Deepika Singh, 2022, 10:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

A Intelligent VM Health Scoring Using Nagios Logs and AI Models

Deepika Singh, Raghavendra Joshi, Nandhini Prabhu, Ravi Kiran
Osmania University, Hyderabad, India

Abstract- This study investigates the role of symbolism in driving cultural transformations, focusing on the interplay between traditional and modern symbols, and the significant influences of media, technology, art, literature, and economic factors. Through a comprehensive methodological approach that includes literature review, case studies, and symbolic analysis, the research uncovers the dynamic processes through which symbols evolve and impact societal norms, identities, and collective consciousness. The findings demonstrate that symbols act as catalysts for societal change, facilitating shifts in cultural narratives and social attitudes. This paper provides valuable insights into the psychological impacts of symbolic transformations and underscores the importance of symbols in understanding cultural evolution and societal development.

Keywords: Symbolic Transformation, Cultural Evolution, Media Influence, Psychological Impact, Traditional and Modern Symbols.

I. INTRODUCTION

Evolution of VM Monitoring in Enterprise IT

As virtualization technologies have become foundational in modern enterprise infrastructure, monitoring virtual machines (VMs) has transitioned from a reactive to a proactive discipline. Legacy monitoring tools like Nagios have long provided critical alerting capabilities based on defined thresholds. However, with increased VM density, multi-tenant environments, and dynamic scaling, these static methods are no longer sufficient. Administrators require a more intelligent, context-aware view of system health to ensure availability, performance, and service-level adherence.

Challenges of Manual and Static Threshold Monitoring

Threshold-based monitoring suffers from inherent limitations—false positives, lack of contextual understanding, and difficulty in adapting to workload variability. A single spike in CPU might be benign in one VM but critical in another. Static alert definitions also fail to capture cumulative health degradation or intermittent anomalies. These shortcomings often lead to alert fatigue, missed incidents, and suboptimal root cause analysis. There's a pressing need for adaptive mechanisms that understand infrastructure behavior patterns in real time.

Why AI-Augmented Health Scoring Matters

AI-enhanced systems offer an opportunity to model VM health using patterns learned from historical data. By applying machine learning (ML) to telemetry from Nagios logs, systems can generate dynamic health scores that reflect true operational risk. These scores act as early warning systems, highlighting performance drift or degradation before critical failures occur. Beyond alerting, health scores can be integrated with ITSM platforms, resource scaling engines, or auto-remediation scripts to drive intelligent operations (AIOps).

Objectives and Scope of the Review

This review focuses on combining log-based monitoring from Nagios with machine learning models to build a real-time, intelligent VM health scoring framework. It covers: telemetry collection, data preprocessing, feature engineering, ML model selection, scoring pipelines, visualization dashboards, and integration with incident response workflows. The article also explores practical use cases and discusses challenges such as data quality, explainability, and system scalability.

II. OVERVIEW OF NAGIOS AND VM TELEMETRY

Architecture and Plugins of Nagios Monitoring

Nagios employs a modular architecture centered around a core daemon (nagios) that handles scheduling, event processing, and alerting. The extensibility of Nagios comes primarily from its plugin framework, where custom scripts and binaries can be used to monitor specific services, system metrics, and application behavior. For virtual machines, plugins such as check_cpu, check_mem, and check_disk are commonly employed to assess host health. These plugins can be executed either locally or remotely through NRPE (Nagios Remote Plugin Executor) or SSH, enabling monitoring across distributed infrastructures. The Nagios configuration model, which includes host and service definitions, contact groups, and escalation policies, allows for fine-grained control over alerting and monitoring policies. In virtualized environments, Nagios plugins are also tailored to interact with hypervisors, containers, and cloud APIs to extract contextual health data. This plugin-driven architecture ensures that Nagios remains relevant in complex, hybrid infrastructures.

Types of Logs and Events Generated

Nagios generates several log files, the most critical of which include nagios.log, status.dat, and performance data logs. These files contain a chronological record of all host and service checks, state transitions, notifications, acknowledgments, and passive checks. Each log entry typically includes a timestamp, host/service name, status (e.g., OK, WARNING, CRITICAL), and a plugin output message. These logs serve as an invaluable source of ground truth when analyzing VM behavior and system health over time. In clustered Nagios deployments, logs are synchronized across nodes to maintain consistency. For long-term storage and analysis, logs can be forwarded to external systems such as a centralized ELK (Elasticsearch, Logstash, Kibana) stack or Splunk. Importantly, these logs are raw and unstructured, requiring significant preprocessing before being used as features in machine learning pipelines. Nonetheless, they provide essential visibility into resource trends, anomaly occurrences, and historical baselines.

Typical VM Health Metrics Tracked (CPU, I/O, Memory, Network)

Virtual machine health is typically evaluated using a standard set of telemetry metrics: CPU utilization, memory consumption, disk I/O, and network throughput. Nagios, through its plugins, periodically collects and evaluates these metrics to determine the operational state of a VM. CPU metrics help identify overutilization or idle wastage, while memory data captures swap usage, caching behavior, and memory leaks. Disk I/O metrics reveal latency, saturation, and read/write errors, which are often precursors to performance degradation. Network metrics, including packet loss, interface errors, and bandwidth usage, can indicate bottlenecks or service disruptions.

While each metric is individually useful, true health scoring requires contextual correlation—for example, high CPU usage might be normal during backups, but problematic during idle hours. Moreover, anomalies in these metrics can be transient or persistent, and distinguishing between the two is key to minimizing false alerts. Health scoring models must account for these complexities to produce reliable outcomes.

Limitations of Rule-Based Alerting and False Positives

Traditional Nagios alerting relies on static threshold configurations, such as CPU usage > 80% for more than five minutes. While straightforward to implement, this approach is inherently brittle in dynamic environments. Static rules fail to accommodate seasonal patterns, workload spikes, or maintenance windows, leading to frequent false positives. Additionally, they cannot differentiate between transient issues and sustained degradation.

As infrastructure complexity grows—with multi-tenant VMs, containers, and variable workloads—rule tuning becomes a manual, error-prone process. These limitations result in "alert fatigue," where administrators either ignore alerts or become overwhelmed, potentially missing critical failures.

Moreover, rule-based alerts operate in isolation and cannot perform pattern recognition across time or multiple metrics. AI-enhanced health scoring addresses these gaps by learning from historical patterns and contextual data, significantly reducing noise while improving precision and recall in fault detection.

III. FROM LOGS TO FEATURES: DATA PREPROCESSING PIPELINE

Parsing Nagios Logs (Host Logs, Service Logs, Alert Histories)

Parsing Nagios logs is the first and most critical step in transforming raw monitoring data into actionable inputs for machine learning models. Nagios logs are plain-text and semi-structured, which poses challenges for automated parsing. Scripts must extract relevant fields such as timestamps, hostnames, service names, state transitions, plugin output, and duration. Host logs document uptime and availability, while service logs focus on application-level metrics like response times or disk usage. Alert history, on the other hand, captures the lifecycle of an event when it occurred, how it was acknowledged, and when it was resolved. Log parsing tools like Logstash, Fluentd, or custom Python scripts using regular expressions or log parsing libraries are typically used to extract structured records. Each parsed event becomes a candidate feature for training models, enabling temporal analysis of fault progression and systemic correlation of events across services and hosts.

Event Correlation and Timeline Reconstruction

Once logs are parsed, the next step involves reconstructing sequences of events that occurred within the same VM or service context. This correlation is essential to understanding causality and identifying patterns that lead to failures. For instance, high disk I/O followed by memory saturation and then a service crash can indicate a workload spike or memory leak. Timeline reconstruction involves aligning events on a common clock, usually based on synchronized timestamps (e.g., UTC). It also includes linking dependent services (e.g., Apache and MySQL) and categorizing alerts by severity, frequency, and recurrence. Event correlation tools such as ELK with Graph plugin, or even purpose-built correlation engines, are used to automate these workflows. The resulting correlated sequences form the basis for temporal features used in models such as LSTMs or time-based classifiers.

Feature Engineering Techniques (Rolling Averages, Histograms, Temporal Flags)

Raw telemetry is rarely suitable for direct use in machine learning. Feature engineering techniques such as rolling averages smooth out noise and capture sustained trends in VM metrics. For example, a rolling average of CPU over 10-minute intervals can highlight consistent overutilization. Histograms are used to capture distributional properties of metrics, such as variance in memory usage or spike frequency. Temporal flags binary or categorical variables indicating time of day, day of the week, or holiday periods help models account for periodicity in workloads. Additionally, features such as "number of alerts per hour" or "duration in critical state" give a quantified sense of alert density and system distress. These engineered features make AI models more robust and capable of generalizing across VMs with different usage profiles.

Labeling Health States for Supervised Learning

To train supervised models, data must be labeled with known health states—e.g., "healthy," "degraded," or "failed." Labeling can be done manually by analyzing postmortem reports, system tickets, or change logs, or automatically using heuristics (e.g., a critical alert followed by downtime equals failure). One challenge is that failures are relatively rare, leading to imbalanced datasets. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) or class-weighted learning are used to address this. Labeling is also complicated by partial failures (e.g., degraded performance without a complete crash) and false positives in Nagios alerts. Thus, labels must be validated using cross-functional inputs from monitoring teams, application owners, and infrastructure logs. Accurate labels are the foundation of trustworthy machine learning models.

IV. AI AND ML MODELS FOR HEALTH SCORING

Regression-Based Scoring Models (e.g., Random Forest Regression)

Regression models are well-suited for computing a continuous health score rather than simple classifications. Random Forest Regression, for example, aggregates multiple decision trees to predict a health index based on input features like CPU load, memory use, and alert density. This approach captures non-linear interactions and handles missing data effectively. Other models like Gradient Boosting Regression and Support Vector Regression can also be used, depending on data characteristics. Regression models output a score typically between 0 and 100, representing system health, where 100 is fully healthy. These scores are intuitive for dashboards and can trigger alerts based on dynamic thresholds. Model interpretability tools like SHAP or feature importance plots help explain how different metrics influence the score, aiding operational trust.

Anomaly Detection Models (Isolation Forest, Autoencoders)

In scenarios where labeled failure data is limited or absent, unsupervised anomaly detection models are invaluable. Isolation Forests detect anomalies by randomly partitioning data and identifying instances that are isolated with few splits. Autoencoders, neural networks trained to reconstruct their input, flag anomalies based on reconstruction error. These models are ideal for identifying unusual behavior in Nagios logs, such as rare event combinations or unseen metric spikes. Anomaly scores from these models can be mapped to health scores or directly used to trigger alerts. They are particularly effective in capturing silent failures or gradual degradation that static rules often miss. When integrated into real-time pipelines, these models help detect novel threats in dynamic VM environments.

Ensemble Models and Voting Systems

Combining multiple models through ensemble learning often yields better performance and resilience. For example, a weighted average of a regression model and an anomaly detector can provide a more nuanced score—balancing known issues with pattern deviation. Voting classifiers or score aggregators allow incorporation of different perspectives (e.g., short-term vs long-term metrics). This reduces the risk of overfitting and ensures robust behavior across diverse workloads. Ensembles are especially useful in hybrid VM setups where Linux, Windows, and containerized systems may behave differently. Moreover, ensemble outputs can be tuned using business rules for example, elevating scores during patient admission hours in healthcare systems.

Explainability in AI-Driven Scoring (LIME, SHAP)

Explainability is crucial for operational trust, especially in regulated industries. LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) provide interpretable insights into why a model produced a certain score. For example, SHAP can show that high memory usage and repeated disk alerts were primary contributors to a low health score. These tools help SREs and system admins validate model decisions and adjust configurations accordingly. In enterprise environments, explainability also aids in compliance with standards like ISO/IEC 27001 or HIPAA, which require traceability of automated decisions. When visualized in dashboards, explanation metrics become educational feedback loops that enhance human-AI collaboration in system monitoring.

V. REAL-TIME HEALTH SCORE PIPELINES

Stream Processing for Live Metric Ingestion

To generate health scores in real-time, it's essential to build a data ingestion pipeline capable of handling continuous streams of VM metrics and Nagios log entries. Tools like Fluentd, Logstash, or Apache Kafka are commonly used to collect and route logs from multiple sources. These systems allow structured data from Nagios (like check results or state transitions) to be streamed directly into a processing engine like Apache Spark or Flink. Stream processing frameworks can apply transformations, parse metrics, and compute intermediate statistics such as moving averages or alert frequencies on the fly. This enables near-instantaneous feedback on a VM's operational state. Integration with monitoring

agents ensures that logs from /var/log, status.dat, and plugin outputs are captured promptly. Ultimately, this streaming infrastructure feeds real-time data into the AI models trained earlier, allowing them to continuously produce and update VM health scores, with latency typically measured in seconds to minutes.

Inference Engines and Score Output

Once the pre-trained models are in place, a dedicated inference engine is used to apply them to incoming data. This engine can be a lightweight Python microservice running Flask or FastAPI, deployed either on centralized servers or directly on VM clusters. As new telemetry arrives, the engine loads the relevant model (regression, anomaly detector, etc.) and computes a health score, which is then emitted via REST API or pushed into a metrics database like Prometheus or InfluxDB. These scores can be aggregated or visualized on Grafana dashboards, color-coded by severity. By assigning each VM a dynamic, real-time score, system operators gain a continuous view of system health without being flooded by raw alerts. Moreover, thresholds can be set to trigger proactive interventions—like scaling a cluster, restarting a process, or flagging a ticket. This approach shifts monitoring from reactive alerts to intelligent, predictive analytics.

Feedback Loops for Model Refinement

The effectiveness of AI-driven health scoring improves over time when feedback loops are implemented. These loops capture operator responses to alerts, postmortem root causes, and system behaviors that occurred after score dips. For instance, if a VM scored "critical" but remained stable, this instance becomes a valuable counterexample for model retraining. Feedback is collected via helpdesk ticket closures, incident comments, or structured admin annotations in dashboards. This feedback can be stored in a metadata layer associated with telemetry records. In weekly or monthly retraining cycles, this enriched dataset helps improve model precision, reduce false positives, and align scoring closer to real-world behavior. Incorporating feedback also makes the scoring system adaptive to infrastructure changes such as new services, updated plugins, or workload migrations without needing constant manual intervention.

VI. INTEGRATION WITH OPERATIONS AND DECISION SYSTEMS

Alert Thresholding Based on Score Dynamics

In contrast to binary thresholds used in traditional monitoring, AI-based health scoring supports dynamic thresholding. This means alerts can be triggered not only when scores fall below a fixed value but also based on the rate of decline or deviation from historical baselines. For example, a 20-point drop in score over an hour may warrant investigation even if the final score remains above 60. Dynamic thresholding reduces alert fatigue and ensures alerts are meaningful in context. Administrators can configure policies such as "alert only if the score drops by 30% within 15 minutes" or "escalate if the score remains below 50 for more than 10 minutes." These conditions are highly customizable and can be aligned with SLAs or SLOs for critical systems. This approach enables more nuanced alerting strategies and better prioritization of incident response in operational workflows.

Dashboard and Visualization Interfaces

Health scores are most effective when visualized intuitively. Dashboards built on platforms like Grafana, Kibana, or Splunk offer real-time heatmaps, historical trends, and correlation graphs. These visualizations enable system administrators to quickly pinpoint distressed VMs, identify clusters with recurring problems, and drill into component-level issues. For example, a Grafana dashboard can display a grid of VMs color-coded by score, along with line graphs of score evolution over time and metrics contributing to score drops. Some dashboards integrate with LIME or SHAP visualizations to show "why" a score changed. Admins can interactively adjust thresholds or overlay annotations (like planned maintenance) for added context. By combining raw metrics with AI-derived scores and their explanations, these dashboards serve as a powerful tool for both real-time monitoring and retrospective analysis.

Integration with Incident Management Platforms
To close the loop between detection and resolution, health scoring systems must integrate with ITSM platforms like ServiceNow, Jira Service Management, or PagerDuty. When scores cross alert thresholds or exhibit significant drops, automated tickets can be generated with contextual information such as the score value, top contributing metrics, and suggested remediations.

Using webhooks or APIs, these systems ensure that predictive alerts reach the right teams with minimal manual effort. Integration with CMDBs helps correlate scores with asset metadata such as VM role, owner, or business criticality—so incidents can be prioritized intelligently. Moreover, updates to the ticket (like resolution status or comments) can be fed back into the scoring model for training. This bi-directional flow enables a self-improving, operationally embedded AI monitoring ecosystem.

VII. CASE STUDIES AND REAL-WORLD DEPLOYMENTS

Academic Hospital Using Nagios and Autoencoders
An academic hospital IT department deployed Nagios to monitor approximately 500 VMs supporting critical applications such as PACS, EHR, and billing systems. They augmented their Nagios setup with Autoencoder-based anomaly detection, feeding it parsed logs and VM metrics like CPU, memory, and disk I/O. By training Autoencoders on months of "healthy" data, the team established baseline behaviors for each VM type.

When reconstruction errors exceeded predefined thresholds, early alerts were generated. Over six months, the hospital saw a 35% reduction in unplanned downtime and nearly eliminated false-positive alerts for transient spikes. Integration with a Grafana dashboard provided real-time visibility into score fluctuations. Medical IT teams appreciated the model's ability to catch performance degradation before service interruptions, allowing for proactive scaling or failover.

Telecom Cloud Provider with Regression-Based Scores
A telecom cloud provider managing over 2000 Linux VMs across multiple regions used Random Forest regression to assign health scores to VMs hosting VoIP, OSS/BSS, and streaming services. Nagios was already used for basic monitoring, but alerts were too noisy and missed context. The regression model incorporated 20+ engineered features—such as alert density, resource utilization trends, and historical uptime—to generate scores from 0 to 100 every 5 minutes. These scores were pushed to Prometheus and visualized in a custom Grafana dashboard. The system allowed SREs to detect gradual degradation in underutilized nodes and automate rolling restarts or workload balancing. The health scores also helped triage incidents, reducing mean time to resolution (MTTR) by 40%. Over time, the model was tuned to prioritize high-traffic or customer-facing nodes, adding value to their NOC operations.

Government Data Center with Explainable AI Models
A government-run data center adopted a SHAP-explained Gradient Boosting model to score VMs supporting citizen services and backend systems. Given the regulatory environment, explainability was crucial. Each health score was accompanied by a SHAP-based breakdown showing which features—e.g., disk errors, swap usage, alert density—contributed to risk. These were embedded in ServiceNow incident tickets to justify preemptive interventions. The team also used LIME for real-time score explanations during live war room meetings. The approach improved cross-team communication and confidence in AI-driven decisions. The model was retrained quarterly with labeled ticket data and system feedback. This explainable AI system enabled the center to remain compliant with government IT mandates while improving uptime and response agility.

VIII. CHALLENGES AND RISKS

Data Quality and Noise in Nagios Logs
One of the most critical challenges in building intelligent health scoring models using Nagios data is poor data quality. Nagios logs, while detailed, are

often inconsistent in formatting across custom plugins or legacy implementations. Timestamp gaps, duplicate events, or missing severity labels can complicate parsing and make downstream feature extraction error-prone. Moreover, Nagios doesn't normalize plugin outputs—some may emit structured text, while others produce free-form error messages. This variability injects noise that can mislead AI models, resulting in poor generalization or false anomaly detection. Additionally, transient metrics such as CPU spikes during backups—are often misinterpreted as genuine issues unless temporal smoothing or contextual filtering is applied. Addressing this requires robust log sanitization pipelines, intelligent outlier filtering, and well-curated plugin configurations. Without this preprocessing hygiene, AI predictions may be skewed or outright incorrect, undermining trust in the health score system.

False Positives and Alert Fatigue in AI Scores

Even with machine learning, false positives remain a major issue. Overly sensitive models may flag routine performance fluctuations as health risks, flooding dashboards with red alerts and undermining user confidence. This problem is especially acute in healthcare or financial sectors where system volatility is natural—like during nightly data syncs or end-of-month processing. If AI-driven health scores consistently trigger alerts during these known patterns, admins begin to ignore them, reintroducing the same fatigue associated with traditional rule-based systems. Balancing sensitivity with specificity is critical. Techniques like threshold learning, alert suppression during maintenance windows, and post-processing with temporal filters can mitigate this. Incorporating human-in-the-loop feedback, where operators label false alerts, helps refine future predictions. Ultimately, AI must augment—not replace—operator intuition, and scoring models should be transparent and adaptable to operational realities.

Model Drift and Infrastructure Evolution

AI models degrade over time—a phenomenon known as model drift. As underlying infrastructure changes (e.g., new VM templates, kernel versions, workload shifts), the statistical patterns captured during model training may no longer hold. For example, introducing a new backup agent might increase CPU usage during specific hours, misleading a static model into interpreting it as a fault. Similarly, replacing hardware or migrating services to containers can invalidate existing telemetry patterns. Model drift leads to inaccurate scoring and increased false negatives or positives. Addressing this requires periodic retraining using fresh data, incorporating change logs from CMDBs, and maintaining a robust monitoring loop to detect scoring anomalies. Automation can help identify when models deviate significantly from expected prediction accuracy, triggering retraining workflows. Failing to manage drift risks eroding the credibility of the health scoring platform and increasing operational risk.

IX. FUTURE TRENDS

Transfer Learning Across Similar VM Types

As organizations grow and diversify their infrastructure, retraining models for each VM or cluster becomes resource-intensive. Transfer learning offers a solution by allowing pre-trained models from one environment (e.g., Apache web servers) to be adapted to similar VMs (e.g., NGINX servers). This involves freezing base layers of the model and fine-tuning only a subset using limited labeled data from the target system. Transfer learning reduces training time, improves convergence, and allows rapid onboarding of new applications or regions. For instance, a regression model trained on Nagios data from Europe-based web servers can be adjusted for US-based servers without needing months of logs. This approach is especially useful in multi-tenant cloud environments where templates are cloned across clients. By reducing dependency on local training data, transfer learning accelerates model deployment and health scoring scalability.

Federated Learning for Privacy-Conscious Environments

In sectors like healthcare and finance, raw telemetry and logs often contain sensitive metadata, making centralized model training a compliance risk. Federated learning solves this by allowing each site

or data center to train models locally and share only anonymized model updates (gradients or weights) with a central server. The central model aggregates these updates and redistributes the refined model to all participants. This decentralized training preserves data privacy while enabling collaborative intelligence. Hospitals across a region, for example, could improve VM health prediction accuracy without sharing sensitive logs. Implementing federated learning requires careful synchronization, differential privacy mechanisms, and robust version control. As adoption grows, this approach will become critical for large enterprises seeking to harness AI without violating data sovereignty or exposing operational telemetry.

Multimodal Models Combining Logs, Metrics, and Topology

Most current health scoring models rely solely on numerical metrics or log sequences. However, richer insights emerge when multiple data modalities are fused—logs, structured metrics, topology maps, and even change tickets. Multimodal models can understand not just "what" is failing, but "why," "where," and "what's impacted." For example, integrating service dependency graphs from CMDBs with metric anomalies can help isolate fault domains and prioritize critical assets. Combining workload schedules with metric drift can distinguish false alerts from real degradation. These models, typically built on transformer-based architectures or graph neural networks, provide a holistic view of VM health. Though still an emerging field, multimodal fusion is poised to become the next frontier in intelligent infrastructure analytics.

X. CONCLUSION

The evolution of virtual machine monitoring from static thresholding to intelligent, AI-driven health scoring represents a paradigm shift in infrastructure operations. By leveraging Nagios logs, system metrics, and modern machine learning models, organizations can gain real-time visibility into system health with unprecedented granularity and foresight. Unlike traditional alert systems, health scoring offers continuous risk assessment, actionable insights, and contextual awareness significantly reducing downtime and improving service reliability. As shown through real-world deployments, from academic hospitals to government data centers, these systems not only enhance technical resilience but also align with regulatory, privacy, and operational goals.

However, challenges remain data quality, model drift, false positives, and integration complexity must be actively managed. The future lies in decentralized learning, multimodal intelligence, and explainable scoring interfaces. Ultimately, AI-augmented health scoring empowers IT teams to shift from reactive firefighting to proactive, predictive operations ushering in a new era of intelligent infrastructure management.

REFERENCE

1. Gan, T., Kumar, A., Ehiwario, M., Zhang, B., Sembroski, C., Jesus, O.D., Hoffmann, O.J., & Metwally, Y. (2019). Artificial Intelligent Logs for Formation Evaluation Using Case Studies in Gulf of Mexico and Trinidad & Tobago. Day 3 Wed, October 02, 2019.
2. Umamaheswari, K., & Sujatha, S. (2018). INSPECT- An Intelligent and Reliable Forensic Investigation through Virtual Machine Snapshots. International Journal of Modern Education and Computer Science, 10, 17-28.
3. Massaro, A., Gargaro, M., Dipierro, G., Galiano, A.M., & Buonopane, S. (2020). Prototype Cross Platform Oriented on Cybersecurity, Virtual Connectivity, Big Data and Artificial Intelligence Control. IEEE Access, 8, 197939-197954.
4. Abdullah, M., Lu, K., Wieder, P., & Yahyapour, R. (2017). A Heuristic-Based Approach for Dynamic VMs Consolidation in Cloud Data Centers. Arabian Journal for Science and Engineering, 42, 3535 - 3549.
5. Lu, Y., Liu, L., Panneerselvam, J., Yuan, B., Gu, J., & Antonopoulos, N. (2020). A GRU-Based Prediction Framework for Intelligent Resource Management at Cloud Data Centres in the Age of 5G. IEEE Transactions on Cognitive Communications and Networking, 6, 486-498.
6. Vijayendren, M.R., Thanuja, M., Sathishkumar, M.P., Kumar, M.S., & Vadivel, M. (2018). International Journal of Intellectual

Advancements and Research in Engineering Computations An Integrated Resource and Service Consolidation Approach using History Logs under Cloud Environment.

7. Battula, V. (2021). Dynamic resource allocation in Solaris/Linux hybrid environments using real-time monitoring and AI-based load balancing. International Journal of Engineering Technology Research & Management, 5(11), 81–89. https://ijetrm.com
8. Madamanchi, S. R. (2021). Disaster recovery planning for hybrid Solaris and Linux infrastructures. International Journal of Scientific Research & Engineering Trends, 7(6), 01–08.
9. Madamanchi, S. R. (2021). Linux server monitoring and uptime optimization in healthcare IT: Review of Nagios, Zabbix, and custom scripts. International Journal of Science, Engineering and Technology, 9(6), 01–08.
10. Madamanchi, S. R. (2021). Mastering enterprise Unix/Linux systems: Architecture, automation, and migration for modern IT infrastructures. Ambisphere Publications.
11. Mulpuri, R. (2021). Command-line and scripting approaches to monitor bioinformatics pipelines: A systems administration perspective. International Journal of Trend in Research and Development, 8(6), 466–470.
12. Mulpuri, R. (2021). Securing electronic health records: A review of Unix-based server hardening and compliance strategies. International Journal of Research and Analytical Reviews, 8(1), 308–315.
13. Tariq, Z., Mahmoud, M.A., & Abdulraheem, A. (2019). Core log integration: a hybrid intelligent data-driven solution to improve elastic parameter prediction. Neural Computing and Applications, 31, 8561 - 8581.
14. Lyu, M.R. (2018). AI Techniques in Software Engineering Paradigm. Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering.
15. Lashari, S.E., & Lashari, Z. (2020). Applications of Artificial Intelligence (AI) in Petroleum Engineering Problems.
16. Castineira, D., Zhai, X., Darabi, H., Valle, M., Maqui, A., Shahvali, M., & Yunuskhojayev, A. (2018). Augmented AI Solutions for Heavy Oil Reservoirs: Innovative Workflows That Build from Smart Analytics, Machine Learning And Expert-Based Systems. Day 2 Tue, December 11, 2018.
17. Rahaman, M.S., Vasant, D.P., Jufar, D.S., & Watada, J. (2020). Feature Selection-Based Artificial Intelligence Techniques for Estimating Total Organic Carbon from Well Logs. Journal of Physics: Conference Series, 1529.
18. Anifowose, F.A. (2009). Hybrid AI Models for the Characterization of Oil and Gas Reservoirs: Concept, Design and Implementation.
19. Sanquetta, C.R., Piva, L.R., Wojciechowski, J., Corte, A.P., & Schikowski, A.B. (2017). Volume estimation of Cryptomeria japonica logs in southern Brazil using artificial intelligence models. Southern Forests: a Journal of Forest Science, 80, 29 - 36.
20. Khazaeni, Y., & Gaskari, R. (2009). Top-Down Intelligent Reservoir Modeling (TDIRM).
21. Porwol, L., Pereira, A.G., & Ojo, A.K. (2019). From VR-Participation Back to Reality - an AI&VR-driven Approach for Building Models for Effective Communication in e-Participation. Irish Conference on Artificial Intelligence and Cognitive Science.,