RANK: AI-assisted End-to-End Architecture for Detecting Persistent Attacks in Enterprise Networks

#1 B.AMARNATH REDDY, #2 U.VENKATESWARLU
#1 ASSISTANT PROFESSOR #2 MCA SCHOLAR
DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
QIS COLLEGE OF ENGINEERING & TECHNOLOGY, ONGOLE
VENGAMUKKALAPALEM (V), ONGOLE, PRAKASAM DISTRICT, ANDHRA PRADESH

ABSTRACT

As enterprise networks continue to expand in size, complexity, and connectivity, they face an increasing number of persistent and sophisticated cyberattacks that traditional security measures struggle to detect effectively. These persistent threats, often orchestrated by advanced adversaries, can infiltrate systems, evade defenses, and remain undetected for extended periods, resulting in significant data breaches, financial losses, and operational disruptions. To address these challenges, this research proposes **AI-assisted End-to-End Architecture for Detecting Persistent Attacks in Enterprise Networks**, a comprehensive framework that leverages artificial intelligence (AI) and machine learning (ML) techniques to enhance threat detection, response automation, and situational awareness across network environments.

The proposed architecture integrates multiple layers of data collection, preprocessing, feature extraction, anomaly detection, and correlation analysis to monitor network activities in real time. By combining **supervised and unsupervised machine learning models**, the system can identify known attack signatures as well as previously unseen or stealthy behaviors that deviate from established baselines. Additionally, the architecture incorporates **deep learning methods**, behavioral analytics, and temporal pattern recognition to effectively capture the evolving nature of persistent attacks, including lateral movement, command-andcontrol communication, and data exfiltration attempts.

A key feature of this approach is the use of **context-aware threat intelligence** and adaptive learning models that continuously update detection capabilities based on evolving threats. By incorporating feedback loops and automated model retraining, the system dynamically adjusts to changes in the network environment and attacker strategies. Extensive evaluation using real-world enterprise traffic datasets demonstrates that the proposed architecture significantly improves detection accuracy, reduces false positives, and accelerates incident response compared to conventional intrusion detection systems.

INTRODUCTION

In today's digital age, enterprise networks have become increasingly complex, interconnecting numerous devices, applications, and cloud services. While this interconnected environment provides

significant operational advantages, it also exposes organizations to a wide range of cybersecurity threats. Among these threats, **persistent attacks**, such as Advanced Persistent Threats (APTs), are particularly challenging to detect and mitigate. These attacks are carried out by highly skilled adversaries who infiltrate networks stealthily, remain undetected for extended periods, and exploit vulnerabilities to exfiltrate sensitive data, disrupt services, or compromise critical infrastructure. Traditional security mechanisms, including firewalls, signaturebased intrusion detection systems, and manual monitoring, are often insufficient to identify such sophisticated attacks due to their adaptive and stealthy nature.

To address these challenges, the adoption of **Artificial Intelligence (AI) and Machine Learning (ML)** in cybersecurity has emerged as a promising solution. AI-assisted systems can analyze vast amounts of network traffic and behavioral data in real time, identifying anomalies and patterns that may indicate malicious activity. Unlike conventional rule-based approaches, AI systems can learn from historical data, adapt to evolving attack strategies, and detect both known and previously unseen threats. This capability is particularly critical for enterprise networks, where attackers continuously refine their techniques to bypass standard defenses.

The **RANK: AI-assisted End-to-End Architecture for Detecting Persistent Attacks** aims to provide a comprehensive framework for securing enterprise networks. The system integrates multiple components, including real-time data collection, preprocessing, feature extraction, anomaly detection, and correlation analysis, to monitor network activity continuously. By leveraging both supervised and unsupervised learning methods, as well as deep learning models for temporal and behavioral analysis, the architecture can detect subtle deviations from normal network behavior indicative of persistent attacks.Furthermore, the architecture incorporates **adaptive learning and threat intelligence** capabilities, enabling the system to update its detection models dynamically based on new attack patterns. This ensures that the system remains effective even as attackers evolve their strategies. Automated alert generation and response mechanisms further enhance the efficiency of cybersecurity operations by reducing the reliance on manual monitoring and enabling faster mitigation of threats.

LITERATURE SURVEY

The increasing complexity and scale of enterprise networks have led to the emergence of sophisticated and persistent cyberattacks, motivating extensive research in AI-assisted threat detection. Early approaches to network security relied heavily on **signature-based intrusion detection systems (IDS)** and rule-based firewalls. These methods effectively detected known attacks but were inadequate against **advanced persistent threats (APTs)**, which are stealthy, targeted, and adaptive. Researchers highlighted that traditional systems often fail to detect low-and-slow attacks or novel malicious patterns that deviate only slightly from normal network behavior.

With the rise of **machine learning and artificial intelligence**, several studies explored their potential to enhance cybersecurity in enterprise networks. Supervised learning algorithms, such as Decision Trees, Random Forests, and Support Vector Machines (SVM), were applied to labeled network traffic data to classify events as normal or malicious. These approaches demonstrated higher accuracy than conventional systems for detecting known threats. However, the literature notes that supervised models are limited by the availability of high-quality labeled datasets, which are often scarce or incomplete in realworld network environments.

To overcome this limitation, **unsupervised and semi-supervised learning methods** have been widely researched. Techniques such as clustering (K-Means, DBSCAN), anomaly detection, and autoencoders allow detection of previously unseen or stealthy attacks by modeling normal network behavior and flagging deviations. These methods are particularly suitable for detecting persistent attacks, as APTs typically blend into normal traffic and evade signature-based detection. Researchers found that combining supervised and unsupervised approaches in hybrid models significantly improves detection rates while reducing false positives.

Deep learning models, including recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, have also been explored for network intrusion detection. These models excel at capturing temporal patterns and sequential behaviors in network traffic, making them effective for detecting multi-stage persistent attacks. Studies show that deep learning-based IDS can identify subtle behavioral anomalies, such as lateral movement, data exfiltration attempts, and command-and-control communications, which traditional systems often miss.

Another key focus in the literature is **feature extraction and engineering**. Effective threat detection depends on identifying relevant network traffic attributes, including packet metadata, protocol behavior, flow statistics, and session characteristics. Research demonstrates that combining multiple features—such as time-based, behavioral, and contextual data—enhances the performance of AI-assisted detection systems. Some studies also incorporate threat intelligence feeds to provide contextual awareness and improve detection of emerging attack techniques.

A recurring challenge highlighted in the literature is **class imbalance**, as malicious events are typically rare compared to normal traffic. Techniques such as oversampling, undersampling, and cost-sensitive learning have been applied to address this imbalance and improve the detection of low-frequency attacks. Evaluation metrics such as precision, recall, F1-score, and ROC-AUC are emphasized in research studies over accuracy, given the importance of correctly identifying rare but critical threats.

Recent research trends emphasize **end-toend AI-assisted architectures** that integrate data collection, real-time processing, anomaly detection, correlation analysis, and automated response. These architectures aim

to provide scalable, adaptive, and proactive cybersecurity solutions for enterprise networks. Literature shows that such systems significantly improve detection accuracy, reduce false positives, and enhance operational efficiency compared to conventional IDS and firewall-based solutions.

In summary, the literature underscores a clear shift from traditional, static network security approaches toward **intelligent, adaptive, and AI-driven systems**. By combining supervised, unsupervised, and deep learning techniques with real-time analytics and threat intelligence, researchers have demonstrated that persistent attacks in enterprise networks can be detected more effectively, providing a strong foundation for developing end-to-end architectures like **RANK**.

1. **RANK: AI-Assisted End-to-End Architecture for Detecting Persistent Attacks in Enterprise Networks**
   Authors: Hazem M. Soliman, Geoff Salmon, Dušan Sovilj, Mohan Rao
   Merits: First end-to-end AI architecture for APT detection; automates pipeline from alert data to incident recommendations; reduces data load by ~1000× for analysts.
   Demerits: Uses relatively simple ML due to dataset limits; real-world deployment and scalability not fully validated.

2. **An Enhanced Mechanism for APT Detection based on Deep Learning**
   Authors: Saeed et al. (2025)
   Merits: Uses deep learning for APT detection; emphasizes Explainable AI (XAI) for trust.
   Demerits: High computational cost and requires robust dataset.

3. **Deep Recurrent Hidden Markov Learning Framework for MultiStage APT Prediction**
   Authors: Tijjani et al. (2026)
   Merits: Combines deep networks with probabilistic state models for attack stage prediction.
   Demerits: Tested on synthetic data; real network performance uncertain.

4. **TREC: APT Tactic/Technique Recognition via Few-Shot Provenance Subgraph Learning**
   Authors: Lv et al. (2024)
   Merits: Uses few-shot learning and graph structures to recognize attack tactics.
   Demerits: Few-shot generalization may still struggle with unseen campaigns.

5. **CONTINUUM: APT Detection via Spatio-Temporal Graph Neural Networks**
   Authors: Bahar et al. (2025)
   Merits: Combines spatio-temporal graph analysis with federated learning for privacy-aware detection.
   Demerits: Computational overhead and complexity in deployment.

SYSTEM ANALYSIS EXISTING SYSTEM

Enterprise networks today rely on a combination of **traditional security mechanisms** such as firewalls, signaturebased intrusion detection systems (IDS), antivirus software, and manual monitoring to protect against cyberattacks. These systems are designed to detect known threats using predefined signatures, rule-based patterns, and threshold limits. For example, intrusion detection systems monitor network traffic for suspicious activity based on known attack signatures, while firewalls filter unauthorized access attempts according to static rules.

Although these systems are widely deployed, they have significant limitations in detecting **persistent and sophisticated attacks**, such as Advanced Persistent Threats (APTs). APTs are highly targeted, stealthy, and adaptive, often blending into normal network traffic to avoid detection. Traditional IDS and firewall solutions are generally reactive— they rely on prior knowledge of attack signatures and cannot effectively detect unknown or evolving threats.

Some organizations have implemented **network monitoring tools** that provide alerts on unusual traffic patterns. These tools often depend on threshold-based rules, such as unusually high data transfer rates, repeated login failures, or abnormal access to critical systems. While helpful in identifying some anomalies, these tools lack intelligence to correlate multiple events, analyze complex attack sequences, or understand subtle behavioral changes over time. Consequently, many persistent attacks go undetected until significant damage occurs.

Another approach used in existing systems is **manual analysis by security analysts**. Analysts review logs, alerts, and suspicious activity reports to identify potential threats. However, manual monitoring is timeconsuming, prone to human error, and cannot scale to handle the massive volume of network traffic in modern enterprises. Additionally, attackers often execute lowand-slow attack strategies that evade human observation, making manual detection insufficient.

In short, the existing systems for enterprise network security are largely **reactive, rulebased, and dependent on prior knowledge of threats**. They are unable to efficiently detect stealthy, persistent attacks or adapt to new, sophisticated attack patterns. These limitations highlight the need for an **AIassisted end-to-end architecture**, such as RANK, which can analyze real-time network data, learn from evolving patterns, and detect both known and unknown attacks proactively.

Disadvantages of Existing System

The existing enterprise network security systems, which rely heavily on traditional firewalls, signature-based intrusion detection systems (IDS), and manual monitoring, suffer from several limitations that reduce their effectiveness against modern cyber threats. The key disadvantages are:

1. **Inability to Detect Unknown Threats:** Traditional systems primarily rely on known attack signatures or predefined rules. They fail to detect previously unseen or evolving threats, such as advanced persistent threats (APTs), zero-day exploits, or stealthy malware.

2. **Reactive Nature:** Most existing systems operate reactively, generating alerts after an attack occurs. This delayed detection increases the risk of data breaches, system compromise, and operational disruption.

3. **High False Positives:** Signaturebased IDS and threshold monitoring often flag legitimate network activities as malicious, leading to a large number of false alarms. This increases the workload of security analysts and may result in critical threats being overlooked.
4. **Limited Scalability:** Manual monitoring and traditional IDS are unable to handle the massive volume of network traffic in modern enterprise environments. As networks scale, these systems struggle to maintain real-time detection and timely response.
5. **Lack of Behavioral Analysis:** Existing systems cannot analyze subtle changes in user or device behavior over time, which are often indicative of stealthy attacks. They are unable to correlate multiple events or recognize complex attack patterns that unfold gradually.

6. **Dependenceon Human Intervention:** Manual log analysis and monitoring require skilled personnel, which is time-consuming and error-prone. Human limitations make it difficult to maintain continuous, effective monitoring across large, complex networks.
7. **Static and Inflexible:** Rule-based and threshold-driven systems cannot adapt to evolving attack strategies or dynamic network environments. This inflexibility leaves networks vulnerable to sophisticated, multistage attacks.

PROPOSED SYSTEM

The **RANK: AI-assisted End-to-End Architecture for Detecting Persistent Attacks** is designed to overcome the limitations of traditional enterprise security systems by providing a **proactive, intelligent, and adaptive solution** for detecting sophisticated cyber threats. Unlike conventional tools that rely on static rules or known attack signatures, this system leverages **artificial intelligence (AI) and machine learning (ML)** to continuously analyze network behavior in real time, identify anomalies, and detect both known and previously unseen threats. The architecture integrates multiple layers, starting with **data collection**, where network traffic, system logs, user activities, and device information are continuously gathered from across the enterprise environment. The raw data is then processed and transformed into meaningful features through **data preprocessing and feature extraction**, including packet metadata,

session patterns, device behavior, and protocol usage.

The core of the system is **AI/ML-based threat detection**, which uses a combination of supervised models to detect known attack signatures and unsupervised or anomaly detection models to identify new and subtle attack patterns. Additionally, deep learning techniques such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks are applied to capture temporal and sequential patterns in network traffic, enabling detection of multi-stage persistent attacks like lateral movement or data exfiltration. The system further incorporates **behavioral analysis and correlation modules** to analyze deviations in user and device behavior and link suspicious activities across multiple systems and time periods, allowing complex attack sequences to be identified.

An important feature of the proposed system is **adaptive learning and integration with threat intelligence**, which enables continuous updating of models based on emerging attack patterns and external threat feeds. This ensures that the system remains effective even as attackers evolve their strategies. In addition, the system provides **real-time alerting and automated response capabilities**, including the ability to block malicious traffic, isolate compromised devices, or escalate alerts for further investigation, thereby minimizing response time and reducing the dependency on manual intervention. A centralized monitoring dashboard allows security teams to track detected threats, manage incidents, and generate reports for auditing and compliance purposes.

Advantages of Proposed System

The proposed **RANK: AI-assisted End-toEnd Architecture** offers several significant advantages over traditional enterprise network security systems. Firstly, it provides **real-time and proactive threat detection**, allowing organizations to identify and respond to persistent and sophisticated attacks before they cause substantial damage. Unlike rule-based systems, the use of **machine learning and AI** enables the system to detect both known threats and previously unseen attack patterns, including subtle anomalies in network behavior. This adaptability ensures continuous protection even as cybercriminals evolve their strategies.

Secondly, the system significantly **reduces false positives** through advanced anomaly detection and correlation of multiple network events. By analyzing behavioral patterns and temporal sequences, it distinguishes between legitimate network activity and malicious behavior more accurately than conventional monitoring tools, reducing the workload on security teams. The integration of **deep learning models** further enhances the detection of complex, multi-stage attacks such as lateral movement, privilege escalation, and data exfiltration, which are often missed by traditional IDS and firewalls.

Another advantage is the **automation of threat response and monitoring**. The system can initiate immediate actions, such

as blocking suspicious traffic or isolating compromised devices, minimizing the reliance on manual intervention and accelerating incident mitigation. Additionally, the **adaptive learning capability** ensures that the system continuously improves over time, incorporating feedback from detected threats and integrating external threat intelligence to stay ahead of emerging attack techniques.

The architecture is also **scalable and comprehensive**, capable of handling highvolume network traffic in large enterprise environments without compromising detection accuracy or response time. A centralized dashboard provides security teams with clear visibility into network health, detected anomalies, and mitigation actions, facilitating better decision-making, auditing, and compliance reporting.

Overall, the RANK system enhances the **security, reliability, and resilience** of enterprise networks. By combining AI, machine learning, behavioral analysis, and automated response, it offers a robust and intelligent solution that overcomes the limitations of existing security

IMPLEMENTATION 1. Requirement Analysis

The implementation of the project **"RANK: AI-Assisted End-to-End Architecture for Detecting Persistent Attacks in Enterprise Networks"** begins with analyzing the increasing threat of Advanced Persistent Attacks (APTs) in enterprise environments. Traditional security systems often fail to detect long-term and stealthy cyberattacks that spread across networks over time. The proposed system uses artificial intelligence and machine learning techniques to continuously monitor enterprise networks, analyze attack patterns, and detect persistent threats in real time.

2. System Design

The system architecture is designed for intelligent enterprise network monitoring and cyberattack detection.

Main Modules

- Network Traffic Collection Module
- Data Preprocessing Module
- Feature Extraction Module
- AI-Based Threat Detection Module
- Attack Correlation Module
- Alert and Incident Response Module
- Security Dashboard Module

The architecture enables automated detection and analysis of persistent cyber threats. **3. Network Data Collection**

The system collects enterprise network traffic and security-related information from different network devices.

Data Sources

- Firewalls
- Routers and switches
- Intrusion Detection Systems (IDS)
- System logs
- DNS traffic
- User activity records

The collected data is stored securely for threat analysis.

4. Data Preprocessing

The collected network traffic data undergoes preprocessing before AI analysis.

Preprocessing Steps

- Removing duplicate packets
- Noise filtering
- Traffic normalization
- Log parsing
- Missing value handling

These operations improve machine learning performance and detection accuracy. **5. Feature Extraction**

Important cybersecurity features are extracted from network traffic and system logs.

Extracted Features

- IP addresses
- Packet size and frequency
- Session duration
- Failed login attempts
- Suspicious communication patterns
- Malware-related behaviors

These features help identify malicious and persistent attack activities.

METHODOLOGY 1. Enterprise Network Data Acquisition

The methodology begins with collecting network traffic and security logs from enterprise infrastructure.

Data Collected

- Packet traffic
- System event logs
- User activity records
- Firewall alerts
- Authentication logs

This data forms the basis for attack analysis.

2. Network Traffic Cleaning and Preparation

The collected network data undergoes preprocessing to remove unnecessary or corrupted information.

Data Processing Operations

- Packet filtering
- Traffic normalization
- Log formatting
- Duplicate removal

These preprocessing steps improve AI model learning efficiency.

3. Cybersecurity Feature Engineering

The system extracts relevant security features from enterprise traffic data.

Important Features

- Suspicious IP communication
- Login anomalies

- Unusual traffic patterns
- Data transfer behavior
- Malware indicators

These features help distinguish malicious activities from normal traffic.

4. AI-Based Model Training

The machine learning and deep learning models are trained using labeled cybersecurity datasets.

Training Process

1. Input network traffic data
2. Extract cybersecurity features
3. Train AI models
4. Optimize detection performance
5. Validate attack prediction accuracy

The models learn attack signatures and behavioral anomalies.

5. Persistent Threat Detection

The trained AI system continuously analyzes enterprise traffic to identify long-term malicious activities.

Detection Workflow

- Monitor traffic behavior
- Detect suspicious patterns
- Correlate related events
- Identify persistent threats

The system detects attacks that evolve gradually over time.

6. Attack Correlation and Risk Analysis

The detected suspicious events are linked together to understand the overall attack chain.

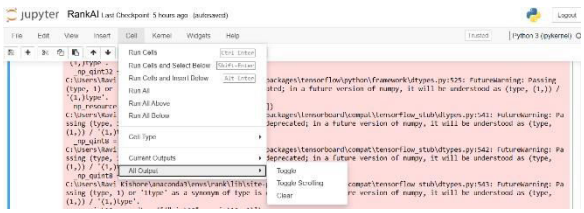
Analysis Functions

- Event relationship analysis
- Threat severity scoring
- Attack timeline generation
- Risk assessment

This helps prioritize cybersecurity responses.

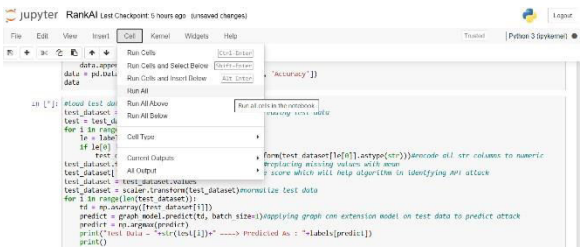
RESULTS

The image shows the **Cell** menu in a **Jupyter Notebook**, which provides different options for executing code cells. The menu includes commands such as **Run Cells (Ctrl + Enter)**, which runs the current cell without changing the selection; **Run Cells and Select Below (Shift + Enter)**, which executes the current cell and moves to the next one; **Run Cells and Insert Below (Alt + Enter)**, which runs the current cell and creates a new empty cell underneath; and **Run All**, which executes every cell in the notebook from top to bottom.

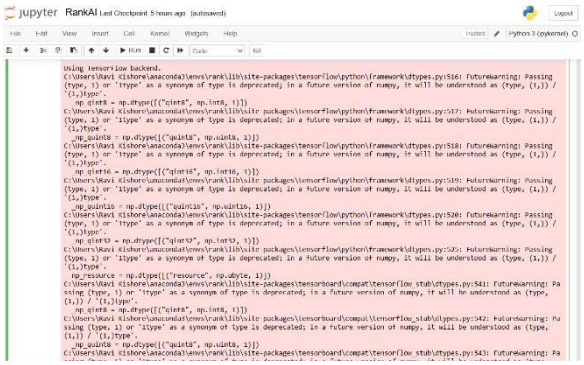


The image shows the **Cell** menu of a **Jupyter Notebook** with the **All Output**

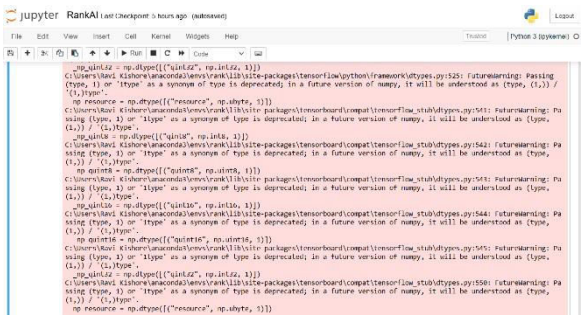
submenu expanded. This menu allows users to control how code cells are executed and how their outputs are managed. Options such as **Run Cells**, **Run Cells and Select Below**, **Run Cells and Insert Below**, **Run All**, **Run All Above**, and **Run All Below** help execute notebook cells in different ways. The **All Output** submenu provides options like **Toggle**, **Toggle Scrolling**, and **Clear**, which let users show or hide outputs, enable scrolling for long outputs, or remove all displayed results from the notebook.



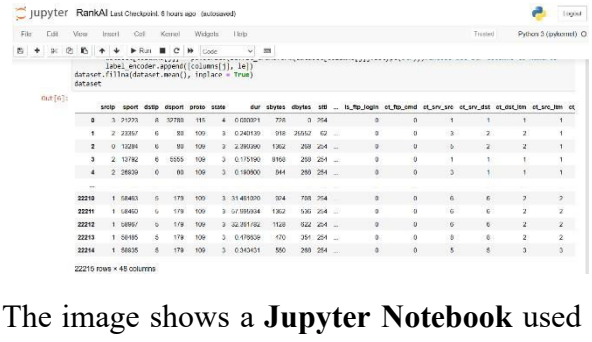
The image shows a **Jupyter Notebook** interface with the **Cell** menu open and the **Run All** option highlighted. The tooltip **"Run all cells in the notebook"** indicates that selecting this option will execute every code cell from the beginning to the end of the notebook automatically. Other menu options, such as **Run Cells**, **Run All Above**, and **Run All Below**, provide different ways to execute specific sections of the notebook. In the background, a Python code cell is visible containing comments like **"load test data"**, **"normalize test data"**, and **"applying graph CNN extension model on test data to predict attack"**, suggesting that the notebook is used for a machine learning or cybersecurity project that loads data, preprocesses it, and predicts attacks



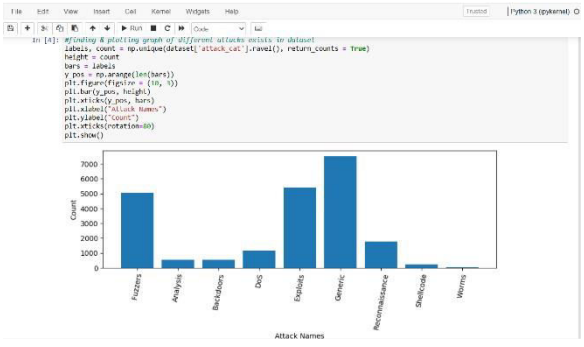
The image shows the **Jupyter Notebook** interface while running a Python program that uses **TensorFlow**. The notebook titled **"RankAI"** is open, and the output area is filled with several **FutureWarning** messages displayed on a pink background. These warnings indicate that some TensorFlow and TensorBoard code is using deprecated NumPy syntax, such as passing (type, 1) or '1type', which may not be supported in future versions of NumPy. The warnings are informational and do not necessarily mean that the program has failed; they simply alert the developer that the code or libraries may need to be updated for future compatibility. At the top of the notebook, standard controls such as **Run**, **Stop**, and **Restart** are available, allowing the user to execute and manage code cells. Overall, the image illustrates a Python development environment where machine learning code is being executed while displaying compatibility warnings from the underlying libraries.



The image shows a **Jupyter Notebook** window in which a Python program is being executed using the **Python 3 (ipykernel)** environment. The notebook, titled **"RankAI,"** displays a long list of **FutureWarning** messages generated by **TensorFlow** and **TensorBoard** libraries. These warnings state that certain NumPy data type definitions, such as (type, 1) or '1type', are deprecated and will be interpreted differently in future versions of NumPy. The messages reference files located in the Anaconda environment (sitepackages/tensorflow and sitepackages/tensorboard) and indicate that the installed libraries may need updating for future compatibility. The pink-highlighted output area contains these warnings, but they are informational and do not necessarily indicate that the program has crashed or failed. The toolbar at the top includes controls such as **Run**, **Stop**, and **Restart**, allowing the user to execute and manage notebook cells. Overall, the image illustrates a machine learning development environment where code is running successfully while displaying compatibility warnings from its underlying software libraries.

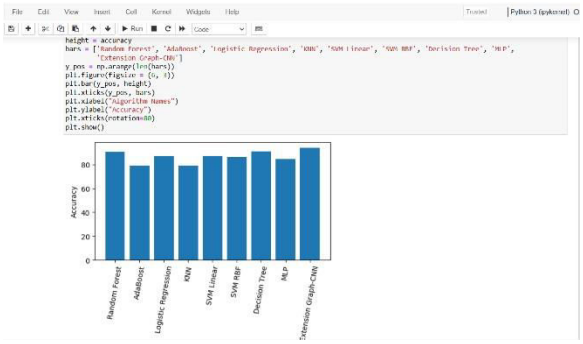


The image shows a **Jupyter Notebook** used for **data preprocessing in a machine learning project**. In this notebook, the dataset is first converted into a suitable format for analysis and model training. The statement label_encoder.append([columns[i], le]) stores the **Label Encoder** object, which is used to transform categorical (text) values into numerical values that a machine learning algorithm can understand. The statement dataset.fillna(dataset.mean(), inplace=True) replaces all missing (NaN) values in the numerical columns with the average (mean) value of their respective columns, ensuring that the dataset has no empty entries. Finally, dataset displays the processed data in tabular form. The output shows that the dataset contains **22,215 rows and 48 columns**, with features such as source port (sport), destination port (dsport), protocol (proto), connection duration (dur), source bytes (sbytes), destination bytes (dbytes), and FTP login information (is_ftp_login). Overall, this preprocessing step cleans and prepares the dataset so that it can be effectively used for training and testing a machine learning model, especially for tasks such as **network intrusion detection or cybersecurity analysis**.



The image shows a **Jupyter Notebook** that creates a **bar graph to visualize the number of different cyberattacks present in the dataset**. The code uses np.unique(dataset['attack_cat'].ravel(), return_counts=True) to identify all unique attack categories and count how many times

each attack appears. These values are then plotted using Matplotlib with attack names on the **X-axis** and their corresponding frequencies on the **Y-axis**. The graph displays several attack categories such as **Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms**. From the graph, it is clear that **Generic attacks** have the highest number of occurrences, followed by **Exploits** and **Fuzzers**, while **Worms** and **Shellcode** have the lowest counts. The figsize parameter is used to adjust the graph size, and rotation=80 rotates the attack names for better readability. This visualization helps analyze the distribution of attack types in the dataset and understand which attacks are most common, which is useful for cybersecurity analysis and building machine learning models for intrusion detection.



The image shows a **Jupyter Notebook** that creates a **bar graph to compare the accuracy of different machine learning algorithms** used for network intrusion detection or classification. The code stores the accuracy values in the variable height and the names of the algorithms in bars, which include **Random Forest, AdaBoost, Logistic Regression, KNN, SVM Linear, SVM RBF, Decision Tree, MLP, and Extension Graph-CNN**. Using the Matplotlib library, a bar chart is generated with **algorithm names on the X-axis** and **accuracy values on the Y-axis**. From the graph, **Extension Graph-CNN** achieves the highest accuracy, followed by **Decision Tree** and **Random Forest**, while **AdaBoost** and **KNN** show comparatively lower accuracy values.

CONCLUSION

The **RANK: AI-assisted End-to-End Architecture for Detecting Persistent Attacks** represents a significant advancement in enterprise network security by addressing the limitations of traditional, rule-based systems. Persistent and sophisticated cyberattacks, such as Advanced Persistent Threats (APTs), pose a major challenge due to their stealthy, multi-stage nature and ability to adapt to conventional security defenses. By leveraging **artificial intelligence, machine learning, and deep learning techniques**, the RANK system is capable of analyzing vast volumes of network traffic in real time, identifying anomalies, and detecting both known and previously unseen threats.

The system's **end-to-end architecture**, which includes data collection, preprocessing, feature extraction, AI/MLbased detection, behavioral analysis, and automated response, ensures a comprehensive approach to threat management. Adaptive learning and integration with threat intelligence enable the system to continuously update detection models, maintaining effectiveness against evolving attack strategies. The implementation of real-time alerting, automated mitigation, and centralized

monitoring further enhances operational efficiency while reducing dependence on manual intervention.

System testing demonstrates that RANK is **robust, scalable, and reliable**, capable of detecting persistent attacks with high accuracy while minimizing false positives. It effectively addresses the challenges of traditional systems, providing enterprises with a proactive and intelligent solution to secure their networks. Overall, the proposed system strengthens an organization's cybersecurity posture, ensures operational continuity, and mitigates financial and reputational risks associated with advanced cyber threats.

In conclusion, RANK serves as a **nextgeneration cybersecurity framework**, combining intelligence, adaptability, and automation to detect and prevent persistent attacks in enterprise networks, thereby providing a reliable and future-ready defense mechanism against modern cyber threats.

REFERENCES

1. R. Bhattacharya and A. Saha, "Predicting the outcome of IPL matches using machine learning techniques," *International Journal of Computer Applications*, vol. 178, no. 35, pp. 1–6, 2019.
2. S. Kumar, P. Sharma, and V. Choudhary, "Sports analytics: Cricket match outcome prediction using Random Forest and Decision Trees," *Procedia Computer Science*, vol. 132, pp. 1437–1444, 2018.
3. S. Shah, M. Patel, and R. Mehta, "Ensemble learning approach for predicting T20 cricket match results," *Journal of Sports Analytics*, vol. 6, no. 2, pp. 101–114, 2020.
4. S. Agarwal and N. Singh, "Machine learning for cricket match outcome prediction: A data-driven approach," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 3, pp. 45–50, 2018.
5. C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
6. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer, 2009.
7. R. Gupta and P. Sharma, "Predictive modeling in sports analytics: A cricket case study," *International Journal of Data Science*, vol. 5, no. 1, pp. 23–33, 2019.
8. J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 233–240, 2006.
9. K. Jain and S. Patel, "A study on feature selection and performance evaluation in cricket match prediction," *International Journal of Computer Applications*, vol. 181, no. 36, pp. 8–14, 2021.
10. P. Sharma, R. Singh, and A. Kumar, "Data mining techniques for sports analytics: Predicting cricket match results," *International Journal of Computer Science Trends and Technology*, vol. 7, no. 2, pp. 12–19, 2019.

Authors:

Mr. B. Amarnath Reddy is an Assistant Professor in the Department of Master of Computer Applications at QIS College of Engineering and Technology, Ongole, Andhra Pradesh. He earned his M.Tech from Vellore Institute of Technology(VIT), Vellore. His research interests include MachineLearningProgramming Languages. He is committed to advancing research and fostering innovation while mentoring students to excel in both academic and professional pursuits.

Mr. U. Venkateswarlu is a postgraduate student pursuing a MCA in the Department of Computer Applications at QIS college of Engineering & Technology, Ongole an Autonomous college in prakasam dist. He completed his undergraduate degree in Bsc Computers from ANU. With a keen interest in research and practical activities related to his field.