. It addresses the persistent challenges of file cluttering & improves user productivity. The positive performance & user feedback underscore its potential as an asset in the realm of digital file management. Further refinements & widespread adoption are expected to enhance its utility in different computing environments.

The comprehensive assessment demonstrated that the model offers a robust & efficient means of categorizing & sorting files across diverse formats. It not only minimizes the time & effort required for manual organization but also enhances digital workspace efficiency. Real-world case studies & benchmarking against existing tools revealed the program's practical utility & competitive advantages. As a versatile tool that accommodates various file types, this stands as an asset in enhancing productivity & decluttering digital environments. Its usability, scalability, & ethical considerations make it a compelling choice for users seeking improved file management.

The proposed model is an extension to the already built feature of the file managers present in the OS's functionality, which doesn't state that the model is to override the already functioning file managers the only point is to integrate it with the file manager for better performance.

For scalability & platform dependency to the model, in terms of Linux based OS python3 comes integrated with the OS so a direct script would be self-sufficient to perform the task & for the windows OS .exe format can also be used to make the script run directly.

For enhance processing & faster execution the model can also be implemented C++ or Java.

The model can also be integrated in various cloud platforms to automate file structuring.

Refrences

[1] Ahmed, I., Lhee, K.S., Shin, H., & Hong, M. (2010). Content- based file-type identification using cosine similarity and a divide-and-conquer approach. *IETE Technical Review, 27(6), 465-477.*

[2] Ahmed, I. & Lhee, K. S. (2008, March). Detection of malcodes by packet classification. In 2008 *Third international conference on availability, Reliability and Security* (pp. 1028-1035). IEEE.

[3] Wang, K., & Stolfo, S. J. (2004, September). Anomalous payload-based network intrusion detection. *In International workshop on recent advances in intrusion detection* (pp. 203-222). Berlin, Heidelberg: Springer Berlin Heidelberg.

[4] Li, W. J., Wang, K., Stolfo, S. J., & Herzog, B. (2005, June). Fileprints: Identifying files types by n-gram analysis. In *Proceedings from the sixth Annual IEEE SMC Information Assurance Workshop (pp. 64-7-1).* IEEE.

[5] McDaniel, M., & Heydari, M. H. (2003, January). Content based file type-detection algorithms. In *36th Annual Hawaii International conference on system sciences, 2003. Proceedings of the* (pp. 10-pp). IEEE.

[6] Srinivasan, N., & Vaidehi, V. (2007, February). Reduction of false alarm rate in detecting network anomaly using mahalanobis distance and similarity measure. In *2007 International Conference on Signal Processing, Communications and Networking* (pp. 366-371). IEEE.

[7] Eller, R. (2003). Bypassing msb data filters for buffer overflow exploits on intel platforms. Online publication at http://community. core-sdi. com/~ juliano/bypass-msb. txt.

[8] Li, B., Wang, Q., & Luo, J. (2006, December). Forensic analysis of document fragment based on SVM. In 2006 International Conference on Intelligent Information Hiding and Multimedia (pp. 236-239). IEEE.

[9] Karresand, M., & Shahmehri, N. (2006, May). Oscar—file type identification of binary data in disk clusters and ram pages. In IFIP International Information Security Conference (pp. 413-424). Boston, MA: Springer US.

[10] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007, August). Bothunter: Detecting malware infection through ids-driven dialog correlation. In USENIX Security Symposium (Vol. 7, pp. 1-16).

[11] Liang, T. P., Moskowitz, H., & Yih, Y. (1992). Integrating Neural Networks and Semi-Markov Processes for Automated Knowledge Acquisition: An Application to Real-time Scheduling. Decision Sciences, 23(6), 1297-1314.

[12] Mena, J. (2003). Investigative data mining for security and criminal detection. Butterworth-Heinemann.

[13] Ahmed, I., Lhee, K.S., Shin, H., & Hong, M. (2010). Content- based file-type identification using cosine similarity and a divide-and-conquer approach. *IETE Technical Review, 27(6), 465-477.*

[14] Ahmed, I. & Lhee, K. S. (2008, March). Detection of malcodes by packet classification. In 2008 *Third international conference on availability, Reliability and Security* (pp. 1028-1035). IEEE.

[15] Wang, K., & Stolfo, S. J. (2004, September). Anomalous payload-based network intrusion detection. *In International workshop on recent advances in intrusion detection* (pp. 203-222). Berlin, Heidelberg: Springer Berlin Heidelberg.

[16] Li, W. J., Wang, K., Stolfo, S. J., & Herzog, B. (2005, June). Fileprints: Identifying files types by n-gram analysis. In *Proceedings from the sixth Annual IEEE SMC Information Assurance Workshop (pp. 64-7-1).* IEEE.

[17] McDaniel, M., & Heydari, M. H. (2003, January). Content based file type-detection algorithms. In *36th Annual Hawaii International conference on system sciences, 2003. Proceedings of the* (pp. 10-pp). IEEE.

[18] Srinivasan, N., & Vaidehi, V. (2007, February). Reduction of false alarm rate in detecting network anomaly using mahalanobis distance and similarity measure. In *2007 International Conference on Signal Processing, Communications and Networking* (pp. 366-371). IEEE.

[19] Eller, R. (2003). Bypassing msb data filters for buffer overflow exploits on intel platforms. Online publication at http://community. core-sdi. com/~ juliano/bypass-msb. txt.

[20] Li, B., Wang, Q., & Luo, J. (2006, December). Forensic analysis of document fragment based on SVM. In 2006 International Conference on Intelligent Information Hiding and Multimedia (pp. 236-239). IEEE.

[21] Karresand, M., & Shahmehri, N. (2006, May). Oscar—file type identification of binary data in disk clusters and ram pages. In IFIP International Information Security Conference (pp. 413-424). Boston, MA: Springer US.

[22] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007, August). Bothunter: Detecting malware infection through ids-driven dialog correlation. In USENIX Security Symposium (Vol. 7, pp. 1-16).

[23] Liang, T. P., Moskowitz, H., & Yih, Y. (1992). Integrating Neural Networks and Semi-Markov Processes for Automated Knowledge Acquisition: An Application to Real-time Scheduling. Decision Sciences, 23(6), 1297-1314.

[24] Mena, J. (2003). Investigative data mining for security and criminal detection. Butterworth-Heinemann.

[25] Asuero, A. G., Sayago, A., & González, A. G. (2006). The correlation coefficient: An overview. *Critical reviews in analytical chemistry*, *36*(1), 41-59.