**INTERNSHIP REPORT**

**On**

**AI/ML FOR NETWORKING**

**BY**

|  |  |
| --- | --- |
| **Reg.no** | **Name** |
| BU22CSEN0101001 | Anthala Anitha |
| BU22CSEN0101754 | Kanasani Bhavana |
| BU22CSEN01001592 | Saketh Kumar Devatha |

**Intel**

**(Duration: 05-15, 2025 to 07-12, 2025)**

A logo with a bird in the center

AI-generated content may be incorrect.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

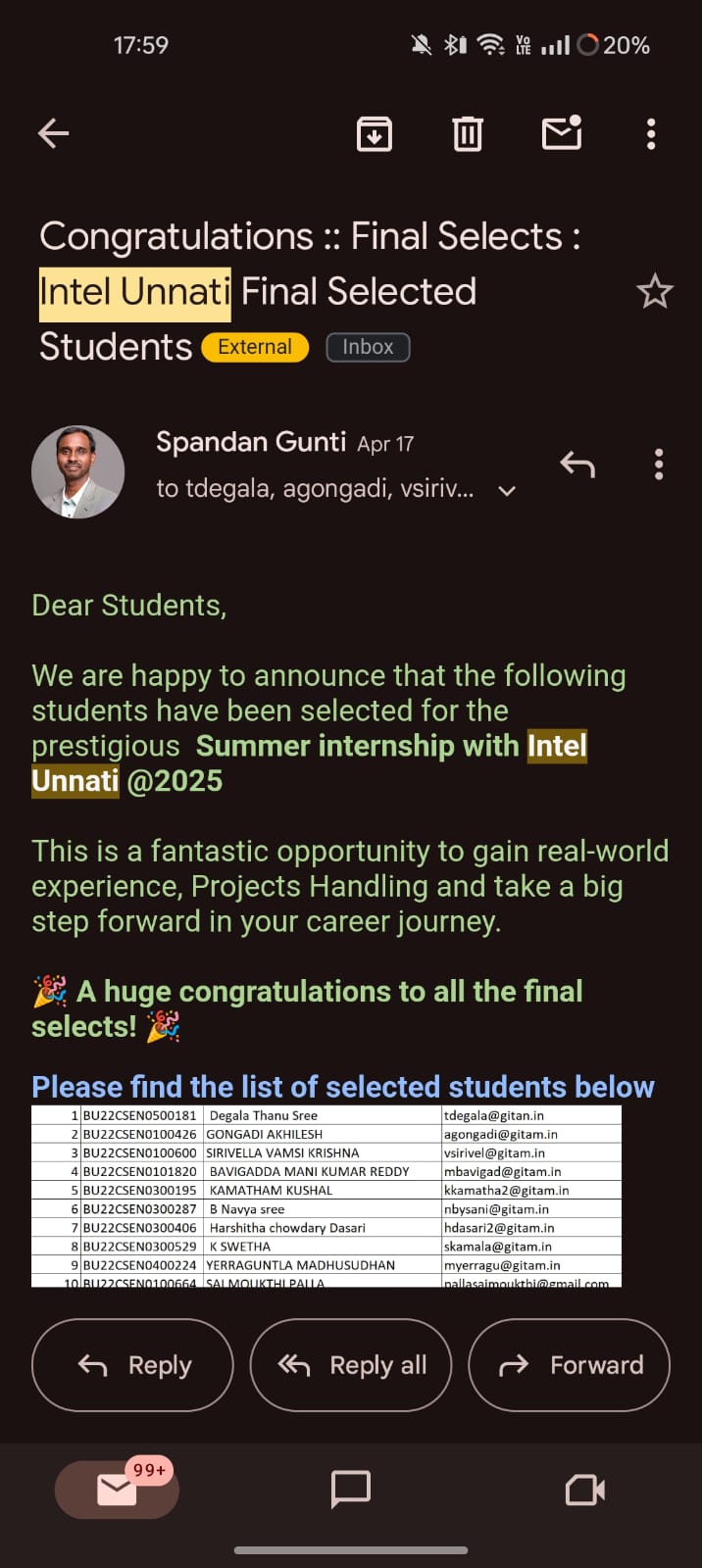
**Gandhi Institute of Technology and Management**

**(DEEMED TO BE A UNIVERSITY)**

**BENGALURU, KARNATAKA, INDIA**

**SESSION:2020-2024**

**Acceptance/Offer Letter from Industry(optional)**



**Contents**

|  |  |
| --- | --- |
| **Title** | **Pg.No** |
| **Abstract** | **7** |
| **Introduction** | **8 -11** |
| **Team** | **11-12** |
| **Dataset Description** | **12-13** |
| **Methodology** | **14-15** |
| **Results and Discussion** | **15-17** |
| **Conclusion** | **17-18** |
| **Appendix** | **18-20** |
| **References** | **21** |

**Abstract**

This project leverages Artificial Intelligence (AI) and Machine Learning (ML) to enhance modern network security by automating the detection of malicious traffic in real time. As cyber threats grow more complex and encrypted traffic increases, traditional rule-based methods often fall short. To address this, we developed an intelligent solution that begins by capturing live network traffic using Wireshark, converting the raw packet data (.pcap) into structured .csv files via CICFlowMeter to extract flow-based features. Using Python and scikit-learn, we trained an ML model to classify traffic as benign or malicious, incorporating SQL Injection (SQLi) payloads to improve detection accuracy. The model achieved over 97% accuracy with minimal false positives and negatives, demonstrating its robustness. For practical deployment, we integrated the model into a Streamlit-based web application, allowing users to input URLs or SQL queries for instant threat analysis. Finally, the app was made publicly accessible through N grok, enabling real-world testing. This project proves that AI/ML can significantly outperform traditional methods by providing a scalable, adaptive, and efficient approach to cybersecurity reducing manual effort, improving detection rates, and staying effective even against encrypted threats.

**1. INTRODUCTION**

1.1 Problem Description

Modern computer networks are experiencing exponential growth in traffic volume, diversity, and complexity due to the increasing number of connected devices, cloud services, and remote access environments. Alongside this growth, cyber threats such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and advanced persistent threats (APTs) have become more sophisticated, frequent, and difficult to detect using traditional techniques.Conventional rule-based network security systems like firewalls, intrusion detection systems (IDS), and deep packet inspection (DPI) rely on predefined signatures and heuristics. These approaches are limited when dealing with encrypted traffic, zero-day vulnerabilities, and polymorphic attacks, often resulting in missed detections or high false positives.

Furthermore, manually analyzing and classifying network traffic is time-consuming and inefficient, especially in high-speed and large-scale networks. There is a growing need for intelligent, automated systems that can adaptively analyze traffic patterns, detect anomalies, and classify threats in real time.To address these challenges, the integration of Artificial Intelligence (AI) and Machine Learning (ML) into networking offers a promising solution. By learning from historical and real-time traffic data, AI/ML models can automatically identify malicious behavior, predict potential attacks, and enhance the overall security posture of the network without human intervention.This project aims to design and implement an AI/ML-based system capable of detecting and classifying network traffic, particularly focusing on malicious SQL injection attacks, through data preprocessing, model training, and real-time prediction in a scalable and user-friendly environment.

1.2 Project Objectives

The main objective of this project is to enhance network security through the application of Artificial Intelligence (AI) and Machine Learning (ML) techniques for intelligent, real-time threat detection and classification.

The specific goals include:

1. Automated Traffic Classification:  
   Develop a machine learning model that can accurately classify network traffic as either benign or malicious, with a special focus on SQL injection attacks.
2. Data Capture and Feature Extraction:  
   Use Wireshark to collect real-time network traffic and convert it into structured datasets using CICFlowMeter for analysis and training.
3. Effective Preprocessing Pipeline:  
   Design a robust preprocessing system that handles obfuscated and complex payloads by preserving important syntactic patterns (e.g., stacked queries, inline comments).
4. Model Training and Optimization:  
   Train and evaluate multiple ML algorithms (e.g., Logistic Regression, Naive Bayes) using real and simulated data to select the most accurate and lightweight model.
5. Real-Time Prediction System:  
   Build an interactive Streamlit-based web application that allows users to input traffic/query samples and instantly receive classification results along with confidence scores.
6. High Accuracy and Low False Positives:  
   Aim to minimize false alarms by fine-tuning model parameters and balancing the dataset for better generalization.
7. Scalability and Usability:  
   Ensure the system is lightweight, fast, and capable of handling high-volume traffic, with a user-friendly interface suitable for security analysts and researchers.
8. Team Collaboration and Documentation:  
   Maintain detailed documentation (README, code, results) and ensure smooth division of work and knowledge sharing among team members.

1.3 Motivation

With the rise in digital communication, cloud computing, and connected devices, modern networks have become a primary target for cyberattacks. Threats like SQL Injection (SQLi), Cross-Site Scripting (XSS), and other forms of malicious traffic can compromise sensitive data, disrupt services, and cause significant financial and reputational damage.

Traditional network security tools—such as rule-based firewalls and signature-based intrusion detection systems—struggle to cope with the dynamic and encrypted nature of today’s traffic. They often fail to detect new, obfuscated, or zero-day attacks, and their performance degrades significantly in high-traffic environments. Manual monitoring is not scalable, time-consuming, and often prone to human error.

This growing gap between traditional security capabilities and modern attack techniques motivated us to explore a smarter solution. By leveraging the power of Artificial Intelligence (AI) and Machine Learning (ML), we can analyze traffic patterns, detect anomalies, and classify malicious activities in real time—without relying on static signatures.

This project is driven by the vision of building a lightweight, intelligent, and scalable system that not only improves the speed and accuracy of threat detection but also reduces the burden on cybersecurity professionals. Our goal is to make network security more proactive, adaptive, and automated, ensuring safer and more resilient infrastructures in the age of digital transformation.

**2. TEAM**

2.1 Team Contribution

* A. ANITHA
  + Captured real-time network traffic using Wireshark.
  + Downloaded and curated relevant SQL injection datasets.
  + Participated in manual testing of the detection model using edge-case payloads.
  + Contributed to the final project documentation and coordinated overall report writing.
* BHAVANA
  + Developed the initial machine learning model using Google Colab, focusing on classifying payloads as malicious or safe.
  + Preprocessed the early dataset and evaluated baseline models like logistic regression and decision trees.
  + Helped validate model predictions and assess accuracy metrics during training.
* SAKETH KUMAR
  + Led the refinement and optimization of the model, especially for logic-based SQLi edge cases.
  + Designed and built the preprocessing pipeline, including TF-IDF vectorization and syntax preservation for complex payloads.
  + Implemented the complete Streamlit web application with real-time detection and batch query support.
  + Authored the README documentation, ran detailed tests, and managed internal task coordination.

**3. Dataset Description**

3.1 Source and Collection Tools

The dataset used in this project was a combination of real-time captured traffic and manually curated SQL injection payloads. The following tools were used for data collection and preparation:

* Wireshark: Used to capture live network traffic, including HTTP requests, simulated attacks, and safe browsing activity.
* Cyclometer: Transformed the .pcap files from Wireshark into structured .csv format with flow-based features suitable for ML training.
* Public SQLi Payload Repositories: Attack payloads were sourced from OWASP, GitHub repositories, and existing security datasets.
* Custom Payload Scripts: For creating logic-based SQLi, stacked queries, and bypass scenarios not available in standard datasets.

3.2 Features Extracted

From the converted .pcap files, CICFlowMeter extracted detailed flow-level features for each network session. These features were used to build the ML model. Some of the key features include:

* Flow ID – Unique identifier for each network flow
* Source IP / Destination IP – Identifies sender and receiver
* Source Port / Destination Port
* Protocol – Typically TCP/UDP/HTTP
* Flow Duration
* Total Forward/Backward Packets
* Packet Length Stats (Min, Max, Mean)
* Flow Bytes/s and Packets/s
* Header Flags (SYN, ACK, etc.)
* Payload Content – Extracted to analyze injected queries
* Label – 0 = Safe, 1 = Malicious

3.3 SQL Injection Types and Payloads

To ensure model robustness, the dataset covered a wide variety of SQL injection (SQLi) attack types, including simple, advanced, and obfuscated payloads:

|  |  |
| --- | --- |
| SQLi Type | Example |
| **Union-Based** | ' UNION SELECT username, password FROM users |
| **Boolean-Based (Blind)** | ' OR 1=1 --, ' AND 1=0 |
| **Time-Based (Blind)** | '; IF(1=1) WAITFOR DELAY '00:00:05' |
| **Error-Based** | ' OR 1=CONVERT(int, (SELECT @@version)) |
| Stacked Queries | '; DROP TABLE users; |
| Authentication Bypass | ' OR 'a'='a in login inputs |
| Obfuscated Payloads | Encoded characters (%27, %20), inline/block comments (/\*\*/, --) |

**3.4 Data Augmentation Techniques**

Since real attack data is often limited, **data augmentation** was applied to ensure balance and diversity in the training set:

* **Manual Injection Variants:** Created different versions of known SQLi queries using alternate keywords, casing, spacing, or comments.
* **Encoding Techniques:** Included payloads with URL encoding, Unicode, and hex formats.
* **Randomized Benign Traffic:** Extracted and labeled normal queries from real web usage to simulate everyday traffic.
* **Class Balancing:** Ensured a 1:1 ratio of safe and malicious queries to avoid training bias.
* **Shuffling and Resampling:** Used to prevent overfitting and introduce variance in training iterations.

These techniques made the dataset more resilient and helped the model generalize better to unseen or disguised attacks.

**4. METHODOLOGY**

4.1 Data Capture using Wireshark

The first step in the process involved capturing real-world network traffic using **Wireshark**, an open-source packet analyzer. The traffic captured included:

* Benign HTTP/HTTPS requests from safe browsing activity.
* Simulated SQL Injection (SQLi) attacks entered through web forms, query strings, and URLs.
* Login and search inputs used to generate realistic request payloads.

This resulted in a .pcap (packet capture) file, which stores all packets in sequence for later analysis.

4.2 Flow Conversion using CICFlowMeter

The .pcap file generated by Wireshark was then processed using CICFlowMeter, a tool that converts raw packet data into bidirectional flow-based features suitable for machine learning. This process involves:

* Aggregating packets into flows based on IP addresses and ports.
* Extracting over 80 features such as flow duration, byte count, packet size, header flags, and protocol types.
* Exporting the results into .csv format for further processing.

4.3 Data Preprocessing and Labeling

The .csv file obtained from CICFlowMeter was cleaned and labeled to prepare it for training:

* Null/Empty Fields: Removed or filled using default values.
* Payload Content: Extracted and tokenized from HTTP and query fields.
* Labeling:
  + 0 for benign traffic
  + 1 for SQL injection payloads
* Syntax Preservation: Important SQL elements like quotes, comments (--, #), semicolons, and obfuscations were preserved for accurate classification.
* Normalization: Case normalization and removal of excessive white spaces while retaining meaningful tokens.

4.4 Model Selection and Training

Multiple machine learning models were tested to find the best fit for SQLi detection:

* Logistic Regression – baseline model for binary classification.
* Decision Tree – interpretable but less robust with textual data.
* Naive Bayes (Multinomial) – selected for final deployment due to its high accuracy and performance with TF-IDF vectorized inputs.

Key steps:

* Dataset split into training and testing sets (e.g., 80:20).
* Model trained using supervised learning with cross-validation.
* Final model saved using joblib for deployment.

4.5 Pipeline Design and TF-IDF Optimization

To improve detection of textual attacks, the TF-IDF (Term Frequency – Inverse Document Frequency) vectorizer was fine-tuned:

* N-gram Range: (1,2) to capture both single words and bigrams.
* Minimum Document Frequency (min\_df): Adjusted to filter out rare/noisy tokens.
* Maximum Features: Set to limit dimensionality and reduce overfitting.
* Custom Tokenizer: Used to handle punctuation and SQL-specific patterns.

4.6 Streamlit Web App Development

The final model was integrated into a real-time detection system using Streamlit, a Python library for building lightweight web applications:

Interface: Accepts user input (single or batch SQL queries).

* Live Prediction: Displays classification as Safe or Malicious with probability/confidence score.
* Additional Features:
  + Payload normalization
  + Error handling for empty or malformed inputs
  + Simple and intuitive layout for non-technical users
* Deployment: Hosted via ngrok for public accessibility and demo purposes.

This web app serves as a proof-of-concept for real-time, client-facing ML-based intrusion detection.

**5. Results and Discussion**

5.1 Model Performance Metrics

After training the final Multinomial Naive Bayes model, its performance was evaluated using standard classification metrics. The dataset was split into training and testing sets (80:20), and results were computed on the test set.

| **Metric** | **Safe (0)** | **Malicious (1)** |
| --- | --- | --- |
| **Precision** | 0.96 | 0.99 |
| **Recall** | 0.99 | 0.93 |
| **F1-Score** | 0.98 | 0.96 |
| **Accuracy** | - | **97% (overall)** |

🔍 **Interpretation:**  
The model demonstrated strong performance across all metrics, with high precision (minimizing false positives) and recall (minimizing false negatives), making it suitable for practical deployment.

5.2 Comparative Analysis of Algorithms

To determine the most suitable model, multiple classifiers were trained and evaluated under identical conditions.

| **Model** | **Accuracy** | **Training Time** | **Inference Speed** | **Remarks** |
| --- | --- | --- | --- | --- |
| Logistic Regression | 93% | Fast | Fast | Decent baseline, struggled with obfuscations |
| Decision Tree | 90% | Medium | Medium | Overfitted on training data |
| Multinomial Naive Bayes | **97%** | Very Fast | Very Fast | Best performance with TF-IDF text input |

**Conclusion:** Naive Bayes offered the best balance of speed, accuracy, and

generalization for short, structured queries like SQL injections.

5.3 Observations on Dataset Behavior

* **Balanced Dataset:** Equal number of safe and malicious queries helped prevent bias during training.
* **Augmented Inputs:** Manually designed SQLi payloads, especially those with inline comments, stacking, and encoding, greatly improved model learning.
* **Payload Sensitivity:** TF-IDF vectorization helped the model detect slight variations in syntax, such as ' OR '1'='1 --, UNION SELECT, and DROP TABLE.
  1. Implications for Real-world Use
* **Effective Intrusion Detection:** The trained model can detect a wide variety of SQLi attacks with high confidence.
* **Real-time Deployment:** Lightweight nature of Naive Bayes + Streamlit ensures fast response time, even on modest hardware.
* **Flexible Input Handling:** Model can be easily retrained with additional payload types (e.g., XSS, RCE).
* **Privacy-Friendly:** Works without inspecting encrypted payloads deeply—ideal for privacy-respecting monitoring tools.

This project demonstrated the successful application of machine learning to real-time network threat detection, specifically targeting SQL Injection attacks. By capturing and transforming network traffic into structured flow features, and using a robust preprocessing and TF-IDF pipeline, we trained a lightweight Naive Bayes classifier that achieved 97% accuracy on a balanced dataset.

The project not only proved the effectiveness of AI/ML in detecting malicious traffic but also delivered a practical and scalable tool via a Streamlit web application. This allows real-time classification of queries in an easy-to-use format, bridging the gap between machine learning research and practical network defense.

**6. Conclusion**

**6.1 Summary of Key Findings**

This project successfully demonstrated how **Artificial Intelligence (AI)** and **Machine Learning (ML)** can be leveraged to enhance network security by automatically detecting malicious SQL injection (SQLi) queries in real-time network traffic. The key outcomes include:

* A fully functional ML pipeline from **Wireshark capture → CICFlowMeter → preprocessing → model training → real-time deployment**.
* **Multinomial Naive Bayes** was identified as the most efficient algorithm, achieving an impressive **97% accuracy**.
* The preprocessing pipeline was carefully optimized to preserve SQL-specific syntax and edge-case payload patterns.
* A lightweight **Streamlit web application** was developed for user-friendly, real-time classification of single or batch inputs.
* Manual testing confirmed the system’s robustness against obfuscated, logic-based, and diverse SQLi attack vectors.

**6.2 Limitations**

Despite the project’s success, several limitations were observed:

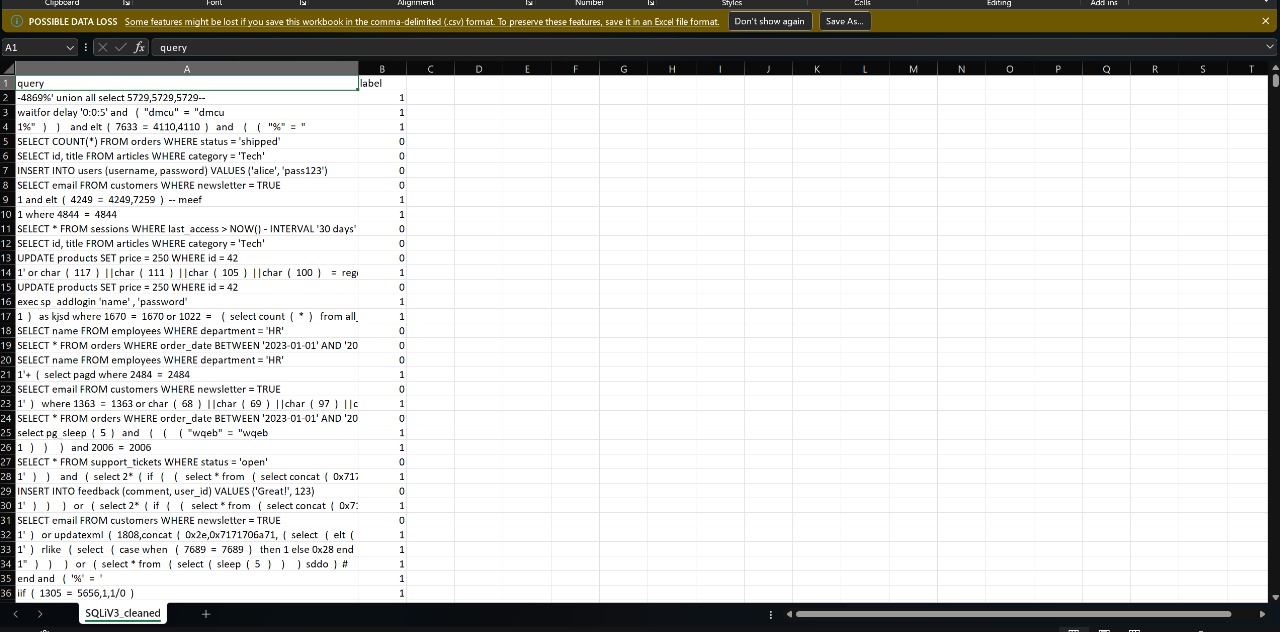
* The focus was **only on SQL Injection attacks**; other threats like XSS, RCE, and CSRF were not included.
* The model is **not trained on encrypted (SSL/TLS) traffic**, limiting its scope for HTTPS-based intrusion detection.
* Performance might degrade in real-world high-throughput networks without additional optimization or deployment architecture.
* The current model is **not explainable**, meaning it does not provide reasons or features behind its predictions.

**6.3 Future Enhancements**

To make the system more robust and adaptable for real-world environments, the following improvements are proposed:

1. **Support for Multiple Attack Types:**  
   Extend the dataset and model to classify other attacks like **XSS**, **Remote Code Execution (RCE)**, and **Cross-Site Request Forgery (CSRF)**.
2. **Encrypted Traffic Analysis:**  
   Implement features to handle **TLS/SSL-encrypted** flows, possibly using metadata or certificate analysis.
3. **Integration with Network Tools:**  
   Deploy the model as part of a **browser plugin**, **firewall**, or **SIEM** (Security Information and Event Management) system.
4. **Explainable AI (XAI):**  
   Integrate tools like **LIME** or **SHAP** to help understand why a particular input was marked malicious.
5. **Cloud & Edge Deployment:**  
   Optimize the model and app for scalable deployment in **cloud environments** or **IoT edge devices**.

**Appendix**

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

**References**

[1] Hussain, A., Hussain, A., Qadri, S., Razzaq, A., Nazir, H., & Ullah, M. S. (2024). Enhancing LAN security by mitigating credential threats via http packet analysis with Wireshark. Journal of Computing & Biomedical Informatics, 6(02), 433-440.

[2] Özekes, S., & Karakoç, E. N. (2019). Makine öğrenmesi yöntemleriyle anormal ağ trafiğinin tespit edilmesi. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 7(1), 566-576.

[3] Salama, M. (2024). Optimization of Regression Models Using Machine Learning: A Comprehensive Study with Scikit-learn. Optimization of Regression Models Using Machine Learning: A Comprehensive Study with Scikit-learn| IUSRJ, 5.

[4] Moscato, R. (2024). Web App Development Made Simple with Streamlit: A web developer's guide to effortless web app development, deployment, and scalability. Packt Publishing Ltd.

[5] Dong, R., Luo, Z., Xue, H., Shao, J., Chen, L., Jin, W., ... & Wang, J. (2025). Development and Validation of an Explainable Machine Learning Model for Warning of Hepatitis E Virus‐Related Acute Liver Failure. Liver International, 45(6), e70129.