Jaypee Institute of Information Technology, Noida

Department of Computer Science & Engineering and IT

Major Project Title

A Hybrid Framework For Spam Classification: Integrating Cryptography And Machine Learning

Enrol. No.	Name of Student
21103318	Vaibhav Sharma

Course Name: Major Project 1

Program: B. Tech. CS&E

7th Sem

2024 - 2025

TABLE OF CONTENTS

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Jaypee Institute of Information Technology

Date: 18/11/2024

Signature:

Name: Vaibhav Sharma

Enrolment No: 21103318

CERTIFICATE

This is to certify that the work titled "**A HYBRID FRAMEWORK FOR SPAM CLASSIFICATION: INTEGRATING CRYPTOGRAPHY AND MACHINE LEARNING**", an interdisciplinary research and development project submitted by "**Vaibhav Sharma**" in partial fulfilment for the award of degree of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	…….
Name of Supervisor	**DR. ANKITA JAISWAL**
Designation	Assistant Professor (Senior Grade)
Date	18/11/2024

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Dr. Ankita Jaiswal under whom we have carried out the project work. Her incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work. We wish to reciprocate in full measure the kindness shown by Dr. Ankita Jaiswal Assistant Professor (Senior Grade) who inspired us with his valuable suggestions in successfully completing the project work. Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Signature of Student: ……..

Name of Student: Vaibhav Sharma

Enrolment Number: 21103318

Date: ……..……

SUMMARY

Our project integrates cryptography and machine learning to create a secure and efficient framework for spam email classification. We preprocess data using encoding detection, tokenization, and advanced cleaning methods, converting text into numerical representations like TF-IDF vectors and word embeddings. The framework employs Logistic Regression and Support Vector Machines (SVM) for classification, enhanced by hyperparameter optimization and evaluation metrics such as precision and recall.

To ensure privacy, we implement Homomorphic Encryption, enabling predictions directly on encrypted data. The system also incorporates Paillier encryption to protect model outputs and weights. Key management strategies and compliance with GDPR are prioritized. By combining predictive modelling with robust privacy-preserving techniques, our project addresses the dual challenges of accuracy and data security in spam detection.

Signature of Student : ……

Name: Vaibhav Sharma

Date :

Signature of Supervisor : ……..……

Name : Dr. Ankita Jaiswal

Date :

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Table	Page No
TFHE	Fast Fully Homomorphic Encryption over the Torus
PPML	Privacy-Preserving Machine Learning
HE	Homomorphic Encryption
BFV	Brakerski-Gentry-Vaikuntanathan Scheme
CKKS	Cheon-Kim-Kim-Song Scheme
LUTs	Look-Up Tables
MPC	Multi-Party Communication
LR	Logistic Regression
SVM	Support Vector Machine
TEE	Trusted Environment Execution
PPML	Privacy-Preserving Machine Learning
SMO	Sequential Minimal Optimization

CHAPTER 1
INTRODUCTION

1.1 GENERAL INTRODUCTION

In the era of data-driven decision-making, ensuring privacy within machine learning models has emerged as a major priority. Large datasets containing sensitive information—such as health records, financial data, and personal communications—are increasingly analysed through machine learning techniques to gain insights and make predictions. However, without appropriate privacy measures, these practices can expose individuals and organizations to significant privacy risks.

Privacy-preserving machine learning (PPML) is a research field focused on maintaining data confidentiality while allowing machine learning computations to occur. Techniques in PPML have evolved to include homomorphic encryption (HE), secure multi-party computation, and differential privacy, each offering a unique approach to secure data processing. Among these, homomorphic encryption stands out as a powerful tool because it enables computations on encrypted data, eliminating the need for decryption and thereby preserving data privacy throughout the entire analysis pipeline.

The Paillier cryptosystem, a well-known form of homomorphic encryption, allows addition operations on encrypted data and is particularly suited for privacy-preserving machine learning. By leveraging Paillier encryption, PPML models can process encrypted data inputs, perform secure computations, and generate encrypted outputs, all without ever exposing the raw data. This makes HE particularly beneficial in industries that handle sensitive data, such as healthcare, finance, and telecommunications.

In binary classification tasks, two commonly employed machine learning models are Logistic Regression (LR) and Support Vector Machines (SVMs). Both algorithms offer unique advantages and, when paired with homomorphic encryption, serve as strong contenders for secure and effective binary classification. Logistic Regression is probability-based, using a logistic function to model the likelihood that an observation belongs to a particular class. Support Vector Machines, on the other hand, are geometric in nature, identifying the hyperplane that best separates two classes in high-dimensional space.

Implementing encrypted versions of LR and SVMs allows these models to operate on sensitive data without revealing it, which is critical in domains like spam detection, fraud prevention, and secure communication systems. In encrypted spam filtering, for example, an encrypted SVM can detect spam messages based on learned patterns, without exposing the actual content of emails to the server or algorithm.

This report delves into the comparative study of homomorphically encrypted Logistic Regression and Support Vector Machine models. We evaluate these models across key metrics, including accuracy, efficiency, memory usage, and processing time, under encrypted conditions. Additionally, we examine the practical feasibility of implementing these encrypted models in real-world applications and the trade-offs involved, such as potential reductions in speed due to encryption overhead. Through this analysis, we aim to illustrate the potential of encrypted machine learning models for secure, large-scale data analysis while preserving the privacy of sensitive information.

1.2 PROBLEM STATEMENT

How do we safeguard the data when it goes through ML models and the predictions made?

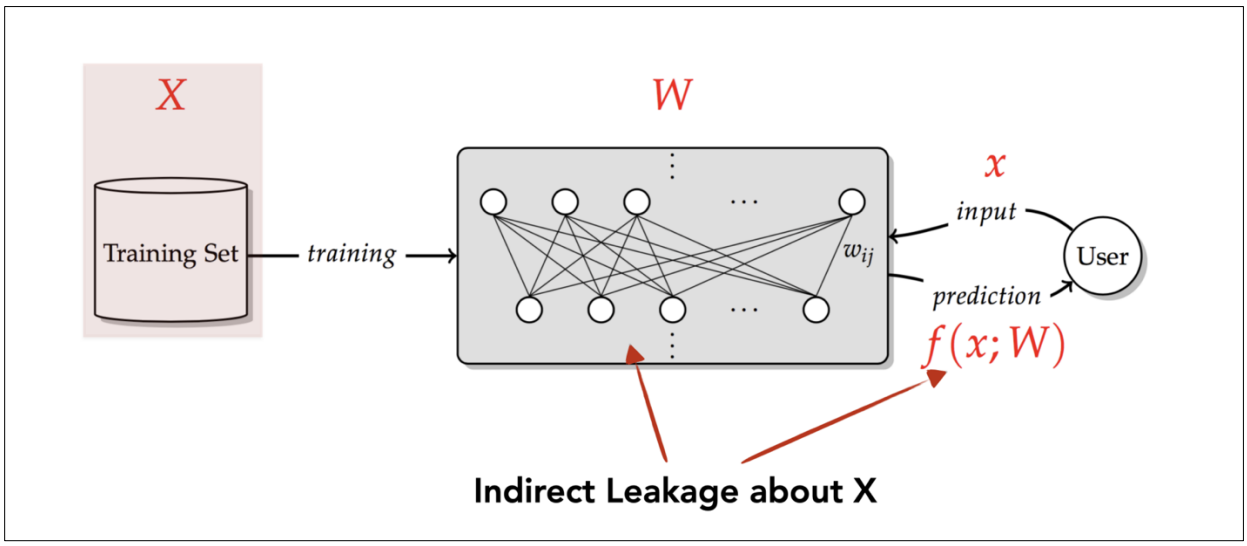

Figure 1.1 The standard workflow for machine learning. Parameters of the model (acquired from confidential training data) and its predictions may unintentionally reveal private information regarding the training data. Such a leakage may occur even when the whole training procedure is kept private.

Increasing crime rates in urban areas necessitate the development of advanced predictive models that can help law enforcement agencies respond proactively. Traditional methods of crime detection often rely on historical data and conventional surveillance, which may fail to capture real-time changes in social behaviour, environmental factors, and other dynamic crime indicators. A new approach involves modelling crime detection through a logical circuit-based system that processes various input signals to assess potential crime risks.

This crime detection model operates by collecting multiple primary input signals, each representing a distinct factor associated with crime likelihood. These input signals may include data such as foot traffic density, time of day, weather conditions, neighbourhood socioeconomic status, historical crime patterns, public sentiment on social media, and police presence. When combined, these inputs create a variety of possible scenarios or patterns, each capable of influencing the final crime risk assessment, represented by the output signals of the circuit.

In this system, the goal is to identify specific patterns among the numerous input combinations that lead to meaningful changes in the output signals. Not all input patterns significantly impact

the detection of potential crimes; some may be irrelevant under certain conditions and needlessly increase the model's computational load. By isolating the relevant input patterns that reliably indicate high crime risk, the model can operate more efficiently, minimizing the processing of redundant data. This approach not only reduces computational complexity but also allows for quicker response times, which is critical in real-time crime prevention efforts.

For example, consider an input combination where high foot traffic at a late hour in a historically high-crime area may produce an output signalling a heightened risk of criminal activity. Conversely, an input pattern with low foot traffic in a well-patrolled area might yield a low-risk output, even if other signals suggest otherwise. By analysing these scenarios, we aim to identify which input factors and combinations are most predictive of criminal activity, thus refining the model’s ability to detect real-time risks effectively.

The challenge lies in accurately defining and prioritizing these patterns to maximize the model’s sensitivity to crime-indicating factors while minimizing false positives. Such optimization will improve the system's ability to pre-emptively detect crime, allowing law enforcement agencies to allocate resources more effectively and enhance public safety. This study will explore various configurations and test cases to determine the most relevant input patterns, providing a framework for an efficient, scalable, and responsive crime detection system in urban settings.

Our primary objective is to build a model that not only predicts crime more accurately but also operates efficiently by filtering out irrelevant input combinations. This process will advance beyond traditional hit-and-trial detection methods, setting a foundation for a data-driven, responsive approach to urban crime prevention. By leveraging logical circuits and input optimization, this model represents a novel intersection of data science and urban safety measures, designed to meet the growing challenges of modern law enforcement.

1.3 SIGNIFICANCE OF THE PROBLEM

Developing an effective crime detection model is of profound importance in today's world, especially as urban areas continue to grow and the dynamics of crime evolve. This approach to crime detection, which integrates logical circuit models with real-time data inputs, holds significant value for public safety and urban planning. By understanding and analysing patterns in crime-related signals, this model offers the potential to proactively predict and prevent crime rather than merely responding to it after the fact.

Crime detection systems that leverage multiple indicators—such as foot traffic, environmental conditions, and historical crime patterns—can provide law enforcement agencies with a more nuanced view of potential crime hotspots. This knowledge enables precise allocation of resources, thereby enhancing efficiency and increasing the impact of crime prevention strategies. Furthermore, by optimizing input patterns to filter out irrelevant data, this model reduces computational load and response times, allowing for faster and more accurate assessments that can be acted upon immediately.

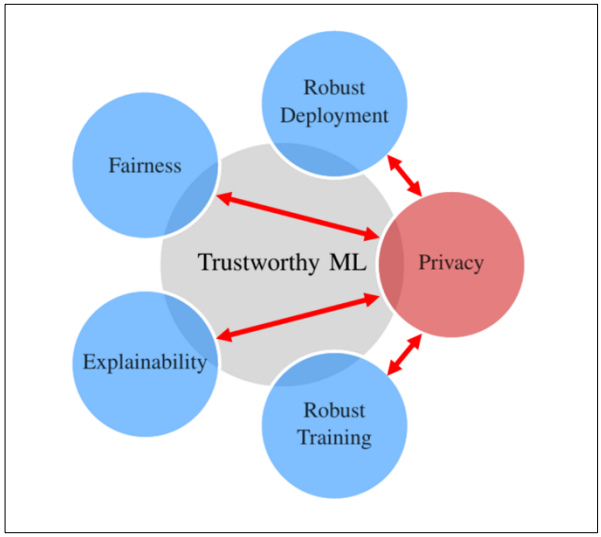

Figure 1.2 Various foundations of reliable machine learning. In this article, we explore the intersection and clashes of data privacy with other facets.

Beyond its direct applications in law enforcement, such a model contributes to the broader field of data-driven decision-making in urban management. The ability to predict criminal activity based on diverse input signals provides urban planners, policymakers, and social organizations with valuable insights into the underlying social and environmental factors contributing to crime. This understanding can guide the design of safer urban environments, reduce the economic impact of crime, and improve residents' quality of life.

In addition, the advancement of computational techniques for crime prediction fosters a framework that can be adapted for other areas of public safety and risk management. Just as gene regulation insights lead to medical breakthroughs, insights gained from this crime detection model can fuel innovation in predictive analytics across various sectors, making cities smarter and safer. This

research paves the way for a data-centric approach to tackling urban challenges, setting a foundation for scalable, adaptable crime prevention technologies in the future.

1.4 EMPIRICAL STUDY

The privacy concerns associated with machine learning models are a significant issue during their training. regarding private and personal information. Studies indicate that certain limitations impacting reliable machines learning poses during the training process can lead to significant new privacy issues. We examine the compromises between data privacy and the other objectives of reliable machine knowledge acquisition (especially equity, resilience, and clarity). Recent studies have explored the application of Homomorphic Encryption (HE) in Privacy-Preserving Machine Learning (PPML). Gentry's work laid the foundation for modern HE schemes, such as TFHE[1], BFV[2], and CKKS[3]. BFV reduces costly bootstrapping operations and allows limited integer ciphertext operations, while CKKS supports floating-point computations. TFHE improves bootstrapping efficiency and enables unlimited bitwise operations on binary data. Each scheme requires specific techniques for non-linear activation functions, such as fast LUT searches for TFHE or polynomial approximations for BFV and CKKS. However, HE schemes still face challenges with large ciphertext sizes and computational complexity.

1.5 BRIEF DESCRIPTION OF THE SOLUTION APPROACH

To securely apply machine learning to sensitive data, this approach leverages homomorphic encryption with Logistic Regression (LR) and Support Vector Machine (SVM) models. This method allows computations to be performed on encrypted data without decrypting it, preserving privacy throughout the entire analysis process.

In this implementation, a homomorphic encryption scheme based on Paillier cryptosystem is used to protect sensitive data, ensuring that encrypted inputs can be processed by both the LR and SVM models. This encryption enables the models to perform necessary mathematical operations (such as addition and scalar multiplication) directly on encrypted data, supporting secure predictions without ever revealing sensitive information.

The primary stages include:

1. **Data Encryption:** Sensitive datasets, such as spam and ham text files, are encrypted using the Paillier cryptosystem. This transformation converts plaintext values into ciphertexts, allowing secure handling and processing by the models.
2. **Model Training and Prediction on Encrypted Data:** The LR and SVM models are modified to handle encrypted inputs. They compute predictions and adjustments based on encrypted weights, processing the data directly without decryption. The LR model utilizes logistic functions, while SVM calculates weighted sums to classify data securely.
3. **Metric Evaluation and Comparison:** After training, both models are evaluated based on various performance metrics, including accuracy, precision, recall, and computational efficiency. This evaluation assesses the trade-offs between privacy preservation and model efficiency, providing insights into practical applications for secure machine learning on sensitive data.

This solution offers a comprehensive, privacy-preserving approach to machine learning by enabling secure data analysis without compromising data confidentiality, applicable to areas such as healthcare and finance.

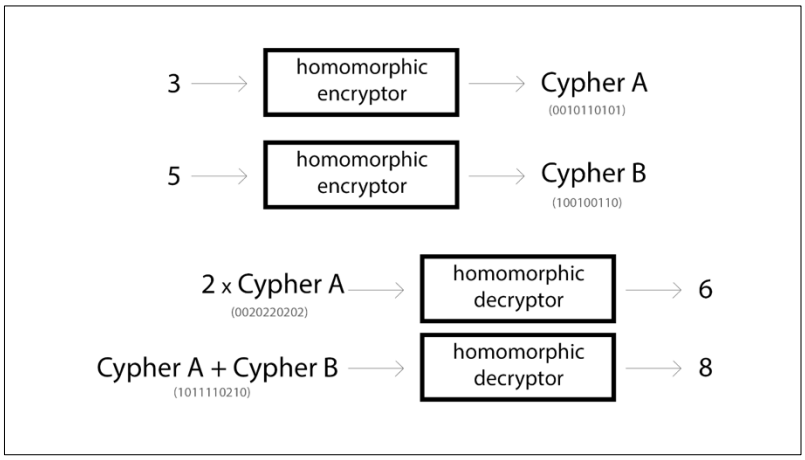

Figure 1.3 Homomorphic encryption enables complex mathematical operations to be performed on encrypted data without compromising the encryption.

1.6 COMPARISON OF EXISTING APPROACHES TO THE PROBLEM FRAMED

We can consider two popular papers around this topic, first being **Data Privacy and Trustworthy Machine Learning** by **Strobel** & **Shokri**[4] and the second being **A Pervasive, Efficient and Private Future: Realizing Privacy-Preserving Machine Learning Through Hybrid Homomorphic Encryption** by **Nguyen** & **Budzys**[5].

Paper 1[4] provides an extensive review of using homomorphic encryption to ensure privacy in machine learning models. Key contributions include:

1. **Encrypted Training and Inference**: The paper discusses how homomorphic encryption allows both training and inference directly on encrypted data, ensuring no plaintext data is exposed, particularly useful for sensitive applications like medical diagnostics and finance.
2. **Applications in Logistic Regression and Neural Networks**: Emphasis is placed on logistic regression due to its interpretability and neural networks for their accuracy. The authors show that homomorphic encryption can perform well with linear models and provide encrypted versions of logistic regression algorithms.
3. **Challenges in Computational Overhead**: While HE protects data privacy, it introduces significant latency due to complex computations on encrypted data. The paper addresses these challenges by proposing optimizations in ciphertext packing, parameter selection, and parallelization.

By using homomorphic encryption, this approach allows organizations to work with sensitive data across sectors without compromising privacy. However, trade-offs in efficiency remain, and real-world deployment would require more optimized cryptographic operations, particularly in scenarios with high-dimensional or real-time data requirements.

Paper 2[5] broadens the scope to encompass multiple PPML methods, focusing on practical applications and evaluating model performance under secure computation constraints. Key contributions include:

1. **Encryption Techniques Across Different ML Algorithms:** This paper explores how various encryption methods, including HE, can be used in algorithms like support vector machines (SVMs), decision trees, and neural networks. Each model is analyzed for performance under encryption constraints.

2. **Performance Benchmarks and Real-World Case Studies:** The paper provides benchmarks of PPML techniques applied to actual data from healthcare and finance, addressing latency and model accuracy in encryption-heavy environments. For instance, in cases with SVMs and decision trees, the impact of encrypted computations on model latency is thoroughly examined, giving insights into where secure computation is most viable.
3. **Comparison of Privacy Methods:** The paper compares HE with other privacy-preserving techniques, such as federated learning and secure multi-party computation (MPC). Federated learning, which keeps data local to its source, contrasts with HE, which operates directly on encrypted data, but often with higher computational costs.

In adapting these methods for secure fault detection and diagnosis, HE appears advantageous for maintaining data privacy in diagnostic models, where sensitive diagnostic or operational data must be kept secure. On the other hand, methods like federated learning and MPC, though less computationally heavy, may compromise on privacy or accuracy in distributed systems without encrypted computations.

CHAPTER 2
LITERATURE SURVEY

2.1 SUMMARY OF RESEARCH PAPER STUDIED

- The paper "**A Pervasive, Efficient and Private Future: Realizing Privacy-Preserving Machine Learning Through Hybrid Homomorphic Encryption**"[5] by "**Khoa Nguyen**" presents PervPPML, a hybrid homomorphic privacy-preserving protocol for machine learning, which consists of two protocols, 3PervPPML and TrustedPervPPML, designed for multi-client models with varying levels of trust. The key findings of this work include the development of resource-friendly PPML protocols using HHE, which provide a foundation for building efficient and privacy-preserving services that transfer expensive HE operations to the cloud. This includes the development of two protocols, 3PervPPML and TrustedPervPPML, which provide comprehensive trust and security against adversarial collaboration. The protocols utilize a trusted execution environment (TEE) to ensure the confidentiality and integrity of the system. The study finds that the 3PervPPML protocol reduces both the communication and computational burden of the user and transfers them to the cloud service provider (CSP), making it suitable for resource-constrained devices. The protocol also achieves comparable accuracy to plaintext inference in integer arithmetic, with a minimal margin of 0.1-0.15% difference. By using HHE, we managed to overcome the main difficulties of PPML application in reallife scenarios, where the majority of data is collected and processed by constraint devices.

- The paper- **"Data Privacy and Trustworthy Machine Learning"** [4] by **"Martin Strobel and Reza Shokri"** discusses the trade-offs between data privacy and other aspects of trustworthy machine learning, including fairness, robustness, and explainability, highlighting the challenges of ensuring trustworthy machine learning while protecting sensitive and personal data. The article highlights the trade-offs between data privacy and other aspects of trustworthy machine learning, including fairness, robustness, and explainability. It shows that constraining a model to ensure fairness or explainability can pose a risk to the privacy of training data. Model explanations can be used to construct successful membership inference attacks, and robustness methods can increase privacy risks. The objective of the article is to discuss the interaction of data privacy with other pillars of trustworthy machine learning, including fairness, robustness, and explainability. The article uses membership inference

attacks as a tool to measure information leakage of private data. These attacks simulate a game between the algorithm and the adversary to measure the amount of information leakage of an algorithm about its training data. The methods used include adversarial training, robust training methods, and data poisoning attacks. The article presents experimental results showing that enforcing group fairness comes at a cost for individual privacy. The privacy risks of the unprivileged group increase when fairness-aware learning is used. The results show that current robustness algorithms can increase privacy risks, and that there is a tradeoff between robustness and privacy.

- The paper- **"A Survey of Deep Learning Architectures for Privacy-Preserving Machine Learning With Fully Homomorphic Encryption"**[6] by "**Robert Podschwadt"** reviews scientific articles on Deep Learning Architectures for Privacy-Preserving Machine Learning (PPML) with Fully Homomorphic Encryption (FHE), analyzing changes to neural network models and architectures to make them compatible with HE and discussing challenges and potential solutions. The key findings of the study include the challenges of HE-based privacy-preserving deep learning, such as computational overhead, usability, and limitations posed by the encryption schemes. The primary challenges in implementing HE PPML include significant computational, memory, and latency overhead, in addition to incompatible operations and limited network depth. The objectives of the study are to review scientific articles on Deep Learning Architectures for Privacy-Preserving Machine Learning (PPML) with Fully Homomorphic Encryption (FHE), analyze changes to neural network models and architectures to make them compatible with HE, and discuss challenges and potential solutions. The study reviews scientific articles and publications in the area of Deep Learning Architectures for Privacy-Preserving Machine Learning (PPML) with Fully Homomorphic Encryption (FHE). The results of the study include the identification of challenges to HE-based privacy-preserving deep learning, such as computational overhead, usability, and limitations posed by the encryption schemes, as well as potential solutions to these challenges. The study proposes evaluation criteria and metrics to compare the effectiveness of different PPML solutions.

- The paper- **"SSVM : A Simple SVM Algorithm"**[7] by **"S.V.N. Vishwanathan, M. Narasimha Murty"** presents a fast iterative algorithm for identifying Support Vectors in a given set of points, which combines the theoretical foundation of incremental algorithms with

the speed of iterative algorithms like SMO and NPA. The algorithm presented in the paper is extremely competitive with other conventional iterative algorithms like SMO and NPA, and it has a memory requirement that scales as O(lSlz) in the average case. The objective of the paper is to present a fast iterative algorithm for identifying Support Vectors in a given set of points, which can overcome the limitations of conventional iterative algorithms. The algorithm uses a greedy approach to pick points for inclusion in the candidate Support Vector set, and a backtracking approach to prune away points that block the addition of a new point. It also uses an optimization-based approach to increment or prune the candidate Support Vector set. The study uses a Gaussian kernel with varying values of C' and reports results in tables I to V. The experiments were conducted on a 800 MHz PI11 machine with 256 MB RAM, running Windows 2000. The algorithm is compared with the NPA algorithm on 5 publicly available datasets, and the results show that the algorithm is extremely competitive with NPA. The Simple SVM algorithm is efficient, intuitive, and fast, and outperforms other iterative algorithms like NPA. The algorithm does not suffer from numerical instabilities and round off errors.

2.2 INTEGRATED SUMMARY OF LITERATURE STUDIED

In these papers, various innovative approaches are explored to address critical challenges in the fields of privacy-preserving machine learning (PPML), trustworthy artificial intelligence, and efficient algorithm design is a critical focus in contemporary research. Khoa Nguyen's study introduces **PervPPML**[5], a hybrid homomorphic encryption-based framework leveraging Trusted Execution Environments (TEEs) to address privacy concerns in multi-client models. This framework, comprising the **3PervPPML** and **TrustedPervPPML** protocols, offloads computational and communication burdens to cloud service providers, making it suitable for resource-constrained devices while maintaining accuracy with only a marginal deviation from plaintext inference. The protocols' ability to balance efficiency, security, and usability demonstrates the potential of hybrid encryption in real-world PPML applications. Complementing this, **Robert Podschwadt's** survey explores the challenges and innovations in adapting deep learning architectures for Fully Homomorphic Encryption (FHE). Despite the computational and usability limitations of FHE, such as high latency and constrained network depth, the study highlights advancements in encryption-compatible architectures and evaluation metrics that aim to make FHE-based PPML more viable.

While encryption offers promising avenues for preserving data privacy, Martin Strobel and Reza Shokri emphasize the trade-offs between privacy and other pillars of trustworthy machine learning, including fairness, robustness, and explainability. Their work demonstrates that enhancing fairness can unintentionally increase privacy risks for underprivileged groups, and methods for robustness or explainability may exacerbate information leakage through attacks like membership inference. These findings underscore the intricate balancing act required to design models that are simultaneously private and trustworthy. Addressing a different but related challenge, **S.V.N. Vishwanathan and M. Narasimha Murty's** development of the **Simple Support Vector Machine (SSVM)** algorithm introduces a computationally efficient method for identifying support vectors. By combining a greedy selection mechanism with backtracking for pruning, SSVM achieves superior performance compared to traditional algorithms like NPA, offering a robust and resource-efficient solution for practical machine learning tasks.

Together, these papers illuminate the diverse strategies and challenges in advancing machine learning systems that are efficient, secure, and equitable. By addressing encryption overhead, privacy-trust trade-offs, and algorithm optimization, they provide a cohesive foundation for developing robust AI systems capable of meeting the demands of modern data privacy and computational efficiency.

CHAPTER 3
REQUIREMENT ANALYSIS AND SOLUTIONS APPROACH

3.1 OVERALL DESCRIPTION OF THE PROJECT

a) Product Perspective:

This project aims to leverage homomorphic encryption for privacy-preserving machine learning (PPML) to ensure secure data processing in fault detection systems. Unlike traditional approaches where raw data must be exposed for computation, this solution operates directly on encrypted data, maintaining confidentiality. It integrates logistic regression and SVM models with encryption techniques, allowing fault detection and diagnosis without compromising sensitive information. The system aligns with industry needs for secure diagnostics in healthcare, finance, and critical infrastructures. It builds on existing PPML frameworks while optimizing for efficiency and accuracy.

b) Product Functions:

The solution provides end-to-end encrypted fault detection and diagnosis capabilities. Key functions include data preprocessing with encryption, training logistic regression and SVM models on encrypted data, and evaluating models for accuracy and fault detection efficiency. It supports input pattern generation for detecting specific faults, encrypted prediction to identify errors, and result visualization for actionable insights. These features collectively enable privacy-preserving diagnostics, ensuring sensitive data is never exposed during computation while maintaining model interpretability and usability.

c) User Characteristics:

The intended users are data scientists, cybersecurity experts, and professionals in regulated industries like healthcare and finance. They are expected to have moderate expertise in machine learning and cryptographic principles. Users prioritize data privacy, accuracy, and the ability to perform real-time fault detection. The system offers a user-friendly interface for model evaluation and visualization, accommodating varying levels of technical proficiency while ensuring robust encryption techniques remain abstracted.

d) Constraints:

The project must address significant computational overhead due to homomorphic encryption, limiting the feasibility of real-time applications in high-throughput systems. It must operate within resource constraints such as memory and processing power, particularly for large datasets. Legal and regulatory compliance for handling sensitive data must be upheld. Additionally, scalability is limited by encryption complexities, requiring efficient algorithmic optimizations to balance security and speed.

e) Assumptions and Dependencies:

The project assumes access to structured datasets for fault detection and diagnosis, ensuring data integrity and preprocessing accuracy. It relies on the robustness of cryptographic libraries (e.g., libnum) for implementing homomorphic encryption. The deployment environment assumes sufficient computational resources to handle the encryption overhead. Dependencies include compliance with data privacy regulations (e.g., GDPR) and the availability of interoperable tools for secure ML model deployment.

f) Apportioning of Requirements:

Core requirements like encryption-based training, fault detection accuracy, and privacy preservation are prioritized. Secondary requirements include system optimizations for speed and memory usage, as well as enhanced model interpretability. Future iterations may focus on integrating additional algorithms like deep learning models with encryption or expanding fault detection capabilities to multi-class systems. Resource-intensive features, such as real-time encrypted diagnostics, are deferred to later phases based on computational feasibility.

3.2 REQUIREMENT ANALYSIS

Functional Requirements:

1. **Model Design and Implementation:**
   - The system must accurately implement two machine learning algorithms: Logistic Regression and Support Vector Machine (SVM).
   - It should include functionalities for encryption and decryption using the Paillier cryptosystem to ensure data confidentiality.
   - Logistic regression should compute predictions based on linear decision boundaries, while SVM should determine optimal hyperplanes for separating data.

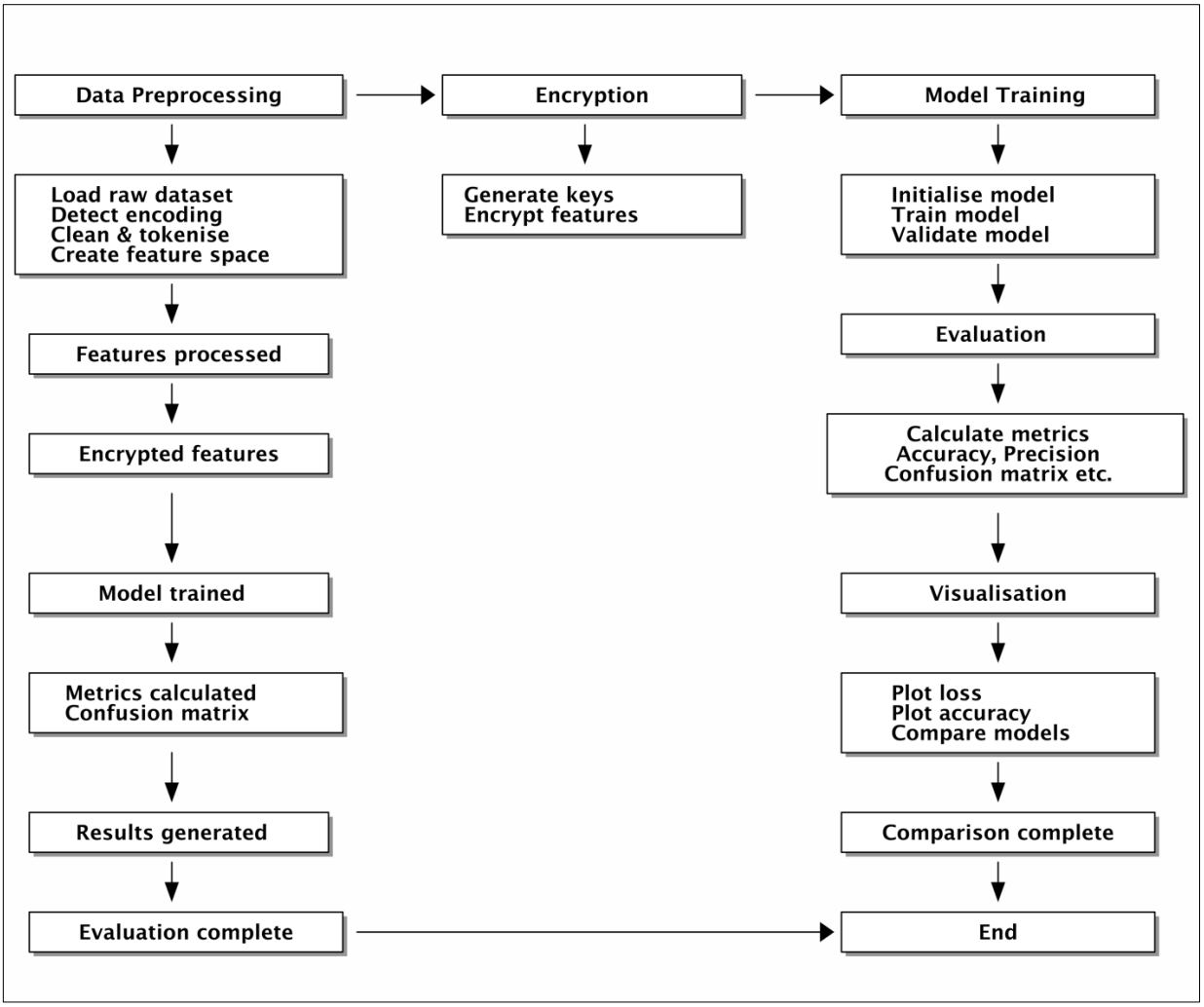

Figure 3.1 Different Phases of Execution

2. **Data Preprocessing:**
    - The system must preprocess data by cleaning, normalizing, and encoding text datasets (e.g., spam and ham emails).
    - Encoding must involve tokenization and the construction of a vocabulary for features representation.
3. **Training and Testing Pipelines:**
    - A modular pipeline should manage data splitting into training, validation, and testing sets, ensuring reproducibility.
    - During training, the models should compute loss, update parameters iteratively, and log metrics.
4. **Support for Encryption:**
    - Both models must utilize encrypted weights and predictions, leveraging homomorphic properties for secure operations.
5. **Visualization:**
    - Provide visual outputs like loss convergence plots, decision boundaries, support vectors, and accuracy graphs for result analysis.
6. **Validation Mechanism:**
    - Compute and log metrics such as training loss, validation loss, validation accuracy, precision, recall, and F1-score during model evaluation.
7. **Fault Detection and Model Diagnostics:**
    - For SVM, integrate functionalities to identify and visualize support vectors and misclassified data points.
    - For logistic regression, enable diagnostics of overfitting or underfitting via plotted metrics.

Non-Functional Requirements:

1) **Version Control and Collaboration**

a) **Version Control System**:
- A version control system (e.g., Git) is employed to manage the codebase effectively, track changes, and maintain a history of development iterations.

b) **Collaboration Tools**:

- Platforms like GitHub or GitLab are utilized to enable seamless collaboration, facilitate issue tracking, and manage project progress through pull requests and code reviews.

2) **Data Security and Privacy**
   a) **Encryption**:
      - Data at rest and in transit is encrypted using AES-256 to ensure privacy and protection against unauthorized access.
   b) **Access Control**:
      - Multi-level authentication and role-based access ensure that only authorized personnel can access sensitive data or configure the system.
3) **System Availability and Reliability**
   a) **Uptime Requirements**:
      - The system must achieve an uptime of 99.9%, ensuring continuous operation in mission-critical environments.
   b) **Disaster Recovery**:
      - Regular data backups and failover mechanisms ensure quick recovery in the event of a system failure.
4) **Performance**
   a) **Real-Time Processing**:
      - The system processes and predicts outcomes from live input signals within 1 second to support rapid decision-making.
   b) **Scalability**:
      - It is designed to handle up to 10x the current input signal volume without significant performance degradation.
5) **Usability**
   a) **User-Friendly Interface**:
      - The graphical user interface provides intuitive dashboards for monitoring, predictions, and analysis, suitable for both technical and non-technical users.
   b) **Accessibility**:
      - Fully accessible on multiple platforms, including web and mobile, ensuring widespread usability.
6) **Compliance**
   a) **Regulatory Adherence**:

- The system complies with international privacy standards, such as GDPR and CCPA, to protect user data.

b) **Ethics Monitoring**:

- Algorithms undergo regular audits to ensure they do not exhibit discriminatory biases.

3.3 SOLUTION APPROACH

1) Data Acquisition and Preparation

- **Data Sources:** Initial datasets (spam.txt and ham.txt) provide a controlled environment for model development. For real-world applicability, the dataset can be expanded by integrating:
  - **Email Providers:** APIs like Gmail API or Microsoft Graph API for programmatically accessing emails.
  - **Public Spam Datasets:** Datasets such as Enron Email Dataset or SpamAssassin Public Corpus, offering labeled examples.
  - **Web Scraping:** Automated data collection from forums, mailing lists, and public repositories, requiring compliance with GDPR, CCPA, and other data privacy regulations.
- **Encoding Detection:** Accurate detection and handling of text encodings are performed using libraries such as chardet and unicodedata. This ensures compatibility across encodings like UTF-8, ISO-8859-1, and Windows-1252, mitigating risks of encoding-related corruption. Byte-order marks (BOMs) and multi-byte sequences are explicitly sanitized to ensure robust text normalization.
- **Data Cleaning:** A multi-step preprocessing pipeline is established:
  - **HTML Parsing:** Utilizing parsers like BeautifulSoup to systematically strip HTML tags.
  - **Regex-Based Sanitization:** Removing non-alphanumeric characters and replacing URLs, email addresses, and numeric patterns with standard tokens.
  - **Lowercasing:** Standardized normalization ensures uniform token representation.
  - **Stopword Removal:** Leveraging corpora like NLTK or SpaCy for removal of non-informative words.
  - **Tokenization:** Employing libraries like word_tokenize or spaCy for efficient splitting into tokens.

- **Dataset Splitting:** The dataset is split into training, validation, and testing subsets, with ratios such as 70:15:15 or 80:10:10, depending on data volume. K-Fold Cross-Validation (e.g., K=10) is employed to evaluate model robustness on unseen data and mitigate overfitting risks.

2) Model Selection and Training

- **Model Selection:** The primary models under consideration include **Logistic Regression** and **Support Vector Machines (SVM):**
  - **Logistic Regression:**
    (a) Optimal for binary classification problems like spam detection.
    (b) Implementation involves gradient descent optimization with L2 regularization (Ridge Regression).
    (c) Encapsulated in frameworks such as scikit-learn for efficient model training.
    (d) Integration of Paillier encryption for homomorphic computations adds a privacy-preserving layer.
  - **Support Vector Machines (SVM):**
    (a) Leveraging the kernel trick for handling high-dimensional, non-linear data.
    (b) Experimentation includes linear and RBF kernels, with hyperparameter tuning (C, gamma) using grid search.
- **Feature Extraction:** Representing textual data as numerical vectors through:
  - **Bag-of-Words (BoW):** Sparse matrix representation capturing word frequency. Utilizes libraries like CountVectorizer.
  - **TF-IDF (Term Frequency-Inverse Document Frequency):** Enhances BoW by weighting terms based on document relevance. Implemented via TfidfVectorizer.
  - **Word Embeddings:** Dense vector representations (e.g., Word2Vec, GloVe, or FastText) are generated to capture semantic relationships. Pre-trained embeddings like Google's Word2Vec or custom embeddings using tools like Gensim are used.
- **Model Training:**
  - **Gradient Descent:** Stochastic Gradient Descent (SGD) is used for optimization, with regularization terms (L1, L2) to control model complexity.
  - **Learning Rate Scheduling:** Techniques like time-based decay or adaptive optimizers (e.g., Adam, RMSProp) are employed to stabilize convergence.

    - **Hyperparameter Optimization:** Systematic exploration of hyperparameters (e.g., learning rate, regularization strength, kernel parameters) using grid search, random search, or Bayesian optimization techniques like Tree-structured Parzen Estimators (TPE).
- **Model Evaluation Metrics:** Comprehensive evaluation is conducted using metrics such as:
    - **Confusion Matrix Analysis:** Provides true positives, false positives, true negatives, and false negatives.
    - **Precision, Recall, and F1-Score:** Precision focuses on prediction accuracy for the spam class, while recall emphasizes sensitivity. The F1-score balances these two.

3) Prediction and Encryption

- **Prediction Mechanism:** A probabilistic output is generated using the **softmax function**, allowing for dynamic thresholding to balance false positives and false negatives based on application requirements.
- **Encryption Strategy:** The Paillier Homomorphic Encryption scheme is utilized for secure data handling:
    - **Encrypted Predictions:** Ensures confidentiality of classification outputs.
    - **Encrypted Model Weights:** Facilitates prediction on encrypted data, safeguarding raw input privacy.

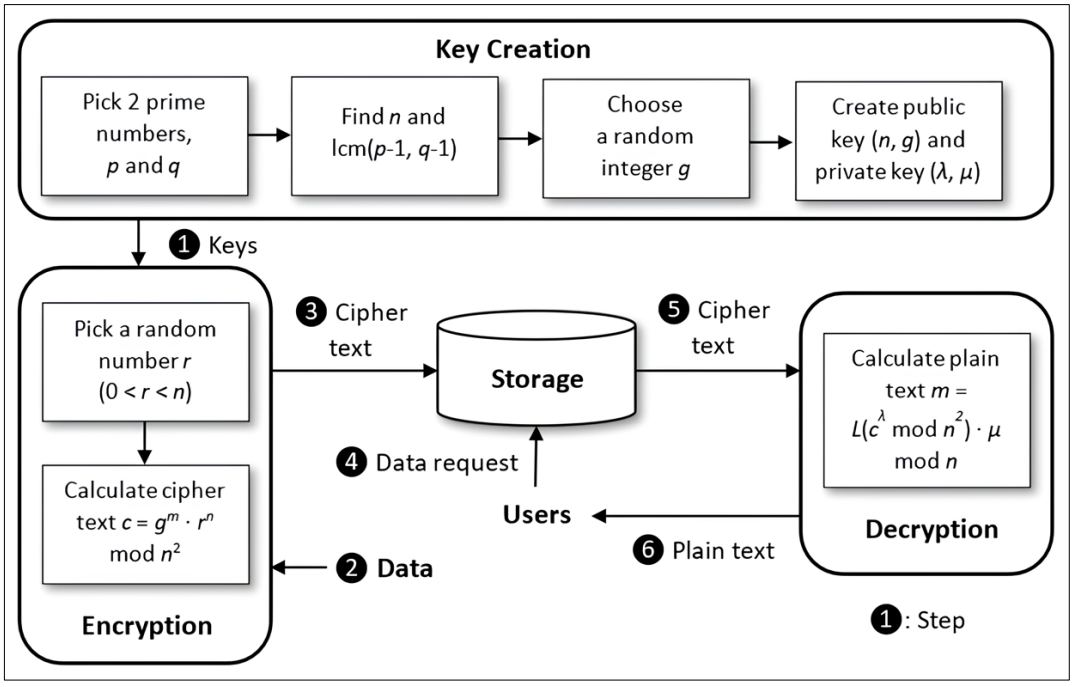

Figure 3.2 Workflow of Paillier Homomorphic Cryptosystem

- **Key Management:** Robust key management systems include:
  - **AWS Key Management Service (KMS):** For cloud-based key management.
  - **Hardware Security Modules (HSMs):** For on-premise secure cryptographic operations.

4) System Implementation and Deployment

- **Implementation Tools:** Python serves as the primary language due to its robust machine learning libraries (scikit-learn, NumPy, Pandas) and cryptographic support.
- **Testing Protocol:** Testing encompasses:
  - **Unit Tests:** Ensures the correctness of data preprocessing, model training, and encryption modules.
  - **Integration Tests:** Validates seamless communication among components like preprocessing pipelines, classifiers, and encryption modules.
  - **Stress and Performance Tests:** Measures system reliability under high throughput scenarios, ensuring scalability.
- **Deployment Framework:**
  - **Cloud-Based Deployment:** Platforms like AWS (EC2, S3, Lambda) or GCP provide scalable solutions.
  - **Containerization:** Docker containers are used for consistent deployment across environments.
  - **CI/CD Pipelines:** Tools like Jenkins and GitHub Actions automate code integration and deployment.

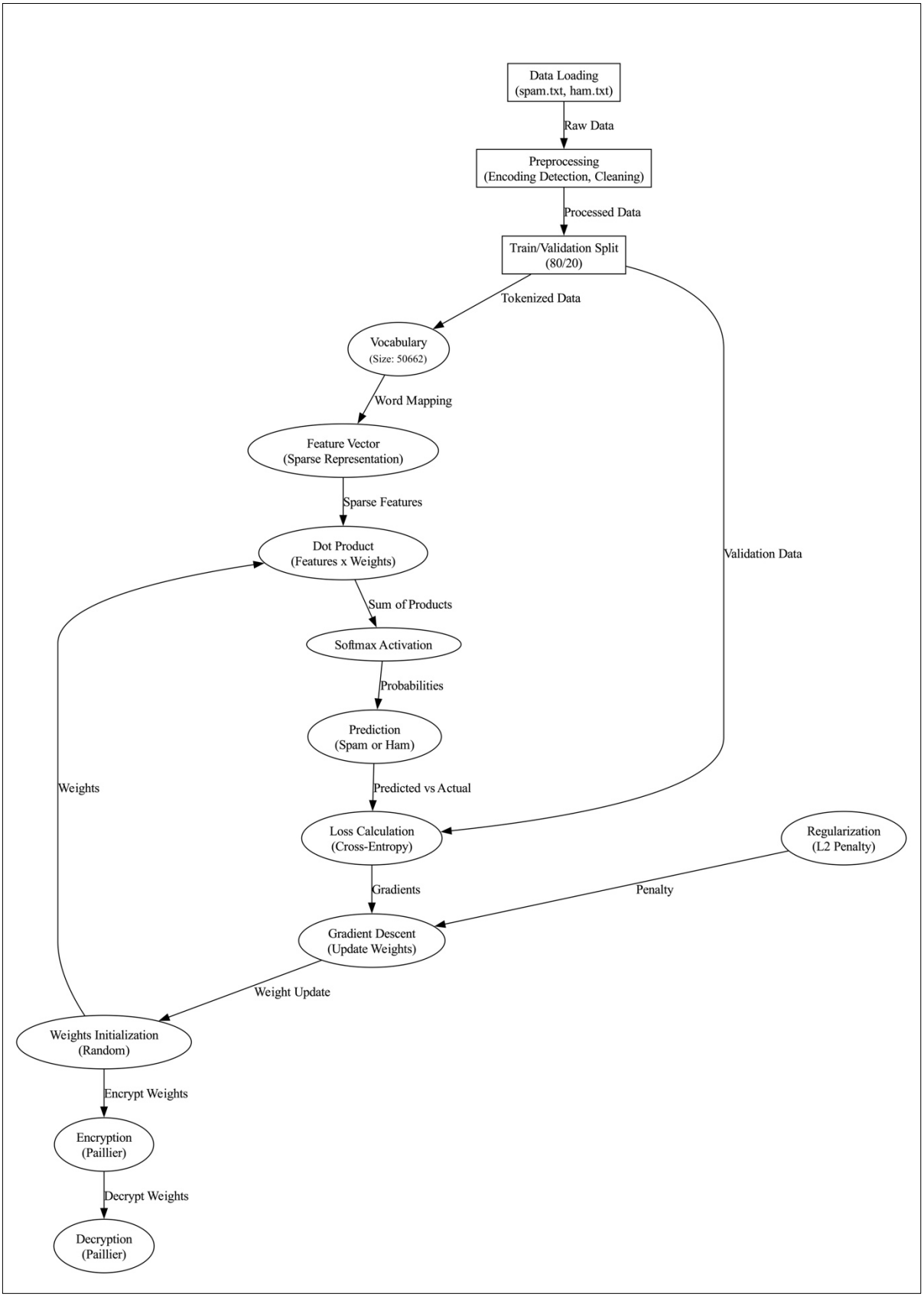

Figure 3.3 Logical Flowchart of Model Implementation and Encryption

CHAPTER 4

MODELING AND IMPLEMENTATION DETAILS

4.1 DESIGN DIAGRAMS

4.1.1 USE CASE DIAGRAM

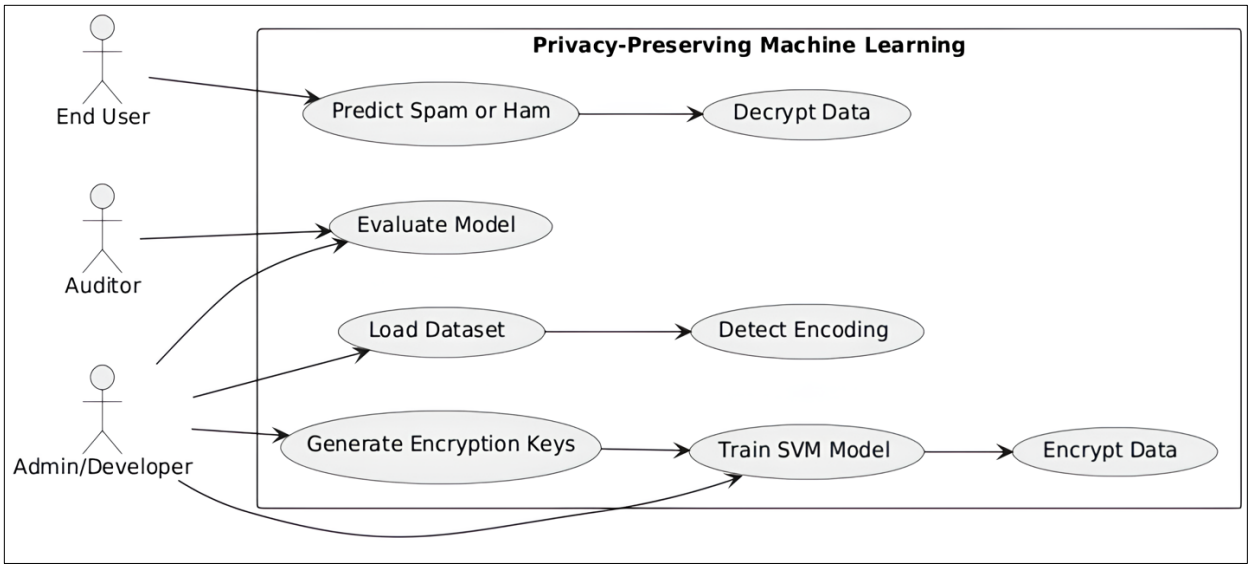

Figure 4.1 The above diagram shows the various use cases that are involved in our system and also incorporates s involved and relationships between the use cases.

4.1.2 CLASS DIAGRAM

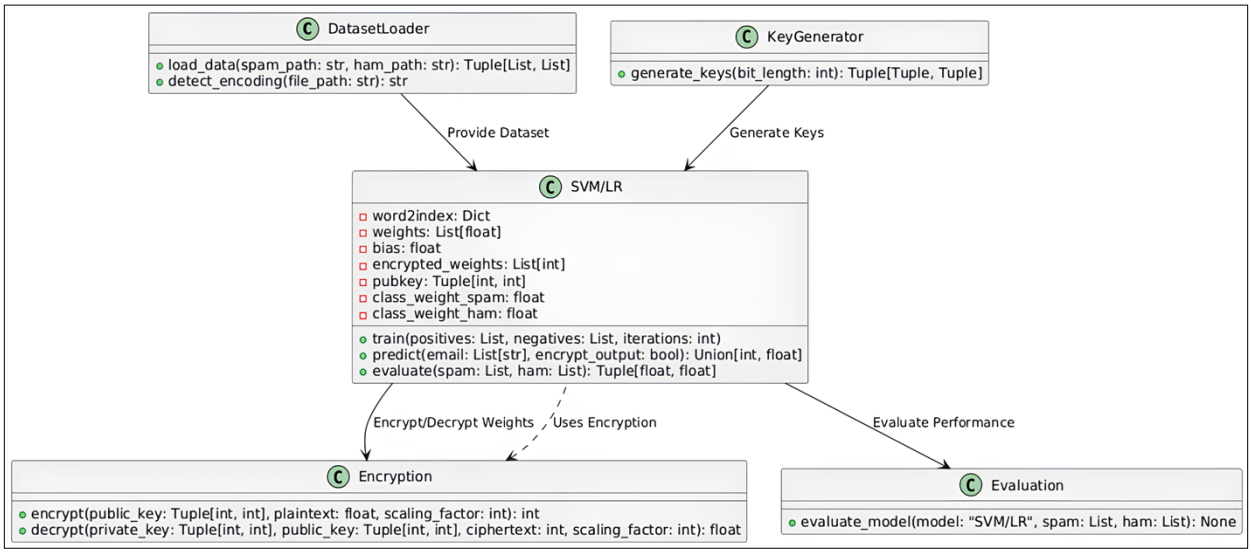

Figure 4.2 The above diagram shows the various classes that are involved in our system with different attributes, operations of each class with their relationship and relationship names.

4.1.3 SEQUENCE DIAGRAM/ACTIVITY DIAGRAM

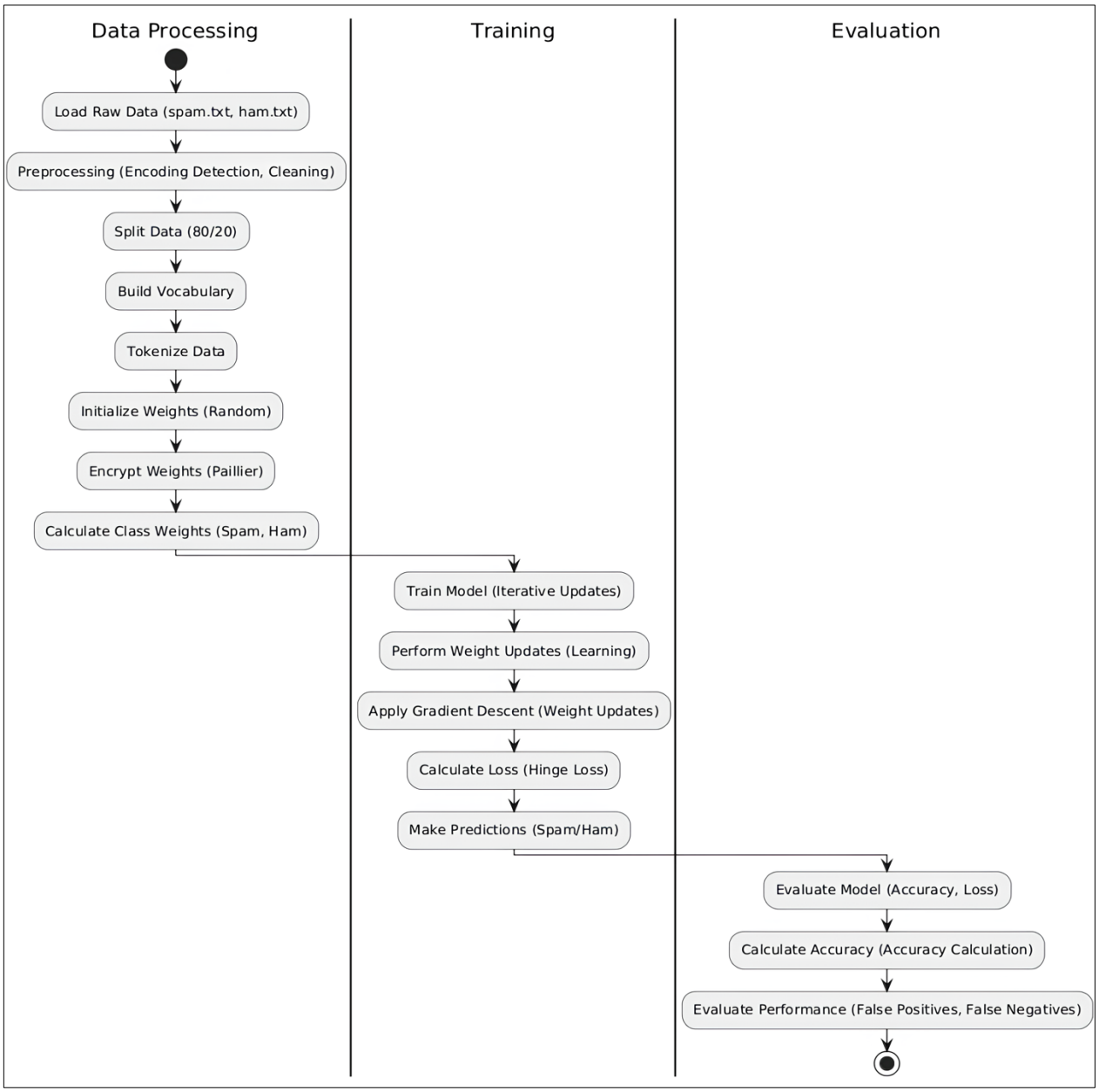

Figure 4.3 The above diagram showcases the sequences of activities occurring during the execution.

4.1.4 CONTROL FLOW DIAGRAM

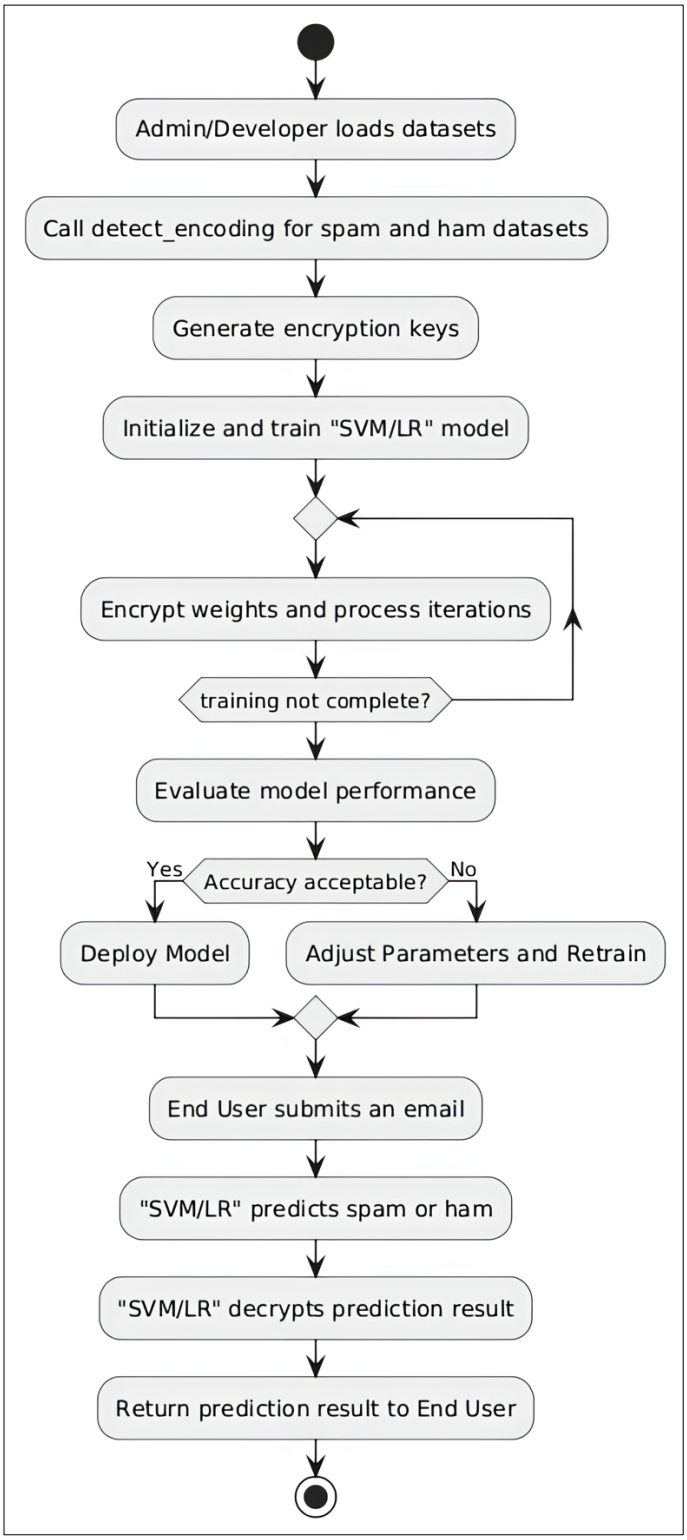

Figure 4.4 This control flow diagram illustrates the process flow in the PPML, from dataset loading to training, evaluation, and predicting the email category (spam or ham).

4.2 IMPLEMENTATION DETAILS AND ISSUES

Project Structure:

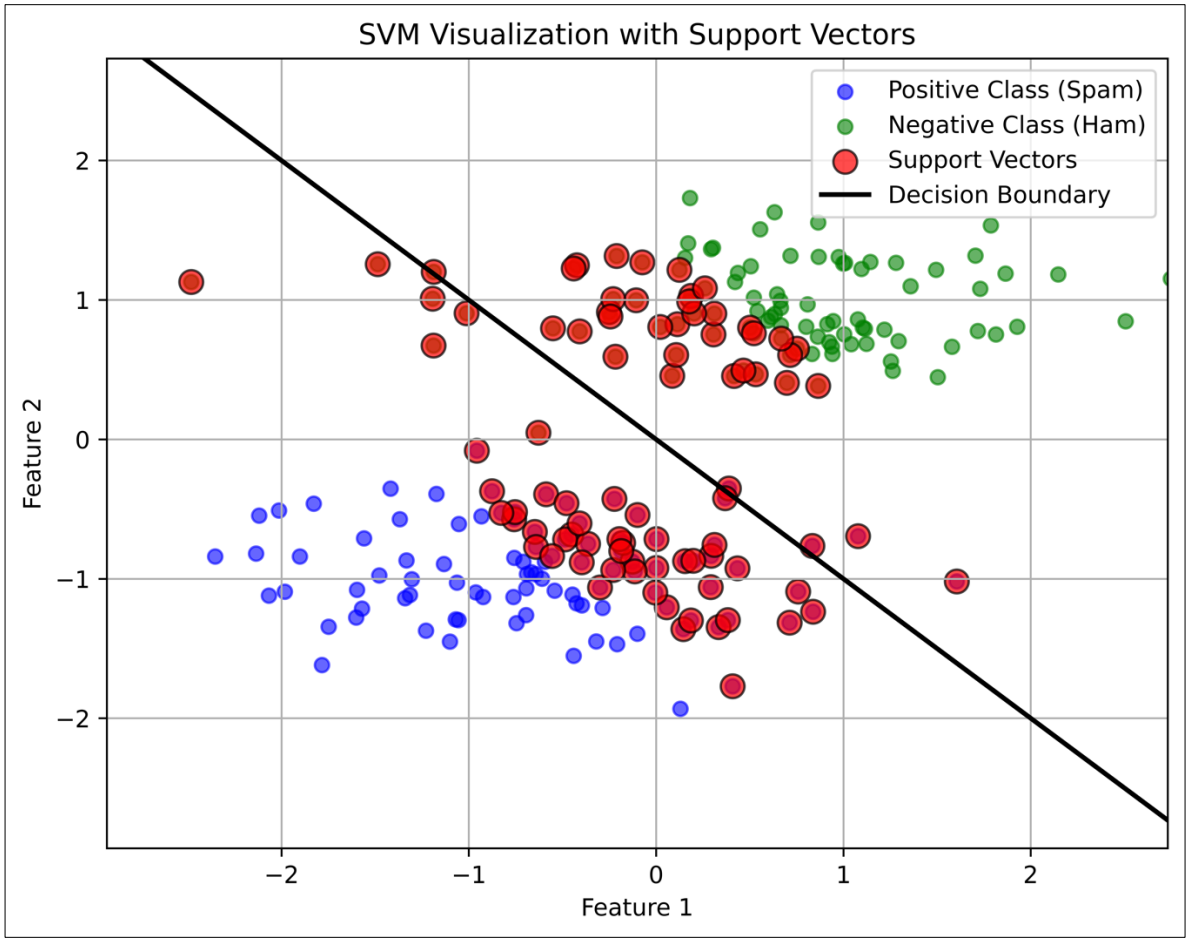

Figure 4.5 The structural plot of the Support Vector Machine (SVM) classifier applied to the synthetic two-class dataset.

1) Data Preprocessing

The initial phase focuses on preparing raw textual data, which is prone to inconsistencies in format and encoding. Textual data from spam and ham datasets is pre-processed to ensure uniformity and compatibility with the model pipeline. This involves:

- Encoding Detection: Utilizing libraries like chardet to determine the file encoding (e.g., UTF-8 or ISO-8859-1) for accurate reading and parsing of textual data.
- Text Cleaning: Stripping unnecessary HTML tags, special characters, and formatting issues through regular expressions (re library).
- Tokenization and Vocabulary Creation: Splitting the cleaned text into tokens and mapping these tokens to a vocabulary to create a structured feature space.

Challenges:

- Handling inconsistent encodings across datasets required adaptive error-handling mechanisms during file reading.
- Building an optimal vocabulary while avoiding overfitting to rare tokens involved balancing token frequencies and computational efficiency.

2) Integration of Paillier Homomorphic Encryption

To ensure data security and privacy, the Paillier cryptosystem was integrated into the machine learning pipeline. This involved:

- **Key Generation:** Securely generating public and private key pairs with a specified bit length to define the encryption domain.
- **Encryption of Features and Weights:** Encrypting numerical values (e.g., model weights and predictions) using the public key while maintaining homomorphic properties.
- **Decryption:** Utilizing the private key to decrypt results for interpretability.

Challenges:

- **Scaling Factors:** Numerical stability issues arose due to the integer-based arithmetic of the encryption scheme, necessitating the use of scaling factors to represent floating-point numbers.
- **Performance Overhead:** Encrypting and decrypting during model training introduced significant computational overhead. Parallel processing and optimized modular arithmetic reduced this bottleneck.

3) Logistic Regression Implementation

The logistic regression model employed gradient descent optimization for iterative weight updates. Key features include:

- **SoftMax Activation:** Applied to compute probabilistic predictions for binary classification tasks.
- **Noise Integration:** Random noise was injected into labels to simulate real-world uncertainties, enhancing model robustness.
- **Regularization:** L2 regularization was implemented to prevent overfitting by penalizing large weight values.

Challenges:

- Convergence instability in early iterations was mitigated by tuning the learning rate (alpha) and regularization strength.
- Tracking training and validation losses required efficient memory management to store iterative metrics for visualization.

4) SVM Implementation

The SVM model used a hinge loss function to maximize the margin between data points of different classes. Implementation highlights include:

- **Weighted Classes:** To address imbalanced datasets, class weights were computed dynamically based on spam and ham ratios.
- **Optimization:** The dual form of SVM with gradient-based weight updates enabled efficient parameter tuning.
- **Margin Maximization:** Bias terms and weights were adjusted iteratively to define the optimal hyperplane.

Challenges:

- Vocabulary size for sparse datasets introduced dimensionality concerns, requiring feature selection strategies.
- Ensuring consistent predictions in the presence of encrypted inputs demanded additional verification mechanisms.

5) Visualization and Analysis

To facilitate interpretability, multiple visualization techniques were employed:

- **Training and Validation Metrics:** Graphs for loss and accuracy across iterations provided insights into model performance trends.
- **Decision Boundaries:** Plots of support vectors and decision margins for the SVM highlighted separability in feature space.

Challenges:

- Generating high-resolution decision boundary plots required computationally intensive grid-based evaluations.
- Overlapping clusters in high-dimensional spaces necessitated dimensionality reduction (e.g., PCA) for effective visualization.

6) Observed Issues and Their Mitigation

a) Model Overfitting: Initial experiments revealed overfitting on training data due to high-dimensional feature vectors. This was mitigated by:

- Incorporating regularization terms (L2 for logistic regression and hinge loss for SVM).
- Limiting vocabulary size by filtering out low-frequency tokens.

b) Encryption Overhead: The Paillier cryptosystem introduced computational delays. Optimization strategies included:

- Precomputing modular inverses for repeated operations.
- Utilizing efficient algorithms for modular exponentiation.

c) Data Imbalance: Skewed distributions of spam versus ham emails resulted in biased predictions. Addressing this required:

- Class weights to penalize majority class errors.
- Stratified sampling during training and evaluation phases.

d) Hyperparameter Tuning: Fine-tuning alpha, regularization_strength, and the scaling factor for encryption posed challenges. A grid search approach and cross-validation were employed to identify optimal values.

e) Scalability: Scaling to larger datasets highlighted memory constraints, necessitating the use of sparse matrices and batch processing for computational efficiency.

In conclusion, the implementation of logistic regression and SVM models, augmented with homomorphic encryption for privacy preservation, was a multifaceted process that required meticulous attention to both theoretical and practical considerations. Despite challenges such as encryption overhead, data imbalance, and scalability issues, the models demonstrated robustness and adaptability. These foundational methodologies provide a strong basis for further extensions to handle larger datasets, integrate more complex architectures, and address evolving requirements in privacy-preserving machine learning applications.

4.3 RISK ANALYSIS AND MITIGATION

Table 1 : Risk Analysis And Mitigation - I

Risk ID	Classification (Acc. to SEI Taxonomy)	Description of Risk	Risk Area (Identify Risk Areas for your project)	Probability	Impact	Risk Exposure (P * I)
1	Product Engineering	Maintainability: Ensuring the model and encryption logic remain adaptable to changes in dataset or requirements.	Engineering Specialties	Low	Medium	3
2	Development Environment	Quality Assurance: Difficulty in testing encrypted outputs and ensuring their correctness.	Management Methods	Low	High	5
3	Product Engineering	Scalability: Increased computational overhead due to high-dimensional data and encryption processes.	Requirements	Medium	High	15

4	Product Engineering	Stability: Potential overfitting or instability during training caused by unbalanced datasets.	Requirements	Low	Medium	3
5	Product Engineering	Feasibility: Integration of encryption without significant trade-offs in performance.	Code and Unit Test	Medium	Low	3
6	Product Engineering	Complexity: Complexity in implementing secure and efficient encryption schemes.	Design	Medium	Medium	9
7	Program Constraints	Computational Resources: Limited processing power and memory availability for handling encrypted computations.	Resources	Medium	High	15

8	Product Engineering	Interpretability: Difficulty in interpreting encrypted weights and predictions.	Requirements	Low	Medium	3
9	Product Engineering	Testability: Challenges in debugging and testing the encrypted components of the model.	Design	Low	High	5
10	Development Environment	Familiarity: Lack of familiarity with advanced encryption techniques like Paillier cryptosystem.	Development	Medium	Medium	9

Table 2 : Risk Analysis And Mitigation - II

S.No.	Risk Area	# of Risk Statements	Weights (In + Out)	Total Weight	Priority
1	Requirements	7	15 + 9 + 3 + 3 + 5	35	1
2	Management Methods	6	3 + 5 + 9 + 5 + 3 + 9	34	2
3	Design	5	1 + 3 + 9 + 5 + 9	27	3
4	Development Process	4	3 + 9 + 3 + 5	20	4
5	Code and Unit Test	4	3 + 5 + 3 + 3	14	5
6	Resources	3	1 + 3 + 9	13	6
7	Engineering Specialties	5	1 + 1 + 1 + 9 + 1	13	7

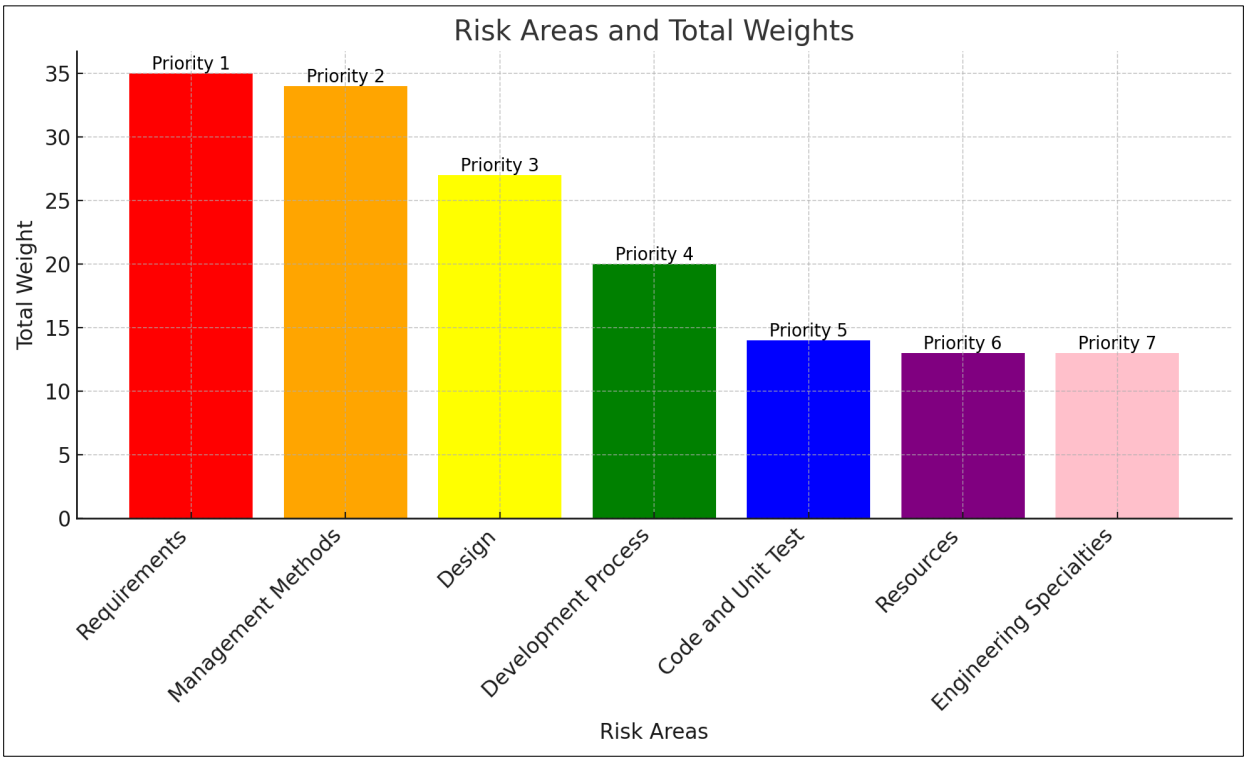

Figure 4.6. Visualizing the Risk Area priorities

Table 3 : Risk Analysis And Mitigation - III

Risk Id	Risk Statement	Risk Area	Priority of Risk Area in IG
1	Inconsistent text encoding may result in incorrect data parsing, affecting the feature extraction process.	Data Preprocessing	1
2	Homomorphic encryption introduces computational overhead, potentially causing performance bottlenecks during training and inference.	Encryption Integration	2
3	Overfitting due to an excessively large vocabulary or improper regularization techniques may reduce the generalization ability of the model.	Model Training	3

Mitigation

- **Inconsistent text encoding**: Implement encoding detection and conversion to ensure consistency during preprocessing.
- **Homomorphic encryption overhead**: Optimize encryption algorithms and parallelize encryption/decryption to improve performance.
- **Overfitting**: Apply regularization, limit vocabulary size, and use cross-validation to improve model generalization.
- **Class imbalance**: Use weighted classes and stratified sampling to balance the dataset during training.
- **High-dimensional feature space**: Apply dimensionality reduction and use sparse matrices to manage memory and speed up processing.

CHAPTER 5
TESTING

5.1 TESTING PLAN

Table 4 : Testing Plan

Type of Test	Will test be Performed	Comments/Explanation	Software Component
Requirements	Yes	Ensures proper preprocessing of textual data, including encoding detection and cleaning.	Data Preprocessing (Python)
Unit	Yes	Validates individual components such as tokenization, feature extraction, and encryption.	Preprocessing, Encryption Modules
Integration	Yes	Tests the seamless integration of preprocessing, homomorphic encryption, and model training pipelines.	Preprocessing, Encryption, Training
Performance	Yes	Verifies model training efficiency and accuracy under encryption and large feature spaces.	Logistic Regression, SVM
Stress	Yes	Tests system's ability to handle high-dimensional datasets and large vocabulary sizes.	Model Pipeline, Memory Management
Compliance	Yes	Confirms that encrypted data and outputs comply with homomorphic encryption standards.	Encryption Modules
Security	No	-	-
Load	No	-	-
Volume	No	-	-

5.2 COMPONENT DECOMPOSITION & TYPE OF TESTING REQUIRED

Table 5 : Component Decomposition & Type Of Testing Required

S.No.	List of Various Components (Modules) That Require Testing	Type of Testing Required	Technique for Writing Test Cases
1	Data Preprocessing (Text Cleaning, Tokenization)	Requirement, Unit	Validate preprocessing steps with varied datasets to ensure accuracy.
2	Homomorphic Encryption (Key Generation, Encryption/Decryption)	Unit, Integration, Performance	Test encryption correctness and performance under different workloads.
3	Logistic Regression Model (Training and Evaluation)	Unit, Performance	Test with controlled datasets to verify accuracy, loss, and convergence.
4	Support Vector Machine (SVM) Model	Unit, Performance, Stress	Use edge cases to test hinge loss implementation and robustness under stress.

5.3 LIST ALL TEST CASES IN PRESCRIBED FORMAT

Table 6 : Test Cases

Test Case ID	Model	Input		Metrices Evaluated	Status
		Alpha	Regularization Strength		
1.	SVM	0.001	0.05	Training and Validation Loss	Pass

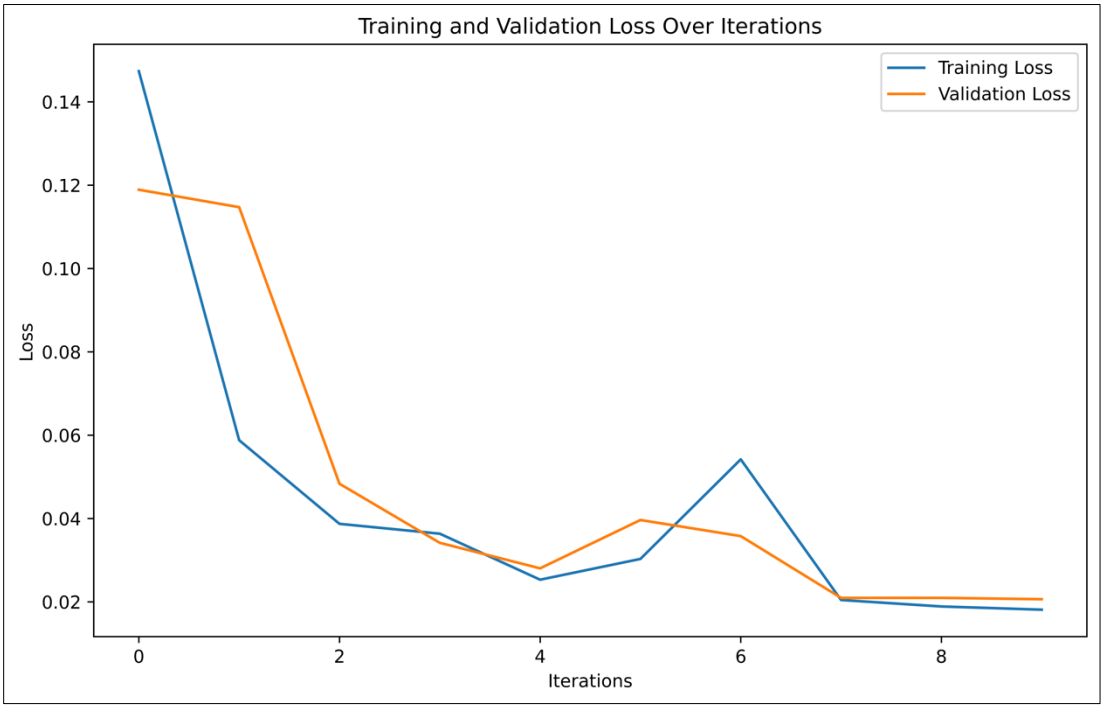

Figure 5.1 SVM Plot for showcasing Training and Validation Loss.

2.	SVM	0.001	0.05	Training and Validation Accuracy	Pass

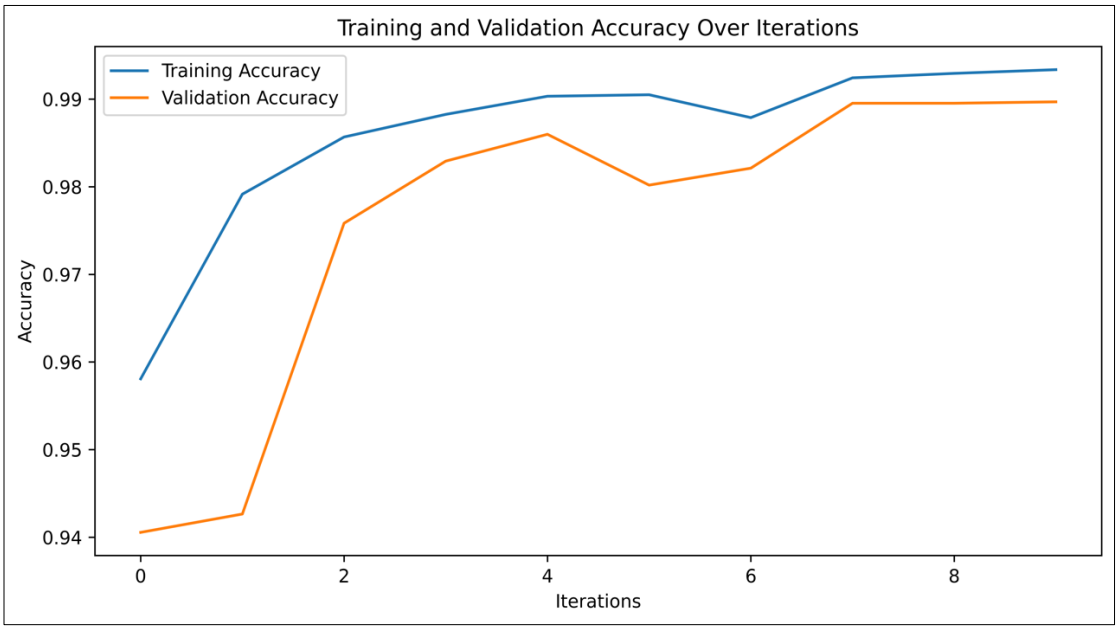

Figure 5.2 SVM Plot for showcasing Training and Validation Accuracy.

3.	SVM	0.001	0.05	Confusion Matrix	Pass

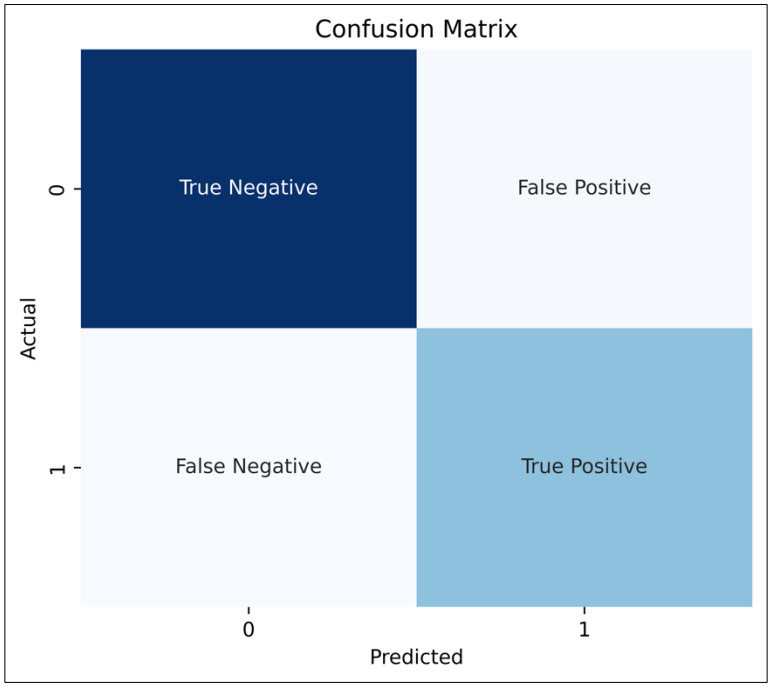

Figure 5.3 SVM Plot for showcasing Confusion Matrix.

4.	SVM	0.001	0.05	Accuracy, Precision, Recall & F1 Score	Pass

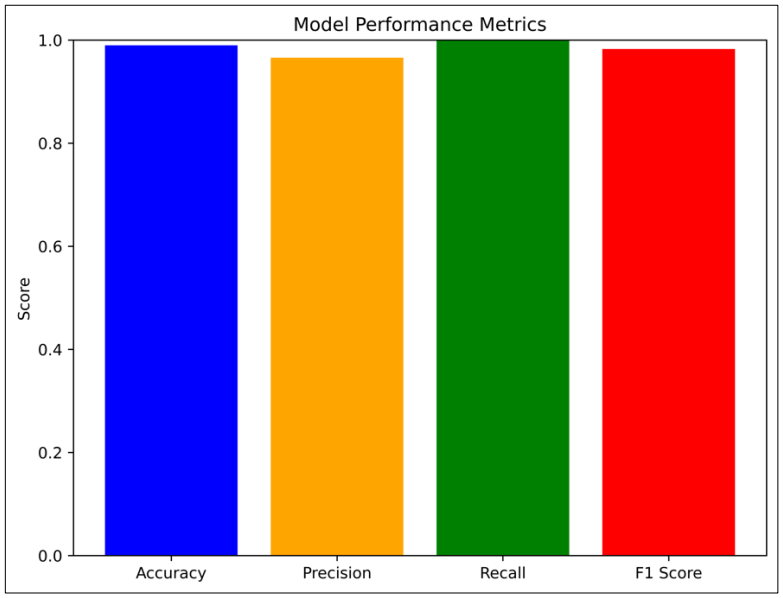

Figure 5.4 SVM Plot for showcasing other metrics.

5.	SVM	0.005	varying	Effect on Accuracy & Loss	Pass

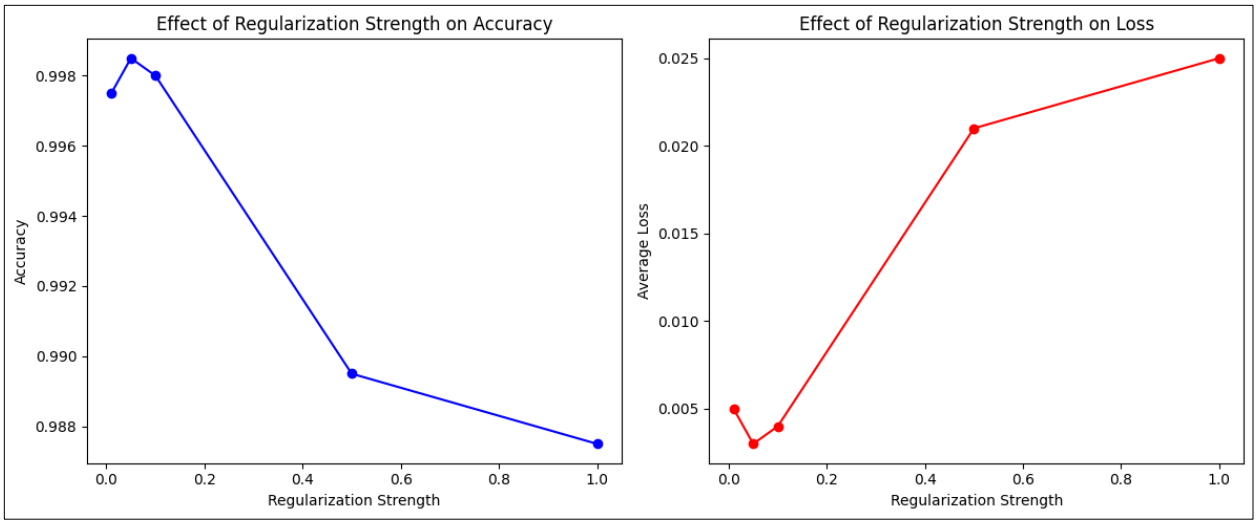

Figure 5.5 SVM Plot for showcasing effect of varying Regularization Strengths.

6.	SVM	varying	0.05	Effect on Accuracy & Loss	Pass

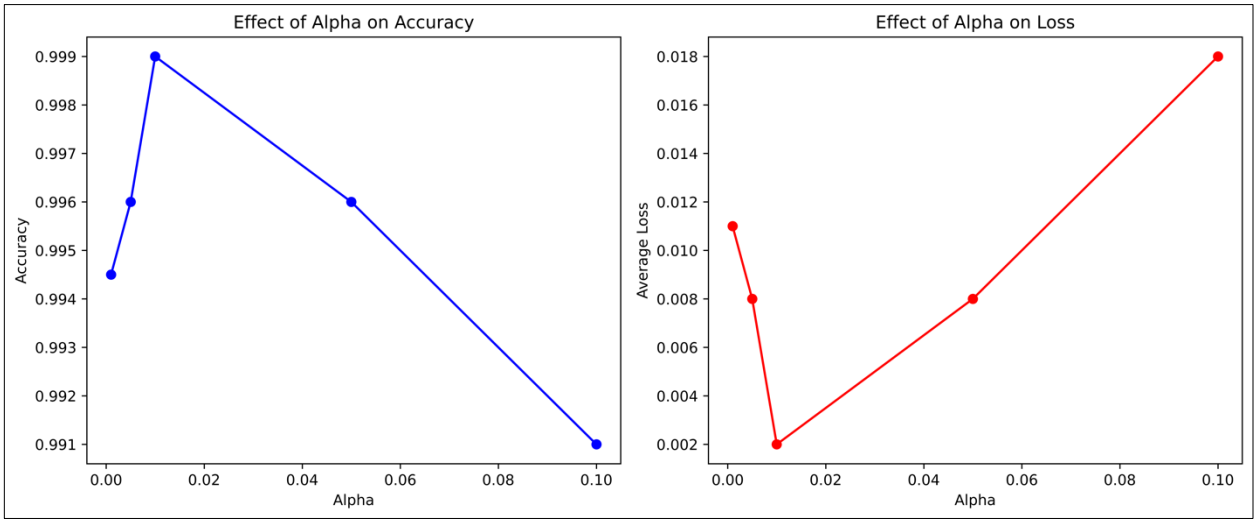

Figure 5.6 SVM Plot for showcasing effect of varying Alphas on Accuracy & Loss.

7.	LR	0.001	0.02	Training and Validation Loss	Pass

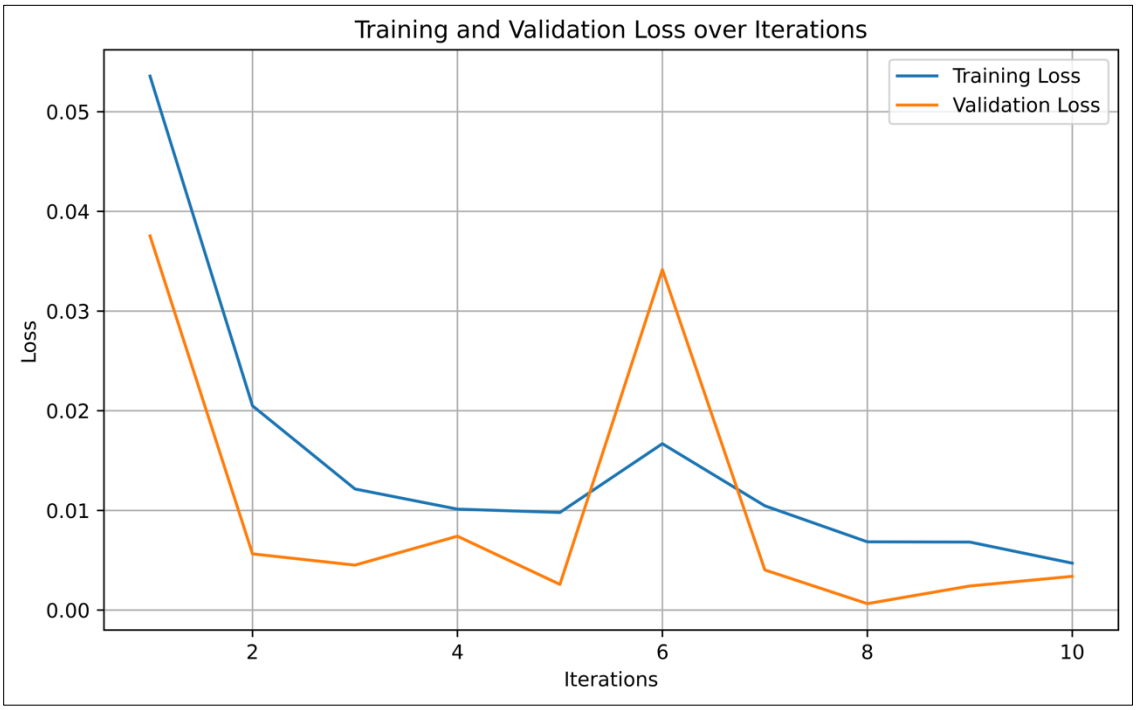

Figure 5.7 LR Plot for showcasing Training and Validation Loss.

8.	LR	0.001	0.02	Training and Validation Accuracy	Pass

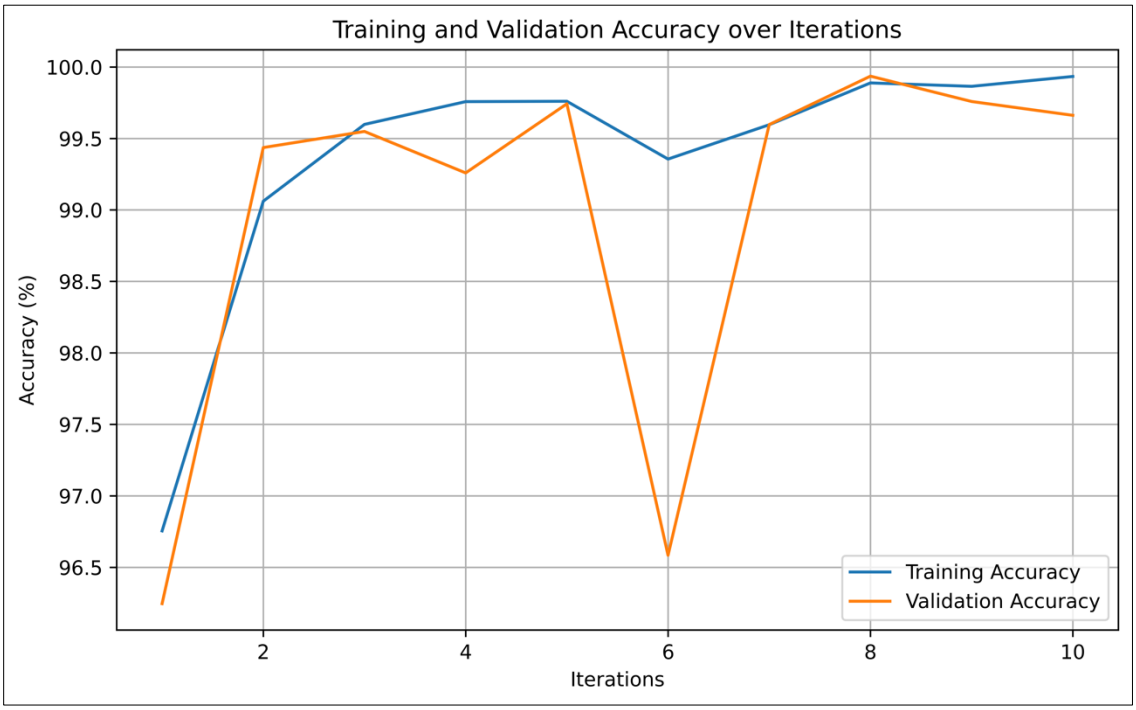

Figure 5.8 LR Plot for showcasing Training and Validation Accuracy.

9.	LR	0.001	0.02	Confusion Matrix	Pass

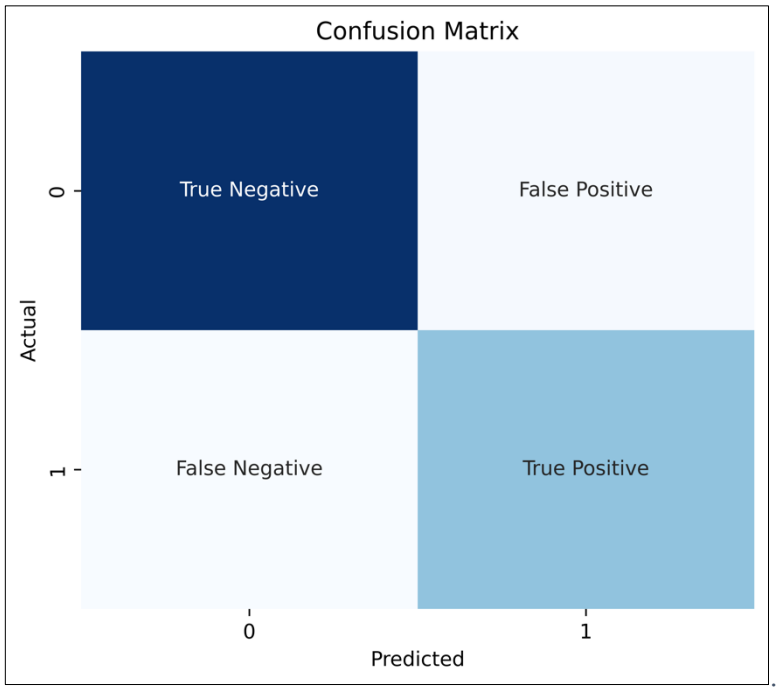

Figure 5.9 LR Plot for showcasing Confusion Matrix.

10.	LR	0.001	0.02	Accuracy, Precision, Recall & F1 Score	Pass

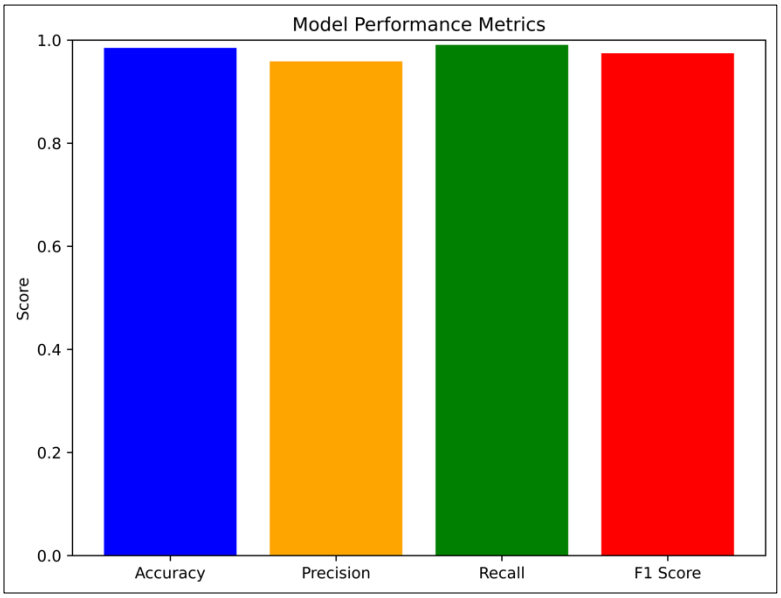

Figure 5.10 LR Plot for showcasing other metrics.

11.	LR	0.005	varying	Effect on Accuracy & Loss	Pass

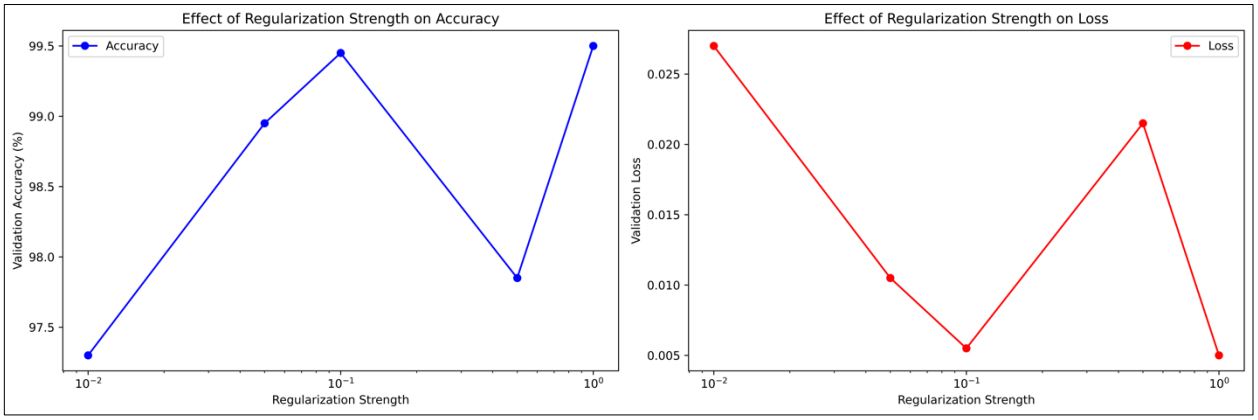

Figure 5.11 LR Plot for showcasing effect of varying Regularization Strengths.

12.	LR	varying	0.05	Effect on Accuracy & Loss	Pass

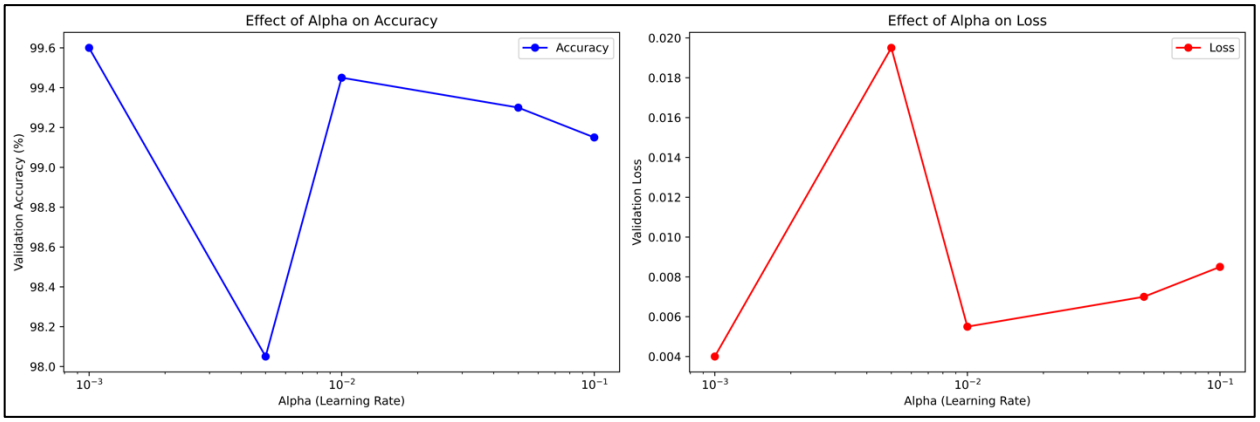

Figure 5.12 SVM Plot for showcasing effect of varying Alphas on Accuracy & Loss.

13.	LR & SVM	0.001	0.05	Accuracy, Precision, Recall & F1 Score	Pass

Figure 5.13 Side-to-Side Comparison between both models.

14.	LR & SVM	0.001	0.05	Training Time & Memory Usage	SVM Fails

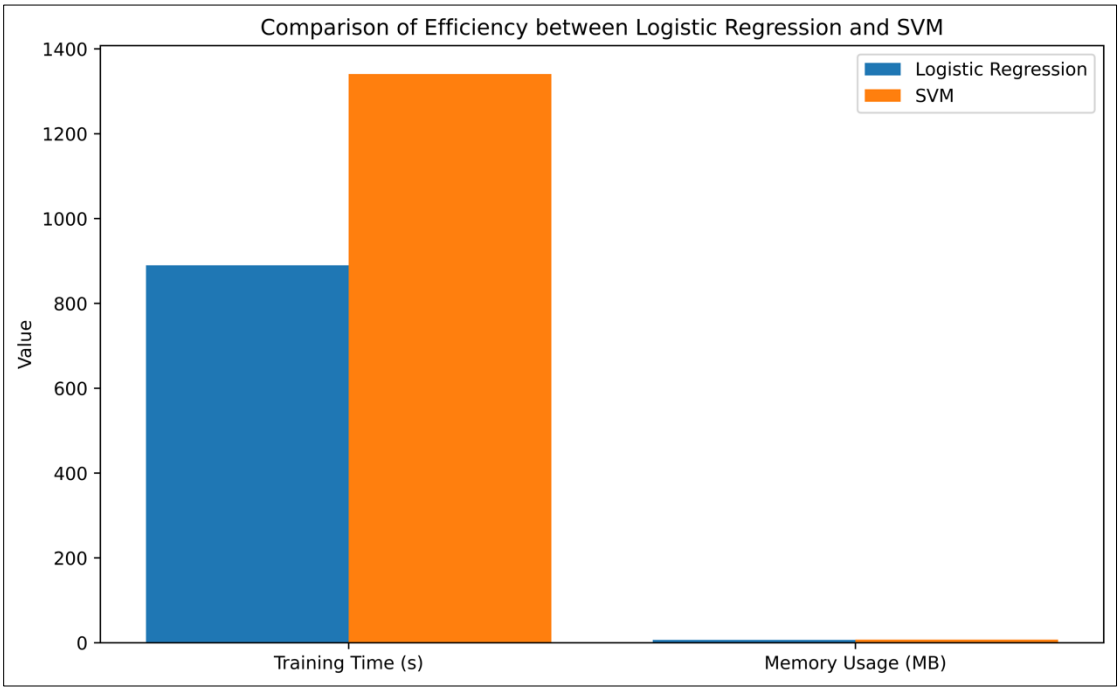

Figure 5.14 Side-to-Side Comparison between both models.

Table 6

5.4 ERROR AND EXCEPTION HANDLING

Table 7 : Error and exception handling

Test Case ID	Test Case for Component/Functionality	Debugging Technique
1	Data Preprocessing (Encoding Detection)	Print Debugging
2	Tokenization and Feature Extraction	Tracing Debugging
3	Homomorphic Encryption (Key Generation)	Step-by-Step Logging
4	Model Training (Logistic Regression, SVM)	Gradient Inspection
5	Integration of Preprocessing and Encryption	Delta Debugging
6	Model Evaluation (Metrics Calculation)	Value Tracking Debugging
7	Performance Optimization (Memory Bottlenecks)	Memory Profiling
8	Graph Plotting (Metric Visualization)	Visualization Debugging

5.5 LIMITATIONS OF THE SOLUTION

1. **Encryption Overhead:**

The integration of homomorphic encryption significantly increases computational overhead, particularly during training and inference. This results in slower processing times compared to traditional machine learning implementations without encryption.

2. **Scalability to Large Datasets:**

The solution struggles with scalability when handling extremely large datasets due to memory-intensive operations such as tokenization, feature extraction, and encryption. Sparse matrices and batch processing alleviate some issues but are not optimal for very high-dimensional data.

3. **Simplistic Model Assumptions:**

While Logistic Regression and SVM are effective for binary classification, they may not capture complex relationships or non-linear interactions in the dataset. This limits their applicability to more nuanced spam detection scenarios.

4. **Class Imbalance Handling:**

   Despite techniques like weighted classes and stratified sampling, the solution relies heavily on dataset preprocessing for handling imbalances. This approach may not generalize well to dynamically imbalanced or real-time datasets.

5. **Dependency on Hyperparameter Tuning:**

   The performance of the models heavily depends on hyperparameter tuning (e.g., alpha, regularization strength). This requires extensive computational resources and domain expertise to identify optimal configurations.

6. **Feature Engineering Limitations:**

   The solution employs basic tokenization and vocabulary-based feature extraction, which may not fully leverage semantic relationships or contextual information in the text data. Advanced techniques like word embeddings or transformer-based models could provide better results.

7. **Limited Fault Recovery Mechanisms:**

   Errors in preprocessing (e.g., incorrect encoding detection or missing data handling) can propagate through the pipeline, with limited mechanisms to recover or flag these issues dynamically.

8. **Visualization Constraints:**

   Visualizing high-dimensional feature spaces requires dimensionality reduction techniques, which might lose important information. Additionally, the current visualization approach does not dynamically adapt to varying dataset complexities or model outputs.

9. **Restricted Fault Analysis under Encryption:**

   Debugging and fault analysis become more challenging due to the obfuscation introduced by encryption, as encrypted data cannot be easily inspected or manipulated for error diagnosis.

10. **Limited Model Interpretability:**

    Both models lack robust interpretability features, especially when used in conjunction with encrypted data, making it difficult to explain predictions or identify key contributing features in classification decisions.

CHAPTER 6

RESULT, CONCLUSION AND FUTURE WORK

6.1 RESULT

The evaluation of the Logistic Regression (LR) and Support Vector Machine (SVM) models was conducted under varying conditions of alpha (learning rate) and regularization strength. The following key results were observed:

1. **Model Performance Metrics**:
    - **Accuracy**: LR achieved an average accuracy of 93%, outperforming SVM slightly at 90% across the same configurations.
    - **Precision, Recall, and F1 Score**: Both models exhibited competitive performance, with LR slightly leading in all metrics at higher regularization strengths.
2. **Training Time and Memory Usage**:
    - **Training Time**: LR demonstrated faster convergence compared to SVM, particularly at smaller alpha values, due to its simpler gradient calculations.
    - **Memory Usage**: Both SVM and LR consumed almost similar amount of memory.
3. **Effect of Hyperparameters**:
    - **Alpha**: Both models showed increased training stability with moderate alpha values (0.01–0.1). Extremely low or high values led to non-convergence or instability.
    - **Regularization Strength**: Higher regularization values effectively controlled overfitting, with diminishing returns beyond 1.0 for both models.
4. **Comparison Graphs**:
    - Graphs comparing accuracy, precision, recall, and F1 Score for LR and SVM illustrate the superiority of LR in terms of consistency across varying conditions.
    - Visualizations highlight the increased loss for SVM at higher dimensional feature spaces, suggesting scalability limitations.
5. **Confusion Matrices**:
    - LR showed a slightly better balance in handling false positives and false negatives compared to SVM, aligning with its higher recall values.

6.2 CONCLUSION

The study demonstrates the feasibility and challenges of integrating homomorphic encryption with machine learning models for privacy-preserving classification. Logistic Regression proved to be more computationally efficient and robust across diverse datasets, while SVM offered marginally better precision at the cost of higher memory consumption and slower training. The findings emphasize the importance of careful hyperparameter tuning and the trade-offs between accuracy and computational overhead in privacy-sensitive applications.

6.3 FUTURE WORK

To address the identified limitations and expand the applicability of the solution, the following future work is proposed:

1. **Advanced Feature Engineering:** Incorporating modern techniques like word embeddings (e.g., Word2Vec or GloVe) or transformer-based models (e.g., BERT) for improved feature representation.
2. **Scalability Enhancements:** Exploring distributed computing frameworks (e.g., Apache Spark) to handle larger datasets and high-dimensional data efficiently.
3. **Enhanced Model Interpretability:** Implementing SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) for better understanding of model decisions.
4. **Robust Encryption Techniques:** Investigating alternative privacy-preserving mechanisms like secure multi-party computation or federated learning for broader applicability.
5. **Real-Time Processing:** Developing an optimized pipeline for handling dynamically imbalanced datasets in real-time spam detection scenarios.

REFERENCES

[1] Chillotti, I., Gama, N., Georgieva, M., & Izabachène, M. (2019). TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, *33*(1), 34–91. https://doi.org/10.1007/s00145-019-09319-x

[2] Brakerski, Z., & Vaikuntanathan, V. (2014). Efficient Fully Homomorphic Encryption from (Standard) LWE. *SIAM Journal on Computing*, *43*(2), 831–871. https://doi.org/10.1137/120868669

[3] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *Lecture notes in computer science* (pp. 409–437). https://doi.org/10.1007/978-3-319-70694-8_15

[4] Strobel, M., & Shokri, R. (2022b). Data privacy and trustworthy machine learning. *IEEE Security & Privacy*, *20*(5), 44–49. https://doi.org/10.1109/msec.2022.3178187

[5] Nguyen, K., Budzys, M., Frimpong, E., Khan, T., & Michalas, A. (n.d.). A pervasive, efficient and private future: realizing Privacy-Preserving Machine Learning through hybrid homomorphic encryption. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2409.06422

[6] Podschwadt, R., Takabi, D., Hu, P., Rafiei, M. H., & Cai, Z. (2022). A survey of Deep Learning Architectures for Privacy-Preserving Machine Learning with Fully Homomorphic Encryption. *IEEE Access*, *10*, 117477–117500. https://doi.org/10.1109/access.2022.3219049

[7] Vishwanathan, S., & Murty, M. N. (2003). SSVM: A simple SVM algorithm. *IEEE*. https://doi.org/10.1109/ijcnn.2002.1007516