International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Machine Learning Framework for Resource Utilization Analysis

Manni Megna Nookala

megna.nookala@gmail.com

Abstract:
Repository management platforms have become an essential component of modern software engineering by providing a centralized environment for storing, managing, and distributing software packages throughout the software development lifecycle. Among these platforms, **Sonatype Nexus** is widely adopted in DevOps and Continuous Integration/Continuous Deployment (CI/CD) environments because of its support for multiple artifact formats, including Maven packages, Docker images, software libraries, and application dependencies. Its centralized architecture streamlines artifact management, enhances collaboration among development teams, and strengthens governance of enterprise software assets. As organizations continuously generate and publish large volumes of software artifacts across multiple repositories, storage utilization increases significantly, creating substantial challenges in repository administration and infrastructure capacity management. Repository administrators must continuously monitor storage growth and perform maintenance activities to ensure sufficient storage availability. However, delayed maintenance or inadequate capacity planning can lead to repository outages, disrupting build pipelines, deployment workflows, and other mission-critical software engineering operations. Although Sonatype Nexus provides built-in cleanup mechanisms to automate repository maintenance, these utilities primarily remove obsolete or unused artifacts and do not predict future repository growth. Since frequently accessed and production-critical artifacts must remain available, storage management becomes increasingly complex as repository utilization expands. Consequently, administrators require reliable analytical techniques to forecast future storage requirements and support proactive infrastructure planning. To address this challenge, this paper presents a **machine learning-based analytical framework** using **Univariate Linear Regression Analysis** to predict repository storage utilization from historical usage data. The proposed model identifies storage growth patterns by deriving a regression equation that establishes the relationship between elapsed time and repository storage consumption. The resulting predictive model estimates future storage requirements with improved accuracy, enabling administrators to schedule infrastructure expansion, optimize repository maintenance activities, and allocate storage resources proactively. Experimental results demonstrate that the proposed approach closely approximates actual repository growth trends, reduces administrative effort, minimizes the risk of storage exhaustion, improves repository availability, and supports effective infrastructure capacity planning for enterprise-scale DevOps environments. By enabling proactive storage forecasting, the proposed framework helps organizations maintain uninterrupted software development and deployment operations while improving resource utilization and operational reliability.

Keywords: Linear Regression, Forecasting, Prediction, Analytics, Storage, Repository, Nexus, Capacity, Utilization, Modeling, Machine Learning, DevOps, Artifacts, Trend Analysis, Regression, Optimization,

Infrastructure, Automation, NXRM.

INTRODUCTION

Nexus Repository is a centralized platform used to store and manage software artifacts [1] , such as libraries, packages, and dependencies, in various formats. It acts as a universal repository manager that supports formats like Maven, npm, NuGet, Docker [2], and more, allowing teams to store, retrieve, and share these artifacts during the software development lifecycle. Nexus [3] stores build artifacts like JAR files, Docker images, etc., making them accessible to development and deployment pipelines. It can proxy external repositories (e.g., Maven Central) and cache dependencies locally, which reduces build times and network usage. Allows teams to store multiple versions of artifacts and manage them through consistent versioning policies. Nexus can act as a private Docker registry [4] for storing and managing container images, ensuring secure access to containerized applications. Nexus integrates seamlessly with continuous integration and continuous deployment tools like Jenkins [5][23] and Bamboo, automating the retrieval and publication of build artifacts. It offers fine-grained role-based access control (RBAC) and can scan repositories for vulnerabilities, ensuring compliance with security policies. Nexus is widely used in DevOps pipelines to manage dependencies [6] [24], store build outputs, and ensure a secure software supply chain. Each business unit will have one nexus instance and the users will start uploading artifacts to corresponding repository. There are snapshot and release repositories. All non prod environment artifacts will be created as snapshot artifacts and the prod artifacts will be created as non snapshot artifacts called release artifacts. ReactJS and Node JS artifacts will be stored in npm hosted repositories.

LITERATURE REVIEW

Sonatype Nexus Repository:

Nexus instances are connected to load balancer as shown in the Fig 1. There are different types of clients available to work with nexus repository manager (NXRM). Based on the dev team request NXRM admin [7] will create the hosted repositories for the respective type of deliveries. For Java development hosted repositories SNAPSHOT , RELEASE will be created, for nodes js npm repositories will be created. To upload the docker images docker registries will be created.

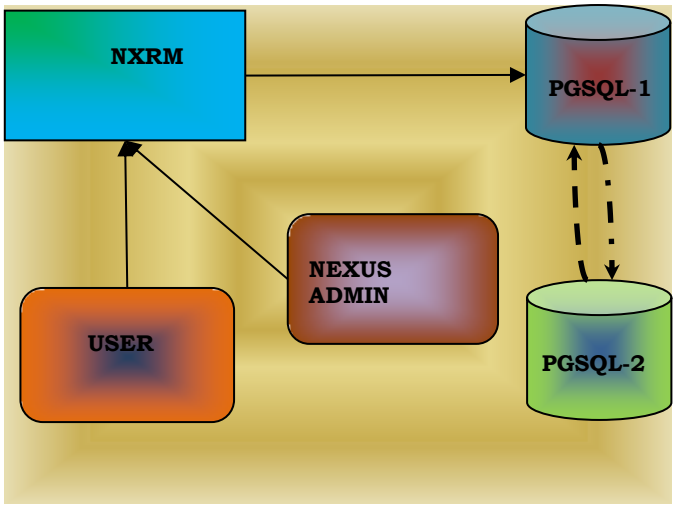


Fig: 1 Sonatype Nexus Repository Management Architecture diagram

The Fig: 1 shows the Sonatype Nexus Repository Management Architecture diagram. User is connecting

with NXRM tool to upload or down load the artifacts. The data is getting stored at postgres SQL [8]. The two instances of PGSQL is for ytaking the backup on predefined intervals. NXRM admin will provide access to respective users. If there are number of users for the same repository he will create Active Directory (AD) Groups and add the number of users to the same AD Groups so that he no need to add each user to the repository for access. The AD group [9] will be given with the corresponding permissions. Java development team will create artifact type as jar file. This will be uploaded to snapshot or release repository based on the version of the artifact. If it is fixed version it will be uploaded to release repository , if it is snapshot version it will be uploaded to snapshot repository. For web development npm packages will be created and these will be uploaded to npm hosted repository. Dot Net development packages will be uploaded to Nuget repository. Docker images will be uploaded to Docker registry.

Artifact Storage:

On daily basis artifacts from the different deveopers will be uploaded to corresponding repositories. The access has been given by the NXRM admin team to work with the uploads. We can not re upload if it is NON SNAPSHOT artifact. If it is SNAPSHOT version it will be re uploaded.

Snapshot Repository:

In the development environment the code needs to be corrected frequently , so we can not finalize the artifact in the initial cycles. For each delivery or the completion of cycle the artifact will be uploaded. So in this case we need to use SNAPSHOT (Ex: 1.0.0-SNAPSHOT) version [10] so that for each upload of the same artifact the time stamp will be appended to the artifact. For example if we are creating the artifact on the date 10/17/2020 21:03 then the artifact name is abc-1.0.0-171020202103. In this case abc is the name of the artifact. If we do the same build again after 5 minutes then the artifact name is 1.0.0-171020202603. I have assumed the same minutes for easy explanation. In this case for each build it will create new artifact name. So the rebuild is happening without any issue. Snapshot repository [11] allows rebuild operation. But this is technically wrong statement, why because we are not uploading same artifact each time. Every time we are uploading different artifact.

Release Repository:

Once all cycles are completed in dev environment i.e If we feel that the build is working fine and we are ready to create the artifact [12] , we can change the version of the artifact as fixed version. In this case it doesn't matter about the date of working. The name of the artifact will be abc-1.0.0. No time stamp will be added to artifact name [13][25]. In this case re upload is not allowed unless and until we need to enable the option to re upload the same artifact. Next release the artifact name will be abc-1.0.1. This will come from the snapshot artifact abc-1.0.1-SNAPSHOT [14].

Repository Proxying:

Different teams are having different repositories in the Nexus instance. Suppose team A wants to use the artifacts of team B then they can use the repositories inside Maven settings.xml [15][26] file so that these repositories will be scanned based on the dependencies mentioned at pom.xml file. We need to mention groupId , artifactId and version at pom.xml if we want to use any artifact. If the artifact is internal to organization then it will be downloaded from the repos mentioned at corresponding tags in settings.xml file. If the artifact is available at outside of the organization then it will throw the error if the corresponding

location is not mentioned at settings.xml file. We need to mention the external urls as proxies inside settings.xml file. Maven will look into local cache for artifact. If it is not available at local cache It will connect with urls internal to organization. If it is not available at internal , then it will connect [16][27] to external proxies to get the artifacts to local. As soon as it finds the artifacts either at local organization or external to organization it will be downloaded to local cache for future reference and it will be used in the subsequent builds.

Version Control:

Suppose the current artifact name is nexus while creating the development branch in the github the version is nexus-1.0.0-SNAPSHOT. It will be uploaded to nexus SNAPSHOT repository as nexus-1.0.0-171020241136. If we do next build then the artifact name is nexus-1.0.0-171020241138 (this buld is after 2 mins). Once we are done with all these cycles if we fix the version by removing the SNAPSHOT then the build version is nexus-1.0.0 and this will be uploaded to releases repository. Once we are done with this artifact if any one in the organization refers the GAV [17][28] of this artifact , this will be downloaded to their local from this nexus release repository. For the next cycle the new git hub branch will created with the version at development as nexus-1.0.1-SNAPSHOT. For eachj development build time stamped artifact will be created like nexus-1.0.1-171020201142 , nexus-1.0.0-171020201145 etc. The release artifact nexus-1.0.1 will be created at release build and it will be uploaded to nexus release repository. We can refer any of these versions using the GAV (GroupId, artifactId and version) parameters at pom.xml file.

Private Docker Registry:

Nexus can act as a private Docker registry for storing and managing container images, ensuring secure access to containerized applications. We need to create docker registry [18][29] at nexus instance as we have created like snapshot and release repositories, and login to this docker registry from the terminal and start pushing the docker images to same registry. We can pull the images back to our location based on our deployments.

Integration with CI/CD Pipelines:

Nexus integrates seamlessly with continuous integration and continuous deployment tools like Jenkins and Bamboo, automating the retrieval and publication of build artifacts. Once the build starts by getting the github code into local , the artifact will get created. This needs to be uploaded to Nexus repository using the Nexus plugin which is already integrated to Jenkins. This should be authenticated using nexus credentials. Once the development team commit [19][30] the code it will automatically trigger the job to run for build so that the code will be checked out to the base of the Jenkins (CI/CD tool). Once the build is successful , this process will create the artifact based on the type of the pom.xml file i.e it might be jar , war or zip .

This will be saved at the local cache of the machine where the build is happening. As part of the next phase of the build the artifact will be uploaded to Nexus repository. If the build created the snapshot artifact , then it will be uploaded to snapshot repository. If it is release artifact , it will be created to release repository. Even we can do the same process [20][31] for npm artifacts or docker images. As soon as the docker image got created it will get pushed to docker registry available at the nexus repository. Please find the architecture of the flow what we have discussed so far on the code commit till artifact uploaded to

NXRM repository at Fig 2.

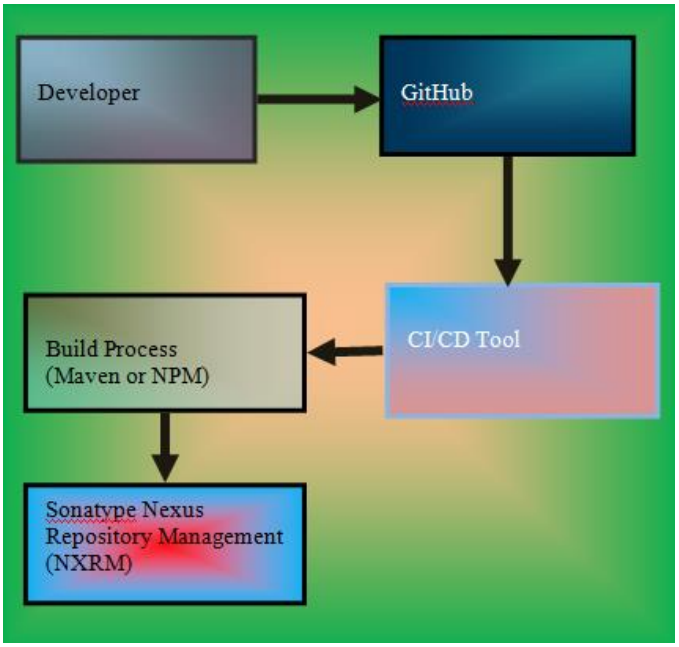


Fig 2: CI/CD Pipeline Integration

Security and Access Control:
We can have admin creds for configuring the NXRM instance. Once we are done with config operation we will be asked to update the creds by providing the password [21] in encrypted form. We can have our own creds. We can integrate Linear Data Access Protocol (LDAP) to the NXRM instance so that we can add AD Groups as well to the instance. For each project team one AD Group will be created and added to instance. User management becomes very easy by adding and removing the members to the ADGroup.

SNo	System IP	Time Stamp	Operation	URL	Size	Status
1	10.0.*	100924	PUT	url	150	success
2	10.1.*	110924	DEL	url	200	success
3	10.2.*	120924	GET	url	250	success
4	10.4.*	130924	PUT	url	300	success
5	10.3.*	140924	GET	url	350	success
6	10.6.*	150924	PUT	url	400	success
7	10.5.*	160924	DEL	url	450	success
8	10.4.*	160924	GET	url	500	success

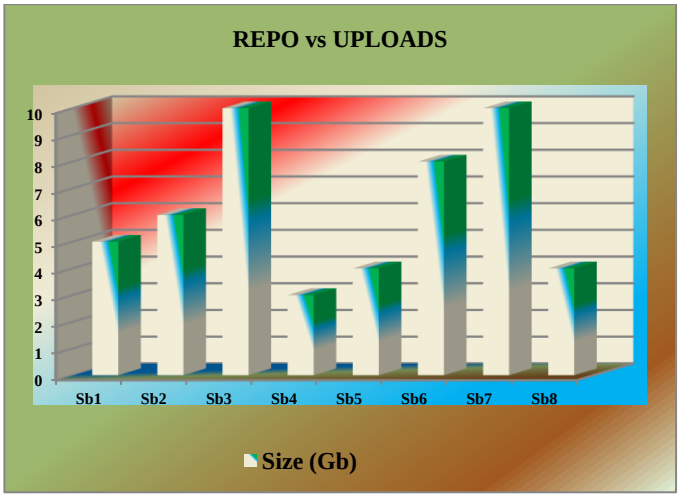
Table 1. Nexus Repository log file entries.

For number of repositories Table 1 shows the different type of metrics like name of the repository , time stamp , sub group of the repository, type of the operation GET, PUT or DELETE , volume of the upload, success or failure of the upload. In this we need to segregate the metrics using the type of the operation. Among these operations only PUT operation [22] will directly influence the space of the disk. We need to

figure out the total volume of the uploads happening on the daily , weekly and monthly analysis . For example we have taken sample values only having PUT operation for each sub group from the repository. Please find the same in Table 2. The values which we have mentioned in the Table 1 might not be related to values from Table 2.

subgroup	URL	Time Stamp	Size (Gb)	Operation	Status
Sb1	url1	100924	5	PUT	success
Sb2	url2	120924	6	PUT	success
Sb3	url3	130924	10	PUT	success
Sb4	url4	140924	3	PUT	success
Sb5	url5	150924	4	PUT	success
Sb6	url6	160924	8	PUT	success
Sb7	url7	170924	10	PUT	success
Sb8	url8	180924	4	PUT	success

Table 2: Business unit vs Upload operations



Graph 1: Business unit vs Upload operations

Please observe the usage on daily, weekly , monthly basis. We need to manually delete the unwanted artifacts like snapshots and very old fixed version volumes. If we didn't delete the same , then the upcoming uploads will not be successful which will effect the huge number of developers. Either admin has to delete them manually or by using admin task available at NXRM or he needs to connect with development team to pick the unwanted artifacts and remove the same. The amount of space gaining in the deletion of unwanted artifacts should be in sync with the amount of volume uploaded to NXRM repositories. Since the entire NXRM instance is shared by number of business units and there will be very high volume consumers or low volume consumers. Either we need to provide separate instance for high volume consumers or have them to provide unwanted artifacts list in high frequency so that admin can delete the same to gain the space for further activities. Failing of any of these actions will lead to abnormal failure of uploads by other business units though they are consuming very less amount of space.

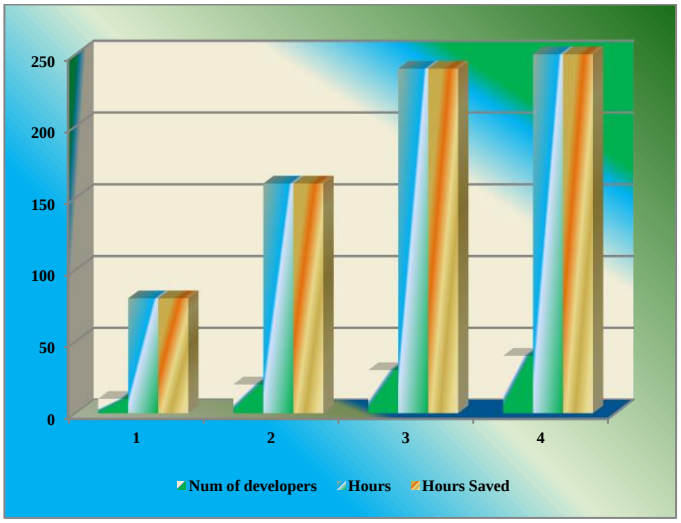
Even though we have huge volume available in the beginning of the NXRM activities , one day we need to provide the solution for any of these issues. Instead of working on this process where the manual

intervention is very much demanding, if we can predict the usage based on the daily , weekly and monthly usage analysis , it would be easy forecast the expected space required for further activities. If we know well in advance that the available space sufficient only for certain period of time , then it is easy for us to take the appropriate action like alerting the developers to provide unwanted artifacts or increasing the disk space.

If we follow this procedure this will avoid the abnormal incidents like NXRM is not available for operations. This will save the huge number of man hours from development team. If there are 20 members in one team and 10 to 15 teams are depending on the same NXRM then 2400 man hours we will save on daily basis. If there are any abnormal activities because of not having this prediction analysis then we need to face the wastage of man hours. Please find the table shows the man hours analysis.

SNo	Num of developers	Hours	Hours Saved
1	10	80	80
2	20	160	160
3	30	240	240
4	40	320	320

Table 3: Number of developers vs Man hours



Graph 2: Number of developers vs Man hours

Graph 2 shows that number of hours saved by the developers once we get any solution to avoid abnormal incidents. What ever the man hours available in each team , 100% savings we can observe.

PROPOSAL METHOD

Problem Statement

Usage of space using number of uploads will lead to abnormal incidents of the NXRM instance. Instead of waiting till the end of space and facing the abnormal incident we need to predict the space usage on daily , weekly or monthly basis will help us to patch the disk space so that we can save the number of man hours.

Proposal

Machine Learning algorithm will be used to predict the usage based on the historical analysis of the space usage. We need to consider past one year log files to analyze and figure out the dependent attributes and independent attributes (Feature Engineering). NXRM log files carries repository info in the form of url , time stamp , type of operation , status of the operation, size of the upload, time required to complete the upload operation. We need to analyze the data python data analytics tools like Pandas and numpy to get the exact repository name from the url which the dev team has used to upload the artifacts to nexus repository manager. Finally we will get the repo name , time stamp , artifact size , type of operation and time required to perform the corresponding activity.

In the second time analysis we need to filter the data using type of operation. With this we will get data for PUT operation (upload operation). Like this we need to get the data for the entire month for each repository, Ex: If there are repositories like TCT , RFR , CFC , PRD , GST, TGB, and SCY. Number of users are available in each business unit (repository name for each business unit), and each user is uploading artifacts to NXRM repository. In the analysis we need to filter the data on repo wise , followed by analysis on monthly basis for each business unit. Please find the data at Table 4 having detailed analysis on what we have described so far.

SNo	URL	Name	Activity	Size(Gb)	Status
1	http://ui	UI	PUT	4	Success
2	http://back	Back	GET	NA	Success
3	http://middle	Middle	DEL	NA	Success
4	http://ui	UI	GET	NA	Success
5	http://back	Back	GET	NA	Success
6	http://ui	UI	GET	NA	Success
7	http://middle	Middle	PUT	6	Success

Table 4: Nexus log file data attributes and values

Ui , middle , back like this they have different type of repo names created inside TCT business unit. The main problem is TCT business unit is facing the space issue. We have started the analysis on getting the repo name and type of operation along with time stamp , size of the artifact for uploading to repository. Monthly wise we will get total volume has been uploaded to each repository. This is how we will get the total space consumption by one business unit (including all repos ui, middle , back end).

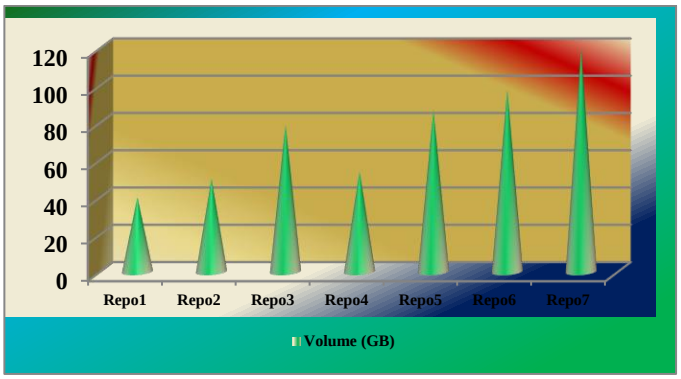
IMPLEMENTATION

We have used python numpy , pandas , Scikit learn to analyze the data . The data has been taken based on daily basis , weekly basis and monthly basis usage analysis. Using this we can get the approx space required to run the NXRM instance without facing the abnormal incidents.

Once we have historical usage for the single business unit on the monthly basis , we can use the same parameters in the machine learning algorithm to predict the dependent variable for future activities. Like this we need to do for all business unit and apply the same feature engineering process and figure out the parameters to apply inside machine learning algorithm (Linear Regression Analysis). This will provide prediction on dependent variable . In our case time is the independent variable and the size of the artifact (total size consumed by all artifacts business unit wise in a month) is the dependent variables.

SNo	Repo Name	Volume (GB)
1	Repo1	40
2	Repo2	50
3	Repo3	79
4	Repo4	54
5	Repo5	87
6	Repo6	98
7	Repo7	120

Table 5: Repo vs Volume consumption

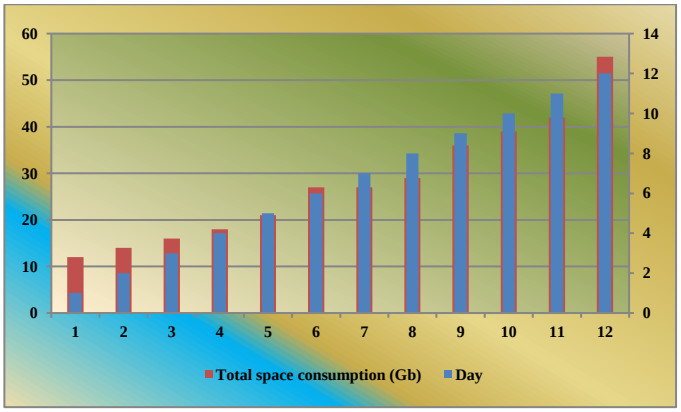


Graph 3: **Repo vs Volume consumption**

Graph3 shows the volume consumption for each repo (Repo1→ R1, Repo2→ R2, Repo3→ R3, Repo4→ R4, Repo5→ R5, Repo6→ R6 and Repo7→R7). Please find the usage data for all repos in one business unit per one month. We need to take the consolidate amount space consumed by one business unit in one month. As part of the analysis we have taken daily basis usage for 12 days, so that we will predict the volume required for next x number of days. Please find the table as follows for 12 days in October month

SNo	Day	Total space consumption (Gb)
1	1	12
2	2	14
3	3	16
4	4	18
5	5	21
6	6	27
7	7	27
8	8	29
9	9	36
10	10	39
11	11	42
12	12	55

Table 6: Space consumption on daily basis



Graph 4: **Day vs Volume consumption**

Calculating the Slope (m) and Intercept (b):
To compute the values of m and b, we use **the** least squares method [18], which minimizes the sum of the squared differences between the actual and predicted values. The formulas for the slope (m) and intercept (b) are

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

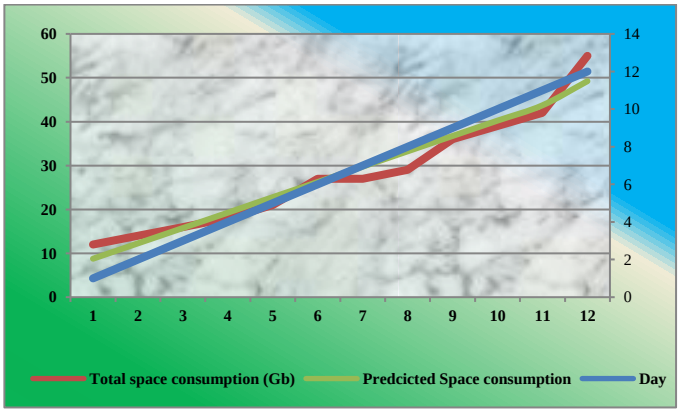
n is the number of data points (days in this case), ∑xy is the sum of the product of each x and y, ∑x is the sum of all x values, ∑y is the sum of all y values, ∑x2 is the sum of the squares of all x values. In this case, based on the provided space consumption data over the months (with numerical encoding for months), we calculated the m and b 3.50 and 5.27. Final linear regression equation [19] for space consumption is y=3.50x+5.27. If x=1 the predicted y is 8.77Gb and if x=2 y value is 12.7 Gb. This allows you to predict future space consumption based on this historical trend.

Fig 3: shows the actual and predicted values for the daily usage of volume based on the regression equation derived in the previous step.

SNo	Day	Total space consumption (Gb)	Predcicted Space consumption
1	1	12	8.769
2	2	14	12.265
3	3	16	15.762
4	4	18	19.258
5	5	21	22.795
6	6	27	26.251
7	7	27	29.748

8	8	29	33.244
9	9	36	36.741
10	10	39	40.237
11	11	42	43.734
12	12	55	49.23

Table 7: Actual vs Predected values



Graph 5: Actual vs Predected values

As per the values from the Table 7 , the predicted values are near by actual values. The difference between actual and predicted values are called residual points. Since the difference actual and predicted values is not having much with these values.

CONCLUSION

NXRM instance has been facing issues with usage of the space . To avoid the abnormal incidents we have prepared the linear regression analysis so that we can predict the space required for the future days. Once we have the prediction analysis we can take the necessary precautions like connecting with dev team to get the list of unwanted artifacts so that we can remove the list to gain the space. If it is not satisfies with the predicted values , then extra space request need to be created so that there will not be any abnormal incidents on NXRM. If we doesn't have this type of analysis , we will not be known until the system goes down. Having this type of analysis saves lot of man hours. We have number of options like connecting with dev team to get the unwanted artifacts to remove and gaining the space or running the cron tasks from the admin tasks. But all these things workout until there is no addition of business. On day to day, business will get increased or the developers are not ready to give the artifacts proportional to uploading the artifacts to NXRM. As long as there is proportionality between upload artifacts and delete artifacts definitely we need to add extra space to disk , else one day we need to face the system hang issue , where all the developers start connecting admin for the help or quick resolution which will not be possible. To avoid all these scenarios the solution which we have provided will help us a lot. Future work includes finding out the solution for two independent variables. In this paper we have taken time is the independent variable and volume is the dependent variable. So this is called univariate linear regression analysis. If we have one more independent variable along with time i.e, number of users in this case as well we need to findout the solution to avoid the abnormal incidents.

REFERENCES:

1. Bondugula, V. K. Designing Effective Lock-Based Concurrency Control in Database Systems. International Journal of Intelligent Systems and Applications in Engineering, Volume 9, Issue 1, 2021.
2. Kanagalakshmi Murugan. Elevating Data Throughput in Distributed Key-Value Systems with Data Distribution. International Journal of Intelligent Systems and Applications in Engineering, Volume 9, Issue 1, ISSN: 2147-6799, 2021.
3. Kanagalakshmi Murugan. Dynamic Multi-Objective Resource Optimization in Big Data Clusters. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, Volume 9, Issue 4, July–August, E-ISSN: 2349-7300, https://doi.org/10.37082/IJIRMPS.v9.i4.232618, 2021.
4. Bondugula, V. K. Managing Transactions in Snapshot Isolation with Adaptive Timeouts. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, Volume 9, Issue 4, https://doi.org/10.37082/IJIRMPS.v9.i4.232466, 2021.
5. Kalesha Khan Pattan. Optimizing Fault-Tolerance in Distributed Systems with AI-Augmented Replica Management. International Journal of Intelligent Systems and Applications in Engineering, ISSN: 2147-6799, 2021.
6. Arunkumar Sambandam. Multimodal Observability for Input Output Bottleneck Detection. Computer Fraud and Security, Volume 2021, Issue 8, 2021.
7. Naveen Kumar Bandaru. Transaction Batching for Low Latency Commit Processing in Distributed Systems. International Journal on Science and Technology, Volume 12, Issue 3, July–September, E-ISSN: 2229-7677, 2021.
8. SaiKrishna Mylavarapu, Optimizing Scalability and Efficiency in Clustered Computing Environments, International Journal on Science and Technology, Volume 12, Issue 3, July-September 2021, E-ISSN: 2229-7677, https://doi.org/10.71097/IJSAT.v12.i3.11229
9. Vijaya Krishna Namala, Reducing Runtime Overhead in Distributed Congestion Monitoring Systems, International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, Volume 9 Issue 4, 2021, ISSN: 2349-7300, https://doi.org/10.37082/IJIRMPS.v9.i4.232961
10. PurnachandraRao, NagamalleswaraRao ,HDFS I/O Operations performance optimization ,Journal of AdvancedResearch in Dynamical and Control Systems (JARDCS), vol-11, 01-SpecialIssue,pp.824-836,2019.
11. PurnachandraRao, NagamalleswaraRao, HDFS Pipeline Reconstruction To Avoid The DataLoss, International Journal of Simulation Systems , Science & Technology(IJSSST), vol-19,Number-16,pp.5.1-5.8, 2018.
12. PurnachandraRao, NagamalleswaraRao , HDFS Write Operation Using Fully Connected Digraph DataNode NetworkTopology, International Journal of Applied Engineering Research (IJAER),vol-12,issue-16,pp.6076-6090,2017.
13. PurnachandraRao, NagamalleswaraRao , HDFS Cache Performance using Set Associative Cache Memory, Journal ofTheoreticalandAppliedInformationTechnology(JATIT),vol-95,issue-16,pp.4034-4048,2017.

14. PurnachandraRao, NagamalleswaraRao, HDFS Logfile Analysis Using ElasticSearch, LogStash and Kibana , Integrated Intelligent Computing, Communication and Security, Springer, DOI: https://doi.org/10.1007/978-981-10-8797-4_20, ISBN978-981-10-8797-4, 2017.
15. PurnachandraRao, NagamalleswaraRao, HDFS Memory Usage Analysis, Proceedings of the International conference on Inventive computing and Informatics, IEEE Explore Complaint – Part Number: CFP17L34-ART, DOI: https://doi.org/10.1007/978-981-13-1927-3_27, ISBN: 978-1-5386-4031-9, 2017
16. PurnachandraRao, NagamalleswaraRao ,HDFS Intra-DataNode Disk Balancer using Dynamic Programming", International Journal of Computer & Mathematical Sciences (IJCMS) ISSN 2347-8527 Volume 6, Issue 8, August 2017.
17. PurnachandraRao, NagamalleswaraRao , HDFS Fair Scheduler Preemption for Pending Request", International Journal of Innovations & Advancement in Computer Science (IJIACS), ISSN 2347-8616, Volume 6, Issue 9, September 2017.
18. Sai Krishna Mylavarapu, Adaptive CPU Resource Management in Distributed Systems, International Journal of Intelligent Systems and Applications in Engineering, Vol. 10 No. 2s ,2022, ISSN: 2147-6799.
19. Vijaya Krishna Namala, Enhanced Commit Protocols for Low Latency Distributed Transactions, Computer Fraud and Security Journal, Volume 2022, Issue 5, 2022.
20. Srinivasa Reddy Kummetha, Enhancing Network Security Through Policy Based Threat Detection, *International Journal of Intelligent Systems and Applications in Engineering*, Vol. 8 No. 4, 2020.
21. Srinivasa Reddy Kummetha, Reducing Computational Overhead In Network Graph Partitioning, International Journal of Innovative Research and Creative Technology, Volume 6 Issue 5 October-2020, E-ISSN: 2454-5988
22. Kalesha Khan Pattan. AI-driven Scaling Strategies for Adaptive Workload Management in Distributed Cloud Systems. International Journal of Technology and Applied Science, Volume 13, Issue 5, https://doi.org/10.71097/IJTAS.v13.i5.1125, 2022.
23. Arunkumar Sambandam. Adaptive Replication for Low Latency Distributed Clusters. International Journal of Intelligent Systems and Applications in Engineering, Volume 10, Issue 2, ISSN: 2147-6799, 2022.
24. Naveen Kumar Bandaru. Correlated Telemetry Driven Throughput Analysis in Distributed Pipelines. Computer Fraud and Security, Volume 2022, Issue 5, ISSN (Online): 1873-7056, 2022.
25. Srinivasa Reddy Kummetha. Minimizing Worst-case Complexity in Context-free Graph Coloring Using Algorithmic Approaches. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, Volume 10, Issue 5, September–October, E-ISSN: 2349-7300, 2022.
26. Srinivasa Reddy Kummetha, Enhancing Network Security Through Policy Based Threat Detection, International Journal of Intelligent Systems and Applications in Engineering, Vol. 8 No. 4, 2020.
27. Srinivasa Reddy Kummetha, Reducing Computational Overhead In Network Graph Partitioning, International Journal of Innovative Research and Creative Technology, Volume 6 Issue 5 October-2020, E-ISSN: 2454-5988
28. Bondugula, V. K. Enhancing Deadlock Management in Distributed Databases Using Serializable Snapshot Isolation. International Journal of Leading Research Publication, Volume 3, Issue 5, https://doi.org/10.70528/IJLRP.v3.i5.1547, 2022.

29. Kanagalakshmi Murugan. Reinforcement Learning Scheduler to Improve Kubernetes Performance and Scalability. International Journal of Leading Research Publication, Volume 3, Issue 5, May, https://doi.org/10.70528/IJLRP.v3.i5.1641, 2022.
30. Srinivasa Reddy Kummetha, Navigating Multiple Threat Blockages with Context-free Coloring, International Journal on Science and Technology, Volume 12, Issue 4, October-December 2021, E-ISSN: 2229-7677
31. Srinivasa Reddy Kummetha , Minimizing Worst-case Complexity in Context-free Graph Coloring Using Algorithmic Approaches, International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, Volume 10, Issue 5, 2022.